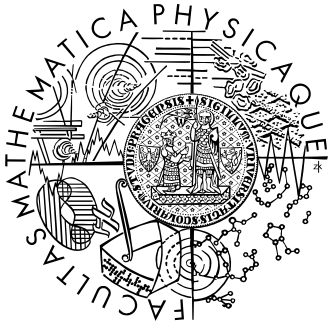
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Prokop Severýn

Šifrovaná VPN (multipoint)

Středisko informatické sítě a laboratoří

Vedoucí bakalářské práce: Dan Lukeš

Studijní program: Informatika, Správa počítačových systémů

2008

Děkuji vedoucímu své práce, Danu Lukešovi za odborné vedení mé práce, trpělivost, čas a rady bez kterých by tato práce nevznikla.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne 29. května 2008 Prokop Severýn

Obsah

Název práce: Šifrovaná VPN (multipoint)
Autor: Prokop Severýn
Katedra (ústav): Středisko informatické sítě a laboratoří
Vedoucí diplomové práce: Dan Lukeš
e-mail vedoucího: lukes@sisal.mff.cuni.cz

Abstrakt: Cílem této práce je vytvořit programové vybavení pro kryptograficky chráněný přenos většího množství dat v prostředí IP sítě. Kryptografické metody jsou zvoleny s ohledem k vyšší rychlosti šifrování při zachování vysoké úrovně zabezpečení. Vzájemnou komunikaci dvou uzlů je možné v případě nemožnosti sestavit přímé IP spojení vést přes jiné účastníky sítě. Součástí programu je také vlastní generátor pseudonáhodných čísel, na jehož kvalitě je založena bezpečnost použitých technik zabezpečení. Program je konzolová aplikace s jednoduchým textovým uživatelským rozhraním.

Klíčová slova: zabezpečené spojení, PRNG, multipoint

Title: Encrypted VPN (multipoint)
Author: Prokop Severýn
Department: SISAL
Supervisor: Dan Lukeš
Supervisor's e-mail address: lukes@sisal.mff.cuni.cz

Abstract: The purpose of this project is to construct a software equipment for a cryptography protected transfer of a biggest information volume in IP network. The cryptography methods are selected for a higher rate of an encryption with currently keeping high level of safety. It is possible to route the communication of two points, over other members of network. The pseudorandom generator is the part of the program, the safety of used algorhitms is based on PRNG quality. The program has a simple text user interface.

Keywords: safe connection, PRNG, multipoint

1 Úvod

Potřeba zabezpečit informaci tak, aby jí porozuměl pouze ten, pro koho je určena, je mezi lidmi přítomna odpradávna. Již za dob antiky se šifrování používalo pro vojenské účely a z této doby také pochází dodnes známá „Caesarova“ šifra. Postupem času vznikla kryptografie jako vědní obor zabývající se utajováním smyslu informace a se vzrůstající hodnotou informace rostl i její význam. Naopak zájem o získání zašifrované informace vedl ke vzniku vědy kryptoanalýzy, která zkoumá možnosti rozšifrování informací bez znalosti potřebných klíčů. S rozvojem počítačů zažily obě tyto vědy nebývalý rozmach a pokrok, takže dnes známe bezpečné asymetrické šifry i rychlé šifry symetrické. Cílem této práce je zkombinovat klady obou těchto technologií a vytvořit software, který bude schopen poskytnout jistoty asymetrické šifry a přitom dokáže v reálném čase přenášet velké množství dat. Dalším úkolem této práce je pro potřeby programu vytvořit kvalitní zdroj náhodných čísel a přidat programu schopnost vypořádat se s problémy v přímého spojení dvou uzlů v IP síti, které je možné obejít přes jiné uzly.

2 Genererátor pseudonáhodných čísel

2.1 Úvod

Náhoda nás v přírodě obklopuje na každém kroku. Z definice se jedná o jev, který může, ale nemusí nastat, přičemž existenci tohoto jevu nemůžeme nikterak ovlivnit ani předpovědět. Při bližším zkoumání se však ukazuje, že i většina přirozených jevů má své zákonitosti a často se dokonce cyklicky opakuje. My však náhodu potřebujeme využít a zaznamenat v systémech používaných lidmi a ty mají ještě výrazně větší sklon k pravidelnosti. Pro člověka je tedy spíše důležitá náhodná veličina, což je proměnná, jejíž hodnota je výsledkem náhodného pokusu. Tuto „náhodu“ v číselné podobě potřebujeme v mnoha situacích, například v simulacích, loteriích a kryptografii. Nejpřímější a také historicky první způsob získávání náhodných čísel je opakovat náhodný pokus a zaznamenávat jeho výsledek, čímž vznikne tabulka náhodných čísel. Jedna z nejznámějších a největších je tabulka náhodných čísel produkovaná od roku 1947 společností RAND Corporation (Research And Development). Pokrok informatiky a rozvoj technologií časem vedl k potřebě větších souborů náhodných čísel a již roku 1964 ve vývojových laboratořích Boeingu technici MacLaren a Marsaglia použili míchající generátor k získávání dalších sad náhodných čísel z původní tabulky od RAND Corporation. Tento jednoduchý program bývá označován jako první PRNG (Pseudorandom number generator) generátor pseudonáhodných čísel. Teorie generování pseudonáhodných čísel od té doby značně pokročila a našla uplatnění v běžných aplikacích, především pro potřeby kryptografie.

Pro program Tunel jsem potřeboval generátor dodávající velké množství dostatečně náhodných čísel pro použití jak k tvorbě klíčů DES, tak i pro samotné šifrování. Možností se nabízí mnoho, jednak generátory softwarové, tak i jistě kvalitnější generátory hardwarové. Hardwarové generátory zaznamenávající náhodné fyzikální jevy jako posloupnost náhodných čísel dodávají nejkvalitnější soubor náhodných čísel, ale protože dosud nejsou běžnou periférii domácích PC, nepřipadalo mi vhodné jejich

využití jako primárního zdroje náhodných čísel pro program Tunel. Naopak velmi snadno dostupná funkce standardní knihovny C random je pro potřeby programu Tunel zcela nedostatečná, pro svou nepříliš kvalitní náhodnost, nehledě na potřebu náhodného semínka. Tabulku náhodných čísel jsem vyloučil z důvodů konečnosti, objemnosti a především z důvodu špatné dostupnosti dobře utajených tabulek

2.2 Semínko (seed) pro PRNG

Jako zdroj semínka programu jsem zvolil metodu využívající délku intervalu mezi dvěma voláními funkce „vlož“, která jsou rozmístěna v různých částech programu. Interval je měřen pomocí čítače procesoru, který musí mít dostatečnou přesnost (alespoň mikrosekund), což je při rychlosti dnešních procesorů naprosto běžné. Z hodnoty vracené čítačem využívám pouze nejnižší platný bit. Program je psán pro multiprocesový operační systém Windows XP, je vícevláknový a jeho běh je závislý na interakci jiných uzlů. Tyto důvody mě opravňují považovat výstup tohoto způsobu získávání náhodného semínka za dostatečně nepredikovatelný pro účely programu Tunel. Přesto v případě nasazení programu v aplikacích s vysokým nárokem na bezpečnost musím doporučit použití kvalitního hardwarového generátoru náhodných čísel jako zdroje semínka pro můj pseudonáhodný generátor.

2.3 Konstrukce PRNG

PRNG programu Tunel je navrhnut na základě modifikované kombinace návrhu pánů G. Marsaglia a A. Zemana, např. MZRAN a MZRAN13. Program obsahuje dvě tabulky vzorců tab1 a tab2. Vždy když je požadována posloupnost pseudonáhodných čísel, program „náhodně“ zvolí po jednom vzorci z každé tabulky. V každém kroku výpočtu se vzorce sčítají modulo 2^32. Program vygeneruje 40000 čísel a z těchto je odzadu naplněn 1Mb zásobník pseudonáhodných čísel a zbytek (tedy počátek) posloupnosti je zahozen. Získávání semínka viz „Semínko pro PRNG“.

tab1[1]**:**

0) a=(69069*x(n-1) + R) mod 2^32
1) a=(x(n-1)*x(n-2)) mod 2^32

tab2[2]**:**

0) b=(x(n-4)-x(n-1) + carry) mod (2^31 - 69)
1) b=(x(n-1)+x(n-2) + carry) mod 2^32
2) b=(x(n-1)+x(n-2) + carry) mod 2^31
3) b=(x(n-2)+x(n-3) + carry) mod 2^31
4) b=(x(n-4)-x(n-5) - carry) mod (2^31 - 1)
5) b=(x(n-2)-x(n-5) + carry) mod (2^32 - 10)

U vzorce tab1/0 musí být konstanta R lichá. Délka periody posloupnosti vzniklé sčítáním dvou generátorů se zpravidla rovná nejmenšímu společnému násobku dílčích period těchto generátorů. Vzhledem k tomu, že každý z dílčích operátorů má délku periody větší než 2^31, a délka generované posloupnosti je 40000, je pro program zcela dostatečná délka periody libovolné kombinace. Vylepšení oproti návrhům tedy spočívá v dynamické volbě konfigurace generátoru pro každou dílčí generovanou posloupnost a dynamické volbě konstanty R.

2.4 Testy

Pro základní ověření kvality vytvořeného PRNG, jsem použil základní sadu testů náhodných posloupností, implementovaných v programu ENT od Johna Walkera[3]. Program provádí testy: informační hustoty, chi-kvadrat, aritmetický průměr, Monte Carlo test pí hodnoty a test sériové korelace. Testoval jsem na vzorku dlouhém 50000 bitů, což je délka posloupnosti

[1] Marsaglia, G., ZemanaA.: A new class of random generátor. The Annals of Applied Probability, No. 1, s. 462-480, 1991.
[2] tamtéž
[3] http://www.fourmilab.ch/random

dostačující programu tunel k vygenerování více než 750 DES klíčů. Následují komentované výstupy testů.

Entropy = 7.996114 bits per byte. Optimum compression would reduce the size of this 50000 byte file by 0 percent.[4]

Tento výsledek vypovídá o prakticky ideální hustotě dat.

Chi square distribution for 50000 samples is 270.04, and randomly would exceed this value 24.73 percent of the times. [5]

Toto považuji za velmi dobrý výsledek, pro srovnání Park-Millerův generátor dosahuje výsledků chi-kvadrát distribuce 212.53 a překračuje ji v 97.53 procentech případů. Náhodná data získaná pomocí vyzařování radioaktivních částic mají distribuční hodnotu 249.51 a překračují ji v 40.98 procentech případů, oba vzorky však na desetinásobně větších datech, než test mého generátoru.

Arithmetic mean value of data bytes is 127.6705 (127.5 = random).
Monte Carlo value for Pi is 3.128285131 (error 0.42 percent).
Serial correlation coefficient is 0.003792 (totally uncorrelated = 0.0). [6]

Výsledky posledních tří testů vypadají také velmi dobře, rozhodně nevylučují náhodnost sekvence a takto mírné statistické odchylky jistě nemohou dopomoci k prolomení šifrování.

[4] Výstup programu ent, Walker John, http://www.fourmilab.ch/random
[5] Výstup programu ent, Walker John, http://www.fourmilab.ch/random
[6] Výstup programu ent, Walker John, http://www.fourmilab.ch/random

3 Tunel – šifrování

3.1 Úvod

Program Tunel řeší problém komunikace dvou počítačů skrze nezabezpečený a nedůvěryhodný komunikační kanál, jímž jsou obecně rozsáhlé počítačové sítě. Tunel zajišťuje důvěrnost dat, tedy utajení informace před neoprávněnou osobou. Tunel zajišťuje důvěrnost dat (tedy utajení informace před neoprávněnou osobou), integritu dat (tj. garantuje, že data jakkoliv pozměněná nebudou považována za pravá) a také autentizaci dat (tj. ověřuje, že obě strany komunikace jsou opravdu tím, za koho se vydávají). Toto program zajišťuje i v případě, kdy existuje „nepřítel“, který má přístup ke všem informacím proudícím skrze informační kanál a může je libovolně pozměňovat. Zajišťuje to vše i v případě pokud „nepřítel“ získá sadu částí k sobě příslušejícího otevřeného a šifrovaného textu. A bez ohledu na složení odesílaného textu (tedy například i text složený ze samých nul). Tyto vlastnosti program splňuje již z předpokladu zadání této práce jako program s veřejným zdrojovým kódem. Od uživatelů požaduje vlastnictví utajeného soukromého RSA klíče a veřejného RSA klíče druhé strany, a to délky 512, 1024, nebo 2048 bitů. Pro dostatečnou úroveň zabezpečení doporučuji použít klíč délky 2048 bitů.

3.2 Šifrovací techniky

Již z důvodu výměny klíčů bylo nutné použít šifru asymetrickou, rozhodl jsem se pro šifru RSA (dle počátečních písmen příjmení autorů Ronald Rivest, Adi Shamir, Leonard Adleman), publikovanou v roce 1978. Šifra je založena na obtížnosti prvočíselného rozkladu velkého přirozeného čísla. Pro tuto úlohu není dosud znám žádný algoritmus pracující alespoň v polynomiálním čase, současně ale není znám ani důkaz jeho neexistence. Se vzrůstajícím výpočetním výkonem počítačů roste riziko prolomení klíčů (metodou faktorizace, jež má exponenciální časovou náročnost) v případě jejich nedostatečné délky. Z tohoto důvodu je doporučeno používání klíčů o

délce 2048 bitů a delších, klíč o délce 512 bitů již od roku 1999 nelze označit za bezpečný. Pro zajištění bezpečnosti před útoky pomocí měnícího se textu je v programu použito vyplňovacího schématu dle PKCS#1.

S rostoucí délkou klíče se zvyšuje čas potřebný na zašifrování a dešifrování zpráv, a to je u programu určeného k přenosu většího množství dat neúnosné. Proto jsem pro přenos dat využil šifru symetrickou, která má značně menší výpočetní složitost než šifry asymetrické. Jako použitou šifru jsem zvolil blokovou šifru TripleDES-EDE v modu CBC a použil metodu solení. Šifra DES je v různých obměnách jedna z nejpoužívanějších šifer současnosti. Roku 1977 byla přijata jako standard FIPS (Federal Information Processing Standard) a později se stala součástí mnoha dalších standardů. Již od svého vzniku podléhala kritice pro svůj příliš krátký klíč, s nímž je spojená nízká úroveň ochrany před útokem hrubou silou. Tato nevýhoda byla experimentálně doložena sestrojením zařízení DES-cracker v roce 1998, které je schopno prohledat celý klíčový prostor o velikosti 2^{56} klíčů do devíti dnů. Od roku 1976 známá komplementárnost šifry, snižuje odolnost proti prolomení hrubou šifrou ještě o další bit. Tuto podstatnou nevýhodu šifry odstraňuje šifra TripleDES, která používá tři klíče DES k trojímu zašifrování otevřeného textu a dosahuje tím efektivní délky klíče 168 bitů. V programu Tunel použitý mód EDE určuje pořadí použití klíčů při šifrování (viz Specifikace).

I po prodloužení klíče na efektivní délku 168 bitů stále zůstaly některé slabiny šifry. Mimo jiné existence skupiny 16 slabých a poloslabých klíčů, které jsou při generování klíčů v programu vyřazovány. Použitý mód CBC částečně odstraňuje další nevýhodu DES, šifruje-li se každý blok dat zvlášť, ke stejnému otevřenému textu získáme stejný text uzavřený. Mód CBC však každý blok otevřeného textu modifikuje předchozím blokem šifrovaného textu ještě předtím, než ho zašifruje. Tím se dosáhne různých šifrovaných bloků pro shodné bloky otevřené. Aby toto platilo i pro dlouhé shodné sekvence dat, je k modifikaci prvního bloku použit náhodný, tzv. inicializační vektor IV. Aby celá sada dat procházejících skrz Tunel nebyla ohrožena ztrátou důvěrnosti (i kdyby „nepřítel“ získal k části šifrovaného textu část textu otevřeného, či prolomil 168 bitový klíč šifry), jsou sady klíčů a IV pro každý směr komunikace různé a v čase dynamicky měněny. Integritu dat zajišťuje program

pomocí kódu MAC. Každý blok otevřených dat se šifruje v modu CBC šifrou DES klíčem jiným, který není shodný ani s jedním ze tří použitých k šifrování TripleDES. IV je použit nulový. Nakonec je blok vzešlý z posledního bloku paketu zašifrován spolu se zprávou. Pokud příjemce po provedení stejného procesu získá jinou hodnotu výsledného bloku než je hodnota bloku přiloženého k paketu, identifikuje paket jako poškozený.

3.3 Realizace programu

Program Tunel je realizován nad TCP síťovým spojením a z vlastností tohoto spojení vychází i návrh jeho protokolu. Program rozpozná chyby síťového spojení, ale nedokáže se z nich zotavit. V případě chyby je tedy nutné uskutečnit celý přenos znovu. TCP spojení jsem zvolil z důvodu jednodušší implementace než v případě použití UDP spojení, nevýhodou je především vyšší režie. Ta je však částečně kompenzována nižší režií vlastního programu.

Program nejprve naváže TCP spojení, poté zašle druhé straně sadu tří klíčů pro TripleDES, jednoho klíče pro MAC a IV. Sadu získá pomocí generátoru náhodných čísel, který je součástí programu. Sadu odesílá zašifrovanou pomocí svého RSA soukromého klíče dle standardu PKCS#1 v2.0. Touto sadou bude následně šifrovat odchozí data. Sadu pro příchozí data přijme od druhé strany a dešifruje ji pomocí veřejného RSA klíče druhé strany. Tuto sadu následně používá pro dešifrování příchozí komunikace. Následující komunikace je již prováděna skrze spojení šifrované pomocí TripleDES a ověřením MAC kódem. Jako první paket odešle blok A1, a jako první paket přijme blok B, jako druhý paket odešle blok B a přijme blok A2. Pokud se A1 a A2 shodují je spojení úspěšně navázáno a dále již odesílá a přijímá uživatelská data. Program po náhodném počtu odeslaných paketů (v rámci daného intervalu) vymění sadu klíčů pro odesílání, a to včetně MAC a IV, opět šifrovaně dle RSAEP. Podrobný formální popis komunikačního protokolu a programu Tunel viz Specifikace.

3.4 Adresář

Součástí programu je adresář, bezpečně uchovávající seznam kontaktů. U každého kontaktu je uvedeno jméno, veřejný RSA klíč a IP adresa. Pro úspěšné navázání spojení se v adresáři musí nacházet kontakt se jménem „me“ a soukromým RSA klíčem uživatele. Adresář je na disk ukládán jako soubor zašifrovaný pomocí TripleDES společně s kontrolním MAC kódem. Klíče jsou získány z SHA-1 otisku hesla zvoleného uživatelem, přičemž velmi slabá hesla adresář nepřijme. Podrobný popis ovládání adresáře viz Uživatelský manuál.

4 Síťový „mód“

4.1 Úvod

Program Tunel může fungovat ve dvou síťových módech, v základním módu a v módu „net“. V základním módu je program Tunel schopen realizovat zabezpečený přenos dat pouze pokud je možné sestavit přímé TCP spojení mezi dvěma spojovanými stroji. Mód „net“ přidává schopnost sestavovat spojení přes jiné stroje, pokud přímé spojení není možné. K nepřímému spojení je možné využít pouze stroje běžící také v módu „net“.

4.2 Spojení

Při volbě způsobu směrování dat se mi přirozeně nabízela varianta ve světě počítačových sítí jednoznačně upřednostňovaná, tedy nesestavovat spojení, ale každý datový paket směrovat zvlášť. Od tohoto modelu jsem se již částečně odchýlil tím, že jsem se rozhodl program Tunel realizovat nad TCP protokolem, který sám simuluje spojované spojení nad paketovou síťí. Stále by sice bylo možné směrovat pakety i nad protokolem TCP, ale už by to bylo poněkud krkolomné a neefektivní. Směrování paketů a nesestavování spojení je v počítačových sítích obecně jistě lepší řešení, ale v případě programu Tunel tomu tak být nemusí. Je třeba vzít v úvahu situace, v kterých program nepřímého spojení využívá. Jsou to situace, kdy není možné doručovat TCP pakety mezi stroji A a B, ale přitom existuje cesta z A do B přes stroje C1, ..., Cn. To nastává např. pokud je počítač A umístěn v podnikové síti, ze které je provoz do internetu veden přes proxy server. V takovém případě může být spojení se strojem B v internetu skrze Tunel blokováno. V podnikové síti se mohou nacházet stroje, které mají do internetu neregulované spojení a běží na nich Tunel. Potom stroj A vybere jeden stroj z této skupiny, přes který bude spojení nejrychlejší, nechť je to C, a sestaví spojení A-C-B. Rychlost přenosu dat mezi A-B a B-C je optimalizována pod úrovní TCP protokolu a jediným úzkým hrdlem se může stát stroj C. Obecně předpokládám, že graf sítě strojů s běžícím programem Tunel bude velmi hustý a pro existující hrany bude platit

trojúhelníková nerovnost. Lze tedy říci, že větší část optimalizace toku dat zůstává realizována i když je Tunel vytvořen jako sestavené spojení nad TCP. Výhodou je, že odpadá režie nutná pro směrování vlastních paketů programu Tunel. Zvolená metoda „spojovaných“ přenosů program předurčuje spíše k jednorázovému přenosu většího objemu dat než k dlouhodobě sestavenému spojení s menším zatížením. S ohledem k tomuto využití je však program i navržen.

4.3 Hledání cesty

Program Tunel prohledává síť paralelně do šířky, každý uzel rozesílá dotaz všem svým sousedům. Nejlepší cesta je vybrána v každém uzlu grafu, takže původní tazatel nevybírá z celého prostoru možných cest. Toto řešení oproti možnosti výběru ze všech cest původním tazatelem (a s tím spojeným předáváním informace o všech cestách) výrazně snižuje síťovou režii. Každý dotaz si nese informaci o své hloubce, maximální hloubka dotazu je shora omezena. Doba, po kterou uzel čeká na výsledek svého dotazu, je také shora omezena. Tím je zabráněno vzniku nekonečných smyček dotazů.

4.4 Výběr cesty

Nejlepší cesta je vybírána na základě uspořádání daném funkcí `TCP_Net::ChooseWay(depth_1, depth_2, time_1, time_2)`, kde depth určují délku nalezené cesty a time čas od odeslání dotazu do přijetí odpovědi. Funkce vrátí false pokud je cesta 1 lepší než cesta 2. Funkce njdříve vytvoří koeficient rozdílu pro časy i hloubky, pro časy je to

q= [time_1 / (time_1 - time_2)]

a pro hloubky obdobně. Poté podle koeficientů zvolí, zda bude pro lepší cestu určující délka cesty nebo rychlost odezvy. Konečný výběr dle koeficientů je nastaven tak, aby vybíral podle času, až na případy, kdy mezi časy je malý rozdíl (q-času < 1), nebo kdy je naopak vyrazně větší rozdíl délek cest, než časů odezvy (q-délky*3 > q-času).

4.5 Bezpečnost

Program nemá kontrolu nad tím, přes jaké uzly budou jeho data směrována (pokud je povolen mód „net“), a proto mohou být směrována i přes nedůvěryhodné uzly. Aby tato skutečnost nikterak neohrozila bezpečnost, přeposílající uzel do datového toku nezasahuje a pouze data předává. Vzhledem k charakteru šifrování nevzniká další riziko tím, že může data (šifrovaná) libovolně číst a pozměňovat.

4.6 Realizace

Program vytvoří jedno „net“ vlákno (běží s nižší prioritou než hlavní vlákno programu), které zajišťuje podporu síťového provozu pro ostatní uzly. Vlákno očekává příchozí žádosti o spojení s jiným strojem, a pro každou formálně správnou příchozí žádost vytvoří další vlákno tzv. „pracovní“ (s nižší prioritou), které tuto žádost zpracuje. Pracovní vlákno vyhodnotí požadavek a v případě, že je to možné, naváže přímé spojení s cílem. Pokud se s cílem nemůže přímo spojit, rozešle požadavek všem kontaktům v adresáři a počká na odpovědi. Podle délky odezvy a délky nalezené cesty, vybere nejvhodnější spojení a tazateli pošle informaci o možnosti spojení. Pokud tazatel spojení potvrdí, pracovní vlákno přeposílá obousměrně komunikaci. V případě že se nepodaří najít cestu k cíli, odešle informaci tazateli a skončí. Podrobný popis protokolu viz Specifikace.

5 Uživatelská dokumentace k programu Tunel

Program Tunel slouží k obousměrnému přenosu dat mezi dvěma PC v IPv.4 síti. Zajišťuje důvěrnost, integritu a autentizaci přenášených dat. Program je určen pro operační systém Windows XP s nainstalovaným TCP komunikačním protokolem. Od uživatele vyžaduje znalost IP adresy PC, s kterým se chce spojit, vlastní soukromý RSA klíč a veřejný RSA klíč uživatele PC s kterým chce komunikovat. Použité RSA klíče musí odpovídat standartu PKCS#1, mohou být délky 512, 1024, nebo 2048 bitů. Pro zachování vysoké míry zabezpečení doporučuji používat klíče o délce 2048 bitů. Program přijímá data ze standardního vstupu, dokud nenarazí na EOF a posílá je skrze šifrované spojení a přijatá data z šifrovaného spojení zapisuje na standardní výstup. Součástí programu je adresář kontaktů, který je šifrován a uložen v souboru adressbook.txt v adresáři programu.

5.1 Editace adresáře

Prvním parametrem příkazové řádky musí být vždy heslo k adresáři, pro spuštění programu v režimu editace adresáře musí být druhým parametrem příkazové řádky "-E".

Požadavky na heslo:

- heslo musí mít délku alespoň 8 znaků;
- heslo musí obsahovat alespoň jeden nealfabetický znak;
- heslo musí obsahovat malá i velká písmena.

Při prvním spuštění programu, nebo při prázdném souboru adressbook.txt, bude heslo použito pro zašifrování adresáře, při dalších spuštění bude heslo použito i k jeho rozšifrování. V adresáři musí být uložena položka se jménem "me" a privátním RSA klíčem uživatele, který bude použit pro dešifrování příchozí komunikace.

Příkazy v editačním režimu adresáře:

- příkaz pro vložení záznamu -I "jméno" "IP adresa" "soubor s RSA klíčem pro odesílání" (Jméno musí mít 1 až 20 znaků, IP 7 až 15 znaků napr. "127.0.0.1", klič zapsán v šestnáctkové soustavě);
- příkaz pro smazání záznamu -D "jméno";
- příkaz pro výpis adresáře –P;
- příkaz pro výpis IP adresy záznamu -S "jméno";
- příkaz pro uložení adresáře a ukončení programu –X.

5.2 Režim přenosu dat

Pokud program není spuštěn v režimu editace adresáře, poběží v režimu pro přenos dat. V tomto režimu může běžet v módech Klient, nebo Server a v obou módech může být zapnut ještě mód „net“.

- Program funguje defaultně v módu klient.
- Parametr příkazové řádky -S, pro běh v módu server.
- Parametr příkazové řádky -A, program se pokusí zvolit mód vhodně sám.

V módu server program čeká na připojení zadaného stroje, v módu klient se připojí k zadanému stroji, který je spuštěn v módu server.

Mód „net“ se spouští parametrem příkazové řádky -N. V tomto módu je spuštěno síťové rozšíření umožňující navázat spojení i se strojem, se kterým nelze navázat přímé IP spojení. V tomto módu program také složí jako spojovací uzel pro spojení mezi jinými stroji.

Pokud program není spuštěn v režimu editace adresáře je třeba jako druhý parametr příkazové řádky zadat jméno záznamu v adresáři, na který se má Tunel připojit.

Další nepovinné parametry příkazové řádky:

- -T"hodnota" - Určuje kolik sekund má program čekat na korektní ukončení spojení pokud vše odeslal a nic nepřijímá. Pokud 0 bude čekat stále. Default je 0;

- -P"hodnota" - kolik setin sekundy bude program čekat před zrušením IP spojení při ukončovaní programu. Default 100;
- -K"hodnota" - Kolik setin sekundy bude program čekat na další data po EOF. Default 0;
- -N povolí "síťový" mód programu;
- -W"hodnota" - kolik sekund bude program běžet před ukončením, pokud přenese všechna uživatelská data. Po Tuto dobu může zprostředkovávat spojení jiných strojů. Default 3600;
- -M"[0|1|2]" - volba pro ukončování programu v módu "síťovém módu". Určuje, jak rychle se má program ukončit po přenesení všech uživatelských dat. Volby jsou: 2(NET_KILL) pro okamžité ukončení, 1(NET_END) poskytne nějaký čas na doběhnutí již vytvořených spojením, 0(NET_RUN) běží dále po čas nastavený pomocí parametru -W. Default NET_RUN.

6 Programátorská dokumentace k programu Tunel

6.1 Struktura programu

Hlavní části programu tunel:

- ***Generátor náhodných čísel*** **–** poskytuje náhodné klíče pro DES a náhodné byty.
  Soubory: myRNG.h, myRNG.cpp
  Interface: `Key_Storage`

- ***TCP spojení*** **–** zajišťuje navázání a obsluhu tcp spojení.
  Soubory: tcp.h, tcp.cpp
  Interface: `TCP, TCP_Net`

- ***Tunel*** **–** hlavní třída programu, zajišťuje přijímání a odesílání paketů, jejich šifrování.
  Soubory: Tunnel.h, Tunnel.cpp
  Interface: `Tunnel`

- ***RSA*** **–** zajišťuje RSA šifrování.
  Soubory: myRSA.h, myRSA.cpp
  Interface: `RSA`

- ***Adresář*** **–** uchovává šifrovaně kontakty na jiné uživatele programu Tunel.
  Soubory: addressbook.h, addressbook.cpp
  Interface: `Book`

- ***Net*** **–** zajišťuje síťovou část programu, používá se pouze v módu „net“.
  Soubory: net.h,net.cpp
  Interface: `NET_THREAD`

- ***Synchro*** **–** zajišťuje synchronizaci přístupu ke kritickým sekcím, metodou pasivního čekání.
  Soubory: synchro.h, synchro.cpp
  Interface: `Lock`

- ***Timer*** **–** umožňuje přesné měření času.
  Soubory: Pro_Timer.h, Pro_Timer.cpp
  Interface: `Pro_Timer`

- ***Main Funkce*** **–** Zajišťuje uživatelský interface a hlavní funkce programu.
  Soubory: main.cpp

- ***Defines*** **–** Předdefinované konstanty programu
  Soubory: defs.h

5.1.1 Generátor náhodných čísel

Struktura – hlavní třída `Key_Storage` generuje náhodné klíče pro DES a vyřazuje z nich slabé klíče. Jako zdroj pseudonáhodných bitů používá objekt třídy `RNG`, popis algoritmů viz specifikace. Pro uchování klíče slouží třída `Key.`

5.1.2 TCP spojení

Zapouzdřuje práci s Win Socket Api, třída TCP slouží k obhospodaření základního spojení a její potomek TCP_Net rozšiřuje funkcionalitu pro režim „net“.

5.1.3 Tunel

Třída `Tunnel` realizuje základní operace komunikace (popis komunikačního protokolu viz Specifikace) mezi dvěma stroji, jako jsou: navázání spojení, odesílání různých typů paketů, výměna klíčů pro nastavení nové sady klíčů pro symetrické šifrování. Využívá instance tříd `Key_Storage, TCP, RSA, Book.`

5.1.4 RSA

Třída pouze zapouzdřuje využití knihovny Crypto++® Library. Umožňuje šifrovat 20 bytový otevřený text do 256 bytového šifrovaného textu a opačně dešifrovat.

5.1.5 Adresář

Třída `Book` implementuje operace nad adresářem. Metoda `load` načte šifrovaná dat ze souboru ./adressbook.txt“, text rozšifruje a záznamy uloží do std::vectoru `_records`. Pro uchování záznamu a operace nad záznamem slouží třída `Record` Je volána při spuštění programu. K šifrování používá knihovnu Crypto++® Library. Při ukončování programu je obsah `_records` opět zašifrován a uložen do souboru „./adressbook.txt“ pomocí metody `save`.

5.1.6 Net

Třída `NET_THREAD` zajišťuje síťovou funkčnost módu „net“. Obsahuje metody pro tvorbu „net“ vlákna a pracovních vláken, spravuje struktury udržující informace o pracovních vláknech `TARG`. Pracuje dle protokolu popsaném ve Specifikaci a vyobrazeném v příloze 2.

5.1.7 Synchro

Třída `Lock` implementuje synchronizaci pomocí objektu WIN API `CRITICAL_SECTION`, tento objekt ošetřuje přístup ke kritickým sekcím na bázi mutexu. Kritické sekce se v programu vyskytují poměrně zřídka.

5.1.8 Timer

Třída `Pro_Timer` využívá Win API funkce ke čtení hodnoty systémového čítače. Neměří v jednotkách času, ale v taktech procesoru, proto jej nelze využít tam, kde je třeba návaznost na jednotky času odvozené od SI.

5.1.9 Main Funkce

Funkce `main()` realizuje jednoduchý uživatelský interface a logiku komunikačního protokolu (viz. specifikace Tunel). Pro tunelování uživatelských dat využívá instanci třídy `Tunnel`. Pokud program běží v „net“

módu, funkce `main`() vytvoří jednu instanci třídy `NET_THREAD` se kterou už kromě ukončování nemusí komunikovat.

7 Závěr

Cílem této práce bylo vytvořit experimentální program pro zabezpečený přenos dat kombinující rychlost symetrické kryptografie a bezpečnost asymetrické kryptografie, který navíc umožňuje obcházet nepropustnosti IP sítě. Domnívám se, že díky poměrně zdařilému PRNG (alespoň dle provedených testů) bylo dosaženo vysoké míry zabezpečení přenášených dat. Pro reálné nasazení aplikace bude jistě třeba dopracovat uživatelské rozhraní a především funkčnost programu v módu „net“ (například schopnost obnovení ztraceného spojení). Zcela jistě bude třeba program podrobit rozsáhlým zátěžovým testům a PRNG dalším testům náhodnosti.

8 Seznam použité literatury

[1] Marsaglia G., Zemana A.: Some portable very-long-period random number generators, Computer in Physics, Vol. 8, NO. 1, 1994

[2] Marsaglia G., Zemana A.: A new class of random generator, The Annals of Applied Probability, No. 1, p. 462-480, 1991

[3] Klíma Vlastimil: Generátory náhodných čísel I-IV. Chip, roč. 1998, č. 3-6

[4] Klíma Vlastimil.: Základy moderní kryptografie I,II,III, 2005

[5] Menezes Alfred J., Van Oorschot Paul C., Vanstone Scott A.: Handbook of Applied Cryptography, 1996

[6] Federal Information Processing Standards Publication 46-3 (DES), 1999

[7] PKCS #1 v2.0:RSA Cryptography Standard

[8] portál Crypto-World http://crypto-world.info/

[9] Bruce Eckel, Thinking in C++ 2nd Edition Volume 1, 2000

[10] Bruce Eckel, Thinking in C++ 2nd Edition Volume 2, 2003

[11] Walker John, http://www.fourmilab.ch/random/

[12] MSDN: Microsoft Developer Network, http://msdn.microsoft.com

[13] Klíma Vlastimil Dr.: cryptography.hyperlink.cz

[14] Sportack Mark A., Směrování v IP

[15] Peterka Jiří, http://www.earchiv.cz/

[16] Pavienský Radek: Přesné měření času ve Windows, Progres, 2000

[17] Builder, http://www.builder.cz

9 Přílohy

Seznam příloh:

Příloha 1 Specifikace
Příloha 2 Schéma komunikace při vyhledávání spojení
Příloha 3 CD-ROM se softwarovým dílem

9.1 Příloha 1 – Specifikace

Generátor náhodných čísel

Generuje náhodné klíče pro 3DES s vyloučením potenciálně slabých klíčů.

POOL - zásobník bitů

Obsahuje dva pooly: pool1 a pool2.

Uspokojení požadavků na náhodné bity:

a) Je-li pool1 zcela plný, uspokojí se požadavek výběrem z poolu1.

b) Není-li pool1 zcela plný nebo nepostačuje-li dostupný počet bitů v poolu1, uspokojí se výběrem z poolu2.

POOL 1

Pool1 je bitová fronta velikosti 1MB.

Plnění poolu 1

a) Průběžně běžící algoritmus „generování náhodných bitů pro pool1"
generuje náhodné bity a ty vkládá na konec poolu1.
Při přeplněníi fronty jsou nejstarší bity odstraňovány.

b) Je-li potřeba větší než je aktuální obsah poolu1, program explicitně volá funkci vlož, a tím umožní vygenerování dalších náhodných bitů podle bodu [a], vstup je ukončen po naplnění poolu1.

Výběr z poolu1

Ze začátku poolu se vybere požadovaný počet bitů, nepostačuje-li dostupný počet požadavku, pak se podle postupu „plněni poolu1" vygeneruje další sada bitů.

Generováni bitů pro pool1

V různých místech programu jsou rozmístěny volání funkce „vlož".
Při každém volání (vyjma prvního) vezmeme rozdíl času tohoto volání a času předcházejícího volání. Nejméně významný bit tohoto rozdílu je jedním náhodným bitem, který je vložen do poolu1.

Časoměr (timer)

-přesnost časoměru nesmí být menší, než nejmenší jednotka, kterou vrací
-musí vracet hodnotu s rozlišením alespoň mikrosekund
-vrací binární číslo (na pozici nejvýznamnějšího bitu nezáleží)
aktuální čas:
-čas od startu systému nebo programu.

POOL 2

Pool2 je bitová fronta velikosti 1MB.

Plnění poolu 2

Vybere se 90 bitů z poolu 1 a algoritmem „generování pseudonáhodných bitů pro pool2" se vygeneruje celý obsah
poolu2.

Výběr z poolu2

Ze začátku poolu se vybere požadovaný počet bitů, nepostačuje-li dostupný počet požadavků, podle postupu „plnění poolu2" se vygeneruje další sada bitů a to se opakuje tak dlouho, dokud není požadavek uspokojen.

Generování pseudonáhodných bitů pro pool2

Vybere 90 bitů z poolu1, 80 z nich rozseká na 5 částí délky 16 bitů, každá je chápána jako binární zápis kladného celého čísla, (na tom, jestli je nejvýznamnější bit vlevo nebo vpravo nezáleží).

Tato čísla tvoří prvních 5 prvků posloupnosti x(n), pokud je některý z nich nulový, vybere se místo něj nový (dalších 16 bitů z poolu1).

Generování dalších prvků posloupnosti

- nejméně významný bit aktuálního času, určí první vzorec z tab1
- druhý, třetí a čtvrtý nejméně významný bit aktuálního času je chápán jako číslo B (stejně jako vise)
- B mod 6 určí vzorec z tab. 2
- zbylých 10 bitů z 90ti vybraných je chápáno jako kladné celé číslo R
- R:=R*2+1

Vzorce se při výpočtu sčítají v modulu 2^32, tedy x(n)=a+b mod 2^32.

tab1

0) a=(69069*x(n-1) + R) mod 2^32

1) a=(x(n-1)*x(n-2)) mod 2^32

tab2

0) b=(x(n-4)-x(n-1) + carry) mod (2^31 - 69)

1) b=(x(n-1)+x(n-2) + carry) mod 2^32

2) b=(x(n-1)+x(n-2) + carry) mod 2^31

3) b=(x(n-2)+x(n-3) + carry) mod 2^31

4) b=(x(n-4)-x(n-5) - carry) mod (2^31 - 1)

5) b=(x(n-2)-x(n-5) + carry) mod (2^32 - 10)

carry je na začátku výpočtu roven 1 a v každém kroku výpočtu se znovu rozhoduje o jeho hodnotě.

Jestli b nepřesáhne modul, nastaví se carry na 0, pokud b modul přesáhne, nastaví se carry na 1.

Tímto postupem se vygeneruje posloupnost 40 000 čísel (min. perioda posloupnosti je 2^62), ty jsou zapsány jako posloupnost bitu do poolu2 (1Mb), a to od konce. Přebytek (tedy začátek posloupnosti) se zahodí.

SKLAD KLIČŮ

Má kapacitu 256 klíčů pro DES.

Plnění Skladu klíčů

Vždy po vkládání (vkládá se vždy celá sada najednou) či vybírání ze skladu je kontrolován počet klíčů ve skladu, pokud jich alespoň 16 chybí, generují se dle postupu „generování sady klíčů".

Generování sady klíčů

Vezme 1024 bitů z poolu, ty se rozdělí na 16 úseků o délce 64 bitů, z každého úseku se podle postupu „generování klíče" vygeneruje 56 bitový klíč.

Ze sady jsou vyhozeny slabé (známé jsou 4) a poloslabé (je jich známo 12) klíče. Sada se vloží do skladu klíčů (Může dojít k tomu, ze všechny klíče budou vyházeny, pak je tedy sada prázdná).

Slabé klíče

Existují 4 slabé klíče K, pro které platí X=EK(X) pro každé X. Dále existuje šest dvojic poloslabých klíčů (K1, K2), pro než platí X=EK2(EK1(X)), pro všechna X.

Jsou to:

0101 0101 0101 0101, FEFE FEFE FEFE FEFE,
1F1F 1F1F 0E0E 0E0E, E0E0 E0E0 F1F1 F1F1,

01FE 01FE 01FE 01FE, FE01 FE01 FE01 FE01,
1FE0 1FE0 0EF1 0EF1, E01F E01F F10E F10E,
01E0 01E0 01F1 01F1, E001 E001 F101 F101,
1FFE 1FFE 0EFE 0EFE, FE1F FE1F FE0E FE0E,
011F 011F 010E 010E, 1F01 1F01 0E01 0E01,
E0FE E0FE F1FE F1FE, FEE0 FEE0 FEF1 FEF1

Generování klíče

Prvních 8 bitů 64 bitového úseku je interpretováno jako číslo R z rozsahu 0-55 (tedy mod 56) a zbylých 56 bitů je cyklicky rotováno o R bitu doleva, výsledné 56 bitové číslo je klíč.

Výběr klíče

Vždy, když je po generátoru požadován klíč, vybere se ze zásobníku klíčů dle postupu „určení pozice klíče" a je ze zásobníku smazán.

Určení pozice klíče

Vezme 8 nejméně významných bitů aktuálního času a chápe je jako číslo R.
R modulo počet klíčů v zásobníku je pořadí klíče v zásobníku.

TUNEL

Program posílá standardní vstup skrze vytvořené šifrované spojení jako proud bitů a stejně tak přijímá ze spojení proud bitů, který posílá na std. výstup (kromě dat potřebných pro svou režii).

IV (Initializing value) - náhodná posloupnost 64 bitů.
Sada klíčů - tři 56 bitové klíče pro 3DES + jeden 56 bitový pro MAC + IV (K1K2K3K4IV).
Šifrování dat je prováděno v CBC modu 3DES ve variantě EDE (3DES-EDE-CBC).
Klíče K1,K2,K3 jsou použity v pozici ST=EK3(DK2(EK1(OT))).
Pro MAC je použit K4.
Data jsou šifrovaná v každém směru jinou sadou klíčů.

Paket

Typ- číslo typu, viz „typy paketu" -8b binárně zapsané kladné celé číslo jako big endian.
Velikost- udává velikost/64b, včetně hlavičky -8b binárně zapsané kladné celé číslo jako big endian.
Data- obsahuje přenášená data - k*64-16b pro k=(1-255)
MAC- MAC kód pro nezašifrovaný paket i s položkou „Typ" - 64b
(pokud nebude souhlasit délka, nebude souhlasit MAC a dle toho se postupuje)

Typy paketu-
1- data
2- potvrzení přijetí paketu čísla X, kde X jsou data, X je binárně zapsané kladné celé číslo jako big endian.

5- měním klíče směrem k tobě, za tímto paketem následuje 2*256 bitů, které po dešifrování RSA tvoří K1K2K3K4IV.

6- ukončuji spojení, ode mě máš vše.

7- ukončuji spojení, mám vše.

N (jiné číslo)- X jsou data, ale N-9 = počet bitů na konci datové části, které jsou balast (max 63 bitu, použiji, pokud nemám dost bitů na plnou délku bloku).

Počítadlo paketů-

Všechny pakety komunikace (kromě paketů typu 4) jsou počítány, a to v každém směru zvlášť

Když počítadlo dosáhne nejvyšší hodnoty, začne počítat zase od jedničky.

Počítadlo je 48 bitové, neznaménkové.

Spojení

Navazování spojení-

Program zahájí vytváření tunelu navázáním TCP spojení.

Pro komunikaci je třeba, aby obě strany znaly veřejné RSA klíče dle standartu PKCS#1 protějšku a měly i své vlastní soukromé RSA klíče.

Uživatel může mít RSA klíč délky: 512,1024,2048 bitů.

Při vytváření tunelu si strany vymění klíče a IV pomocí RSA, dle standartu PKCS#1 v2.0.

Každá strana pošle sadu klíčů (KO1,KO2,KO3,KO4,IVO) pro směr odesílání.

Klient spojí klíče a IV do jednoho bloku v pořadí KO1KO2KO3KO4IVO,

blok zašifruje dle RSAEP a odešle.

Dále už odesílá pakety šifrované 3DES KO1,KO2,KO3,IVO s MAC kódem dle KO4.

Jako data prvního odchozího paketu odešle 48 náhodných bitů, nechť je to posloupnost A.

Od druhé strany obdrží blok, který dešifruje dle RSADP,

tím získá (KP1,KP2,KP3,KP4,IVP) pro dešifrovaní příchozí komunikace a kontrolu MAC kódu.

Dále už příchozí pakety dešifruje 3DES KP1,KP2,KP3,IVP a kontroluje MAC dle KP4.

Jako data prvního příchozího paketu obdrží 48 náhodných bitu, nechť je to posloupnost B.
Odešle B jako data druhého odchozího paketu.
Pokud přijme jako data druhého příchozího paketu A, je tunel vytvořen.
Pokud jsou data druhého příchozího paketu nerovná A, ohlásí chybu a zruší spojení.

Velikost paketu

pokud jsou na vstupu stále data, vytváří pakety maximální velikosti (tj. 16Kb, max velikost paketu tohoto tunelu, ne TCP paketu).
pokud data na vstupu nestačí k naplnění paketu na max velikost (tj. když narazí na EOF), doplní paket na velikost nejbližšího celého násobku 64 balastem (ten vezme z generátoru), informaci o velikosti balastu uloží do typu paketu.

Odesílaní dat

Vstupní data jsou vložena do "paketu".
Paketu je spočítán MAC kód (pomoci K4).
MAC je k paketu připojen.
celý paket i s hlavičkou a MAC je zašifrován 3DES.
Paket je odeslán skrz TCP spojeni.

Příjem dat

Díky známým velikostem předchozích paketů ví jakým bitem z TCP streamu začíná další paket.
Dešifruje tedy první 64bitovy blok paketu, který obsahuje informaci o velikosti.
Celý příchozí paket načte z TCP streamu (zná počátek, velikost a tedy i konec).
Příchozí paket dešifruje pomocí platných klíčů.
Zkontroluje MAC kód dle K4.
Je-li MAC v pořádku

- Je-li to datový paket posle data na výstup.
- Není-li to datový paket, tak dle postupu "režie".

Není-li MAC v pořádku, paket zahodí a ohlásí bezpečnostní chybu spojení.

Ukončovaní spojení

Pokud na vstupním streamu přijde EOF, program počká K sekund a pokud do té doby nenásledují další data, ukončí spojení dle postupu "režie".

Pokud do K sekund na vstupním streamu najde další data, pokračuje dal v komunikaci přes vytvořené spojení.

K, nastaví uživatel. Implicitně 1, pokud je nastaveno K>=100, program EOF ignoruje.

Režie

- Přijde-li paket typu 2, označím si pakety do čísla X jako potvrzené.
- Přijde-li paket typu 5, dešifruji blok 256 bitu za paketem pomoci soukromého klíče RSA, rozdělím ho na K1 K2 K3 K4 IV, v příchozím směru přejdu na tyto klíče. 2
- Přijde-li paket typu 6, vím že druhá strana odeslala vše a tak poslu paket 2 kterým potvrdím poslední příchozí paket.
- Pote co odešlu vše a přijde mi potvrzeni všech mnou odeslaných paketů, pošlu paket typu 7 a ukončím spojení.
- Přijde-li paket typu 7, ukončím spojení.
- Vždy po přijetí 500-ti (mohu i častěji pokud zrovna neodesílám žádná data) paketu odešlu paket typu 2, kde X je číslo posledního úspěšně přijatého paketu.
- Pokud nesouhlasí u přijatého paketu MAC, nahlásím chybu a ukončím spojení.
- Vždy po odeslání Z paketů (nepočítají se opakovaně zasílané) vezmu novou sadu klíčů, ověřím jejich různost, pošlu ji zašifrovanou RSA veřejným klíčem druhé strany za paketem typu 5, přejdu ve směru ode mne na nové klíče.
- Ukončuji-li spojení, přestanu odesílat data a pošlu paket typu 6, čekám na potvrzující paket typu 2 posledního odeslaného datového paketu a na paket typu 6. Až přijdou pošlu potvrzovací paket typu 2 a odešlu paket 7.
- Spojení mohu ukončit po uplynutí času T od odeslání paketu 6, pokud nebylo ukončeno dříve.

Z - po přechodu na nové klíče z generátoru vezmu 8 bitu a interpretuji je jako R, Z=R*5

T - nastaví uživatel, deafault 0 tedy čekám libovolně dlouho

Adresář kontaktů

Obsahuje veřejné klíče kontaktů, v adresáři je i soukromý klíč uživatele. Adresář je umístěn na disku zašifrován spolu se svým kódem MAC, pomoci 3DES-EDE-CBC.

Uživatel si zvolí alespoň 8 znakové heslo (program nepřijme triviální hesla, bude chtít zastoupení speciálního znaku, velká a mala písmena).

Z druhého až posledního znaku hesla je vytvořen jeden sha-1 otisk.

Z prvního až předposledního znaku je vytvořen druhý sha-1 otisk.

Za první otisk je přiřazeno prvních 72 bitu z druhého otisku, tak dostanu posloupnost 232 bitu.

Tato posloupnost je K1K2K3IV pro 3DES a MAC.

Při otevírání adresáře je po rozšifrování zkontrolován kód MAC.

Pokud MAC nesouhlasí, je uživateli oznámeno, že zadal nesprávné heslo a je požádán o správné.

NET

Mód DIRECT

Pokud není zapnut mód „NET" běží program v módu „DIRECT", všechny „síťové" funkce jsou vypnuty a program funguje pouze pro přímé spojení mezi dvěma uzly.

Mód „NET“ (síťový mód)

Program vytvoří dvě hlavni vlákna, DIRECT pro realizace vlastního přenosu dat a NET vlákno pro podporu "sítového" provozu ostatních uzlu.

Net komunikační pakety

Komunikační pakety jsou vždy 8 bytové.

Paket M1 -
FOUNDING(1B),IP_cile(4B),hloubka_dotazu(1B),port_pro_odpoved(2B)
Paket M2 -
RECEIVING(1B),[FOUND|NOWAY](1B),hloubka_dotazu(1B),port_pro_odpoved(2B)
Paket M3 -
RECEIVING(1B),[WANTUSE|END](1B)
FOUNDING = 0
RECEIVING = 1
FOUND = 1
NOWAY = 0
WANTUSE = 1
END = 0

Spojení

Spojením dále rozumím vždy tcp spojeni, tam kde není explicitně uvedeno spojení, ale z logiky věci vyplývá, rozuměj, naváže tcp spojeni a přijme, či odešle zprávu.

Vlákno NET

Naslouchá na portu BASE_LISTEN_PORT (čeká na zadosti o spojeni M1)
Přijme každé příchozí spojení, pokud je první přijatý paket nového spojeni M1, vytvoří nové pracovní vlákno, kterému předá přijatou M1.
Uzavře spojení a dále naslouchá na portu BASE_LISTEN_PORT.
- Vlákno běží s nižší prioritou než hlavní vlákno.
- V procesu běží pouze jedno vlákno NET.

Pracovní vlákno

- Pracovní vlákno je vytvořeno za účelem obsloužení jedné žádosti o spojení.
- Vlákno běží s nižší prioritou než vlákno NET.
- Počet pracovních vláken je shora omezen.

- Při vytvoření dostane zprávu M1, kterou má obsloužit, port pro odpověď ze zprávy je P1 a adresa odesilatele A1, hloubka H1.
- Pracovní vlákno projde adresář, a pokud v něm je IP_cile, pokusí se připojit na jeho BASE_PORT.
- Je-li úspěšné, pošle na A1:P1 zprávu s hodnotou druhého bytu "FOUND" a hloubkou H1+1 a portem P3.
- Přijde-li od A1 do shora omezeného času zprava M3 s hodnotou druhého bytu "WANTUSE", začne "tunelovat" (nechť BASE_PORT = P4 a cil je A2), jinak skončí.

- Pokud je hloubka H1>= MAX_DEPTH-2, odešle na A1:P1 zprávu M2 s hodnotou druhého bytu "NOWAY" a skončí.
- Pokud cíl v adresáři není, nebo se s ním spojit nelze, rozešle všem kontaktům
- z adresáře dotaz M1 na port BASE_LISTEN_PORT s inkrementovanou hloubkou a portem pro odpověď P2.
- Poté na P2 naslouchá a přijímá příchozí spojení, od kterých přijme zprávu M2.
- Ze všech odpovědí, které mají hodnotu druhého bytu zprávy M2 "FOUND", vybere dle hloubky a času mezi odesláním M1 a přijetím M2 nejlepší cestu k cíli, nechť tato má hloubku H2 a následující uzel je A2 a port pro odpověď je P4. Odpovědi přišlé po nastaveném čase už nepřijímá.
- Pokud nepřijal žádnou zprávu s hodnotou druhého bytu "FOUND", odešle na A1:P1 zprávu M2 s hodnotou druhého bytu "NOWAY" a skončí.
- Jinak odešle na A1:P1 zprávu s hodnotou druhého bytu "FOUND" a hloubkou H2 a portem P3.
- Naslouchá na portu P3 po shora omezený čas.
- Přijme-li zprávu M3 s hodnotou druhého bytu "END", nebo vyprší-li čas, odešle na A1:P4 M3 s hodnotou druhého bytu "END" a skončí.
- Přijme-li zprávu M3 s hodnotou druhého bytu "WANTUSE", odešle na A2:P4 M3 s hodnotou druhého bytu "WANTUSE" a začne "tunelovat".

Tunelování

Všechna data přijatá od spojeni na portu P3 přepošle na A2:P4 a všechna data přijatá od A1:P4 přepošle na spojení na portu P3.
Pokud ani z jedné strany spojení nepřijdou data po shora omezený čas, vlákno skončí.

9.2 Příloha 2 – Schéma komunikace při vyhledávání spojení