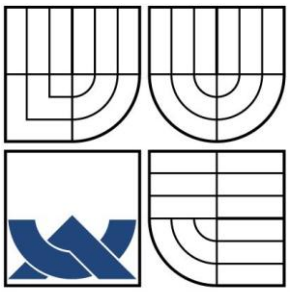
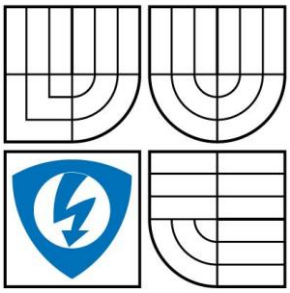


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKACNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

KOMUNIKAČNÍ PROTOKOL PRO MESSENGER A JEHO ZABEZPEČENÍ

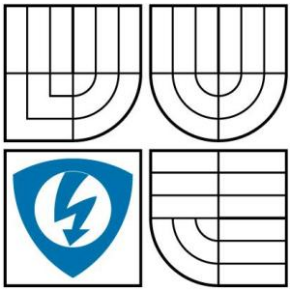
COMMUNICATION PROTOCOL FOR MESSENGER AND HIS SECURITY

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — ONDŘEJ BÍLÝ
AUTHOR

VEDOUCÍ PRÁCE — ING. PETR MLÝNEK
SUPERVISOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor

Teleinformatika

Student:	Ondřej Bílý	***ID:***	97986
Ročník:	3	***Akademický rok:***	2008/2009

NÁZEV TÉMATU:

Komunikační protokol pro messenger a jeho zabezpečení

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte komunikační protokol mezi počítači, který bude realizovat vlastnosti spojové vrstvy a realizujte zabezpečení komunikace.

DOPORUČENÁ LITERATURA:

[1] Osterloh, H.: TCP/IP : kompletní průvodce : použitelný pro veškeré operační systémy, SoftPress, Praha 2003.

[2] Michael Welschenbach, David Kamer. Cryptography in C and C++. Apress, 2005. ISBN 1-59059-502-5.

Termín zadání: 9.2.2009 ***Termín odevzdání:*** 2.6.2008

Vedoucí práce: Ing. Petr Mlýnek

prof. Ing. Kamil Vrba, CSc.
předseda oborové rady

UPOZORNĚNÍ:

Anotace

Bakalářská práce pojednává o tématu Komunikační protokol pro messenger a jeho zabezpečení. V jejím úvodu jsou rozebrány základní poznatky z oblasti kryptografického zabezpečení dat.

Další dvě části se zaměřují na nejpoužívanější architektury v oblasti komunikačních sítí, které slouží k pochopení posíláni dat přes tyto sítě.

Předposlední část se zabývá vytvořením komunikačního kanálu mezi aplikacemi v informačních sítí.

Závěrečná část obsahuje popis tří jednoduchých aplikací, které demonstrují použití různých kryptografických ochran a různých protokolů pro posílání zpráv mezi uživateli.

Klíčové slova

Kryptografie, architektura TCP/IP, referenční mode ISO/OSI, sokety, programovací jazyk Java, UDP, TCP.

Abstract

Bachelor‘s discuss on theme Communications protocol for messenger and his security. The introduction analysis the basic points of cryptographic security of data.

Next two parts deals with the two most used architectures in the area of information network, which helps in understanding of sending data by this network.

Penult deals with the creation of the communication channel in between the applications in the information network..

The final part includes description of three uncomplicated applications. This demonstrates the use of various cryptographic securities and various protocols for the sending of information between the users.

Keywords

Cryptography, architecture TCP/IP, reference model ISO/OSI, sockety, Java programming language.

Prohlášení

Prohlašuji, že svůj semestrální projekt na téma Komunikační protokol pro messenger a jeho zabezpečení jsem vypracoval samostatně pod vedením vedoucího semestrálního projektu a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.
Jako autor uvedeného semestrálního projektu dále prohlašuji, že v souvislosti s vytvořením tohoto projektu jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ..... …….....
podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Petru Mlýnkovi, za velmi cenné rady při zpracování bakalářské práce.

Obsah

Seznam obrázků

1 Úvod

Bakalářská práce v úvodní části pojednává o různých způsobech zabezpečení informačních sítí. Zejména se zaměřuje na problematiku šifrování, které vychází z vědy zvané kryptografie. Šifrování dat je nejlepší způsob, jak zabezpečit přenášená data v informačních sítí. Největší výhodou tohoto způsobu zabezpečení je snadná implementace a nízká cena. Druhá a třetí část pojednává o dvou nejpoužívanějších architekturách informačních sítí. Ve třetí části se zabývám, jak prakticky vytvořit zabezpečený přenos textových zpráv mezi dvěma počítači.

Aplikací messenger lze komunikovat mezi uživateli po celém světě prostřednictvím informačních sítí, a to, v reálném čase. V počátcích vývoje informačních sítí komunikovalo prostřednictvím messengerů jen pár jedinců, v současnosti jsou už neodmyslitelnou součástí každého pracujícího jedince s počítačem a internetem. První messengery sloužily jen ke komunikaci v textové podobě. Ale dnešní messengery nabízejí mnohem více implementovaných funkcí, jako například audio a video komunikaci, posílání souborů, hraní her, kreslení a mnoho dalšího.

V současné době existuje mnoho různých messengerů, které pracují na rozdílných komunikačních protokolech. U některých protokolů můžeme najít zabezpečení přenášených dat, ale většina protokolů toto zabezpečení neobsahuje, což v některých případech může být značným rizikem.

Referenční model ISO/OSI a model TCP/IP vznikly paralelně. V bakalářské práci se zabývám jednotlivými výhodami a nevýhodami komunikace při použití těchto protokolů. Tato teorie mě posloužila k naprogramování komunikace mezi počítači a vytvoření zabezpečeného komunikačního kanálu.

2 Zabezpečení dat

2.1 Zabezpečení chyb při přenosu

Při přenosu dat se velmi často stávalo, že se k příjemci dostala jiná data než ta, které odesílatel poslal. Důsledkem toho byla úvaha zabezpečit posílaná data. Prvním pokusem v této oblasti bylo přidání paritního bitu, tj. přidáním jednoho bitu k původní zprávě. Tak může vzniknout sudá nebo lichá parita, to záleží na celkovém počtu jedniček v celé zprávě. Příjemce musí vědět, jestli má očekávat sudou, nebo lichou paritu. Jestliže příjemce čeká lichou paritu a dojde mu sudý počet jedniček, tak si odvodí, že ve zprávě došlo k chybě, avšak neví přesně v jakém bitu, obecně v lichém bitu. Má-li zpráva očekávanou paritu neznamená to, že byl přenos bezchybný. Pomocí jednoho paritního bitu nelze rozpoznat chybu v sudém bitu. Zabezpečení paritním bitem lze použít tam, kde pravděpodobnost výskytu chyby v jednom bitu je malá a chyba ve více bitech je téměř vyloučena.

Základní myšlenkou bezpečnostních kódů je transformace původního znaku na znak jiný podle předem přesně definovaných pravidel. Při použití bezpečnostních kódů narůstá počet bitů původního znaku, a to, takovou mírou, že čím lepší zabezpečení je použito, tím je také potřeba přidat více bitů přidat. Základní zabezpečení lichou nebo sudou paritou dokáže detekovat chybu v jednom bitu. Pro opravu tohoto jednoho bitu je potřeba použít v 8 bitovém slově navíc dalších 5 bitů.

Existují dva typy bezpečnostních kódů:

- Detekční kódy (error-detection codes), které umožňují pouze rozpoznat (detekovat) chybně přijatý znak.
- Samoopravné kódy (self-correcting codes), které umožňují rozpoznat chybně přenesený znak a dokáží ho i opravit.

V praxi se používá zabezpečení celých zpráv, nikoliv jen jednoho znaku. Při detekci chyby ve zprávě nelze přesně lokalizovat špatně přenesený znak, ale celá tato zpráva se označí jako chybná a je přenesena znovu[5].

2.2 Zabezpečení dat proti útokům

Máme tři základní způsoby jak zabezpečit data:

a) minimalizace nositelů hrozeb
b) blokování mechanismů hrozeb
c) minimalizace dopadů hrozeb

Příklad jak minimalizovat počet nositelů hrozeb může být omezení vstupu do prostorů, kde se tato data nacházejí. Mezi blokování mechanismů hrozeb můžeme například zařadit rušičky radiových vln, nebo také firewally. Mezi mechanismy jak zajistit minimalizaci dopadů hrozeb patří zálohování dat na více serverů[2].

2.2.1 Zabezpečení dat pomocí kryptografie

Nejdůležitější a nejpoužívanější vědou k zajištění bezpečnosti komunikačních systémů je kryptografie. Tato technika se snadno implementuje a je velice odolná vůči útokům. Pomocí kryptografie převedeme zprávu na jinou zprávu, která je řešením nějakého matematického problému. Matematické problémy tohoto typu jsou vytvářeny tak, aby útočník nemohl tyto zprávy řešit v reálném čase. Uživatel, ke kterému jsou zprávy posílány, zná nějakou tajnou informaci, podle které může zprávu řešit v reálném čase. Problematikou vytváření kryptografických ochran se zabývá věda kryptografie a problémem řešení kryptografických ochran se zabývá věda kryptoanalýza .

Zprávy jsou přenášeny přes otevřený kanál, ke kterému má přístup i útočník. Proto je nutné zprávy upravit do takové podoby, aby je útočník nemohl přečíst. Tato úprava zprávy se nazývá šifrování a provádí ji odesilatel. Na straně příjemce dochází k dešifrování, po dešifrování se zpráva vrátí do původní podoby. K šifrování je na straně odesilatele potřeba šifrovací klíč K_E (key encryption) a na straně příjemce dešifrovací klíč K_D (key decryption). Tyto klíče mohou, ale i nemusí být shodné. Útočník může zachytit přenášená data, ale nemůže je dešifrovat, protože nemá klíč. (Obrázek 1).

Obrázek 1. Kryptografický systém

V současné době existuje mnoho různých kryptosystémů, které jsou rozděleny do čtyř úrovní bezpečnosti, od Level 1 (nejnižší) až po Level 4 (nejvyšší). Každý uživatel si může vybrat, který bude vyhovovat jeho potřebám a výpočetní síle, kterou používá [2].

2.2.2 Kryptografické systémy

Máme dva základní druhy kryptografických systému symetrický a asymetrický. Symetrický systém má tajný klíč, kdežto asymetrický má veřejný klíč.

V symetrickém systému musí obě komunikující strany držet klíč v tajnosti. A to, z důvodu toho, že dešifrovací klíč je buď stejný nebo velice podobný jako šifrovací klíč. Tyto systémy jsou velice rychlé a zaručují důvěrnost i autentičnost přenášených zpráv. Ale nastává zde problém s distribucí klíče k příjemci. Symetrické systémy dále dělíme na blokové a proudové. Proudové šifry jsou rychlejší, avšak z hlediska zabezpečení jsou výkonnějším nástrojem šifry blokové [2].

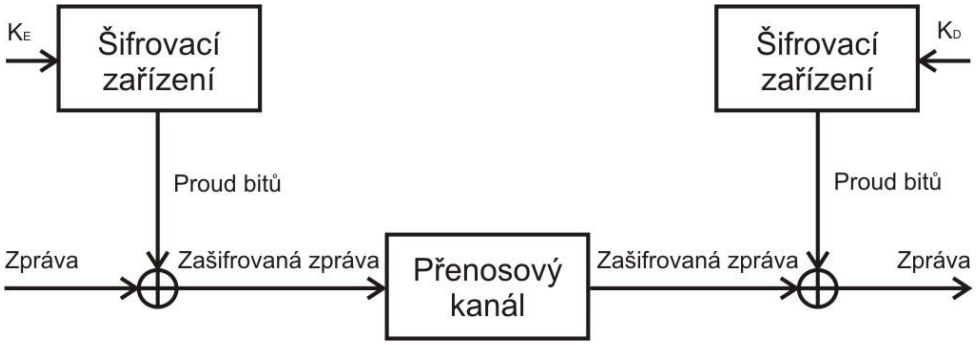
2.2.2.1 Symetrické kryptografické systémy

Proudové šifry:

U proudových šifer závisí hodnota zašifrovaného bitu, pouze na daném bitu a šifrovacím klíči. Tento šifrovací způsob je postaven na jednoduchém principu. Šifrovací zařízení produkuje synchronně proud bitů podle šifrovacího klíče K_E. Tento proud je

zaveden na sčítačku, kde se sčítá v aritmetice modulo dva (exkluzivní součet XOR) postupně z proudem bitů posílané zprávy . Tak vznikne zašifrovaná zpráva.

Dešifrování se provádím stejným způsobem jako šifrování. Dešifrovací zařízení opět synchronně produkuje proud bitů podle dešifrovacího klíče K_D, který je totožný s šifrovacím klíčem K_E. Tím vznikne původní zpráva (Obrázek 2)[2].

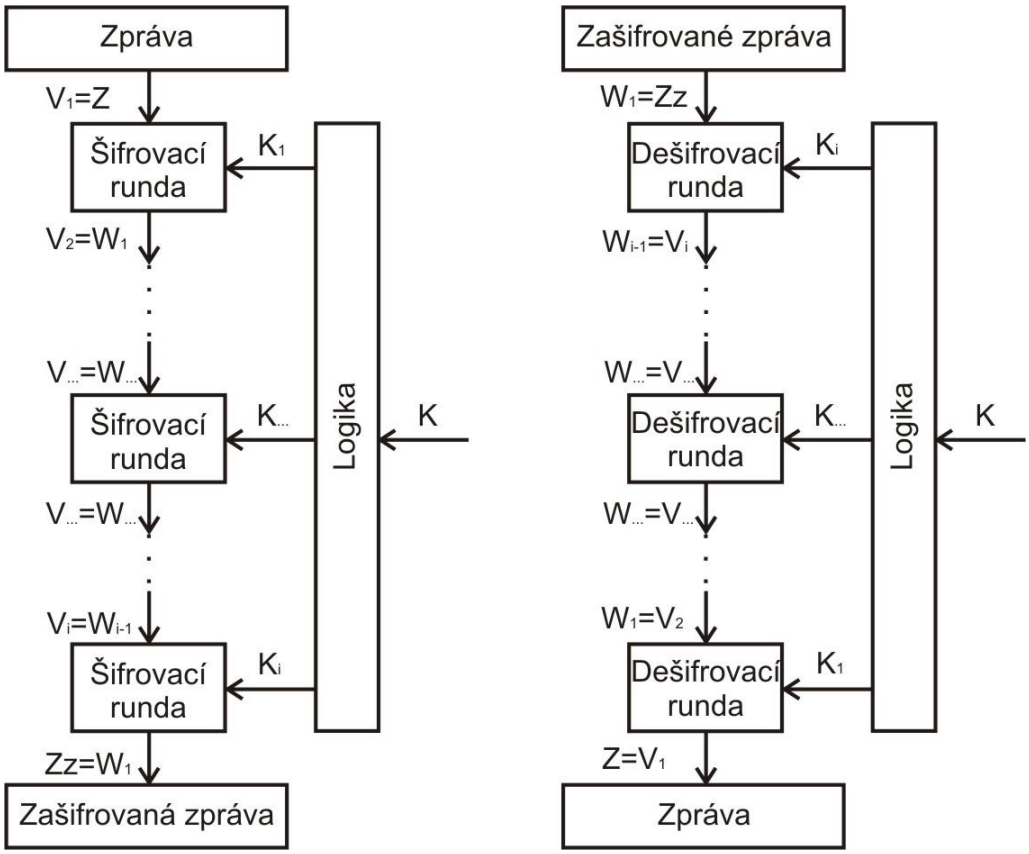


Obrázek 2. Princip proudové šifry

Blokové šifry:

Blokové šifry fungují tím způsobem, že se zpráva rozděluje na bloky o dané délce. Každý z těchto vzniklých bloků se zašifruje klíčem K_E. Spojením všech bloků vznikne zpráva, která je připravena k odeslaní. Její délka zpravidla bývá 64, 128 nebo 256 bitů.

Bloková šifra se skládá s *n* elementárních blokových šifrátorů, tzv. rund. Každá runda vstupního bloku V_i, přiřazuje na základě rundovního klíče K_i výstupní blok W_i. Výstupní blok W_i je vstupní blok následující rundy. Rundovní klíče K_i se vytváří na základě šifrovacího klíče K. Dešifrování probíhá podle stejného schématu, ale v opačném pořadí (Obrázek 3)[2].



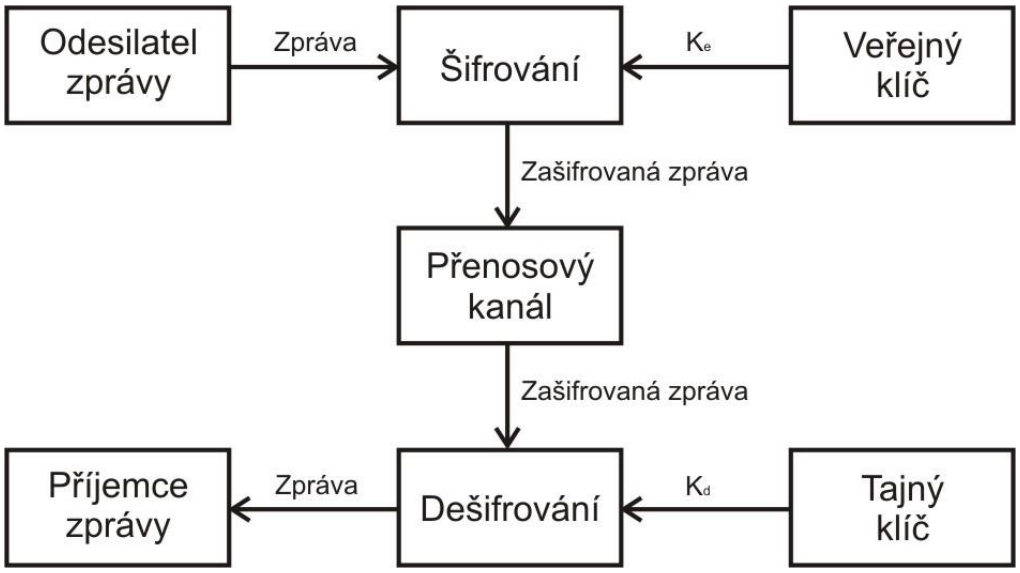
Obrázek 3. Blokový šifrátor a dešifrátor

2.2.2.2 Asymetrické kryptografické systémy

Asymetrické systémy mají k dispozici dva klíče, jeden veřejný a jeden tajný. Z veřejného klíče nelze odvodit klíč tajný. Výhodou tohoto systému je snazší distribuce klíčů, ale za cenu toho, že je tento systém pomalejší než systém symetrický. Asymetrický systém se v praxi používá k podepisování zpráv a k posílání symetrického klíče[2].

Způsob komunikace v asymetrickém kryptografickém systému

Každý z dvojce komunikujících systémů, má dva klíče, jeden veřejný *e* a jeden tajný *d*. Při posílání zprávy se použije k jejímu zašifrování veřejný klíč *e*, kdežto k dešifrování zprávy je zapotřebí použití tajného klíče *d* (Obrázek 4)[2].



Obrázek 4. Princip komunikace mezi dvěma uživateli v asymetrickém kryptografickém systému

2.2.2.3 Rozdělení kryptografickým systému podle utajení

Mezi nejdůležitější parametry kryptografických systémů patří stupeň utajení. Jsou používány tyto tři stupně utajení:

a) dokonalé utajení
b) ideální utajení
c) reálné utajení

V současné době existuje pouze jeden systém, který je schopen realizovat dokonalé utajení. Tento systém produkuje tzv. proudovou šifru s náhodným klíčem. Klíč musí být stejně dlouhý jako zabezpečovaná zpráva a musí být pro každou novou zprávu použit nový, zcela náhodný. Tento systém se nazývá dokonalá šifra. U tohoto systému není schopen kryptoanalytik rozluštit zašifrovanou zprávu.

Při použití systému s ideálním utajením nemůže kryptoanalytik danou zprávu jednoznačně rozluštit. Avšak oproti dokonalému utajení, může kryptoanalytik zjistit, kterým klíčem toto šifrování nevzniklo.

V běžné praxi se používají kryptografické systémy s reálným utajením, a proto je lze čistě teoreticky dešifrovat bez znalosti klíče. Aby to nebylo pro kryptoanalytika tak jednoduché, proto se používají kryptosystémy s dlouhým klíčem. Například pro délku klíče 78 bitů existuje $2^{78} \approx 3{,}0 \cdot 10^{23}$ klíčů. Vyluštění takového klíče je pro kryptoanalytika takřka nemožné [2].

2.2.3 Možnosti útoků na komunikační sítě

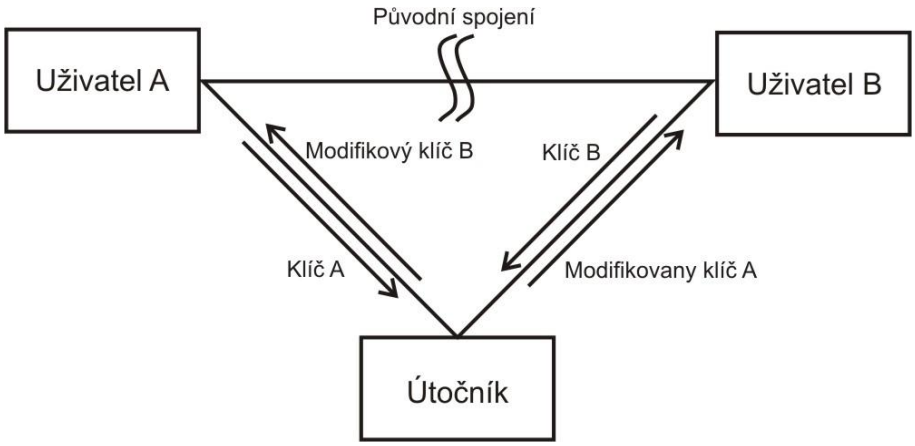
Jak jsme si řekli výše, jedním z nejdůležitějším parametrem v kryptografických systémech je stupeň utajení. Tyto stupně máme tři, a to, dokonalé utajení, ideální utajení a reálné utajení. V praxi se k zabezpečení komunikačních sítí nepoužívá jediného spolehlivé utajení, a to, ideálního utajení. Zejména pro velmi složitou realizaci. Proto se v praxi používají kryptografické systémy s horším stupně utajení, ale snadnější realizací. Největší nevýhoda použití kryptografických systému s horším utajením je to, že jsou teoreticky napadnutelné útočníkem.

Nejméně efektivní metoda jak napadnout komunikační síť, je za použití metody tzv. hrubé síly. Tato metoda spočívá v tom, že prohledává množinu všech možných řešení kryptografického problému. Typickým příkladem je zachycení šifrované zprávy v symetrickém kryptografickém systému, kde se útočník zprávu snaží rozšifrovat různými hodnotami klíčů, dokud se na dešifrovacím zařízení neobjeví smysluplný výsledek. Pak také nalezl klíč k řešení kryptografického problému. Kryptosystémy se tomuto útoku brání velkým počtem možných klíčů. Počet možností roste s délkou klíče a to hodnotou 2^k (kde *k*, je délka klíče). Tento útok lze aplikovat pouze pro klíče do délky 64 bitů a proto se pro symetrické kryptografické systémy doporučuje používat klíče o délce nejméně 80 bitů.

Další metodou útoku na komunikační síť jsou tzv. statické kryptoútoky. Jestliže útočník zjistí nějakou statickou závislost, tak ji může použít k redukci počtu možných řešení kryptografického problému. Jako příklad může posloužit tzv. substituční metoda. V této metodě se každé písmenko abecedy nahrazuje jiným písmenkem. Při dešifrování lze využít toho, že každé písmenko se vyskytuje ve větě různě často. Tato metoda výrazně rychleji konverguje než metoda hrubé síly, lze ji použít i na moderní šifry. Prevence proti této metodě je použití kvalitních kryptografických ochran.

Jeden z nejnebezpečnějších útoku na komunikační sít se nazývá metoda „mužem uprostřed“. Příklad je takový, že mezi dvě komunikující strany se dostane útočník. Komunikace pak probíhá podle tohoto schématu: Uživatel A zašle uživateli B svůj veřejný klíč. Útočník tuto komunikaci zachytí a místo klíče uživatele A pošle uživateli B svůj upravený klíč. Uživatel B se domnívá, že tento klíč, který dostal patří uživateli A. Tak uživatel B zase na oplátku pošle uživateli A svůj veřejný klíč. Útočník opět tuto

komunikaci zachytí a pošle uživateli A svůj modifikovaný klíč. Tím pádem útočník pak může číst celou komunikaci svými tajnými klíči (Obrázek 5). Tomuto útoku lze zabránit použitím spolehlivé autentizace uživatelů.



Obrázek 5. Ukázka útoku „mužem uprostřed“

Jeden z často používaných útoků, je útok opakováním. Jeho princip spočívá v tom, že útočník nějakou určitou zabezpečenou komunikaci uloží na záznamové médium, a tuto komunikaci přehraje příjemci znova. Může dojít například k zopakování záznamu z kamery nebo bankovního převodu. Prevence před tímto druhem útoku je například opatřením záznamu dobou provedení.

Další významná metoda útoku na informační systémy jsou tzv. implementační kryptoútoky, které fungují na využití vlastností technického řešení kryptosystému. Z chování kryptosystému lze získat mnoho cenných informací pro řešení kryptografického problému. Známým typem takového útoku je časová analýza čipové karty při šifrování. Útočník může odhalit z velikosti proudového odběru s jakými bitovými hodnotami čip v jednotlivých fází pracuje. To mu umožní opět redukovat čas k vyřešení kryptografického problému.

Existuje více obecných či úplně specifických útoků na informační systémy, ale je třeba mít na paměti, že při kvalitním použití kryptografických ochran má útočník malou šanci ho napadnout[2].

2.2.4 Bezpečnostní komunikační protokoly

Bezpečné komunikační protokoly jsou protokoly, které se používají k bezpečnému přenášení dat v informačních systémech, ke kterým by mohl mít přístup útočník. Každý z těchto protokolů je kryptografickou aplikací.

Protokol IPSec

Bezpečnostní protokol IPSec (Internet Protocol Security) slouží k bezpečnému přenášení informací v sítí, na síťové vrstvě (IP vrstva). Zajišťuje autentičnost přenášených informací a jejich důvěrnost. Je sestaven z dvou dílčích protokolů AH (Authentication Header), který zajišťuje autentičnost celého paketu a ESP (Encapsulating Security Payload), který zajišťuje důvěrnost a autentičnost pouze přenášených dat.

Tento protokol pracuje ve dvou fázích, nejdříve se po navázání spojení použije tzv. Main mód, kde se IKE (internet key exchanger) stará především o autentizaci obou komunikujících stran. Aby se spojení mohlo uskutečnit je nejdříve potřeba si domluvit, jaké kryptografické údaje se budou používat. Pak teprve může nastat tzv. Quick mód, který slouží k samotnému přenášení dat.

K vlastnímu šifrování posílaných zpráv se používají různé druhy blokových šifer, jako například DES, tripleDES, AES nebo Blowfish[2].

Protokol SSL

Bezpečnostní protokol SSL (Secure Societ Layer) slouží k bezpečnému přenášení informací v sítí, kde je použito protokolu TCP. Zajišťuje autentičnost přenášených informací a jejich důvěrnost. Těchto vlastností je používáno nejvíce v síti Internet, například pro internetové převody peněz, nákupy přes internet, nebo ověřování identity.

SSL vrstva je podvrstvou, která se nachází mezi aplikační a transportní vrstvou. K zabezpečení přenášených zpráv používá různé symetrické šifry jako například DES, tripleDES, AES, RC2. K autentizaci, nebo k přenosu symetrického klíče využívá asymetrických šifer, jako například RSA, DSS nebo KAE.

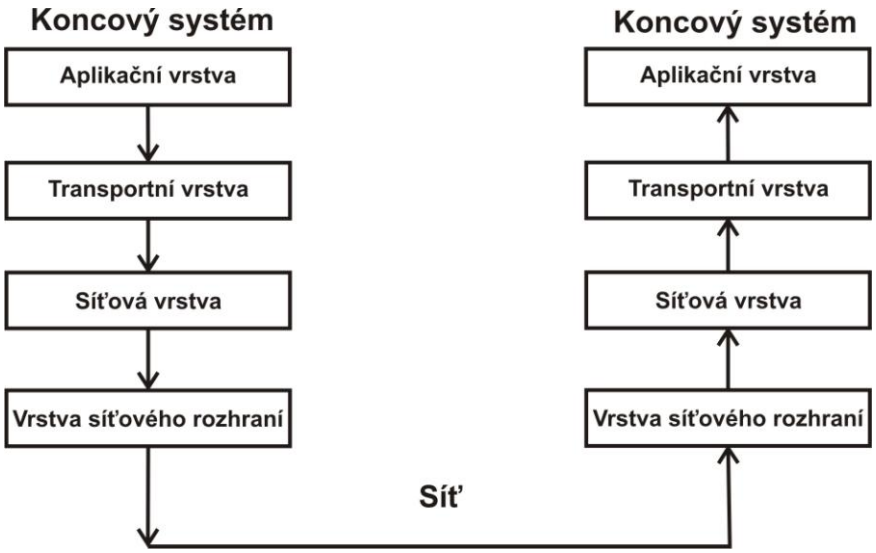
SSL vrstva se dělí ještě na další dvě podvrstvy:

- SSL Handshake Protocol
- SSL Rekord Protocol

SSL Handshake Protocol tento protokol slouží k výměně klíčů mezi účastníky. SSL Rekord Protocol zabaluje data z vyšší vrstvy do objektu record. Tento objekt se pak přenáší po síti[2].

3 Architektura TCP/IP

TCP/IP je sada komunikačních protokolů, která obsahuje, nejen jak je v názvu této architektury, protokoly TCP (Transmission Control Protocol) a IP (Internet Protocol), ale i řadu dalších. Architektura TCP/IP má čtyři vrstvy, aplikační, transportní, síťovou a vrstvu síťového rozhraní. Každá z těchto vrstev má svoji specifickou funkci. Při odeslání dat probíhají data od vrstvy nejvyšší k nejnižší, která je pošle k příjemci. U příjemce probíhají data od vrstvy nejnižší k vrstvě nejvyšší. Základní filozofie vrstvového modelu je taková, že každá vrstva využívá služeb vrstvy nižší a poskytuje své služby vrstvě vyšší (Obrázek 6).



Obrázek 6. Architektura TCP/IP a způsob komunikace

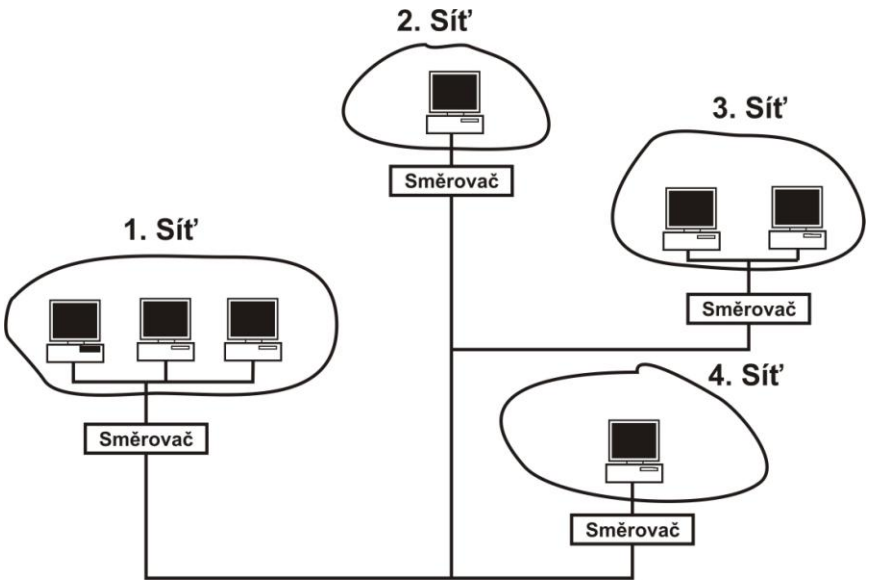
3.1 Vrstva síťového rozhraní

Vrstva síťového rozhraní má na starost vše, ohledně ovládaní přenosové linky s přímím odesíláním a přijímáním datových paketů. V architektuře TCP/IP není tato vrstva blíže specifikována, protože je závislá na použité přenosové technologii. Tuto vrstvu může tvořit jak jednoduchý síťoví ovladač, když je daný uzel připojen k lokální síti, nebo tvoří dvojbodové spojení. Ale může také obsahovat velmi složitý systém s vlastním linkovým protokolem. Tato vrstva bývá také často označováno jako Ethernet vrstva, protože je často zapojována právě do těchto sítí[5].

3.2 Vzájemné propojování sítí v architektuře TCP/IP, síťová vrstva

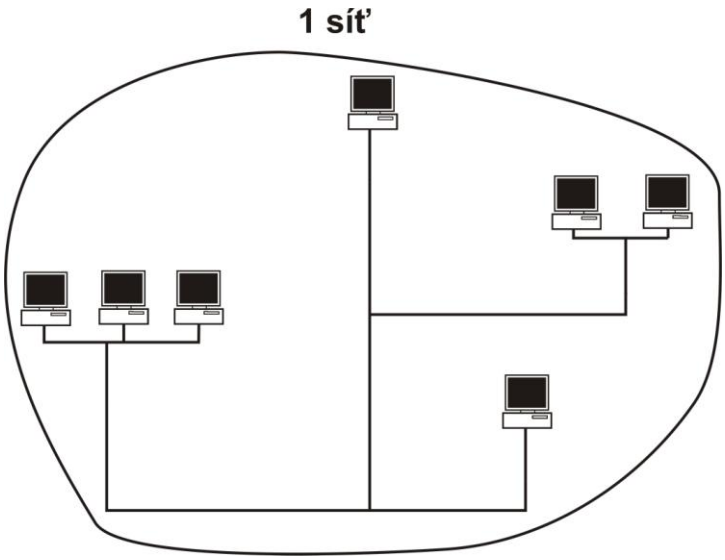
Vzájemné propojení sítí v architektuře TCP/IP je jedna z nejdůležitějších myšlenek této architektury. Představa je taková, že umožní propojení a komunikaci jakéhokoliv počítače s jiným počítačem, a to, i když neexistuje přímé spojení. K zajištění této komunikace jsou použity mezilehlé uzly. Propojení dvou, či více sítí se nazývá internet. Z hlediska uživatele je tato vnitřní struktura propojených sítí neviditelná a neměla by nějak ovlivňovat vlastní komunikaci.

O vzájemné propojení sítí v architektuře TCP/IP se stará, stejně jako u referenčního modelu ISO/OSI, vrstva síťová (Obrázek 7), která bývá často označováno jako IP vrstva, podle nejvýznamnějšího protokolu, který se na této vrstvě používá. Zařízení, která na této vrstvě pracují, se nazývají IP směrovače[5].



Obrázek 7. Pohled na síť z hlediska síťové vrstvy

Vrstva transportní, která je bezprostředně vyšší než vrstva síťová, vidí celou síť jako jednu homogenní síť (Obrázek 8).



Obrázek 8. Pohled na síť z hlediska transportní vrstvy

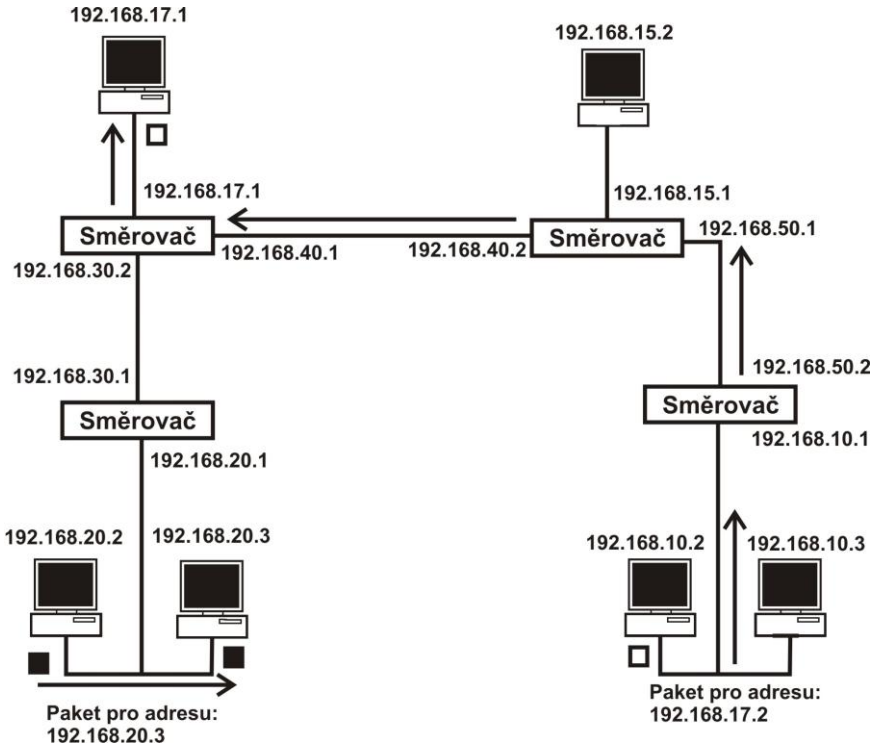
Proto se o problematiku směrovaní v propojených sítí stará vrstva síťová, která má mnoho jiných úkolů než jenom směrování, vyrovnávání odlišnosti různých propojených sítí (maximální velikostí rámců, odlišnosti v komunikaci, o to jestli se má jednat o spojoví, nebo nespojový charakter, odlišnosti adresace apod.). Proto musí směrovač pro každou síť, na kterou je připojen, mít samostatný ovladač na úrovni vrstvy síťového rozhraní. Tento ovladač pak dokáže oprostit síťovou vrstvu od způsobu ovládání příslušné sítě, ale směrování stále řeší vrstva síťová. Na úrovni síťové vrstvy se vytvoří jednotná abstrakce všech dílčích sítí, která umožní pracovat s jednou virtuální sítí se stejnými vlastnostmi. Tyto vlastnosti jsou jednotné adresování, formátování datových paketů (IP datagramy), nebo způsob poskytování přenosových služeb tj. datagramového spojení (nespolehlivý a nespojový přenos) .

K adresaci v síťové vrstvě se používají IP adresy. V dnešní době se používají IP adresy verze 4 které jsou 32 bitové. Adresa se skládá se z čísla pro podsíť a z čísla pro jednotlivé počítače v dílčí síti. IP adresa je pouze abstraktní adresa, která musí být převedena na fyzickou adresu, která má 48 bitů a se kterou dále pracuje vrstva síťového rozhraní[5].

3.3 Adresování

Z praktického hlediska nelze v internetu provést komunikaci koncových zařízení, a to, z důvodů fyzických tak i logických. Hlavní důvod je adresování, které se zakládá na adresování podsítí, nikoliv adresací přímo koncového zařízení z důvodu minimalizace velikosti směrovacích tabulek.

Koncové zařízení pozná IP adresy ostatních koncových zařízení v dílčí síti se kterými může přímo komunikovat. Jestliže, IP adresa neodpovídá žádnému koncovému zařízení v této dílčí síti, jsou tyto data odeslána ke směrovači, který dále rozhodne kam budou tato data odeslána. Směrovač se podívá jen na tu část adresy, kde je adresa sítě a data pošle k dalšímu směrovači. S adresou koncového zařízení se zabývá až poslední směrovač, který je s tímto koncovým zařízením spojen (Obrázek 9). Proto je nutné 32 bitovou IP adresu co nejlépe rozdělit na dvě části. A to tak, aby se vyhovělo požadavkům dané sítě. Pro tyto požadavky vzniklo základní rozdělení adresného prostoru na tři skupiny IP adres: pro velké počty koncových zařízení (třída A), pro střední počet koncových zařízeni (třída B) a pro malý počet koncových zařízení (třída C)[5].

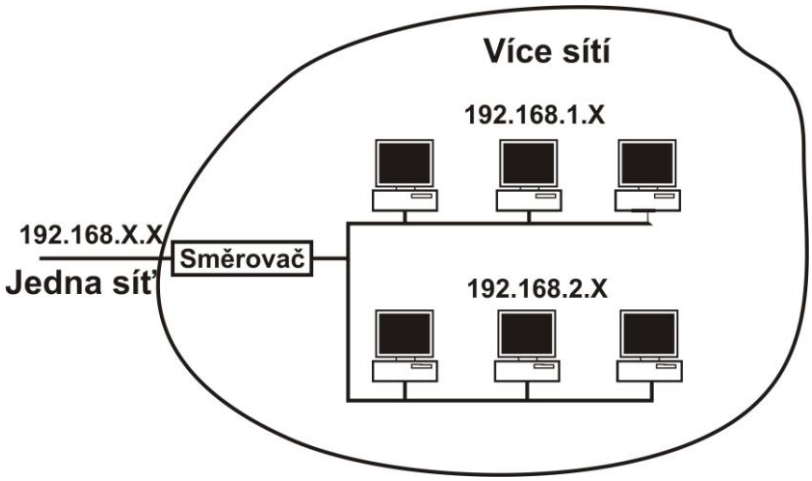


Obrázek 9. Ukázka směrování v síti TCP/IP

IP adresy jsou zapisovány ve dvojkové soustavě, ale pro jejich srozumitelnost jsou převáděna na tzv. tečkovanou desítkovou notaci. To znamená že 32 bitovou IP adresu rozdělíme na 4 části po 8 bitech (oktety). Například číslo 11000000 10101000 00000001 00000101 je číslo 192.168.1.5. V Internetové síti může existovat pouze jedna jediná IP adresa sítě, tuto adresu přiděluje DDN (Defense Data Network) Network Information Center, aby nedocházelo ke konfliktům adresovaní. Po udělení jedinečné IP adresy sítě může uživatel přiřazovat svým dílčím sítím svoje vlastní IP adresy[5].

3.4 Subnetting

Směrování v architektuře TCP/IP vychází pouze z části IP adresy, která patří adrese dílčí sítě. Tato skutečnost zmenšuje směrovací tabulky, oproti tomu kdyby byly použity IP adresy koncových zařízení. Ale ani tato skutečnost nezmenšuje směrovací tabulky natolik, aby jejich velikost byla přijatelná pro celý Internet, proto existuje další způsob jak zefektivnit směrování. Pro představu uvedeme příklad. Existuje firma, která má mnoho dílčích sítí (adresy třídy C). Každá z těchto dílčích sítí je viditelná i vně sítě, tzn. každá tato sít zabírá ve směrovacích tabulkách samostatnou hodnotu. Ale kdyby se tato firma starala sama o směřování mezi svými dílčími sítěmi, tak by tato firma mohla vystupovat navenek pouze s jednou IP adresou (adresa třídy B). Což by značně zmenšilo velikost směrovacích tabulek. Tohoto je docíleno maskami podsítě. Maska podsítě nám rozděluje sítě na podsítě a koncová zařízení. Určuje, která část IP adresy patří pro síť a která část je určena pro koncové zařízení (Obrázek 10). Maska podsítě se zapisuje stejně jako IP adresy až na to, že platné hodnoty jsou ty, které mají z levé strany jedničku a z pravé strany nulu tzn. pokud se při výčtu z levé strany objeví nula pak musí následovat jen nuly. Jedničky v masce určují síťovou část a nuly část pro koncová zařízení. Například maska sítě 11111111 11111111 11111111 00000000 nebo li 255.255.255.0 nám určuje, že prvních 24 bitů je určeno pro síťovou část a 8 bitů je určeno pro koncová zařízení[4][5].



Obrázek 10. Ukázka efektivnosti subnettů

3.4.1 Transportní vrstva a její protokoly

V architektuře TCP/IP existuje mnoho různých protokolů pro přenos dat. Základní rozdělení je na spolehlivé protokoly, které mají větší požadavky na přenos v síti, kde jejich realizace probíhá na vrstvě transportní nebo aplikační a na nespolehlivé protokoly, kde jejich realizace probíhá na vrstvě síťové.

Teď vystává otázka jestli je lepší provozovat spolehlivou službu na vrstvě transportní, nebo na vrstvě aplikační. Výhoda provozu spolehlivých služeb na transportní vrstvě tkví v tom, že prostředky transportní vrstvě realizované jednou entitou, můžou používat všechny entity aplikační vrstvy. Ale při provozu spolehlivých služeb na aplikační vrstvě, si každá aplikační entita zajišťuje sama spolehlivý přenos. Ve skutečnosti se používají obě tyto varianty, protože některé aplikace nepotřebují spolehlivý přenos, nebo nepovažují prostředky v transportní vrstvě za tolik spolehlivé.

V transportní vrstvě se nejvíce používají protokoly TCP (Transmission Control Protocol) a UDP (User Datagram Protocol)[5].

3.5 Aplikační vrstva

Aplikační vrstva v architektuře TCP/IP vychází z toho, že si každá samostatná aplikace zajistí služby, které sama potřebuje a služby, které nižší vrstva nenabízí. V poslední době se toto pravidlo porušuje a některé jednoduché mechanismy se implementují přímo do vrstvy aplikační.

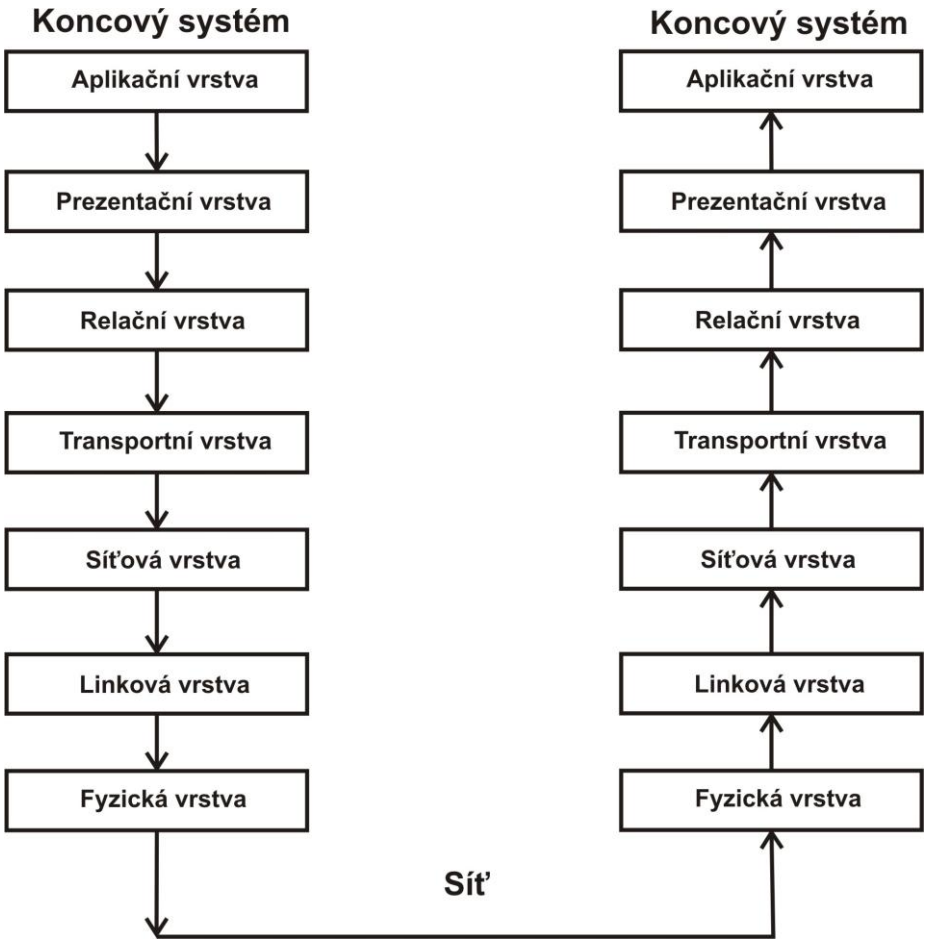
Na samostatných aplikacích je zajištění kódování jednotlivých znaků, vyrovnání se s rozdílnostmi operačních systémů, rozdílnosti hardwarového vybavení apod.

Většina aplikací v této vrstvě pracuje na principu klient - server. Klient si žádá o jednotlivé služby a je iniciátorem komunikace, kdežto server jen poskytuje služby na žádost klienta.

Nejpoužívanější protokoly na této vrstvě jsou Telnet, FTP, HTTP, DHCP, DNS. Pro rozlišení různých aplikačních protokolů se využívá značení portu a protokolu, který je použit na transportní vrstvě. Například protokol FTP používá port TCP/21 (pro řízení komunikace) nebo TCP/20 (pro vlastní přenos dat)[5].

4 Referenční model ISO/OSI

Jedná se o sedmivrstvový model, kde komunikace stejně jak u architektury TCP/IP probíhá od nejvyšší vrstvy k nejnižší vrstvě (Obrázek 11). U referenčního modelu ISO/OSI je velice kritizováno rozložení funkcí, protože spodní tři vrstvy mají přesně vymezenou funkci a mají „hodně“ práce, tak naopak vrchní tři vrstvy nemají přesně vymezené funkce a nemají tolik práce. To řeší model TCP/IP, kde místo tří vrchních vrstev je pouze jedna vrstva (aplikační vrstva). Tvůrci referenčního modelu předpokládali, že každá aplikace bude potřebovat mít všechny služby zaimplementované v některé z vrstev, zatímco tvůrci TCP/IP předpokládali, že tyto služby poskytnou sami aplikace. Proto má model ISO/OSI o tři vrstvy více. Tvůrci navrhli tento model jako spolehlivý a spojově orientovaný (tzn. před samotnou komunikací se musí navázat spojení mezi uživateli a po skončení komunikace se toto spojení zruší). Počítalo se s tím, že se každá vrstva bude starat o spolehlivý přenos, ale to nebylo moc šťastné řešení, protože na sítí vzniká velký provoz jenom proto, že se každá vrstva snaží zajistit sama spolehlivý přenos, což způsobuje zpomalení toku dat v celé síti. Možnost nespolehlivých služeb byla přidána až později pod tlakem modelu TCP/IP, který tyto služby nabízí. A později se také opustila filozofie spojově orientovaného přenosu, který je lepší pro přenášení hlasu, ale pro přenášení dat není vhodný. Nespojový přenos je takový přenos, kde jsou jednotlivé zprávy označeny adresou příjemce a jednotlivá zprávy prochází sítí nezávisle na sobě. Tudíž každá zpráva může k příjemci dojít jinou cestou[5].



Obrázek 11. Referenční model ISO/OSI a způsob komunikace

4.1 Vzájemné propojování sítí v referenčním modelu ISO/OSI

Je samozřejmé, že můžeme v sítích založených na referenčním modelu ISO/OSI propojovat více sítí dohromady. Z toho vyplývá, že mezi spojením koncových účastníků může existovat více mezilehlých sítí . Pro transportní vrstvu, která zajišťuje komunikaci mezi koncovými uživateli, není tento fakt nějak viditelný. Proto se o vzájemné propojení sítí stará vrstva síťová, která je přímo pod vrstvou transportní .

Samotná síťová vrstva se ještě dělí na tři další podvrstvi:

- podvrstva přístupu k podsíti (subnet access sublayer), používající protokol SNDAP (SubNetwork Dependent Access Protocol)

- podvrstva přizpůsobení podsítě (subnet enhancement sublayer), s protokolem SNDCP (SubNetwork Dependent Convergence Protocol)
- podvrstva řízení vzájemně propojených podsítí (internet sublayer), s protokolem SNICP (SubNetwork Independent Convergence Protocol)

Nejnižší z těchto podvrstev, přístup k podsíti se stará o směrování jednotlivých datových buněk, ale jenom v rámci jedné sítě, která má stejná pravidla pro přenos (pravidla pro přenos, směrování, adresování apod.).

Prostření podvrstva přizpůsobení podsítě slouží k přizpůsobení sítí pracujících na různých mechanismech, aby šli tyto sítě pak vzájemně propojit.

Nejvyšší podvrstva řízení vzájemně propojených sítí zajišťuje už přímé doručování jednotlivých datových buněk od odesílatele k příjemci[5].

4.2 Popis a princip funkcí jednotlivých vrstev

4.2.1 Fyzická vrstva

Fyzická vrstva slouží k přenosu jednotlivých bitů k vedlejšímu uzlu. Její základní funkce jsou: kódování, časování, zajištění velikosti pracovní napěťové úrovně, zapojení konektorů, modulace a aktivování a udržovaní fyzického spojení. Fyzická vrstva poskytuje elektrické a mechanické vlastnosti přenosovému mediu. Na této vrstvě pracují fyzické zařízení jako například huby, opakovače a síťové adaptéry[5].

4.2.2 Linková vrstva

Linková vrstva realizuje přenos celých rámců. Má také za úkol starat se o nastavení parametrů přenosu linky, kontrolu správně přenesených rámců, to oznamuje odesilateli, pokud rámec byl přenesen špatně, tak si vyžádá opětovné přenesení celého rámce. Musí také poznat, kde rámec začíná a kde končí, ale i jednotlivé části včetně hlavičky a adres, které rámec obsahuje. I tato vrstva stejně jako fyzická dokáže svoje data (rámce) přenášet pouze k nejbližšímu uzlu. Na této vrstvě pracují fyzické zařízení jako například přepínače a mosty[5].

4.2.3 Síťová vrstva

Hlavním úkolem síťové vrstvy je postarat se o směrovaní v síti tzn. nalezení co nejlepší cesty k cíly přes mezilehlé uzly, tzn. že tato vrstva si už uvědomuje topologii sítě. Datové části, které tato vrstva posílá se nazývají pakety. Síťová vrstva v každém uzli rozhodne, ke kterému dalšímu uzlu by měl být paket poslán, poté ho předá linkové vrstvě a ta za pomoci fyzické vrstvy tento paket pošle dále. Na této vrstvě pracují jako fyzické zařízení směrovače[5].

4.2.4 Transportní vrstva

Transportní vrstva zajišťuje přenos dat mezi koncovými uzly. Jejím hlavním úkolem je poskytnou vyšším vrstvám kvalitu služeb, kterou potřebují. Transportní vrstva stojí mezi poskytovateli přenosových služeb a jejich uživateli. Je to také poslední vrstva, která může měnit nespojový charakter na spojový a naopak.

Pro transportní vrstvu je velice podstatné, znát kvalitu služeb, kterou ji může poskytnou síťová vrstva. Tyto kvality služeb se rozdělují na tři kategorie:

a) bez chyb při přenosu paketů a bez výpadků spojení
b) bez chyb při přenosu paketů, s výpadky spojení
c) s chybami při přenosu paketů, s výpadky spojení

U kategorie A nedochází k téměř žádným výpadkům sítě a jen malým ztrátám paketů. U těchto sítí má transportní vrstva nejmenší práci. Nejčastěji to bývají malé lokální sítě.

U kategorie B dochází k výpadkům spojení, ale ztráta paketů je na úrovni kategorie A. Transportní vrstva se s těmito výpadky dokáže vyrovnat a skrýt to před vyššími vrstvami. U těchto sítí má transportní vrstva více práce, než li tomu bylo u kategorie A. Do kategorie B spadají veřejné datové sítě fungující na doporučení X.25 .

Kategorii C označuje transportní vrstva za nespolehlivou a musí si spolehlivost zajistit sama. K zajištění těchto mechanismů má transportní vrstva nejvíce práce[5].

4.2.5 Relační vrstva

Relační vrstva je pátou vrstvou referenčního modelu ISO/OSI. Tato vrstva obstarává nejméně funkcí z celého referenčního modelu ISO/OSI. Umožňuje vytvoření a ukončení relačního spojení, oznamování výjimečných stavů[5].

4.2.6 Prezentační vrstva

Hlavním úkolem prezentační vrstvy je konverze přenášených dat, tzn. například překlad kódu EBCDIC (Extended Binary Coded Decimal Interchange Code), na kód ASCII (American Standard Code for Information Interchange), a opačně. Tato vrstva také může zajišťovat šifrování dat, které ale může být také realizováno na vrstvě transportní, či fyzické. Může zde být i zajištěna komprimace dat, pro minimalizaci přenášených dat[5].

4.2.7 Aplikační vrstva

Aplikační vrstva se v průběhu času vyvíjela. Nejdříve byla představa taková, že samotné uživatelské aplikace budou zasahovat do této vrstvy, tato část která zde měla zasahovat se nazývá aplikační entita. To znamenalo, že jednotlivé aplikace si musely sami zajišťovat všechny služby fungující na aplikační vrstvě. Referenční model ISO/OSI tyto služby přesně nespecifikoval, jen vymezil, že se má jednat o takové služby, které jsou potřeba pro vzájemnou komunikaci mezi otevřenými systémy a nejsou zajišťovány na nižších úrovních. Referenční model ISO/OSI nespecifikoval ani žádné protokoly. Jednotlivé protokoly vznikaly až postupem času, kdy se začalo pracovat na implementacích síťových aplikacích různého druhu. Tady se také ukázalo, že mnoho aplikací používá stejný mechanismus, tudíž není nutné pro každou aplikaci vytvářet nový. Zde se změnila představa o tom, že služby na aplikační vrstvě byly svěřeny aplikačním entitám, které nejsou součástí jednotlivých aplikací, ale jsou součástí síťového programového vybavení. Další změna této představy spočívala v tom, že se takto koncipované aplikační entity rozdělí na menší a každá z nich bude mít specifickou funkci[5].

5 Sokety

Sokety byly vytvořeny roku 1980 na Berkeleyské univerzitě pro použití v operačních systémech Unix. Sokety umožňují vytvářet aplikace, které slouží ke komunikaci v síti mezi jednotlivými zařízeními za pomoci různých síťových protokolů, například TCP/IP. V polovině devadesátých let vytvořila firma Microsoft svoji verzi soketů nazvanou Windows Soket.

Sokety jsou vlastně něco jako telefony, využívají vlastních numerických adres. Jakmile se oba sokety spojí může začít komunikace. Komunikace probíhá jako server – klient. Počítače, které jsou jako server ponechávají komunikační port otevřený a jsou připraveny pro neočekávané volání. Typický průběh komunikace je takový, že klient určí číslo portu požadovaného serveru tak, že toto číslo vyhledá v databázi DNS (Domain Naming System). Po úspěšném připojení na server, server komunikaci přesune na jiný port, aby přijímací port nechal volný pro další volání[4].

5.1 Módy soketů

Soket může pracovat ve dvou módech a to:

- v blokujícím módu
- v neblokujícím módu

Blokující mód

Tento mód pracuje v synchronním režimu, tj. vyčkává na příslušnou událost například přijetí dat, po přijetí dat může program pokračovat. Aby událost neblokovala celý program, tak se explicitně stanoví doba čekání, nebo se vytvoří nové vlákno, které pracuje paralelně a nezávisle[4].

Neblokující mód

Tento mód pracuje v asynchronním režimu, tj. soket může vykonávat více funkcí naráz, například přijímat i odesílat[4].

5.2 Protokoly a porty

Každá síťová služba pracuje se specifickým portem a transportním protokolem. Porty jsou zde proto, aby bylo možné rozeznat a připojit se k jednotlivým síťovím službám a provozovat je v jednom časovém okamžiku pro více připojení. Často používané služby mají port pevně stanovený, například služba HTTP používá port TCP/80, nebo služba FTP používá port TCP/20. Při vytváření nového soketu je nutné specifikovat příslušný protokol na transportní vrstvě a zvolit si volné číslo portu, které je v rozsahu 1024 až 49151.

5.3 Způsob komunikace klient - server

Způsob komunikace pomocí soketů probíhá způsobem klient – server, tj. klient posílá požadavky pro server a ten na ně odpovídá. Server sám neinicializuje žádné požadavky ke klientovi, jen čeká na jeho požadavky.

Na straně klienta je nutné nastavit IP adresu serveru a příslušný port, na který se chce klient připojit.

Na straně serveru je nutné nastavit port, na kterém bude naslouchat požadavkům od klienta. Po nalezení příslušného portu klientem se tento port uvolní a komunikace se přesune na jiný port, aby příslušný port pro danou službu byl stále k dispozici i ostatním klientům.

6 Vlastní messenger a jeho zabezpečení

K naprogramování vlastního messengera a jeho zabezpečení jsem si vybral programovací jazyk Java a prostředí NetBeans. Pro tuto volbu jsme se rozhodl zejména proto, že jazyk Java nabízí mnoho užitečných funkcí pro vytváření aplikací se soketama, může být spuštěn i na rozdílných operačních systémech. S vývojovým prostředím NetBeans, jsem se setkal v průběhu mého studia.

6.1 Popis jednotlivých aplikací

Vytvořil jsem dvě odlišné aplikace. Každá z nich pracuje s jiným zabezpečením i s jinými protokoly na transportní vrstvě, respektive program pracující s SSL (Secure Sockets Layer) pracuje na vrstvě mezi vrstvou aplikační a transportní.

6.1.1 Aplikace s UDP sokety s AES šifrováním

Aplikace v sobě skrývá funkce jak klienta, tak i serveru. Tento způsob provedení má své výhody, ale i svá úskalí. Největší výhoda tohoto provedení spočívá v tom, že není potřeba žádný centrální prvek, ke kterému se klienti připojují. Když by tento centrální prvek existoval a byl by nedostupný, tak by selhala veškerá komunikace mezi klienty. Naopak nevýhodou tohoto řešení je, že klient může v jeden okamžik komunikovat pouze s jedním dalším klientem. Jedná se o komunikaci peer-to-peer.

Aby aplikace mohla současně posílat a přijímat zprávy, je třeba vytvořit dvě vlákna. Jedno vlákno pro příjem druhé pro odesílání, tyto dva úkony pak na sobě pracují paralelně na sobě nezávisle.

Pro zabezpečení posílaných zpráv jsem zvolil symetrickou blokovou šifru, která zaobaluje zprávu do stejně velkých bloků, které se pak posílají.

Způsob zabezpečení komunikace

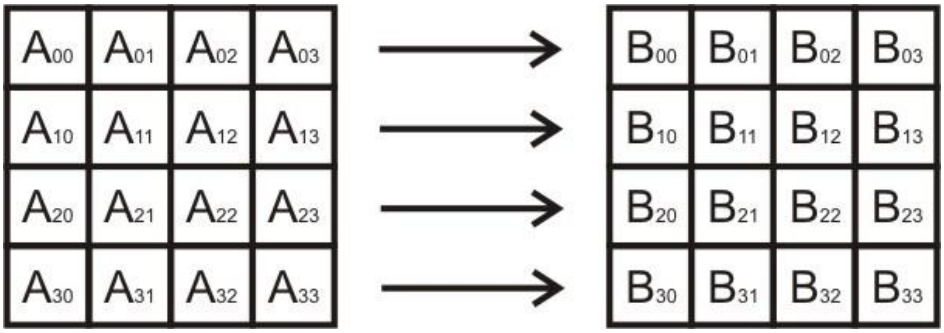
Jako zabezpečení jsem pro tuto aplikaci použil šifrovací algoritmus AES (Advance Encryption Standard), tento standard vychází ze staršího standardu DES (Data Encryption Standard). Tento algoritmus jsem si vybral zejména proto, že je při dostatečně délce klíče velice bezpečný, a také ho lze provozovat bezlicenčně. Tato šifra byla navrhnuta tak, aby ji nebylo možné prolomit za použití útoku hrubé síly a aby

vydržela nejméně 20 let (šifra AES byla schválena jako standart roku 2002). Délka klíče je 128, 196 nebo 256 bitů a zpráva se dělí do bloků o délce 128 bitů.

Princip šifrování pomocí AES

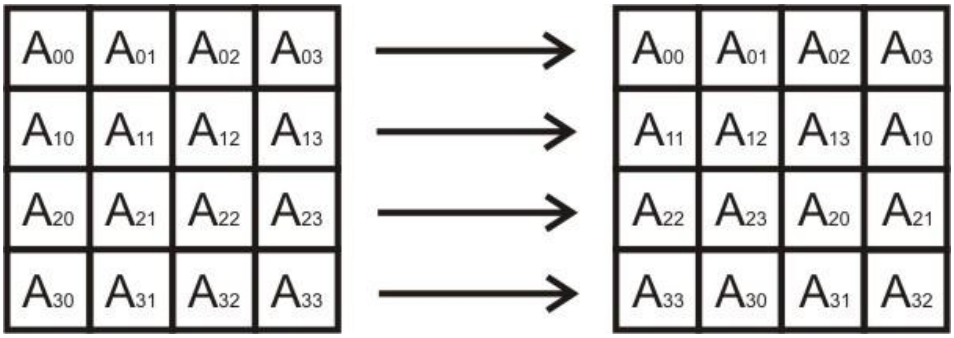
Šifrování pomocí AES probíhá ve čtyřech krocích.

1. Každý jednotlivý byte je nahrazen jiným bytem podle předem stanoveného klíče (Obrázek 12).



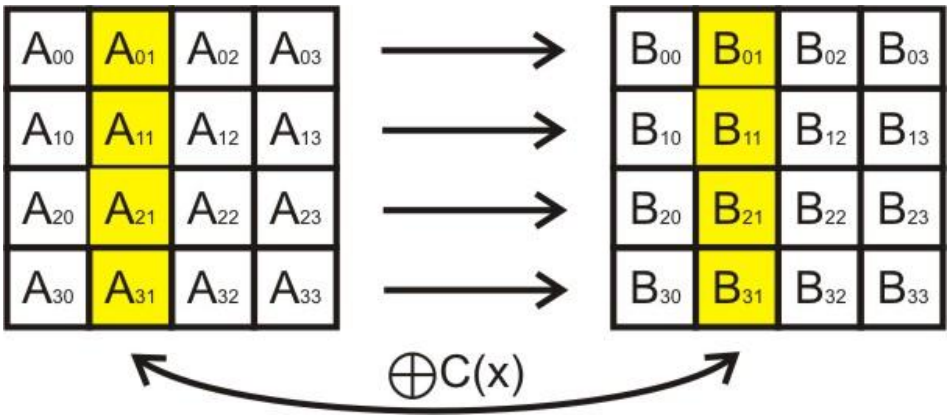
Obrázek 12. Ukázka transformace bytů u AES šifrování

2. Pak se v této matici jednotlivé prvky přehází podle schématu (Obrázek 13).



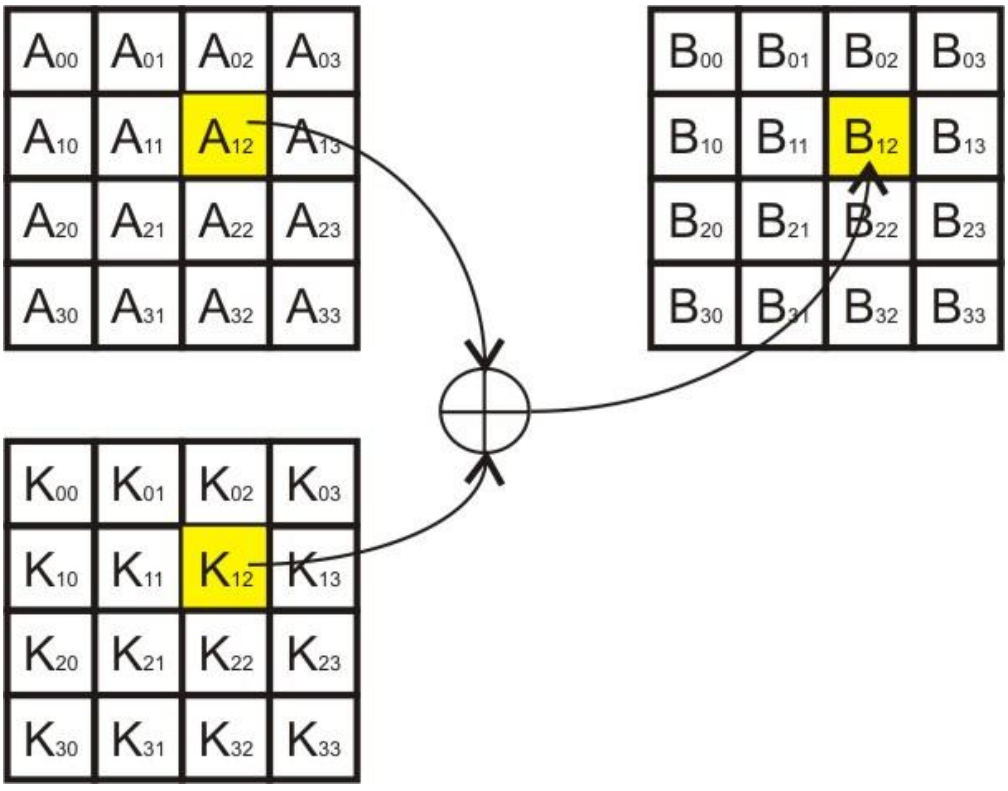
Obrázek 13. Ukázka přehození jednotlivých prvků u AES šifrování

3. Ve třetím kroku dochází k proházení pořadí sloupců a zároveň je každý sloupec vynásoben polynomem c(x) (Obrázek 14).



Obrázek 14. Ukázka přeházení a násobení sloupců u šifrování AES

4. V posledním kroku je každý byt zkombinován se subklíčem, který se získává z původního klíče převodem z Rijndaelovy tabulky. Každý byte subklíče je zkombinován s příslušným bytem zprávy a tím vznikne zašifrovaná zpráva (Obrázek 15)[4][6].



Obrázek 15. Ukázka zkombinování každého bytu se subklíčem u AES šifrování

Ukázka šifrování a dešifrování AES algoritmu

Šifrování:

```
SecretKeySpec key = new SecretKeySpec("1234567891234567".getBytes("UTF-8"), "AES");
String nameAlgorithm = "AES/ECB/PKCS7Padding";
Cipher cipher = Cipher.getInstance(nameAlgorithm);
cipher.init(Cipher.ENCRYPT_MODE, key);
byte cipheredText[] = cipher.doFinal(readText);
```

Nejdříve byl vytvořen objekt *SecretKeySpec,* kde se do konstruktoru předá hodnota klíče a typ šifrování. Já jsem použil 128 bitový klíč, který by měl být pro tento druh aplikace dostačující. Avšak klíč, který jsem si zvolil není příliš bezpečný, protože by měl být vygenerován náhodně. Pak je nutné objektu *Cipher,* který reprezentuje samostatný objekt pro šifrování dat říct, jaký šifrovací algoritmus, režim provozu blokových šifer a výplň zbytku 128 bitového bloku má použít. Nyní je potřeba říct objektu *Cipher*, jaké mechanismy (šifrovací mód nebo dešifrovací mód) a klíč má pro šifrování použít a to za pomoci funkce *init*. Nakonec se data zašifrují pomocí funkce, kterou obsahuje objekt *Cipher* (*doFinal()).*

Dešifrování se provádí analogicky, jen se místo šifrovacího módu, použije mód dešifrovací.

Dešifrování:

```
SecretKeySpec key = new SecretKeySpec("1234567891234567".getBytes("UTF-8"), "AES");
String nameAlgorithm = "AES/ECB/PKCS7Padding";
Cipher decryption = Cipher.getInstance(nameAlgorithm);
decryption.init(Cipher.DECRYPT_MODE, key);
byte decrypt[] = decryption.doFinal(som,0,recived.getLength());
```

Sokety UDP

Výhoda použití UDP soketů spočívá v rychlosti posílání dat a v jednoduchosti vytvoření těchto soketů. Problém UDP soketů je takový, že používá UDP protokol, který je nespolehlivý a nespojový. Odeslaná data vůbec nemusí dojít k příjemci, anebo můžou dojít poškozená. Použití těchto soketů lze s výhodnou snadné implementace používat v malých lokálních sítí, kde je malá pravděpodobnost ztráty dat.

Ukázka Java kódu, posílání a přijímání zpráv při použití UDP soketů

Odesílání zpráv:

```
DatagramSocket client = new DatagramSocket(serverAddress,port);

BufferedReader read = new BufferedReader(new InputStreamReader(System.in));

readLine = read.readLine();
readText = readLine.getBytes();

DatagramPacket sendData = new DatagramPacket(readText, readText.lenght, serverAddress,
port);
client.send(sendData);
```

Nejdříve, aby mohlo dojít ke komunikaci mezi klientem a serverem, je potřeba vytvořit objekt *DatagramSocket*, kde se do jeho konstruktoru předá hodnota IP adresy a portu příjemce. Další řádky reprezentují načítání dat z klávesnice a uložení do proměnné typu *byte*. Soket typu *DatagramSocket* odesílá data za pomocí objektu *DatagramPacket*, který očekává ve svém konstruktoru čtyři hodnoty: odesílaná data, která musí být typu *byte*, délku těchto dat, IP adresu příjemce a port příjemce. Nakonec, aby mohlo dojít k samotnému odeslání zprávy, se použije funkce *send*, kterou obsahuje objekt *DatagramSocket*.

Příjem zpráv:

```
DatagramSocket server = new DatagramSocket(port);

DatagramPacket recivePacket = new DatagramPAcket(recive, recive.lenght);
server.recive(recivePacket);
```

V prvním kroku se musí vytvořit objekt typu *DatagramSocket*, který naslouchá na definovaném portu. Pro přijmutí datagramu je potřeba vytvořit objekt *DatagramPacket*, kde se do konstruktoru předá proměnná, která musí být typu *byte*. Tento vytvořený objekt se použije na dalším řádku, kde se zavolá odkaz na *DatagramSocket*, který obsahuje funkci *recive*. Tato funkce slouží k přijímání dat, když jsou k dispozici.

6.1.2 Aplikace s SSL sokety

Tato aplikace je navrhnuta zcela jinak než předchozí . Obsahuje centrální prvek (server), na který se připojují jednotlivý klienti. To znamená, že veškerá komunikace probíhá

přes tento centrální prvek, toho lze výhodou využít k připojení více než dvou klientů, kteří spolu mohou navzájem komunikovat. Nevýhoda tohoto řešení je, že při výpadku tohoto centrální prvku, se přeruší veškerá komunikace mezi jednotlivými klienty.

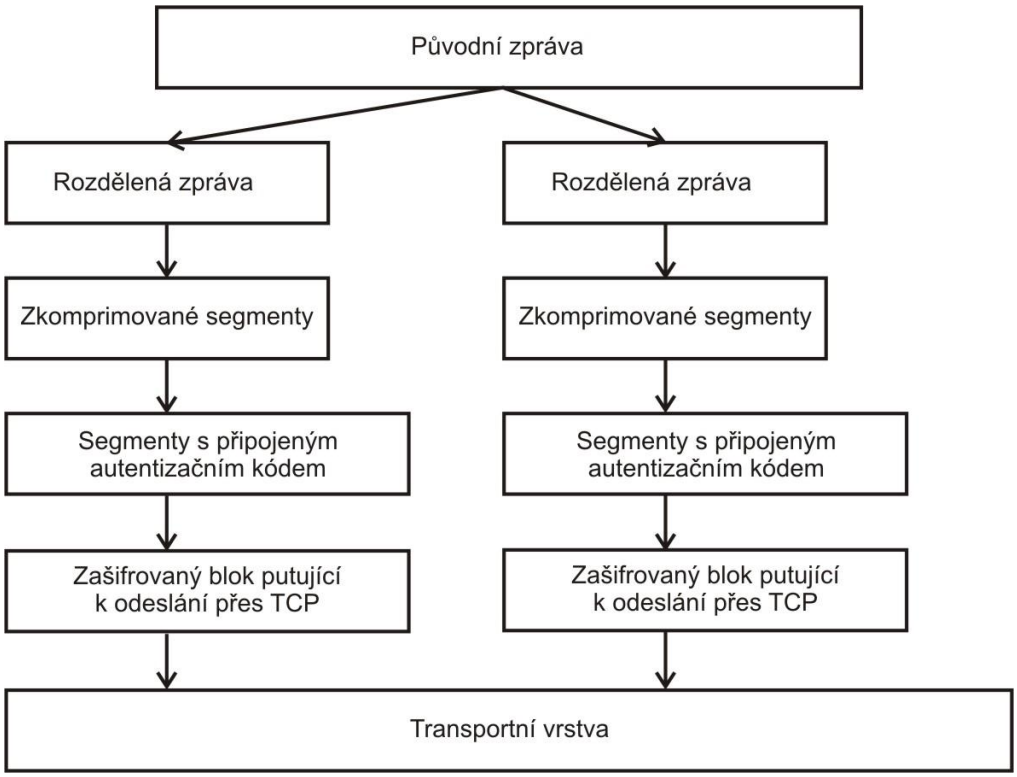
Server pro každého klienta vytvoří samostatné vlákno, což by mohlo být při velkém počtu uživatelů dosti náročné na hardwarové vybavení. Ale pro malý počet uživatelů je to snadnější a efektivnější řešení.

Klientská aplikace opět obsahuje dvě vlákna, jedno pro příjem a druhé pro odesílání zpráv, tato implementace zabraňuje blokování zpráv.

Způsob zabezpečení komunikace

Jako zabezpečení jsem pro tuto aplikaci použil SSL (Secure Socket Layer). SSL slouží k zajištění bezpečného přenášení zpráv v komunikačních sítí, které používají protokol TCP.

Jeho základní prvkem je SSL Record Protocol. Funguje tak, že nejdříve rozdělí původní zprávu na segmenty o maximální délce 2^{14} bytu. Tyto vytvořené segmenty jsou bezztrátově zkomprimovány a následně se k nim připojí autentizační kód. K výpočtu autentizačního kódu slouží hašovací funkce typu MD5 nebo SHA-1. Po připojení autentizačního kódu vznikne zašifrovaný blok, který je přes protokol TCP přenášen po síti (Obrázek 12)[2].



Obrázek 16. Princip SSL Record Protocol

Ukázka Java kódu na straně klienta

Vytvoření zabezpečeného připojení pomocí SSL:

```
KeyStore ks = KeyStore.getInstance(KeyStore.getDefaultType());

char[] password = "changeit".toCharArray();

InputStream in = (new
FileInputStream("C:\\Java\\SSLCentralClient\\src\\sslcentralclient\\keystore.jks"));
ks.load(in, password);
in.close();

KeyManagerFactory kmf = KeyManagerFactory.getInstance("SunX509", "SunJSSE");
kmf.init(ks, password);

TrustManagerFactory tmf = TrustManagerFactory.getInstance("PKIX", "SunJSSE");
tmf.init(ks);

ctx = SSLContext.getInstance("TLS");
ctx.init(kmf.getKeyManagers(), tmf.getTrustManagers(), null);

socketFactory = ctx.getSocketFactory();
socket  = (SSLSocket) socketFactory.createSocket(serverAddress, port);
```

K vytvoření SSL spojení je nutné mít k dispozici příslušný certifikát. Certifikát jsem vytvořil pomocí Javového programu keytool. Takto vytvořený certifikát se pomocí funkce *load()* objektu *KeyStore* načte k dalšímu použití. Objekt *KeyStore* slouží jako „sklad“ certifikátu.

Pro domluvení, jaké se bude používat šifrování posílaných zpráv, slouží objekt *KeyManagerFactory,* který má v konstruktoru uloženo, jaký typ certifikátu se používá a kdo jej vytvořil. Pak je také nutné vytvořit objekt *TrustManagerFactory,* který ověřuje důvěryhodnost příslušného certifikátu a použité šifrování.

Ještě než je vytvořen samostatný *SSLSocket,* je potřeba specifikovat s jakým protokolem bude pracovat. Já jsem použil protokol TLS, což je jen modernější verze SSL protokolu. Nakonec je potřeba vytvořit objekt *SSLSocketFactory,* který vytvoří objekt *SSLSocket,* do jehož konstruktoru se předá příslušná IP adresa a port serveru, na který se bude aplikace připojovat.

Posílání zpráv:

```
SSLSocket socket;
socket  = (SSLSocket) socketFactory.createSocket(IPAddress, port);

BufferedReader readKeyboard = new BufferedReader(new InputStreamReader(System.in));
BufferedWriter write = new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream()));

    String line;
    while ((line = readKeyboard.readLine()) != null) {
       write.write(line);
       write.newLine();
       write.flush();
    }
```

Nejdříve je potřeba vytvořit objekt typu *SSLSocket,* kde se do jeho konstruktoru předá číslo portu a adresa příjemce. Objekt SSL*Socket* slouží k navázání spojení mezi klientem a serverem. Funkce *getOutputStream()* slouží k vytvoření kanálu mezi odesilatelem a příjemcem, do kterého se pomocí objektu *BufferedWriter* zapisuje proud dat. Cyklus *while* slouží k načítaná textu po řádcích a následné odeslání zprávy k příjemci.

Příjem zpráv:

```
SSLSocket socket;
socket  = (SSLSocket) socketFactory.createSocket(IPAddress, port);

BufferedReaderread = new BufferedReader(new
InputStreamReader(this.socket.getInputStream()));

String line = null;
while ((line = read.readLine()) != null) {
        System.out.println("Prijata zpráva:"+line);
        System.out.flush();
}
```

Pro příjem zpráv při použití SSL soketů, se opět použije objekt *SSLSocket*, ale teď místo funkce zápisu do soketu, se použije funkce *getInputStream(),* která slouží ke čtení ze soketu.

Ukázka Java kódu na straně serveru

```
KeyStore ks = KeyStore.getInstance(KeyStore.getDefaultType());

char[] password = "changeit".toCharArray();

InputStream in = (new
FileInputStream("C:\\Java\\SSLCentralServer\\src\\sslcentralserver\\keystore.jks"));

ks.load(in, password);
in.close();

KeyManagerFactory kmf = KeyManagerFactory.getInstance("SunX509", "SunJSSE");
kmf.init(ks, password);

TrustManagerFactory tmf = TrustManagerFactory.getInstance("PKIX", "SunJSSE");
tmf.init(ks);
ctx = SSLContext.getInstance("TLS");
ctx.init(kmf.getKeyManagers(), tmf.getTrustManagers(), null);

SSLServerSocketFactory serverSocketFactory = null;
SSLServerSocket serverSocket = null;
SSLSocket socket = null;

serverSocketFactory = ctx.getServerSocketFactory();
serverSocket = (SSLServerSocket) serverSocketFactory.createServerSocket(port);

while (true) {
        socket = (SSLSocket) serverSocket.accept();
        ChatHandler handler = new ChatHandler(socket);
        handler.start();
      }
```

Na straně serveru probíhá nastavení zabezpečení obdobně jako u klienta. Až na to, že server vytváří *SSLServerSocket*, který čeká na příchozí komunikace od klienta. Po přijmutí komunikace od klienta vytvoří server, pro každého klienta nové vlákno které následně spustí. Tuto funkci reprezentují poslední tři řádky kódu.

Obsluhovací vlákno:

```
static ArrayList handlers = new ArrayList(10);

synchronized(handlers) {
            handlers.add(this);
        }
BufferedReader read = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
BufferedWriter write = new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream()));

while(!(line = read.readLine()).equalsIgnoreCase("/q")) {
            for(int i = 0; i < handlers.size(); i++) {
                synchronized(handlers) {
                    ChatHandler handler = (ChatHandler)handlers.get(i);
                    handler.write.write(line);
                    handler.write.newLine();
                    handler.write.flush();
                }
            }
 }
```

Nejdříve je potřeba vytvořit seznam, do kterého se uloží každé nově příchozí spojení (uživatel) k tomuto slouží objekt *ArrayList*. Toto přidání do seznamu je nutné synchronizovat, aby nedocházelo k zablokování jednotlivých vláken. Pak je nutné vytvořit mechanismus, který bude procházet seznam klientů a bude jim posílat příchozí zprávy k tomuto slouží cyklus *for* a funkce *write*.

7 Závěr

Cílem bakalářské práce bylo vysvětlit základní mechanismy přenosu dat a jejich zabezpečení v informačních sítí. A také vytvoření aplikace, která bude zprostředkovávat zabezpečený přenos posílaných textových zpráv přes tyto sítě.

V praktické části jsem se zaměřil na vytvoření vlastní aplikace, která bude zajišťovat bezpečný přenos textových zpráv v informační síti. Jako programovací jazyk jsme si zvolil jazyk Java, protože obsahuje mnoho užitečných funkcí pro práci se síťovými aplikacemi.

V bakalářské práci jsem naprogramoval dvě aplikace. Funkce obou aplikací je stejná. Uživatel zadá do konzole text, který se šifrovaně pošle přes informační síť k příjemci, avšak každá z těchto aplikací pracuje s různými transportními protokoly a s různým šifrováním posílaných zpráv.

První aplikace k transportu zpráv používá UDP sokety a k zabezpečení je použita symetrická bloková šifra AES. Tato šifra je při použití dostatečně dlouhého a náhodné klíče velice bezpečná. Výhoda této aplikace je snadná implementace a rychlost posílání a přijímání zpráv, protože nevyžaduje žádné zpětné potvrzování správně přijatých zpráv. Ale právě toto nepotvrzování je i největší nevýhoda, protože si klient nemůže být jist, že zpráva došla v pořádku na místo určení. Další nevýhoda této aplikace je distribuce klíčů.

Druhá aplikace k transportu i zabezpečení zpráv používá SSL sokety. Použití protokolu SSL je velice bezpečné a probíhá zde i potvrzování správně přijatých zpráv. Největší výhoda tohoto řešení je snadná distribuce klíčů, dobré šifrování a zaručení autentičnosti uživatelů.

První aplikaci lze s výhodou snadné implementace použít v malých sítí, kde není potřeba zcela jisté doručení zpráv, ale je zde potřeba kvalitní šifrování. Jako příklad využití této aplikace lze uvést malé firemní sítě. Druhá aplikace se hodí pro větší sítě, kde je potřeba kvalitní zabezpečení i autentizace klientů. Tuto aplikaci lze využít například v sítích státní zprávy.

8 Použitá Literatura

[1] BOUŠKA, Petr. TCP/IP - adresy, masky, subnety a výpočty. Www.SAMURAJ-cz.com [online]. 2005 [cit. 2007-09-05]. Dostupný z WWW: <http://www.samuraj-cz.com/clanek/tcpip-adresy-masky-subnety-a-vypocty/>.

[2] Burda, K.Bezpečnost informačních systémů. Skripta FEI VUT v Brně. Brno: MJ Servis, 2005, 104 stran.

[3] HAAN, Laurent. Advanced Encryption Standard (AES). Dostupné na internetu: http://www.progressive-coding.com/tutorial.php?id=0

[4] MANDÍK, Petr. Sokety: Zprostředkovatelé síťové komunikace. COMPUTERWORLD [online]. 2001 [cit. 2001-10-20]. Dostupný z WWW: <http://archiv.computerworld.cz/cwarchiv.nsf/clanky/3BEC01559F035156C1256B480047D233?OpenDocument>.

[5] PETERKA, Jiří. Co je čím ... v počítačových sítích. EArchiv.cz : archiv článků a přednášek Jiřího Peterky [online]. 1991 [cit. 1991-11-01]. Dostupný z WWW: <http://www.earchiv.cz/i_coje.php3>.

[6] ZABALA, Enrique. Rijandeal Cipher. Dostupné na internetu: http://www.cs.bc.edu/~straubin/cs381-05/blockciphers/rijndael_ingles2004.swf

9 Seznam příloh

Veškeré přílohy se nacházejí na DVD nosiči. Jeho obsah je:

- elektronický text práce ve formátu pdf
- aplikace, které byly spouštěny v operačním systému Windows XP, v programovacím prostředí NetBeans IDE 6.1
- zdrojové kódy a knihovny potřebné ke spuštění aplikací

Abecední seznam použitých zkratek:

AES	Advanced Encryption Standard
AH	Authentication Header
ASCII	American Standard Code for Information Interchange
DDN	Defense Data Network
DES	Data Encryption Standart
DNS	Domain Name Systém
EBCDIC	Extended Binary Coded Decimal Interchange Code
ESP	Encapsulating Security Payload
IP	Internet Protocol
ISO	International Standards Organization
NAT	Network Address Translation
OSI	Open Systems Interconnection
RSA	Rivest, Shamir, Adleman
SNDAP	SubNetwork Dependent Access Protocol
SNDCP	SubNetwork Dependent Convergence Protocol
SNICP	SubNetwork Independent Convergence Protocol
SSL	Secure Sockets Layer

TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol