VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

VÝKONOVÁ A BEZPEČNOSTNÍ ANALÝZA TECHNOLOGIE WIREGUARD

PERFORMANCE AND SECURITY ANALYSIS OF WIREGUARD TECHNOLOGY

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — Oliver Varga

AUTHOR

VEDOUCÍ PRÁCE — Ing. Ondřej Krajsa, Ph.D.

SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Oliver Varga ***ID:*** 223335

Ročník: 3 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

Výkonová a bezpečnostní analýza technologie Wireguard

POKYNY PRO VYPRACOVÁNÍ:

V práci proveďte podrobnou analýzu technologie Wireguard, popište jednotlivé použité protokoly a šifrovací algoritmy. Proveďte porovnání s technologiemi IPSec a OpenVPN. Porovnání zaměřte na bezpečnost, dosažitelné přenosové rychlosti, zpoždění a jitter při přenosu.

DOPORUČENÁ LITERATURA:

[1] WireGuard: Next Generation Kernel Network Tunnel [online]. 2020 [cit. 2022-11-18]. Dostupné z: https://www.wireguard.com/papers/wireguard.pdf

[2] Standards for Efficient Cryptography SEC 1. [online]. 2009 [cit. 2022-12-10]. Dostupné z: http://www.secg.org/sec1-v2.pdf

Termín zadání: 6.2.2023 ***Termín odevzdání:*** 26.5.2023

Vedoucí práce: Ing. Ondřej Krajsa, Ph.D.

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

Abstrakt

Táto práca sa zaoberá popisom technológie virtuálnych privátnych sietí WireGuard, jej porovnaním s technológiami IPsec a OpenVPN na základe nimi používaných šifrovacích algoritmov zabezpečujúcich bezpečnosť a taktiež porovnáva ich technické parametre ako sú oneskorenie, jitter a prenosové rýchlosti. Práca zahŕňa popis technológie WireGuard, porovnanie spomínaných technológií a nakoniec výsledky meraní technických parametrov. Popísaná je taktiež nami zvolená metodika testovania parametrov, použitý hardvér k získaniu výsledkov a topológia siete. Nakoniec sú získané výsledky z testovaní porovnané medzi sebou.

Klíčová slova

WireGuard, IPsec, OpenVPN, site-to-site, iperf3, Mikrotik, jitter, oneskorenie, prenosová rýchlosť.

Abstract

This thesis deals with the description of WireGuard virtual private networks technology, its comparison with the technologies IPsec and OpenVPN based on the encrpytion algorithms they use to ensure security and also compares their technical parameters such as latency, jitter and transfer speeds. The work includes a description of the WireGuard technology, a comparison of the mentioned technologies and finally the results of the measurements of their technical parameters. The methodology chosen by us for testing the parameters, the hardware used to obtain the results and the network topology are also described. Finally, the results obtained from the tests are compared.

Keywords

WireGuard, IPsec, OpenVPN, site-to-site, iperf3, Mikrotik, jitter, latency, transfer speed.

Bibliografická citace

VARGA, Oliver. *Výkonová a bezpečnostní analýza technologie Wireguard* [online]. Brno, 2023 [cit. 2023-05-23]. Dostupné z: https://www.vut.cz/studenti/zav-prace/detail/151235. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce Ondřej Krajsa.

Prohlášení autora o původnosti díla

Jméno a příjmení studenta:	*Oliver Varga*
VUT ID studenta:	*223335*
Typ práce:	*Bakalářská práce*
Akademický rok:	*2022/23*
Téma závěrečné práce: *technologie Wireguard*	*Výkonová a bezpečnostní analýza*

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce 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 uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce 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 části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

.

V Brně dne: 20. května 2023 ----------

podpis autora

Poděkování

Ďakujem vedúcemu bakalárskej práce Ing. Ondřejovi Krajsovi, Ph. D. za účinnú metodickú, pedagogickú a odbornú pomoc a ďalšie cenné rady pri spracovaní mojej bakalárskej práce.

V Brně dne: 20. května 2023 ----------

podpis autora

Obsah

1. TECHNOLÓGIA WIREGUARD 11

1.1 SMEROVANIE A SMEROVACIA TABUĽKA 12
1.2 ROAMING DIZAJN A KONCOVÉ BODY 13
1.3 TOKY ODOSIELANIA A PRIJÍMANIA 15
1.4 KRYPTOGRAFIA A POUŽITÉ PROTOKOLY 16
1.4.1 Voliteľný predom zdieľaný kľúč 17
1.4.2 Princíp cookies 18
1.5 PRINCÍP SPRÁV 20
1.5.1 Správa iniciátor – príjemca 21
1.5.2 Správa príjemca naspäť iniciátorovi 22
1.5.3 Cookie funkcie MAC 23
1.5.4 Cookie reply správa 23
1.5.5 Transportné dáta – odvodenie kľúča a správy o transportných údajoch 24

2. POROVNANIE WIREGUARD S IPSEC A OPENVPN 26

3. TESTOVANIE 28

3.1 METODIKA TESTOVANIA 28
3.2 TOPOLÓGIA SIETE – SITE-TO-SITE 29
3.3 POUŽITÉ KOMPONENTY 29

4. VÝSLEDKY TESTOVANÍ TECHNICKÝCH PARAMETROV VPN 31

4.1 TESTOVANIE BEZ POUŽITIA VIRTUÁLNEJ PRIVÁTNEJ SIETE – STATICKÉ SMEROVANIE MEDZI LOKÁLNYMI SIEŤAMI (RB-493G) 31
4.2 TESTOVANIE S POUŽITÍM WIREGUARD VIRTUÁLNEJ PRIVÁTNEJ SIETE (RB-493G) 33
4.3 TESTOVANIE S POUŽITÍM IPSEC VIRTUÁLNEJ PRIVÁTNEJ SIETE (RB-493G) 35
4.4 TESTOVANIE S POUŽITÍM OPENVPN VIRTUÁLNEJ PRIVÁTNEJ SIETE (RB-493G) 37
4.5 TESTOVANIE BEZ POUŽITIA VIRTUÁLNEJ PRIVÁTNEJ SIETE – STATICKÉ SMEROVANIE MEDZI LOKÁLNYMI SIEŤAMI (RB941-2ND) 39
4.6 TESTOVANIE S POUŽITÍM WIREGUARD VIRTUÁLNEJ PRIVÁTNEJ SIETE (RB941-2ND) 41
4.7 TESTOVANIE S POUŽITÍM IPSEC VIRTUÁLNEJ PRIVÁTNEJ SIETE (RB941-2ND) 42
4.8 TESTOVANIE S POUŽITÍM OPENVPN VIRTUÁLNEJ PRIVÁTNEJ SIETE (RB941-2ND) 45

5. POROVNANIE NAMERANÝCH HODNÔT 47

6. ZÁVER 48

SEZNAM OBRÁZKŮ

4.1 Zaťaženie procesora Mikrotik RB-493G, statické smerovanie 31
4.2 Iperf3 ukážka výstupu testu, statické smerovanie 32
4.3 Zaťaženie procesora Mikrotik RB-493G, WireGuard 33
4.4 Iperf3 výstup, WireGuard 34
4.5 Zaťaženie procesora Mikrotik RB-493G, IPsec 35
4.6 Iperf3 výstup, IPsec 36
4.7 Bottleneck procesora Mikrotik, IPsec odosielajúci 36
4.8 Zaťaženie procesora Mikrotik RB-493G, OpenVPN 37
4.9 Iperf3 výstup, OpenVPN 38
4.10 Zaťaženie procesora Mikrotik RB941-2nD, statické smerovanie 39
4.11 Iperf3 výstup, statické smerovanie RB941-2nD 40
4.12 Zaťaženie procesora Mikrotik RB941-2nD, WireGuard 41
4.13 Iperf3 výstup, WireGuard, RB941-2nD 42
4.14 Zaťaženie procesora Mikrotik RB941-2nD, IPsec 43
4.15 Iperf3 výstup, IPsec, RB941-2nD 44
4.16 Zaťaženie procesora Mikrotik RB941-2nD, OpenVPN 45
4.17 Iperf3 výstup, OpenVPN, RB941-2nD 46

SEZNAM TABULEK

1.1 Konfigurácia 1a smerovacej tabuľky 12
1.2 Konfigurácia 2a smerovacej tabuľky 13
1.3 Konfigurácia 2b smerovacej tabuľky 14
1.4 Konfigurácia 1b smerovacej tabuľky 14
1.5 Správa iniciátora príjemcovi 21
1.6 Správa od príjemcu naspäť iniciátorovi 22
1.7 Cookie Reply Správa 24
5.1 Tabuľka priemerných hodnôt – RB-493G 47
5.2 Tabuľka priemerných hodnôt – RB941-2nD 47

Úvod

V dnešnej dobe, kedy sa prakticky väčšina procesov spojených s našími každodennými požiadavkami a povinnosťami odhráva online, stúpajú nároky na samotnú bezpečnosť týkajúcu sa či už osobných, ale aj finančných informácií. Jednou z možností, ako tieto informácie chrániť ponúkajú takzvané virtuálne privátne siete, ktoré vytvárajú zabezpečené spojenie na menej zabezpečených sieťach. Medzi služby, ktoré VPN poskytujú, patrí napríklad skrytie reálnej IP adresy a lokácie. Využívané sú častejšie veľkými spoločnosťami, ktoré nimi zabezpečujú svojim zamestnancom bezpečený prístup k podnikovým aplikáciám počas práce z domu, ale aj mnoho iných výhod. Jednou z týchto virtuálnych sietí je aj technológia WireGuard, ktorou sa táto práca zaoberá.

Cieľom práce bolo vykonať analýzu technológie WireGuard, popísať jej jednotlivé algoritmy a použité šifrovacie algoritmy. Ďalej bolo potrebné porovnať ju s technológiami IPsec a OpenVPN z hľadiska bezpečnosti. Poslednou požiadavkou bolo pomocou testovacích scenárov a vhodným testovacím nástrojom odmerať technické parametre spomínaných VPN technológií, ako sú jitter, prenosová rýchlosť či oneskorenie.

V prvej kapitole a jej podkapitolách práca popisuje technológiu WireGuard a jej fungovanie. V ďalšej časti, teda druhej kapitole, je vykonané porovnanie zadaných technológií. V ďalšej časti práce je popísaná metodika merania jednotlivých parametrov, topológia siete a taktiež použitý hardvér a testovacie nástroje V predposlednej časti, teda v štvrtej kapitole, sú nami namerané údaje týkajúce sa technických parametrov odprezentované a v poslednej časti sa práca venuje vzájomnému porovnaniu získaných výsledkov testov.

1. TECHNOLÓGIA WIREGUARD

WireGuard je bezpečný VPN tunel, ktorý funguje na tretej vrstve referenčného modelu ISO/OSI. Spočiatku implementovaný ako jadrové virtuálne sieťové rozhranie pre Linux, požadovaným cieľom jeho vzniku bolo nahradenie technológií IPsec a OpenVPN vo väčšine prípadov použitia, pričom by, pri zachovaní optimálneho výkonu, stále zostal rýchlejší, bezpečnejší a jednoduchší na používanie ako konkurencia.

Rozhranie vychádza z fundamentálnych princípov zabezpečených tunelov, a teda samotnou podstatou je asociácia medzi IP adresou zdroja tunela a verejným kľúčom rovesníka (peer public key). Výmena kľúčov a ustanovenie zašifrovaných kanálov medzi klientami sú zabezpečené NoiseIK protokolom. Všetka tvorba relácií je spracovávaná transparentne pre používateľa. Vzájomná autentizácia prebieha v štýle podobnom OpenSSH, teda používajú sa krátke vopred zdieľané statické kľúče, ktoré sú vo svojej podstate bodmi na eliptickej krivke Curve25519 [2].

Protokol zabezpečuje mimo iné aj vysokú úroveň skrývania identity a vyznačuje sa taktiež vysokou mierou doprednej bezpečnosti (perfect forward secrecy). Používaním kryptografického algoritmu ChaCha20Poly1305 [8] nedochádza k významnému obmedzeniu prenosových rýchlostí. Tento algoritmus zohráva úlohu pri zapúzdrovaní paketov v UDP pomocou overeného šifrovania. Docielené je taktiež aj zmiernenie útokov odmietnutia služby (denial of service), a to za pomoci vylepšeného prístupu k súborom cookie viazaných na IP. Výrazne zlepšenie je patrné aj pri mechanizmoch súborov cookie IKEv2 [3] a DTLS [4], ktoré majú za úlohu šifrovanie a autentifikáciu. Výsledný dizajn umožňuje zaobísť sa bez pridelovania zdrojov v reakcii na prijaté pakety. Jednoduchosť protokolu WireGuard nakoniec spočíva v samotnej implementácii, kedy pre operačné systémy Linux stačí menej ako 4 000 riadkov kódu, čo v konečnom dôsledku znamená značné zjednodušenie potrebného manažmentu alebo prípadnej údržby.

WireGuard sa vyhýba zložitému vrstveniu ako je tomu napríklad pri protokole IPsec, kde je využívaná takzvaná transformačná vrstva xfrm. Pri jej používaní používateľ vyplní jadrovú štruktúru tak, že zadá akú šifrovú súpravu alebo aké kľúče, poprípade môže tiež špecifikovať možnosti transformácií, ako je napríklad kompresia, bude používať ktorý selektor paketov prechádzajúcich preddefinovaným operačným prostredím. Vo všeobecnosti je potom za aktualizáciu týchto dátových štruktúr zodpovedný používateľský priestorový daemon, a to na základe výsledkov výmeny kľúčov, kedy vo väčšine prípadov je toto docielené a manažované protokolom IKEv2 [3], čo je sám o sebe zložitý protokol, a preto je náročnosť a obsiahlosť samotného kódu značná. Ikeď toto spomínané vrstvenie IPsecu môže byť zo sémantického hľadiska v poriadku, ako aj riešenie oddelenia transformačnej vrstvy od vrstvy rozhrania môže byť múdre z pohľadu sieťovania, spôsobuje problémy vychádzajúce z jeho náročnosti a zvyšuje aj nároky na správnu implementáciu a nasadenie. Namiesto toho WireGuard ponúka riešenie vo virtuálnom rozhraní, ktoré pri konfigurácii môže byť pomenované

ľubovoľne, napríklad wireguard1. Toto rozhranie môže byť spravované pomocou nástrojov ako `ip` a `ifconfig`. Konfigurácia rozhrania pre správne fungovanie tunela nie je zložitá a spočíva v nastavení privátneho kľúča a rozličných verejných kľúčov rovesníkov, s ktorými bude prebiehať zabezpečená komunikácia. Voliteľne je možné konfigurovať aj predzdieľaný symetrický kľúč. Konkrétne operácie, ako napríklad výmena kľúčov, spojenie, prerušenie spojenia, obnovenie spojenia, zisťovanie možných spojení a podobne, sú uskutočňované mimo zraku používateľa (správcu), pričom zostávajú transparentné a spoľahlivé, a vďaka tomu z pohľadu spravovania WireGuard rozhranie pôsobí bezstavovo. Zaručené je, že pakety prichádzajúce z rozhrania WireGuard budú zašifrované a overené aj pri bežnom nastavení pravidiel brány firewall. V konečnom dôsledku vďaka jednoduchosti a priamočiarosti protokolu sú šance na zlyhanie alebo nesprávnu konfiguráciu oveľa menšie ako pri IPsec.

Nakoniec, WireGuard ako VPN technológia používa špecifické kryptografické nástroje a protokoly. Je nadizajnovaný tak, aby bol odolný voči zraniteľnostiam a útokom pomocou fixnej sady kryptografických primitív. Jedná sa o Noise [5] protokol zabezpečujúci výmenu kľúčov, Curve25519 [2] pre ECDH, HKDF [6] pre rozširovanie kľúčov, ChaCha20 [7] verzia podľa RFC7539 [8] a Poly1305 [8] pre overené šifrovanie a nakoniec protokol BLAKE2s [9] na používaný pri procese hashovania.

1.1 Smerovanie a smerovacia tabuľka

Základným princípom každej bezpečnej VPN je asociácia medzi rovesníkmi a ich pridelenými IP adresami použitými ako zdrojové IP adresy. V technológii WireGuard sa každý rovesník vyznačuje verejným kľúčom dlhým 32 bajtov. To znamená, že vo WireGuard existuje jednoduché mapovanie asociácií medzi verejnými kľúčmi a množinou autorizovaných IP adries [1].

Tabuľka 1.1 Konfigurácia 1a smerovacej tabuľky

Verejné kľúče rozhraní	***Súkromné kľúče rozhrania***	***Načúvajúci UDP Port***
dec1...QnbV	aiKm…sR9j	*41414*
Verejné kľúče rovesníkov	***Povolené zdrojové adresy***	
jan0…4xEf	*10.192.122.7/32, 10.192.124.0/24*	
XBnR…1qln	*10.192.122.8/32*	
N7op…iJpO	*10.10.10.198/32*	

Samotné rozhranie má svoj vlastný verejný kľúč a priradený UDP port, na ktorom načúva. Nasleduje list rovesníkov, kde každý rovesník je charakterizovaný vlastným verejným kľúčom a zoznamom autorizovaných zdrojových IP adries.

Pri prenose odchádzajúceho paketu cez rozhranie WireGuard, wireguard1, Tabuľka 1.1 je použitá ako zdroj informácií pre určenie, ktorý verejný kľúč bude použitý

na šifrovanie. Napríklad, paket určený cieľovej IP adrese 10.192.122.8 bude zašifrovaný pomocou bezpečnej relácie odvodenej od verejného kľúča príslušného rovesníka, teda *XBnR...1qln*. Ak rozhranie wireguard1 naopak zašifrovaný paket prijíma, po dešifrovaní a verifikovaní bude prijatý iba v prípade, že sa jeho zdrojová IP adresa zhoduje s adresou pridelenou verejnému kľúču použitému v zabezpečenej relácii na jeho dešifrovanie. Napríklad, ak je paket dešifrovaný z *jan0...4xEf*, nebude zahodený len v prípade, ak má zdrojovú IP adresu zodpovedajúcu 10.192.122.7 alebo bližšie špecifikovanému rozsahu, napríklad od 10.192.124.0 do 10.192.124.255. [1]

Tento jednoduchý prístup autorov WireGuardu k smerovaniu umožňuje spoľahnúť sa aj na základné nastavenia firewallu. Keďže WireGuard je tunel striktne založený na fungovaní iba na tretej vrstve, vo všeobecnosti, každý paket prechádzajúci jeho rozhraním má zaručenú nielen autentickosť zdrojovej IP adresy, ale aj garantovanú doprednú bezpečnosť prenosu. Takéto fungovanie, teda identifikácia rovesníkov už na tretej vrstve, umožňuje oproti iným bežným zabezpečeným tunelom, ako napríklad IPsec, omnoho prehľadnejší a jednoduchší sieťový návrh.

V situácii, kedy sa WireGuard rovesník rozhodne smerovať všetku komunikáciu cez iného WireGuard rovesníka, napríklad prípad, kedy používateľ oprávňuje *dec1...QnbV* poslať pakety rozhraním wireguard1 s ľubovoľnou IP adresou zdroja, kedy všetky odchádzajúce pakety z wireguard1 budú zašifrované zabezpečenou reláciou vychádzajúcou z tohto verejného kľúča a ďalej odoslané na jeho koncový bod(endpoint), by mohla tabuľka smerovania vyzerať napríklad takto [1]:

Tabuľka 1.2 Konfigurácia 2a smerovacej tabuľky

Verejné kľúče rozhraní	***Súkromné kľúče rozhrania***	***Načúvajúci UDP Port***
kv7Q...jLMx	*15uT...CnhE*	*21841*
Verejné kľúče rovesníkov	***Povolené zdrojové adresy***	
dec1...QnbV	*0.0.0.0/0*	

1.2 Roaming dizajn a koncové body

Pre WireGuard rovesníkov je dôležité, aby bolo navzájom medzi nimi možné uskutočnovať výmenu šifrovaných paketov UDP na špecifické koncové body internetu. Každý z rovesníkov vyskytujúcich sa v smerovacej tabuľke môže vopred voliteľne upresniť vonkajšiu IP adresu a UDP port svojho koncového bodu. Toto upresnenie adresy a portu je voliteľné, pretože aj v prípade, kedy nie sú špecifikované stačí, aby WireGuard obdržal správne overený rovesníkov paket a na určenie koncového bodu použije okrajovú vonkajšiu IP adresu zdroja.

Verejné kľúče slúžia ako identifikátory konkrétnych rovesníkov. Na druhú stranu, okrajová vonkajšia IP adresa zdroja zašifrovaného paketu umožňuje rozpoznať vzdialený koncový bod rovesníka. Toto dáva rovesníkom možnosť voľného pohybu medzi rozličnými vonkajšími IP adresami, napríklad ako je to možné medzi mobilnými sieťami pomocou implementácie SSH známej ako Mosh [10]. Môžu teda svojvoľne meniť svoju IP adresu bez toho, aby stratili môžnosť komunikovať s ostatnými v sieti. Toto je možné, pretože v konfigurácii je uvedená len inicializačná adresa, pomocou ktorej príde k prvotnému odoslaniu dát. Po prijatí prvej odpovede sa WireGuard zaoberá už len IP adresami, z ktorých mu prichádzajú len najnovšie overené dáta. Zmenu adresy si ostatní rovesníci všimnú a podľa nej potom riadia následné smerovanie. Po uplatnení týchto rozšírení by teda uvedená tabuľka smerovania mohla vyzerať nasledovne [1]:

Tabuľka 1.3 Konfigurácia 2b smerovacej tabuľky

Verejné kľúče rozhraní	***Súkromné kľúče rozhrania***	***Načúvajúci UDP Port***
kv7Q…jLMx	*15uT…CnhE*	*21841*
Verejné kľúče rovesníkov	***Povolené zdrojové adresy***	***Koncový bod***
dec1...QnbV	*0.0.0.0/0*	*192.158.1.38:41414*

V prípade, kedy *dec1...QnbV* na adrese 192.158.1.38:41414 obdrží zašifrovaný paket od hostiteľa *kv7Q...jLMx*, aktualizuje sa jeho tabuľka s novou informáciou o koncovom bode, na ktorý bude potom odosielať pakety s odpoveďami. Tento koncový bod môže mať adresu napríklad 192.95.5.64:21841 [1]:

Tabuľka 1.4 Konfigurácia 1b smerovacej tabuľky

Verejné kľúče rozhraní	***Súkromné kľúče rozhrania***	**Načúvajúci UDP Port**
dec1...QnbV	*aiKm…sR9j*	*41414*
Verejné kľúče rovesníkov	***Povolené zdrojové adresy***	***Koncový bod***
jan0…4xEf	*10.192.122.7/32, 10.192.124.0/24*	
XBnR…1qln	*10.192.122.8/32*	
N7op…iJpO	*10.10.10.198/32*	*192.158.1.33:21841*

Je potrebné uvedomiť si, že port, na ktorom načúvajú rovesníci, je vždy rovnaký ako port zdroja odosielaných paketov. Tým je zaručená nielen výrazná jednoduchosť, ale

aj dôveryhodné prechádzanie cez NAT. Vďaka tejto roamingovej vlastnosti majú rovesníci vždy najaktuálnejšie informácie o vonkajších IP adresách a UDP portoch.

Takéto fungovanie zabezpečuje nielen minimálnu potrebnú konfiguráciu, ale aj značné pohodlie. Zatiaľ čo z pohľadu man-in-the-middle útočníka je možné pozmeniť neoverené vonkajšie zdrojové IP adresy, nebude takýto útočník schopný dešifrovať alebo úmyselne upraviť payload. Samozrejme, útok odmietnutia služby by mohol byť pre takéhoto útočníka jednoduchý, stačilo by, aby zahadzoval pôvodné pakety, avšak hostitelia, ktorí nie sú schopní dešifrovať a následne potom odpovedať na pakety sú rýchlo zabudnutí.

1.3 Toky odosielania a prijímania

Keď spojíme roamingový dizajn a dizajn smerovania WireGuardu, ktoré sú vysvetlené v predchádzajúcich podkapitolách 1.1 a 1.2, dostaneme sa k samotnému fungovaniu prijímania a odosielania paketov prostredníctvom wireguard1 rozhrania použitím tabuliek z konfigurácie 1a.
Prípad, kedy sa lokálne vygenerovaný (prípadne preposlaný) paket bude prenášať cez odchádzajúce rozhranie wireguard1 [1]:

1. Najprv na rozhranie wireguard1 dorazí plaintextový paket.
2. Je skontrolovaná cieľová adresa tohto paketu, teda 192.168.87.21. Táto adresá zodpovedá rovesníkovi TrMv...WXX0 v tabuľke. V prípade, že adresa nezodpovedá žiadnemu z rovesníkov, paket je zahodený a odosielateľ je o tom informovaný ICMP odpoveďou „no route to host“ a zároveň je vrátená aj hodnota `-ENOKEY`.
3. Paket je zašifrovaný pomocou ChaCha20Poly1305 [8]. Pri šifrovaní je použitý symetrický šifrovací kľúč spolu s nonce počítadlom zabezpečenej relácie, ktoré sú asociované s identifikovaným rovesníkom.
4. K paketu sa po zašifrovaní pridá hlavička obsahujúca rozličné polia.
5. Nakoniec sú paket s priradenou hlavičkou odoslané vo forme UDP paketu do internetového UDP/IP koncového bodu asociovaného s rovesníkom. Ak nie je rozpoznaný koncový bod, paket je zahodený a informuje o tom ICMP paket odpoveďou a vrátená je tiež aj hodnota `-EHOSTUNREACH`.

Paket dorazí na UDP port 41414, čo je port, na ktorom načúva rozhranie wireguard1 [1]:

1. Na vyhovujúcom porte je prijatý paket spolu s pridelenou hlavičkou a zašifrovaným payloadom.
2. Na základe hlavičky WireGuard rozpozná asociáciu s rovesníkovou reláciou a skontroluje správnosť počítadla správ. Následne sa ju pokúsi overiť a dešifrovať pomocou symetrického kľúča relácie. Nerozpoznanie rovesníka alebo neúspešné overenie končia zahodením paketu.

3. Paket je overený, zdrojová IP adresa vonkajšieho UDP/IP paketu aktualizuje koncový bod konkrétneho rovesníka.
4. Po dešifrovaní payloadu má rozhranie k dispozícii plaintextový paket. Ak sa nejedná o IP paket, je zahodený. Kontroluje sa či IP adresa zdroja vnútorného paketu korešponduje s údajmi uvedenými v smerovacej tabuľke. Ak je teda zdrojová IP adresa dešifrovaného paketu napríklad 192.168.31.28, paket zodpovedá smerovaniu rovesníka. V prípade, že smerovaniu nezodpovedá, je zahodený.
5. Ak v celom procese nebol paket zahodený, dostáva sa ďalej do fronty prijímania na rozhraní wireguard1.

1.4 Kryptografia a použité protokoly

Aby bolo možné zahájiť posielanie paketov, je potrebné, aby pred tým došlo k výmene kľúčov. Táto výmena je realizovaná TLS 1.3 1-RTT (one round trip time) handshake protokolom, kedy iniciátor komunikácie posiela správu odpovedajúcemu a ten naspäť odosiela správu iniciátorovi. Po uskutočnení tejto výmeny môže iniciátor odosielať príjemcovi zašifrované správy pomocou zdieľaného páru symetrických kľúčov. Jeden z tejto dvojice kľúčov slúži k odosielaniu, druhý k prijímaniu správ. Po prijatí prvej zašifrovanej správy môže príjemca ďalej odosielať vlastné zašifrované správy iniciátorovi. Bližšie fungovanie tohto procesu výmeny je popísané pre KEA+C protokol [11]. Je dôležité poznamenať, že táto správa o podaní rúk je uskutočňovaná asychrónne s prenosom dátových správ. Aby sa docielilo zmiernenie útokov na odmietnutie služby, tieto správy sú riadané podľa „IK“ predlohy Noise protokolu [5] spolu s novou cookie konštrukciou. Výsledkom takto zostaveného protokolu je nakoniec robustný systém, ktorý je zabezpečený a naviac spĺňa požiadavky zabezpečenia autentifikovanej výmeny kľúčov (AKE) podľa [11]. Ďalšími kľúčovými vlastnosťami sú vysoká miera doprednej bezpečnosti, utajovanie statických verejných kľúčov, odolnosť proti útokom odmietnutia služby a v neposlednom rade je prítomná schopnosť vyhnúť sa odcudzeniu identity na základe zisteného kľúča.

Wireguard je navrhnutý s cieľom vyhnúť sa ukladaniu akéhokoľvek stavu pred samotnou autentifikáciou a neodpovedať na neoverené pakety. Takýmto návrhom je zabezpečené to, že WireGuard zostáva pre neautentifikovaných rovesníkov a rôzne sieťové skenery neviditeľný, vďaka čomu je možné vyhnúť sa hneď niekoľkým typom útokov. WireGuard je na druhú stranu možné implementovať aj bez použitia dynamickej alokácie pamäte, a to ani pre overené pakety. Tento predstavený dizajn avšak vyžaduje, aby bola vždy prvá správa doručená odpovedajúcemu účastníkovi použitá na overenie iniciátora komunikácie, s čím prichádza možnosť zneužitia útočníkom, kedy je mu umožnené prevedenie útoku prehraním (replay attack), kde môže zopakovať prvotnú správu o podaní rúk. Takto oklame obeť, ktorá nevedome obnoví svoj efemérny

(ephemeral) kľúč, čím príde k zneplatneniu relácie iniciátora. Správa avšak aj napriek takémuto útoku zostane utajená a nenaruší sa ani jej autentickosť. Aby sa vyhlo útokom prehraním, je vo WireGuard použitá 12-bytová časová značka TAI64N [12], ktorá sa pridáva do prvej správy, kde je následne zašifrovaná a autentifikovaná. Odpovedajúci následne uchováva informáciu iba o najväčšej časovej značke prijatej od rovesníka a ďalej zahadzuje pakety s rovnakou alebo menšou hodnotou. Pri možnom narušení tohto stavu nedochádza k žiadnemu problému ani možnosti rozvrátiť aktuálne prebiehajúcu reláciu medzi iniciátorom a odpovedajúcim rovesníkom pomocou útoku prehraním, pretože odpovedajúci práve znovu začal a aktuálne nemá žiadnu zabezpečenú reláciu k narušeniu. Akonáhle iniciátor znovu naviaže zabezpečenú reláciu s odpovedajúcim rovesníkom, bude použitá časová značka s vyššou hodnotou, čím sa znehodnotí značka predošlá. Výhodou tejto časovej značky je jej samotná implementácia, keďže sa jedná o big-endian. Je možné teda porovnávať dve dvanásť bytové časové značky pomocou štandardnej funkcie programovacieho jazyka C, `memcmp()`. Aby zostala zachovaná miera popierateľnosti, prvotné správy vo WireGuard používajú výsledok Diffie-Hellmanovho výstupu zo statických kľúčov oboch rovesníkov pre autentifikáciu. Pri kompromitácii jedného zo statických kľúčov môže útočník síce zostaviť iniciačnú správu, nebude však schopný dokončiť proces podania rúk (handshake). Na druhej strane tým ale získa možnosť zabrániť všetkým budúcim spojeniam, pretože vytvorená iniciačná správa obsahuje maximálnu hodnotu časovej značky. Takýto scenár sa spočiatku môže podobať na zvyčajné zraniteľnosti vychádzajúce z odcudzenia identity na základe kompromisu medzi kľúčmi (KCI útok) – na tie nie je WireGuard zraniteľný – v skutočnosti ide avšak o niečo iné. V tomto prípade samotný útočník pomôže odhaliť a odvrátiť ním využitú zraniteľnosť, pretože kompromitovaný kľúč už nemôže byť ďalej používaný rovesníkmi. V prípade, že by presnosť TIA64N časovej značky predstavovala bezpečnostné riziko v podobe nežiadúceho úniku informácií, implementácia dovoľuje skrátiť 24 bitov z jej nanosekundovej časti.

1.4.1 Voliteľný predom zdieľaný kľúč

WireGuard zakladá na nutnosti výmeny statických verejných kľúčov rovesníkov pred zahájením akejkoľvek komunikácie. Dôvernosť všetkých prenášaných informácií ďalej závisí od zabezpečenia Curve25519 ECDH funkcie. Pri navrhovaní zabezpečenia autori nezabudli ani na kvantovú výpočtovú techniku a s cieľom zabrániť jej možným budúcim pokrokom, technológia WireGuard podporuje mód, v ktorom si ktorékoľvek páry rovesníkov môžu navzájom dodatočne vopred zdielať jeden 256-bitový symetrický šifrovací kľúč. Podstatou takéhoto vopred zdieľaného kľúča je pridať ďalšiu vrstvu symetrického šifrovania, čím sa zvýši úroveň ochrany. Myslená hrozba spočíva v scenári, kedy útočníci odchytávajú zašifrovanú komunikáciu s nádejou, že jedného dňa bude možné prelomiť Curve25519 a im tak bude umožnené dešifrovať predom nazbierané

dáta. Z hľadiska správy kľúčov sú v dnešnej dobe dopredu zdieľané symetrické šifrovacie kľúče problematické. Pri takomto zdieľaní kľúča môže prísť k jeho kompromitácii, avšak myšlienkou navrhnutého zabezpečenia je, že v čase, kedy budeme pomocou kvantových výpočtov schopní prelomiť Curve25519, bude takto predom zdieľaný symetrický kľúč dávno zabudnutý. Najdôležitejšie avšak je, že ak by aj nakoniec ku kompromitácii predzdieľaného kľúča došlo, samotné kľúče eliptickej krivky Curve25519 stále poskytujú dostatočnú ochranu. Nakoniec výsledkom takéhoto voliteľného spojenia predom zdieľaných symetrických kľúčov a kryptografickej eliptickej krivky je dosiahnutý prijateľný kompromis aj pre tých najviac podozrievavých používateľov.

1.4.2 Princíp cookies

Aby bolo možné pri zostavovaní komunikácie overiť autentickosť handshake správy, je potrebné, aby sa na krivke Curve25519 vynásobili jej body. Tento proces násobenia aj napriek tomu, že samotná krivka Curve25519 je na väčšine procesorov rýchla, pôsobí značne zaťažujúco pre procesor. Dochádza počas neho k vystaveniu protokolu potenciálnej hrozbe v podobe útoku odmietnutia služby. Aby sa bol teda odpovedajúci rovesník – príjemca správy – schopný vyhnúť takémuto útoku, kedy je pod určitým zaťažením spôsobeným procesom násobenia, má možnosť odmietnuť spracovanie handshake správy (iniciačnej alebo správy slúžiacej ako odpoveď) a namiesto toho odpovedať pomocou cookie reply správy obsahujúcej cookie. Iniciátor takto získané cookie potom môže použiť k opätovnému odoslaniu správy, kde požaduje jej následné prijatie príjemcom. Keď je teda, napríklad server, pod určitým zaťažením, nemôže odpovedať kým neobdrží platný paket. Preto je vyžadované, aby počas zaťaženia všetky správy obsahovali MAC funkciu, ktorá kombinuje verejný kľúč prijímajúceho a voliteľne aj predom zdieľaný kľúč ako MAC kľúč.

Spomínané cookie je v podstate výsledkom výpočtu MAC funkcie iniciátorovej zdrojovej IP adresy pomocou tajnej náhodnej hodnoty meniacej sa každé dve minúty, ktorú si uchováva odpovedajúci rovesník. Táto meniaca sa hodnota pri výpočte slúži ako MAC kľúč. V situácii, kedy iniciátor opätovne odosiela správu, posiela ju spolu s MAC funkciou jeho správy a použije toto cookie ako MAC kľúč. Príjemca správy sa následne (pri zaťažení) na základe platnosti, teda prípad, kedy ide o správnu MAC funkciu používajúcu cookie ako kľúč, tejto správy rozhodne či ju prijať a spracovať. Vďaka tomuto mechanizmu je zaručený aj dôkaz o vlastníctve IP, pretože odosielané správy sú spajané s iniciátorovou IP adresou, čo ďalej umožňuje obmedzovanie rýchlosti pomocou klasických algoritmov obmedzujúcich rýchlosť IP.

Takáto schéma avšak predstavuje tri hlavné problémy. Prvý z problémov sa spája s problematikou opísanou na začiatku kapitoly 1.4, a teda so samotným návrhom, kedy nedochádza k posielaniu odpovedí na neautentifikované správy. Odosielanie cookie reply správy by túto vlasnosť porušovalo. Ďalší problém predstavuje odosielanie cookie, ktoré

by v žiadnom prípade nemalo byť odosielané v podobe čistého textu, pretože by sa tým otvorila možnosť pre útok človekom medzi, ktorý by ho mohol použiť na odosielanie podvodných správ. Nakoniec tretím problémom je fakt, že samotný iniciátor by sa mohol ocitnúť pod útokom odmietnutia služby, kedy sú mu odosielané podvodné cookie, ktoré by mu ďalej znemožňovali správne vypočítať MAC funkciu jeho správy. Odpoveďou autorov WireGuardu na tieto problémy je použitie dvoch MAC funkcií (`msg.mac1` a `msg.mac2`). Výpočet týchto funkcií je vysvetlený neskôr v kapitole 1.5.3.

Aby v situácii popísanej ako prvý problém zostal odpovedajúci rovesník počas zaťaženia mlčať, je v každej správe prítomná `msg.mac1` funkcia používajúca jeho verejný kľúč. Takéto fungovanie prinajmenšom zabezpečí, že pre vyvolanie akejkoľvek odpovede, rovesník posielajúci správu musí poznať partnera, s ktorým komunikuje, čo je možné na základe poznania jeho verejného kľúča. Je vhodné poznamenať, že platná `msg.mac1` musí byť prítomná v každej situácii, nielen v situácii počas zaťaženia. Ikeď verejný kľúč rovesníka sám o sebe nie je tajný, je stále dostačujúci v rámci tohto modelu útoku, ktorého podstatou je zabezpečiť neviditeľnosť určitých procesov a služieb, a teda poznatok o tomto kľúči slúži ako dostatočný dôkaz o znalosti jeho existencie. Aj napriek tomu však prvá MAC funkcia `msg.mac1` dáva možnosť pasívnemu útočníkovi pokúšať sa o uhádnutie, pre ktorý verejný kľúč je daný paket počas komunikácie zamýšľaný, čím sú do určitej miery oslabené schopnosti skrývania identity. Neznamená to však, že správnym odhadom by prišlo ku kryptografickému dôkazu, pretože pri procese tvorenia MAC nie je použitý žiadny súkromný materiál.

Druhý problém, teda odosielanie MAC funkcií v podobe čistého textu, je riešený pomocou autentifikovaného šifrovania s pripojenými dátami (AEAD). Pri prenose sú AEAD spolu s rozšírenou náhodnou hodnotou nonce aplikované na súbor cookie, pričom ako symetrický šifrovací kľúč slúži verejný kľúč odpovedajúceho rovesníka.

AEAD je použité aj pri riešení posledného spomínaného problému, tentoraz sa však pri šifrovaní cookie aplikuje AEAD pole s dodatočnými údajmi, z anglického (additional data). Dôvodom je pridanie ďalšej formy autentifikácie prvej MAC funkcie iniciačnej správy, ktorá vyvolala cookie reply správu. Zmyslom takéhoto riešenia je znemožniť útočníkovi posielať prúdy neplatných cookie odpovedí iniciátorovi s cieľom zabrániť jeho autentifikácii pomocou platného cookie.

Po vyriešení všetkých spomínaných problémov prichádza na rad použitie druhej MAC funkcie (`msg.mac2`). Tá je pridaná za použitia preneseného cookie ako MAC kľúča. V čase, kedy je odpovedajúci rovesník pod vplyvom zaťaženia, bude prijímať iba správy navyše obsahujúce túto druhú MAC funkciu.

Teda po zhrnutí, rovesník v roli odpovedajúceho, po spočítaní MAC funkcií a ich následnom porovnaní s tými prijatými v správach, musí vždy odmietnuť správy obsahujúce neplatné prvé MAC funkcie (`msg.mac1`). Dodatočne, ak sa práve nachádza v procese, kedy je pod vplyvom zaťaženia, je možné, aby odmietol aj neplatné `msg.mac2`. Ak v tejto situácii (pri zaťažení) obdrží správu s platnou `msg.mac1`, ale neplatnou

`msg.mac2`, má možnosť odpovedať pomocou cookie reply správy. Schéma cookie reply správy je vysvetlená v 1.5.4. Konečným výsledkom takéhoto fungovania je zlepšenie cookie mechanizmov prítomných v DTLS [4] a IKEv2 [3].

1.5 Princíp správ

Pre WireGuard sú typické štyri druhy správ, pričom každá z nich má predponu jednobajtového indetifikátora typu správy `msg.type`. Ide o nasledovné správy:

- Iniciačná správa handshake – začína proces vytvárania bezpečného spojenia
- Odpoveď na iniciačnú správu handshake (handshake response) – ukončuje podanie rúk, po nej nasleduje možnosť vytvoriť bezpečné spojenie
- Odpoveď na jednu z uvedených správ v predošlých bodoch – posiela zašifrované cookie
- Zapúzdrený a zašifrovaný IP paket - prenášaný zabezpečeným spojením vyjednaným podaním rúk

Ďalej v rámci kapitoly budú popísané schémy jednotlivých správ a budú použité rozličné symboly a operátory – binárny operátor || reprezentuje zreťazenie operandov, ďalej binárny operátor := predstavuje priradenie pravého operandu k operandu ľavému. Iniciátor podania rúk je označený ako dolný index i, naopak označenie odpovedajúceho rovesníka je dolný index r. Ktorýkoľvek z nich je ďalej označený pomocou dolného indexu $*$. Pre správy, ktoré môže vytvárať ako iniciátor, tak aj odpovedajúci, platí, že $(m, m') = (i, r)$, ak správu vytvára rovesník v roli iniciátora a naopak, ak správu vytvára odpovedajúci rovesník, platí, že $(m, m') = (r, i)$. Obaja z rovesníkov si lokálne spravujú aj niekoľko premenných, ktoré následne pri zostavovaní správ používajú, ide o [1]:

- I_* 32-bitový index, lokálne reprezentuje druhého rovesníka
- S_*^{pub}, S_*^{priv} hodnoty pre statický verejný a statický súkromný kľúč
- E_*^{pub}, E_*^{priv} hodnoty pre efemérny verejný a efemérny súkromný kľúč
- Q hodnota nepovinného vopred zdieľaného symetrického kľúča; ak nie je prítomný mód vopred zdieľaných kľúčov, hodnota je nastavená na 0^{32}
- H_*, C_* hodnota výsledku hashu, hodnota kľúča reťazenia

Ďalej je použitá anotácia $\hat{n}$, ktorá vracia hodnotu (n+16) – Poly1305 [8] autentifikačná značka pridaná k n. Symbolom ϵ je vyjadrený prázdny bitový reťazec nulovej dĺžky, 0^n predstavuje celý nulový (`0x0`) bitový reťazec dĺžky n bajtov a ρ^n reprezentuje náhodný bitový reťazec dĺžky n bajtov. Nakoniec nech je τ dočasná premenná a κ dočasný šifrovací kľúč. Všetky hodnoty celých čísel sú typu little-endian. Dodatočne sú ešte použité nasledujúce funkcie a konštanty [1]:

- **DH(PRIVATE KEY, PUBLIC KEY)** – vracia 32-bitový výstup Curve25519 násobenia bodov súkromného a verejného kľúča
- **DH-GENERATE()** – generuje náhodný Curve25519 súkromný kľúč a k nemu odvodí verejný, vráti pár 32 bajtových hodnôt (PRIVATE, PUBLIC)
- **AEAD(KEY, COUNTER, PLAIN TEXT, AUTH TEXT)** – ChaCha20Poly1305 AEAD poďla RFC7539 [8] s nonce skladajúcej sa z 32 bitov núl nasledované 64-bitovou hodnotou COUNTER
- **XAEAD(KEY, NONCE, PLAIN TEXT, AUTH TEXT)** – XChaCha20Poly1305 AEAD s 24 bajtovou náhodnou NONCE, vytvorené pomocou HChaCha20 [23] a ChaCha20Poly1305 [8]
- **HASH(INPUT)** – BLAKE2S(INPUT, 32), vráti 32 bajtový výstup
- **MAC(KEY, INPUT)** – KEYED-BLAKE2S(KEY, INPUT, 16), kľúčovaná MAC verzia BLAKE2S funkcie, vracia 16 bajtový výstup
- **HMAC(KEY, INPUT)** – HMAC-BLAKE2S(KEY, INPUT, 32), obyčajná BLAKE2S hash funkcia, vracia 32 bajtový výstup
- **KDF_n(KEY, INPUT)** – nastaví $\tau_0 :=$ HMAC(KEY, INPUT), $\tau_1 :=$ HMAC(τ_0, 0x1), $\tau_i :=$ HMAC(τ_0, $\tau_{i-1} \| i$) a vráti n-ticu 32 bajtových hodnôt (τ_1,..., τ_n). Toto je HKDF [6] funkcia
- **TIMESTAMP()** – vracia TAI64N časovú značku [12] aktuálneho času (12 bajtový výstup), prvých 8 bajtov je big-endian celé číslo počtu sekúnd od 1970 TAI a posledné 4 bajty predstavujú big-endian celé číslo počtu nanosekúnd od začiatku tejto sekundy
- **CONSTRUCTION** – „Noise_IKpsk2_25519_ChaChaPoly_BLAKE2s“ UTF-8 reťazec, 37 bajtový výstup
- **IDENTIFIER** – „WireGuard v1 zx2c4 Jason@zx2c4.com“ UTF-8 reťazec, 34 bajtový výstup
- **LABEL-MAC1** – „mac1----“ UTF-8 reťazec, 8 bajtový výstup
- **LABEL-COOKIE** – „cookie--“ UTF- reťazec, 8 bajtový výstup

1.5.1 Správa iniciátor – príjemca

V tomto scenári sa snaží iniciátor naviazať spojenie pomocou správy msg [1]:

Tabuľka 1.5 Správa iniciátora príjemcovi

type:=0x1 *(1 bajt)*	reserved:=0^3 *(3 bajty)*
sender:=I_i *(4 bajty)*	
ephemeral *(32 bajtov)*	
static *($\widehat{32}$ bajtov)*	
timestamp *($\widehat{12}$ bajtov)*	
mac1 *(16 bajtov)*	mac2 *(16 bajtov)*

Funkcie `mac1` a `mac2` budú vysvetlené neskôr v podkapitole 1.5.1. I_i je generované náhodne (ρ^4) v čase odoslanie správy a slúži k spojeniu nasledujúcich odpovedí s reláciou, ktorá začala touto správou. Zostávajúce funkcie sú vypočítané nasledovne [5]:

$C_i := \texttt{HASH}(\texttt{CONSTRUCTION})$
$H_i := \texttt{HASH}(C_i \| \texttt{IDENTIFIER})$
$H_i := \texttt{HASH}(H_i \| S_r^{pub})$
$(E_i^{priv}, E_i^{pub}) := \texttt{DH-GENERATE}()$
$C_i := \texttt{KDF}_1(C_i, E_i^{pub})$
$\texttt{msg.ephemeral} := E_i^{pub}$
$H_i := \texttt{HASH}(H_i \| \texttt{msg.ephemeral})$
$(C_i, \kappa) := \texttt{KDF}_2(C_i, \texttt{DH}(E_i^{priv}, S_r^{pub}))$
$\texttt{msg.static} := \texttt{AEAD}(\kappa, 0, S_i^{pub}, H_i)$
$H_i := \texttt{HASH}(H_i \| \texttt{msg.static})$
$(C_i, \kappa) := \texttt{KDF}_2(C_i, \texttt{DH}(S_i^{priv}, S_r^{pub}))$
$\texttt{msg.timestamp} := \texttt{AEAD}(\kappa, 0, \texttt{TIMESTAMP}(), H_i)$
$H_i := \texttt{HASH}(H_i \| \texttt{msg.timestamp})$

Po prijatí takejto správy rovesník vykoná rovnaké operácie tak, aby boli jeho premenné konečného stavu totožné. Preto pri výpočtoch zamení operandov.

1.5.2 Správa príjemca naspäť iniciátorovi

Rovesník v roli odpovedajúceho posiela správu po spracovaní prvej, predom popísanej v 1.5.1. Aplikuje rovnaké operácie výpočtov, aby sa dostal do rovnakého stavu. I_r je generované náhodne (ρ^4) v čase odoslanie správy a slúži k spojeniu nasledujúcich odpovedí s reláciou ako tomu bolo v prvom prípade. Posiela takúto správu `msg` [1]:

Tabuľka 1.6 Správa od príjemcu naspäť iniciátorovi

`type:=0x2` *(1 bajt)*	`reserved:=`0^3 *(3 bajty)*
`sender:=`I_r *(4 bajty)*	`receiver:=`I_i *(4 bajty)*
`ephemeral` *(32 bajtov)*	
`empty` ($\hat{0}$ *bajtov)*	
`mac1` *(16 bajtov)*	`mac2` *(16 bajtov)*

Kde funkcie `mac1` a `mac2` sú vysvetlené v neskoršej časti kapitoly (1.5.3) a ostatné výpočty sú nasledovné [5]:

$(E_r^{priv}, E_r^{pub}) := \texttt{DH-GENERATE}()$

$C_r := \texttt{KDF}_1(C_r, E_r^{pub})$

$\texttt{msg.ephemeral} := E_r^{pub}$

$H_r := \texttt{HASH}(H_r \| \texttt{msg.ephemeral})$

$C_r := \texttt{KDF}_1(C_r, \texttt{DH}(E_r^{priv}, E_i^{pub}))$

$C_r := \texttt{KDF}_1(C_r, \texttt{DH}(E_r^{priv}, S_i^{pub}))$

$(C_r, \tau, \kappa) := \texttt{KDF}_3(C_r, Q)$

$H_r := \texttt{HASH}(H_r \| \tau)$

$\texttt{msg.empty} := \texttt{AEAD}(\kappa, 0, \epsilon, H_r)$

$H_r := \texttt{HASH}(H_r \| \texttt{msg.empty})$

Znova, ako v predošlom prípade, iniciátor po obdržaní tejto správy uskutoční rovnaké operácie tak, aby boli jeho premenné konečného stavu totožné, čo dosiahne pozmenením operandov DH funkcie. Táto správa je veľkosťou menšia ako správa prvá.

1.5.3 Cookie funkcie MAC

V podkapitolách vysvetľujúcich správy, 1.5.1 a 1.5.2, sú prítomné dve funkcie, `msg.mac1` a `msg.mac2`. Pre konkrétnu handshake správu – $\texttt{msg}_\alpha$ predstavuje všetky bajty správy `msg` pred `msg.mac1` a $\texttt{msg}_\beta$ reprezentuje všetky bajty správy `msg` pred `msg.mac2`. Najposlednejšie obdržané cookie pred sekundami $\tilde{L}_*$ je reprezentované L_*. Nakoniec, hodnota $\texttt{HASH}(\texttt{LABEL-MAC1} \| S_{m'}^{pub})$ funkcie môže byť vopred vypočítaná. Vyplnenie týchto mac funkcií je nasledovné [1]:

$\texttt{msg.mac1} := \texttt{MAC}(\texttt{HASH}(\texttt{LABEL-MAC1} \| S_{m'}^{pub}), \texttt{msg}_\alpha)$

ak $L_m = \epsilon$ or $\tilde{L}_m \geq 120$:

$\texttt{msg.mac2} := 0^{16}$

inak:

$\texttt{msg.mac2} := \texttt{MAC}(L_m, \texttt{msg}_\beta)$

1.5.4 Cookie reply správa

V prípadoch, kedy rovesník obdrží platnú prvú mac funkciu a následne neplatnú druhú mac funkciu a zároveň sa nachádza pod zaťažením, má možnosť poslať cookie reply správu. V takom prípade je $I_{m'}$ určené podľa poľa `msg.sender` zo správy, ktorá túto cookie reply správu vyžiadala. Názorné zobrazenie [1]:

Tabuľka 1.7 Cookie Reply Správa

`type:=0x3` *(1* bajt)	`reserved:=`0^3 *(3 bajty)*
`receiver:=`$I_{m'}$ (4 bajty)	
`nonce:=`ρ^{24} *(24 bajtov)*	
`cookie` $(\widehat{16}$ *bajtov)*	

Tajná náhodná premenná, R_m, mení svoju hodnotu každé dve minúty, A_m označuje zreťazenie vonkajšej IP zdrojovej adresy rovesníka označeného dolným indexom a UDP zdrojového portu. *M* je hodnota `msg.mac1` správy, na ktorú odpovedá. Zbytok cookie poľa sa vyplní takto[1]:

$$\tau := \mathrm{MAC}(R_m, A_{m'})$$

$$\texttt{msg.cookie} := \mathrm{XAEAD}(\mathrm{HASH}(\texttt{LABEL-COOKIE} \parallel S_m^{pub}), \texttt{msg.nonce}, \tau, M)$$

Výsledkom je, že cookie reply správa je zviazaná s relevantnou správou, a tým je znemožnený akýkoľvek útok, kedy by boli rovesníkovi posielané podvodné cookie reply správy. V situácii, kedy je takáto správa platná, po jej prijatí si príjemca uloží cookie spolu s časom prijatia.

1.5.5 Transportné dáta – odvodenie kľúča a správy o transportných údajoch

Po ukončení výmeny spomínaných správ medzi iniciátorom a príjemcom dochádza k výpočtu kľúčov iniciátora a jeho odpovedajúceho pre odosielanie a prijímanie správ trasportných dát.

WireGuard rovesníci si počas fázy podávania rúk vymenia medzi sebou Diffie-Hellman verejné kľúče a následne ešte uskutočnia postupnosť Diffie-Hellman operácií, kedy zahashujú výsledky do zdieľaného tajného kľúča. Po ukončení fáze podávania rúk je takto vytvorený tajný kľúč ďalej používaný pri odosielaní zašifrovaných transportných správ [5].

WireGuard používa koncept transportných kľúčov na zabezpečenie komunikácie medzi rovesníkmi. Tvorba týchto kľúčov spočíva v derivácii zo zdieľaných statických verejných kľúčov a efemérnych (dočasných) kľúčov relácie generovaných počas procesu podávania rúk. Takáto derivácia sa skladá z niekoľkých krokov [5]:

- Každá relácia má svoju konkrétnu krátkodobú dvojicu kľúčov pre každé podanie rúk
- Lokálny statický verejný kľúč, vzdialený statický verejný kľúč a dočasné verejné kľúče relácie sú skombinované čím vytvoria zdieľané tajomstvo

- Toto zdieľané tajomstvo je použité ako vstup do funkcie derivovania kľúčov (KDF), ktorej výstupom sú transportné kľúče použité pre šifrovanie a overovanie správnosti dát. KDF zabezpečuje, že čo i len minimálna zmena vstupnej hodnoty do výraznej miery pozmení hodnotu výstupu
- Nakoniec sú teda takto odvodené transportné kľúče použité k šifrovaniu a overovaniu prenášaných dát medzi rovesníkmi

2. POROVNANIE WIREGUARD S IPSEC A OPENVPN

Porovnávanie virtuálnych privátnych sietí WireGuard, IPsec a OpenVPN je v nasledujúcej časti vykonané na základe nimi používaných šifrovacích algoritmov a kryptografických primitív.

WireGuard je VPN protokol dosahujúci vysoké rýchlosti, pričom si stále zachováva svoju jednoduchosť a zároveň poskytuje efektívne a zabezpečené pripojenie. Aj napriek týmto kladným vlastnostiam je to protokol, ktorý má veľmi malú réžiu. Používa nástroje najmodernejšej kryptografie. Ide o relatívne novú technológiu, a preto nebola zatiaľ dôkladne preverená, ako napríklad OpenVPN. Za zmienku stojí aj fakt, že WireGuard je implementovaný do samotného linuxového jadra od verzie 5.6. V čase písania tejto práce nemá WireGuard žiadne známe závažnejšie zraniteľnosti.

Základným kameňom protokolu ju symetrická kryptografia postavená na ChaCha20 [8], ďalej je pre proces dohody o kľúči dôležitá eliptická krivka Diffie-Hellman (ECDH) (Curve25519 [2]). Hashovanie je v protokole zaobstarávané kryptografickou hashovacou funkciou BLAKE2s [9]. V neposlednom rade je využívaná funkcia SipHash24 [13], ktorá poskytuje kľúče pre hashovaciu tabuľku. Ďalej je použité podanie rúk (handshake) založené na UDP a samotná výmena kľúčov je zabezpečená doprednou bezpečnosťou. Nakoniec, odvodzovanie kľúčov je zaručené pomocou jednoduchej funkcie HKDF [6].

IPsec predstavuje sadu niekoľkých protokolov a v dnešnej dobe je považovaný za štandard v oblasti zabezpečenia internetovej komunikácie. Zabezpečuje dôvernosť a integritu autentifikácie. Rozsahovo sa jedná o zďaleka robustnejšiu kódovú základňu ako je tomu pri WireGuard, čo značne sťažuje jeho potrebný manažment a správu. Narozdiel od technológie WireGuard, IPsec mal časom objavené dokonca aj vyriešené niektoré chyby zabezpečenia. Patrila medzi ne napríklad zraniteľnosť spôsobujúca odmietnutie služby CVE-2018-5389 alebo zraniteľnosť umožňujúca útočníkom znížiť silu zabezpečenia pozmenením šifrovej súpravy, tiež známa ako FREAK útok.

IPsec ponúka rozsiahle množstvo použiteľných kryptografických a šifrovacích algoritmov, čo prináša aj riziko, že pri nedostatočne znalej implementácii môže prísť k nesprávnemu definovaniu algoritmu, ako je známe napríklad pre RSA [14] algoritmus (jeho 1024-bitové kľúče už tiež nie sú považované za bezpečné).

Protokol IPsec IKEv2 [3] ponúka širokú škálu možností implementácie kryptografických protokolov. Z nich ide napríklad o AES [15] spolu s ChaCha20-Poly1305 taktiež pre zaručenie dôvernosti. Naopak ako algoritmus výmeny kľúčov je v RFC3526 [16] špecifikovaný Diffie-Hellman [17] alebo v RFC4753 [18] algoritmus ECDH [19]. Nakoniec pre algoritmy autentifikácie je možné použiť, napríklad ECDSA [20] alebo PSK [21].

OpenVPN je open-source VPN protokol, ktorý nie je špecifikovaný žiadnymi štandardmi (RFC štandardy) a chýba mu dokonca aj kompatibilita s IPsec, no aj napriek

tomu je v dnešnej dobe značne rozšírený. Pre zabezpečenie komunikácie používa vlastnú sadu protokolov SSL/TLS pre výmenu kľúčov. Poskytuje úplnú diskrétnosť, autentifikáciu a integritu. Štandardne prebieha komunikácia pomocu UDP, no je možné použiť aj TCP. Na šifrovanie sa používajú protokoly z knižnice OpenSSL [22]. Patria medzi ne napríklad AES [15], IKEv2 [3] s 256 bitovými kľúčmi.

Systém OpenVPN je tiež stabilný a rýchly najmä cez bezdrátové a mobilné siete alebo akékoľvek iné siete, ktorých spoľahlivosť nemusí byť vždy 100% a môže dochádzať ku stratám paketov. Pre takéto nespoľahlivé siete má OpenVPN implementovaný mód s použitím TCP namiesto UDP, čím je ale obetovaná časť jeho výkonu, čo je spôsobované neefektívnosťou zapúzdrovania TCP v rámci TCP.

3. TESTOVANIE

3.1 Metodika testovania

Predmetom testovania je vyhodnotenie výkonových parametrov technológií WireGuard, IPsec a OpenVPN a ich následné porovnanie. Testovanými parametrami teda budú – jitter, oneskorenie, dosahované rýchlosti a vyťaženie procesora

- **Jitter** – predstavuje zmeny veľkosti oneskorenia paketov v dôsledku zmien smerovania alebo chovania routeru
- **Oneskorenie** (lattency) – časový rozdiel medzi pokynom k začatiu prenosu a samotným začatím prenosu

Testovanie bolo vykonané podľa nami zvolenej topológie siete (podkapitola 3.2), pričom ako nástroj použitý na meranie parametrov priepustnosti siete bol `iperf3`[24], do testovania bolo taktiež zakomponované aj sledovanie vyťaženie procesorov nami zvolených sieťových prvkov (podkapitola 3.3), pričom tento parameter bol meraný proprietárnym nástrojom spoločnosti Mikrotik, takzvaný *Profile*.

Nástroj `iperf3` podporuje ladenie parametrov týkajúcich sa časovania, vyrovnávacej pamäte a protokolov, ktoré nám dovolia prispôsobiť test našim sieťovým prvkom a vlastnostiam siete. Z protokolov je možné použiť ako UDP, tak TCP, čo je pre naše testovanie postačujúce. Funguje na princípe zostavenia spojenia server-klient, kde zo strany klienta sú na stranu servera odosielané dáta a následne meria rozličné parametre, ktoré si môže sám testujúci špecifikovať podľa potreby. Samotné testy v našom prípade mali v priemere 10 minút.

Po zostavení testovacích scenárov, teda implementovaní konfigurácií virtuálnych privátnych sietí zodpovedajúcich nami zvolenej topológie siete (podkapitola 3.2) a následnom vykonaní konkrétnych testov na všetkých troch VPN protokoloch, budú získané výsledky medzi sebou porovnané. Porovnaním dostaneme lepšiu predstavu o výkonnosti meraných VPN technológií. Je avšak nutné poznamenať, že výsledky testov môžu byť ovplyvnené rozličnými faktormi. Jeden z faktorov predstavujú aj použité šifrovacie algoritmy, kedy v prípade protokolu IPsec máme na výber z viacerých možností. Nakoniec bude každý test vykonaný viac krát, aby nami získané výsledky mali čo najvyššiu mieru spoľahlivosti a výpovednej hodnoty.

Ako prvú sme otestovali virtuálnu privátnu sieť WireGuard, pretože práca sa z najväčšej časti venuje práve jej, kedy sme potom na základe výsledkov a pozorovaní vývoja testu v reálnom čase odvodili testovacie kritériá, ktorými sme sa riadili vo zvyšných testoch ostatných technológií.

Prvotne stanovené kritérium sa týkalo strátovosti paketov počas prenosu, kedy sme si ustanovili približnú hranicu 5-10 %. Od tohoto kritéria sa následným testovaním ďalej odvodilo kritérium veľkosti šírky pásma, kedy nedochádzalo k neprípustnému

zahadzovaniu paketov – takúto veľkosť predstavovala hodnota 60 Mbit/s pre oba použité sieťové prvky podľa podkapitoly 3.3.

3.2 Topológia siete – Site-to-site

Topológia siete použitá pri testovaní nami zadaných parametrov predstavovala situáciu tiež známu ako Site-to-site virtuálna privátna sieť. Takéto označenie predstavuje spojenie medzi viacerými sieťami, kedy by sa mohlo napríklad jednať o sieť spoločnosti, v ktorej spolupracuje viacero fyzicky vzdialených pobočiek, ktoré potrebujú medzi sebou navzájom komunikovať prostredníctvom siete Internet, pričom je potrebné, aby takto naviazané spojenie bolo bezpečné a dochádzalo k šifrovaniu takto vymieňaných dát.

Site-to-site virtuálne privátne siete sú taktiež bežne používané pre účely replikácie údajov a zálohovania dátových centier. Vytvorením bezpečných spojení medzi primárnymi a sekundárnymi dátovými centrami, je zaručené, že replikácia a zálohovanie kritických dát prebieha v reálnom čase, čo značne znižuje redundanciu dát a poskytuje možnosti obnovy po incidente.

Cloudové pripojenie: Pri pripájaní sa k poskytovateľom cloudových služieb môže sieť VPN takéhoto typu vytvoriť bezpečné a súkromné prepojenie medzi lokálnou a cloudovou infraštrukúrou.

Sieťová konektivita internetu vecí (IoT): Site-to-site v takomto prípade poskytuje zabezpečné pripojenie a následnú správu zariadení internetu vecí medzi rozličnými lokáciami. Ďalej je takto zabezpečená centralizovaná kontrola a monitorovanie týchto zariadení, pričom je dodržané utajenie a integrita prenášaných dát medzi zariadeniami aj samotným centrálnym riadiacim systémom.

V prípade nášho testovania teda išlo o scenár, kedy spájame dve lokálne siete s cieľom zabezpečiť ich vzájomnú komunikáciu naprieč Internetom, ktorá je bezpečná a šifrovaná.

3.3 Použité komponenty

Počas testovania technických parametrov spomínaných virtuálnych privátnych sietí boli použité sieťové prvky spoločnosti Mikrotik. Konkrétne sa jednalo o sieťové karty RB-493G a sieťové karty RB941-2nD. Na oboch komunikujúcich stranách sa vždy vyskytovali rovnaké sieťové prvky, teda išlo o site-to-site testovanie medzi RB-493G – RB493G a následne rovnako pre RB941-2nD.

Hardvérové špecifikácie nami použitých prvkov boli nasledovné:

- MikroTik RB941-2nD – router s dostačujúcim hardvérovým vybavením pre naše testy, ktorý disponuje štyrmi 10/100 Mbit/s ethernetovými portami, výkonným 650 MHz procesorom a 32MB RAM

- MikroTik RB-493G – prenosové rýchlosti 10/100/1000 Mbit/s, 1 jadrový procesor s frekvenciou 680 MHz a 256 MB pamäťou RAM

Testovanie bolo uskutočnené na vytvorenej virtuálnej privátnej sieti medzi dvoma lokálnymi sieťami, kedy v jednej z nich bol používateľ reprezentovaný počítačom s operačným systémom Windows 10, pričom v druhej lokálnej sieti išlo o používateľa s operačným systémom Debian Linux.

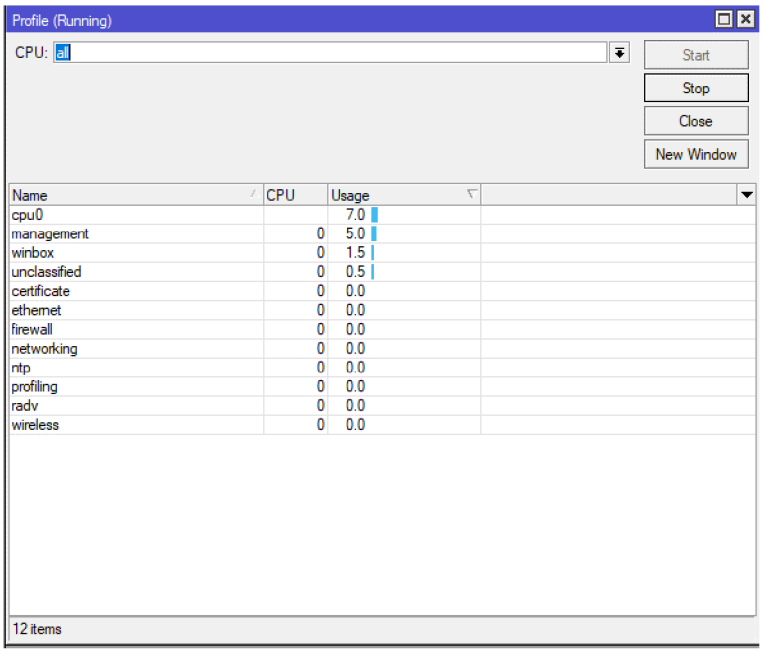
MikroTik RB-493G sa nachádzal počas testovania vo verzii 7.9, pričom bola použitá aj takzvaná Fasttrack funkcia [25], ktorej význam spočíva v znížení zaťaženia procesora zariadenia MikroTik, pričom sa teoreticky zlepšuje využitie dostupnej šírky pásma. Na druhej strane, zariadenia MikroTik RB941-2nD, boli použité vo verzii 7.5, taktiež s nakonfigurovanou funkciou Fasttrack.

4. VÝSLEDKY TESTOVANÍ TECHNICKÝCH PARAMETROV VPN

Najprv budú v rámci kapitoly predstavené výsledky testovania s použitím Mikrotik zariadení RB-493G pre konkrétne virtuálne privátne siete. Uvedené budú taktiež aj výsledky testovania, kedy pre komunikáciu medzi lokálnymi sieťami v našej topológii nebola použitá žiadna virtuálna privátna sieť a komunikácia bola sprostredkovaná len pomocou statického smerovania medzi našimi sieťovými prvkami.

Ďalej v rámci kapitoly budú predstavené výsledky týkajúce sa sieťových prvkov Mikrotik RB-941-2nD s rovnakými kritériami (popis ich stanovenia v podkapitole 3.1) ako tomu bolo pri RB-493G.

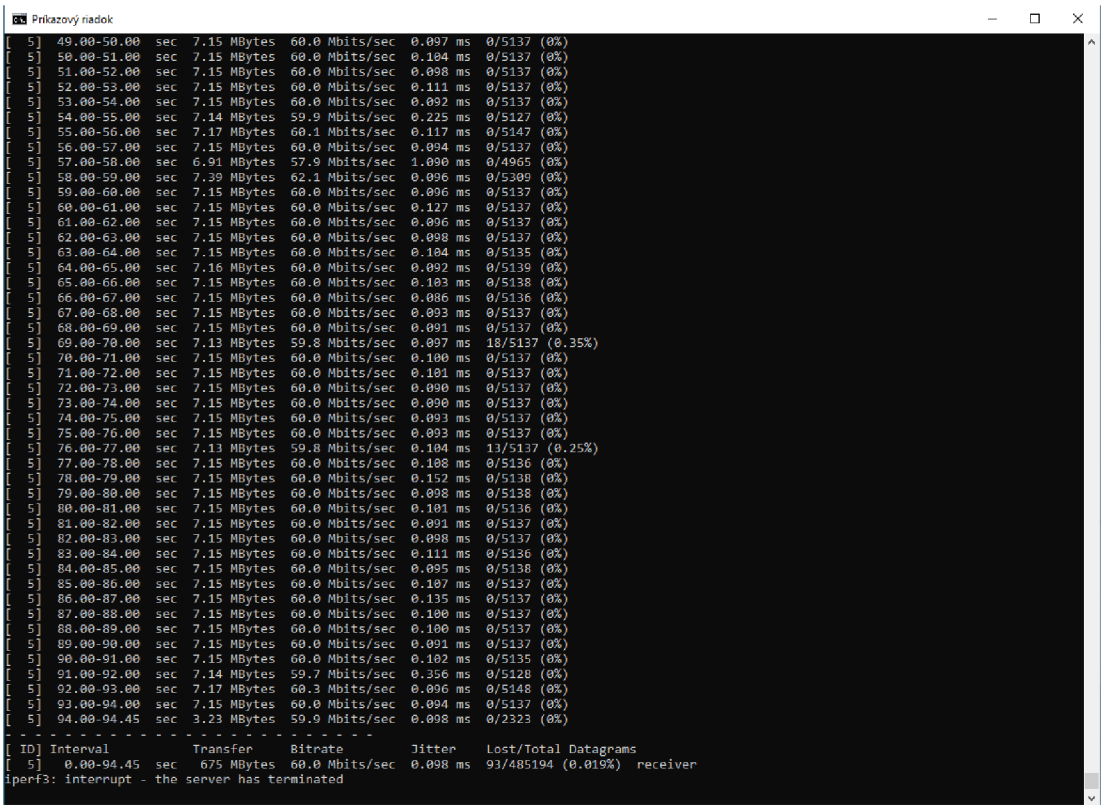
4.1 Testovanie bez použitia virtuálnej privátnej siete – statické smerovanie medzi lokálnymi sieťami (RB-493G)



Obrázok 4.1 Zaťaženie procesora Mikrotik RB-493G, statické smerovanie

Testovanie vlastností siete a taktiež hardvérového zaťaženia pri použití nástroja `iperf3` pri použití statického smerovania dovoľujúce komunikáciu medzi lokálnymi sieťami v našej testovacej topológii. Na obrázku *4.1* môžeme vidieť, že pri konfigurácii so statickým smerovaním dochádza len k minimálnemu zaťaženiu procesora nami použitého

sieťového prvku Mikrotik RB-493G. Toto zaťaženie sa počas celého testu pohybovalo len v rozmedzí do 10 % s občasnými výstupmi až na 80 % celkového zaťaženia na strane príjemcu komunikácie, `iperf3 servera`. Na strane odosielajúceho, teda takzvaného `iperf3 klienta`, dochádzalo len k minimálnemu navýšeniu zaťaženia procesora oproti strane prijímajúceho, kedy občasné výkyvy predstavovali až 85 %.

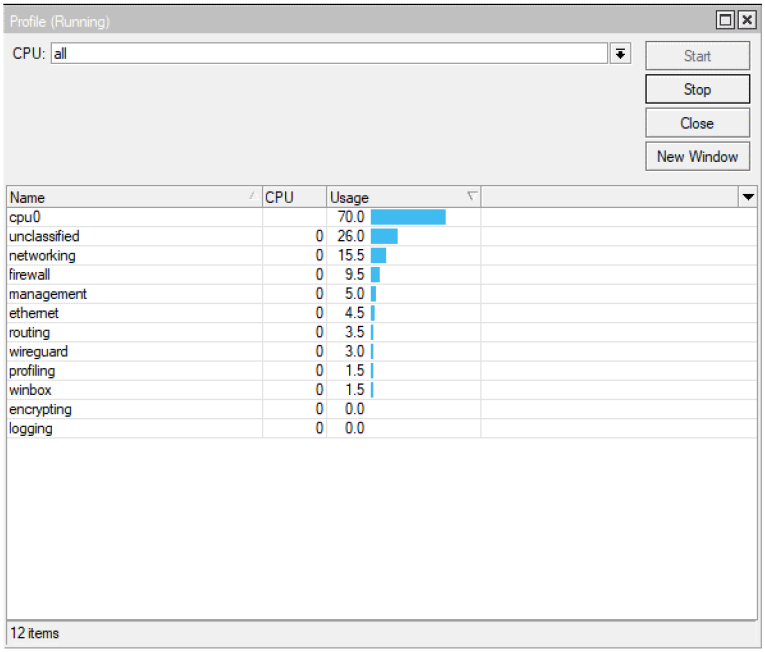


Obrázok 4.2 Iperf3 ukážka výstupu testu, statické smerovanie

V prípade statického smerovania a jeho testovania pomocou nástroja `iperf3`, kedy maximálna šírka pásma predstavovala 60 Mbit/s podľa nami stanovených kritérií (podkapitola 3.1), sme obdržali výsledky vypovedajúce o takmer 100% využití dostupnej šírky pásma.

`Iperf3` test bol v takomto testovacom scenári v priemere schopný dosahovať až 59,6 Mbit/s prenosové rýchlosti. V prípade parametra jitter sa jednalo o priemerné hodnoty 0,096 milisekúnd. Nakoniec priemerné hodnoty parametra oneskorenia predstavovali 2,156 milisekúnd.

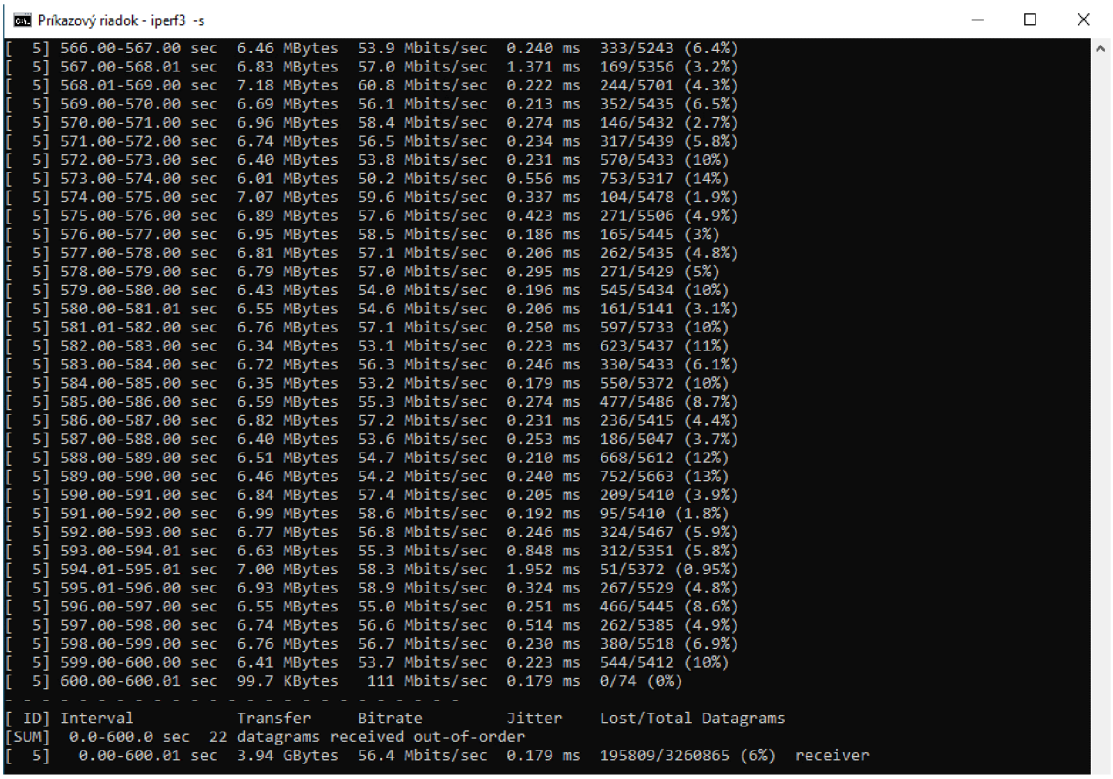
4.2 Testovanie s použitím WireGuard virtuálnej privátnej siete (RB-493G)



Obrázok 4.3 Zaťaženie procesora Mikrotik RB-493G, WireGuard

Ďalšie z testov boli uskutočnené na implementácii virtuálnej privátnej siete WireGuard medzi dvoma lokálnymi sieťami. Počas testovania dochádzalo k prípustným hodnotám zahadzovania datagramov `iperf3` testu (vysvetlené v podkapitole 3.1) pri špecifikovaní šírky pásma na 60 Mbit/s.

V takomto scenári sme sa stretli s vyťažovaním procesora na strane prijímajúceho v hodnotách okolo 70 %, kde proces *unclassified* a *wireguard* (na oboch stranách komunikácie) z toho dokopy tvorili okolo 30 %, počas celého trvania testovania. Na strane odosielajúceho (klient) potom išlo až o 100% vyťaženie procesora Mikrotiku.

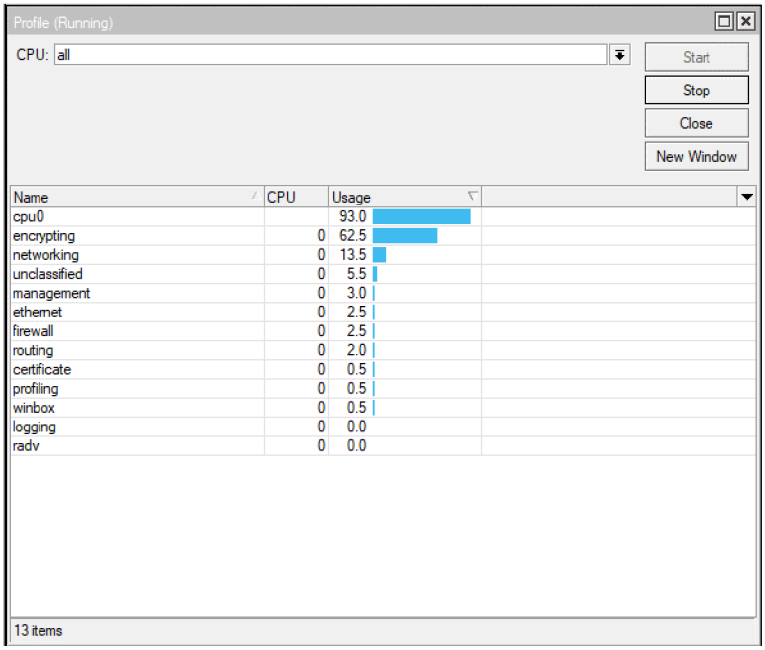


Obrázok 4.4 Iperf3 výstup, WireGuard

Po skončení testovania technológie WireGuard sme obdržali výsledky, kde prijímajúca strana oznamuje priemerné hodnoty prenosovej rýchlosti vo výške až 56,4 Mbit/s, pričom priemerná hodnota na strane odosielajúceho predstavovala 59 Mbit/s.

Priemerná hodnota parametra jitter nakoniec bola 0,179 milisekúnd a v prípade meraného oneskorenia sa jednalo o priemernú hodnotu 2,917 milisekúnd.

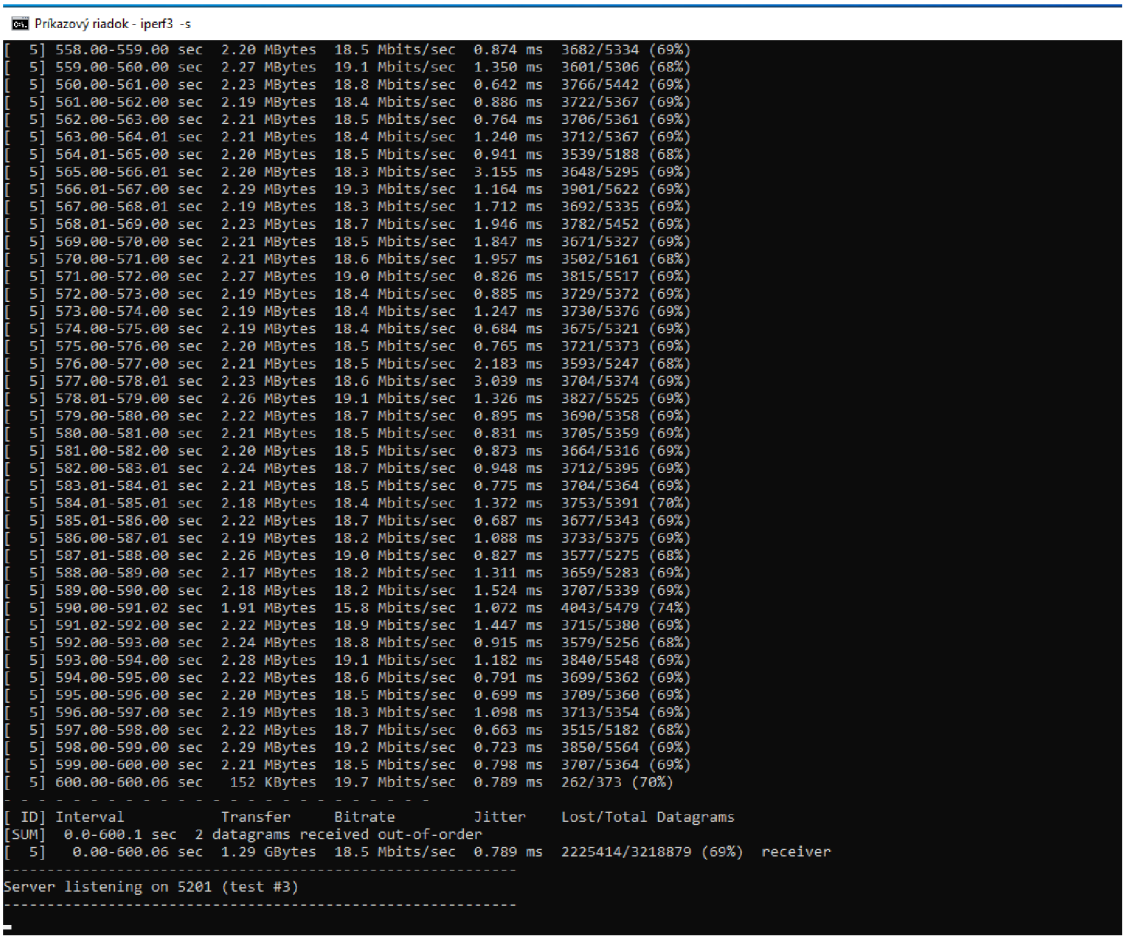
4.3 Testovanie s použitím IPsec virtuálnej privátnej siete (RB-493G)



Obrázok 4.5 Zaťaženie procesora Mikrotik RB-493G, IPsec

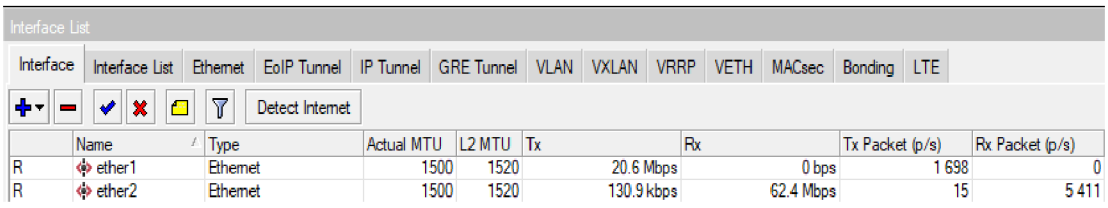
Šifrovací algoritmus technológie IPsec použitý počas nášeho testovania bol AES-256-cbc [15], ako hashovaciu funkciu sme zvolili sha2-256 bit [27]. Diffie-Hellman groupa 14 [26] a IKEv2 [3] návrh boli zvolené ako zvyšné parametre. Bolo tiež zvolené overovanie pomocou digitálneho podpisu s využitím certifikátov pre obe strany.

Pri takejto konfigurácii našej virtuálnej privátnej siete IPsec dochádzalo k zaťaženiu procesora Mikrotiku na strane prijímajúceho v medziach 90-100 %, kedy značnú časť z toho predstavoval proces šifrovania *encrypting* (obrázok *4.5*). Na strane odosielajúceho sme potom mohli sledovať vyťaženie procesora neustále na 100 %. Ako tomu bolo aj v prípade prijímajúceho, najväčší podiel na zaťažení mal proces šifrovania.



Obrázok 4.6 Iperf3 výstup, IPsec

Konečné výsledky testovania takejto konfigurácie VPN IPsec predstavovali v priemere 18,5 Mbit/s prenosové rýchlosti na strane prijímajúceho, pričom strana odosielajúceho oznamovala priemer 59 Mbit/s.



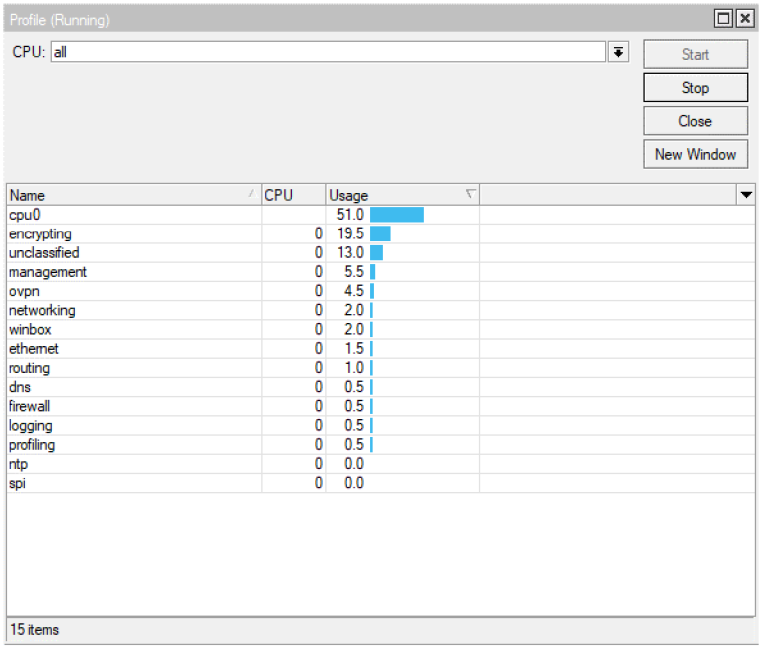
Obrázok 4.7 Bottleneck procesora Mikrotik, IPsec odosielajúci

V tomto prípade sme sa teda stretli s bottleneckom na procesore Mikrotika (obrázok *4.7*) zo strany odosielajúceho, kedy nebol schopný podobných 59 Mbit/s šifrovať a zároveň prenášať na druhú stranu. Toto sa odzrkadlilo aj na výslednej priemernej hodnote obdržaných datagramov prijímajúcim, kedy išlo o hodnotu 31 %.

Oznámená priemerná hodnota parametra jitter na konci testu – 0,789 milisekúnd.

Nameraná priemerná hodnota oneskorenia v takomto scenári predstavovala hodnotu 3,492 milisekúnd.

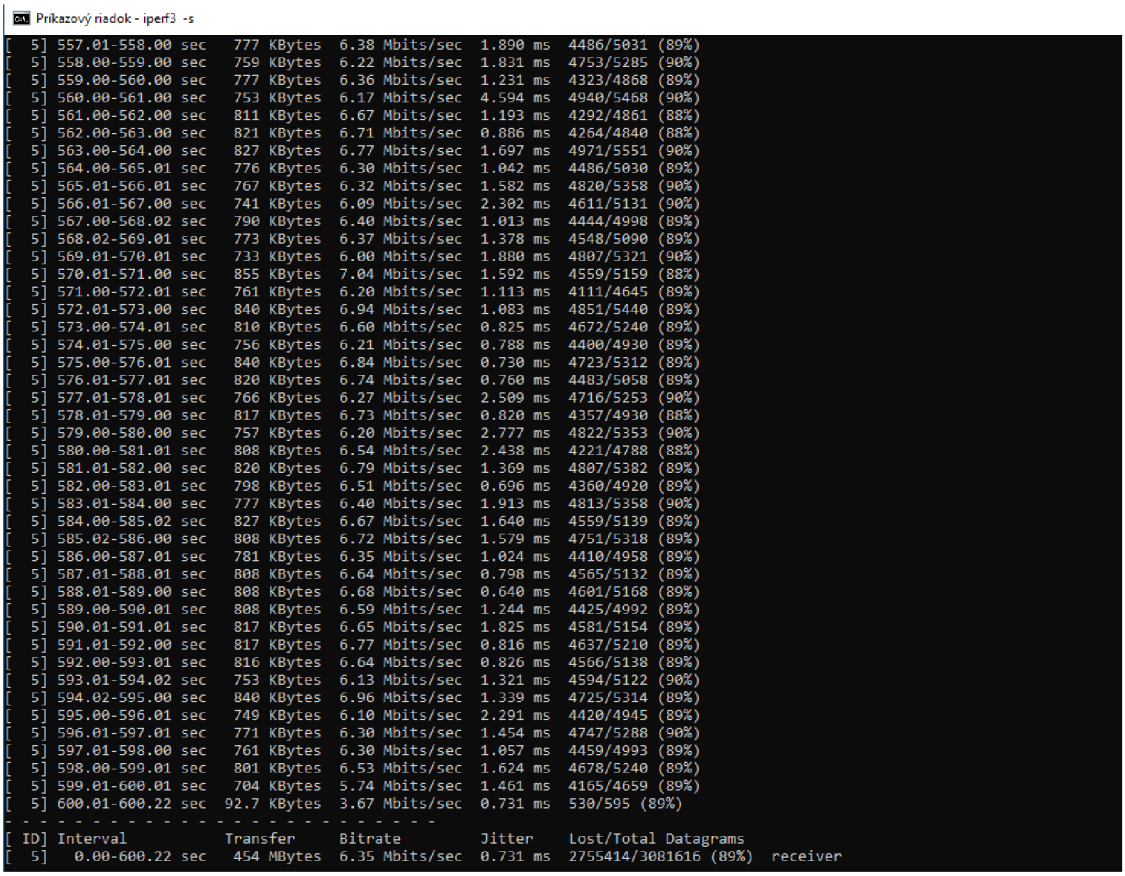
4.4 Testovanie s použitím OpenVPN virtuálnej privátnej siete (RB-493G)



Obrázok 4.8 Zaťaženie procesora Mikrotik RB-493G, OpenVPN

Algoritmy systému OpenVPN použité počas testovania boli AES-256 [15] pre šifrovanie a ako protokol overovania sme zvolili sha2-256 bit [27].

Rovnako ako v prípade testovania technológie IPsec, aj pri tomto testovaní sa opäť stretávame s bottleneckom zo strany procesora Mikrotiku odosielajúceho. Zatiaľ čo na strane prijímajúceho sa vyťaženie pohybuje v priemere okolo 50 %, strana odosielajúceho zaznamenáva neustále vyťaženie 100 %. Teda tento procesor nie je schopný zároveň šifrovať a ďalej posielať dostupnou maximálnou rýchlosťou.



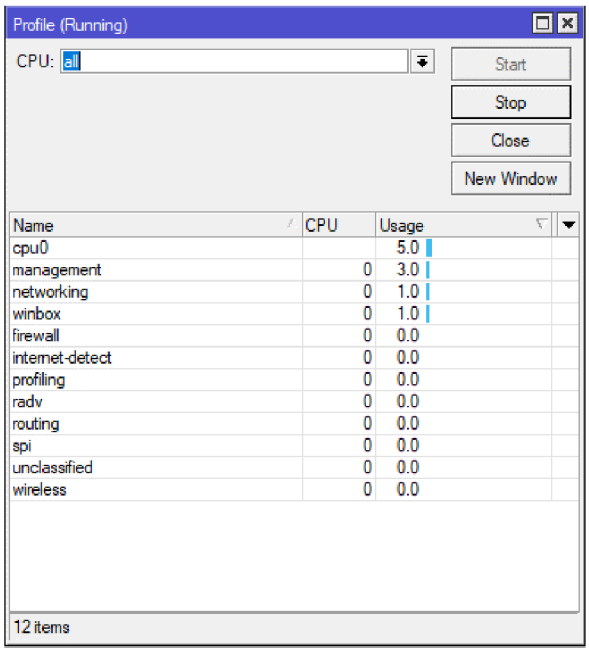
Obrázok 4.9 Iperf3 výstup, OpenVPN

Pri nastavenej šírke pásma 60 Mbit/s dochádza počas testu k značnému zahadzovaniu datagramov, strane prijímajúceho prijme asi 11 % z celkového počtu odoslaných. Toto je spôsobené spomínaným bottleneckom procesora.

Výsledná priemerná hodnota prenosovej rýchlosti nakoniec predstavuje 6,35 Mbit/s.

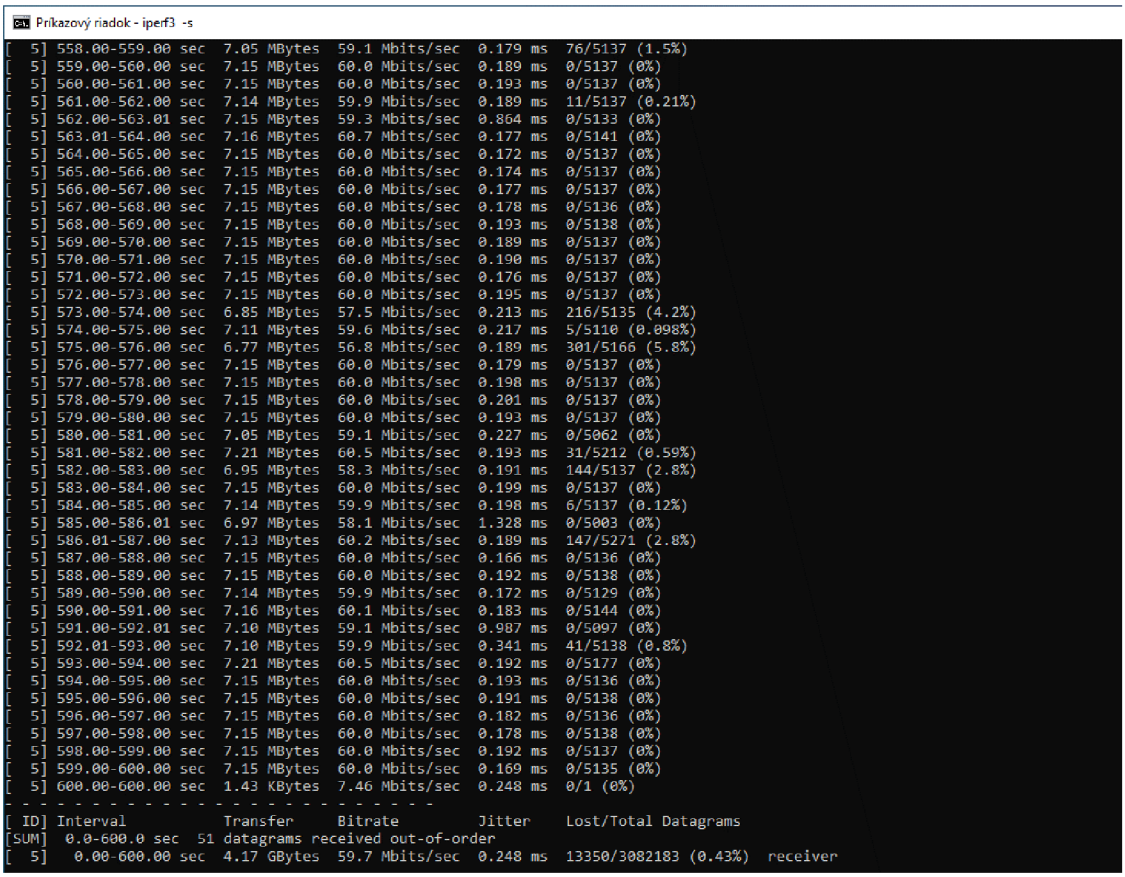
Oznámená priemerná hodnota parametra jitter je 0,731 milisekúnd a nakoniec priemerné oneskorenie bolo 4,359 milisekúnd.

4.5 Testovanie bez použitia virtuálnej privátnej siete – statické smerovanie medzi lokálnymi sieťami (RB941-2nD)



Obrázok 4.10 Zaťaženie procesora Mikrotik RB941-2nD, statické smerovanie

Podobne ako v prípade testovania statického smerovania s použitím sieťových prvkov RB-493G (podkapitola 4.1), aj tu zaznamenávame len minimálne vyťaženie procesorov na oboch stranách komunikácie, pričom významnú časť testu išlo len o hodnoty do 10 % s občasnými výkyvmi do 30 %.



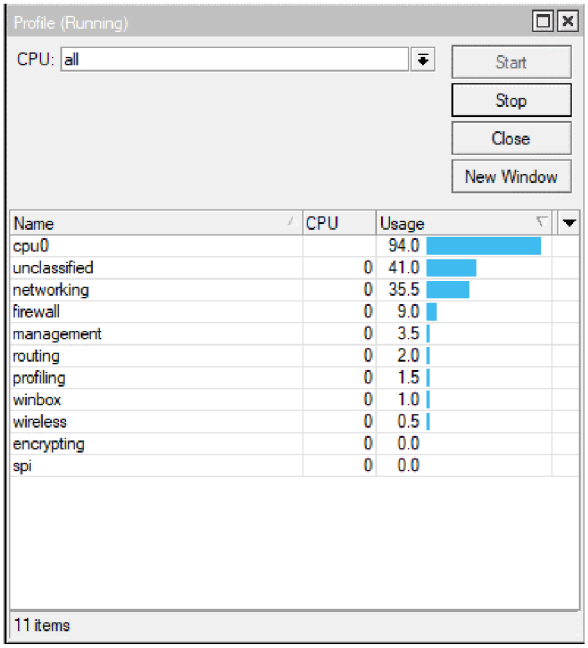
Obrázok 4.11 Iperf3 výstup, statické smerovanie RB941-2nD

Priemerné prenosové rýchlosti 59,7 Mbit/s, takmer 100% zúžitkovanie ponúknutej šírky pásma. Strátovosť veľmi nízka, prijímajúci obdrží takmer všetky odoslané datagramy.

Priemerná hodnota parametra jitter nakoniec 0,248 milisekúnd.

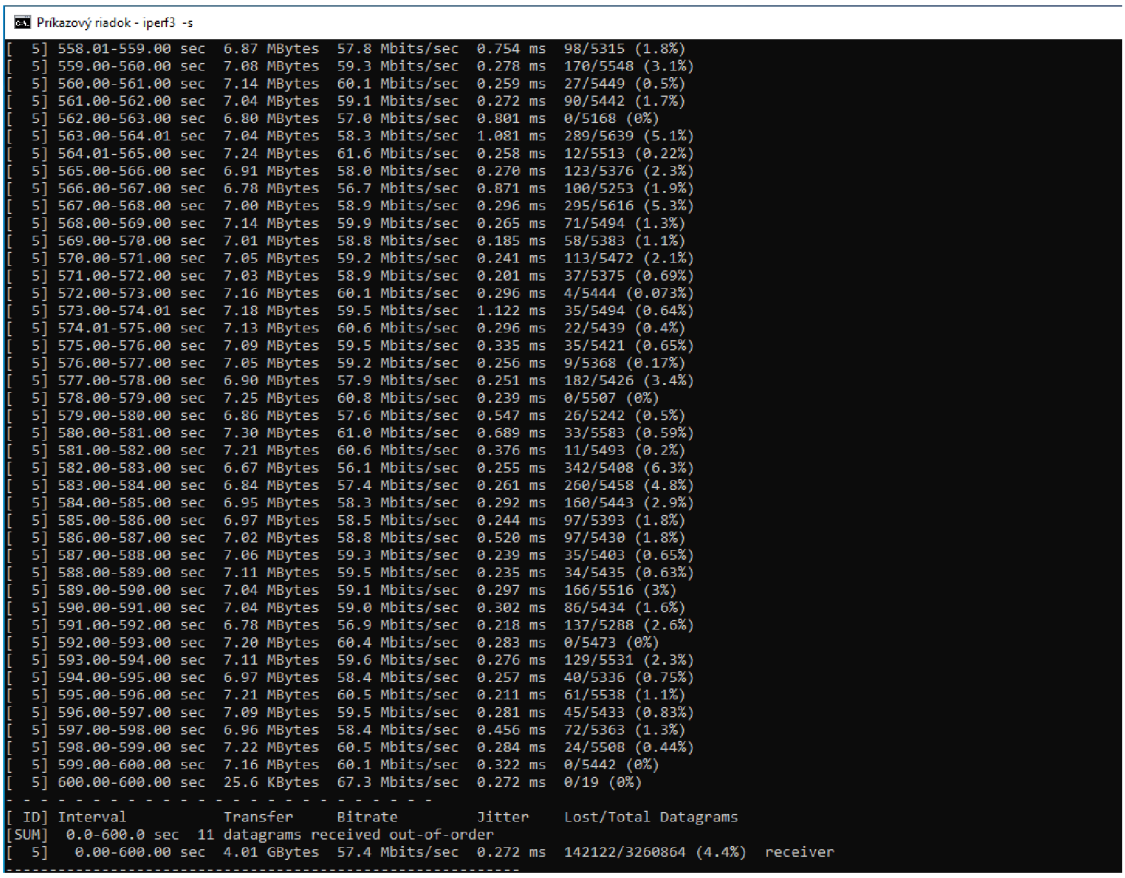
Nameraná priemerná hodnota oneskorenia pri takomto scenári predstavovala 2,214 milisekúnd.

4.6 Testovanie s použitím WireGuard virtuálnej privátnej siete (RB941-2nD)



Obrázok 4.12 Zaťaženie procesora Mikrotik RB941-2nD, WireGuard

Pri tomto testovaní sa už stretávame so značným zaťažením procesorov Mikrotiku na oboch stranách. Narozdiel od testovania technológie WireGuard na RB-493G (podkapitola 4.2) dochádzalo k väčšiemu zaťaženiu procesora na strane prijímajúceho, a to v medziach od 90 do 100 % počas celého trvania testu. Značný podiel na takomto zaťažení zohrával proces *unclassified* (50-60 %), pod ktorým sa môže nachádzať wireguard, no router ho v našom prípade nevie klasifikovať. Procesor na strane odosielajúceho sa nachádzal v neustálom vyťažení 100 %.



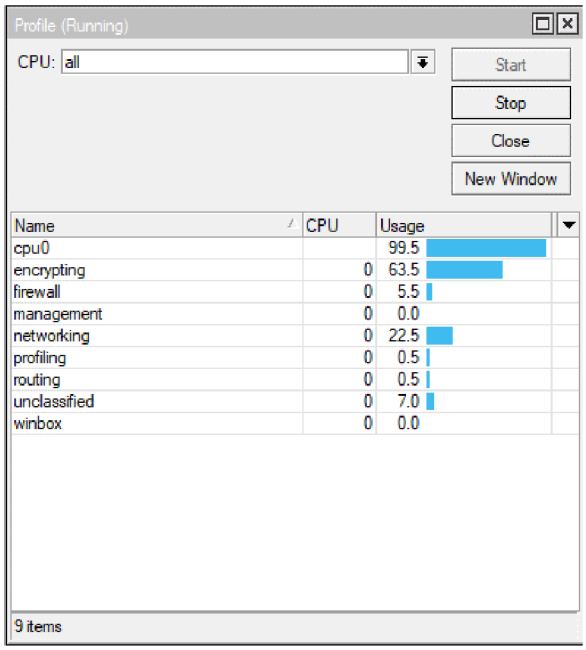
Obrázok 4.13 Iperf3 výstup, WireGuard, RB941-2nD

Výsledná priemerná hodnota prenosovej rýchlosti v takomto testovacom scenári predstavovala 57,4 Mbit/s. Aj napriek vysokým hodnotám zaťažovania procesora na strane odosielajúceho nedochádzalo k neprípustným hodnotám zahadzovania testovacích datagramov ako tomu bolo v prípadoch 4.3 a 4.4.

Priemerná hodnota pre jitter nakoniec 0,272 milisekúnd.

Priemerná hodnota pre sledovaný parameter oneskorenia – 3,062 milisekúnd.

4.7 Testovanie s použitím IPsec virtuálnej privátnej siete (RB941-2nD)

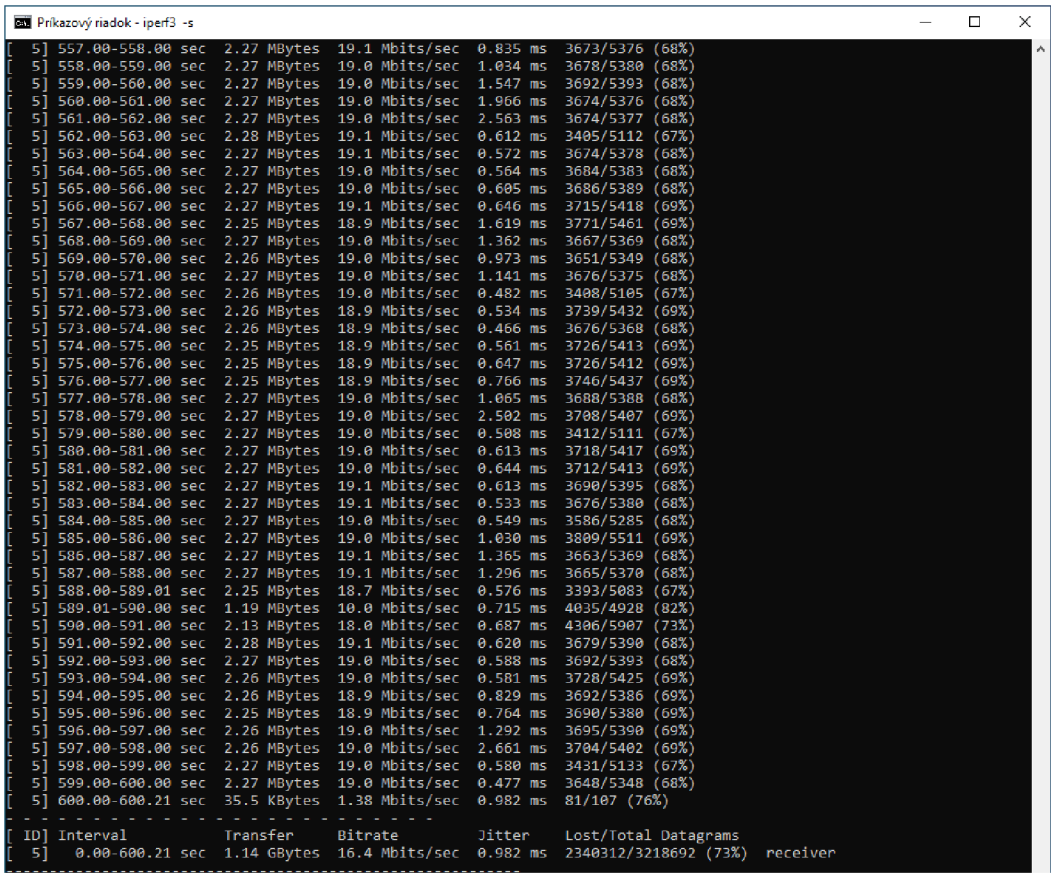


Obrázok 4.14 Zaťaženie procesora Mikrotik RB941-2nD, IPsec

Šifrovacie algoritmy a návrh použité v tomto scenári boli rovnaké ako v prípade 4.3, s jedinou zmenou týkajúcou sa použitého spôsobu overovania. Tu sa jedná o metódu predzdieľaného kľúča.

V tomto scenári dochádzalo k takmer 100% vyťaženiu procesorov na oboch stranách počas celého trvania testu. Významnú časť (priemerne 65 %) z toho tvoril proces *encrypting*.

Na strane odosielajúceho sme sa začali potýkať s obrovskými nestabilitami a celkovým bottleneckom procesora Mikrotiku. Počas testovaní dochádzalo k častým výpadkom a degradáciám rýchlostí, ktoré potom len málokedy boli schopné vrátiť sa k maximálnym hodnotám. Odosielajúca strana pod takýmto zaťažením dokonca zhadzovala aj grafické uživateľské rozhranie Mikrotiku a následnými pokusmi o vrátenie ho späť tak do určitej miery ovplyňovala aj samotné výstupy testov.



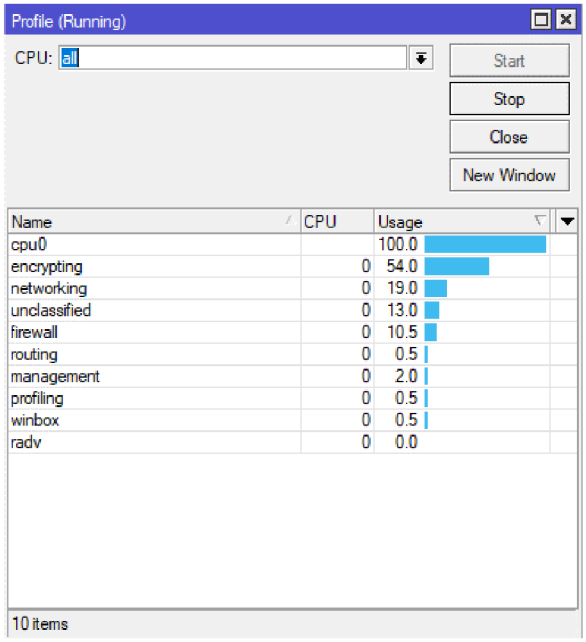
Obrázok 4.15 Iperf3 výstup, IPsec, RB941-2nD

Výsledná priemerná hodnota prenosových rýchlostí 16,4 Mbit/s. Viditeľná aj obrovská strátovosť, kedy prijímajúci obdržal len 27 % odosielaných datagramov.

Priemerná hodnota meraného parametra jitter nakoniec 0,982 milisekúnd.

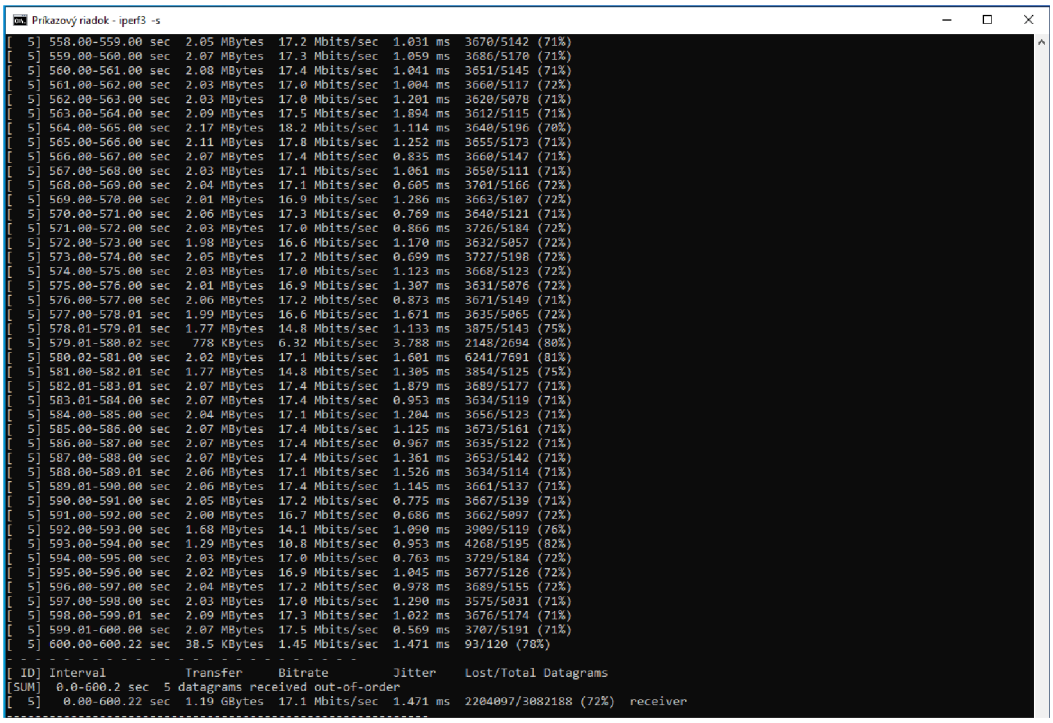
Pre hodnotu oneskorenia išlo v priemere o 3,782 milisekúnd.

4.8 Testovanie s použitím OpenVPN virtuálnej privátnej siete (RB941-2nD)



Obrázok 4.16 Zaťaženie procesora Mikrotik RB941-2nD, OpenVPN

Počas celého trvania testu na strane odosielajúceho neustále maximálne vyťaženie procesora, pričom až 65 % z toho tvoril proces šifrovania *encrypting*. Strana prijímajúceho taktiež zaznamenávala neustále maximálne hodnoty využitia procesora. Narozdiel od merania s použitím konfigurácie IPsec (podkapitola 4.7) sme sa však nestretli s nestabilitou Mikrotiku ani častými pádmi grafického uživateľského rozhrania.



Obrázok 4.17 Iperf3 výstup, OpenVPN, RB941-2nD

Na konci testovania sme obdržali výslednú priemernú hodnotu prenosových rýchlostí 17,1 Mbit/s na úkor až 72% strátovosti.

Meraný parameter jitter na konci testu zaznamenáva priemernú hodnotu 1,471 milisekúnd.

Výsledná priemerná hodnota pre oneskorenie – 4,613 milisekúnd.

5. POROVNANIE NAMERANÝCH HODNÔT

V rámci tejto kapitoly sú nami namerané priemerné hodnoty sledovaných parametrov reprezentované v tabuľkách a na ich základe je uskutočnené záverečné porovnanie technológií, ktorými sa práca zaoberala.

Tabulka 5.1 Tabuľka priemerných hodnôt – RB-493G

	jitter [ms]	oneskorenie [ms]	prenosová rýchlosť [Mbit/s]
Statické smerovanie	0,096	2,156	59,6
WireGuard	0,179	2,917	56,4
IPsec	0,789	3,492	18,5
OpenVPN	0,731	4,359	6,35

Tabulka 5.2 Tabuľka priemerných hodnôt – RB941-2nD

	jitter [ms]	oneskorenie [ms]	prenosová rýchlosť [Mbit/s]
Statické smerovanie	0,248	2,214	59,7
WireGuard	0,272	3,062	57,4
IPsec	0,982	3,782	16,4
OpenVPN	1,471	4,613	17,1

Na základe nami nameraných výsledkov pre oba scenáre môžeme usúdiť, že technológia WireGuard naozaj lepšie využíva poskytnuté parametre siete, ale aj hardvérové vybavenie sieťových prvkov narozdiel od IPsec a OpenVPN. Potvrdzuje sa tým predpoklad, ktorý mohol vyplývať z teoretickej časti práce aj samotného zámeru autorov WireGuard.

Na druhej strane sme testovaním zistili, že technológia OpenVPN nepredstavuje takmer žiadnu konkurenciu pre WireGuard.

Síce v testoch s použitím RB941-2nD môžeme vidieť len rozdiely v rádoch desiatok Mbit/s medzi prenosovými rýchlosťami OpenVPN a IPsec, pripisujeme takéto výsledky bottlenecku a neustálym problémom s prvkom Mikrotik.

6. ZÁVER

Cieľom bakalárskej práce bolo analyzovať technológiu WireGuard, ďalej ju porovnať s technológiami OpenVPN a IPsec a nakoniec uskutočniť testy, pomocou ktorých by boli otestované spomínané technológie a ich parametre (jitter, oneskorenie, prenosové rýchlosti).

V prvej časti práce sa nachádza opis virtuálnej privátnej siete WireGuard a jeho fungovanie.

V ďalšej časti je WireGuard porovnaný s technológiami IPsec a OpenVPN na základe nimi používaných šifrovacích algoritmov.

V nasledujúcej kapitole práca popisuje metodiku a kritériá testovania výkonu jednotlivých technológií s použitím konkrétneho hardvéru a odpovedajúcej konfigurácie topológie siete a taktiež predstavuje namerané výsledky pre konkrétne virtuálne privátne siete, prípadne scenár, kedy virtuálna privátna sieť nie je implementovaná.

Predposledná kapitola predstavuje namerané výsledky uskutočnených testov pre každú z virtuálnych privátnych sietí.

Posledná kapitola sa venuje porovnaniu nami nameraných výsledkov týkajúcich sa technických parametrov virtuálnych privátnych sietí, ktorými sú jitter, oneskorenie pri prenose a dosiahnuteľné prenosové rýchlosti, ktoré sú relevantné pre nami vopred zvolené parametre siete a kritériá testovania a taktiež použitý testovací hardvér a softvér.

LITERATURA

[1] *WireGuard: Next Generation Kernel Network Tunnel* [online]. 2020 [cit. 2022-11-18]. Dostupné z: https://www.wireguard.com/papers/wireguard.pdf

[2] *Curve25519: new Diffie-Hellman speed records* [online]. 2006 [cit. 2020-11-18]. Dostupné z: https://cr.yp.to/ecdh/curve25519-20060209.pdf

[3] *Internet Key Exchange Protocol Version 2 (IKEv2)*. 2010 [cit. 2022-11-18]. Dostupné z: https://www.rfc-editor.org/rfc/rfc5996.txt

[4] *Datagram Transport Layer Security Version 1.2*. [online]. 2012 [cit. 2022-11-18]. Dostupné z: https://www.rfc-editor.org/rfc/rfc6347.txt

[5] *The Noise Protocol Framework*. [online]. 2018 [cit. 2022-11-18]. Dostupné z: http://noiseprotocol.org/noise.pdf

[6] *Cryptographic Exctraction and Key Derivation*: *The HKDF Scheme*. [online]. 2010 [cit. 2022-11-18]. Dostupné z: https://eprint.iacr.org/2010/264.pdf

[7] *ChaCha, a variant of Salsa20*. [online]. 2008 [cit. 2022-11-20]. Dostupné z: https://cr.yp.to/chacha/chacha-20080128.pdf

[8] *ChaCha20 and Poly1305 for IETF Protocols*. [online]. 2015 [cit. 2022-11-20]. Dostupné z: https://www.rfc-editor.org/rfc/rfc7539.txt

[9] *BLAKE2: simpler, smaller, fast as MD5*. [online]. 2013 [cit. 2022-11-20]. Dostupné z: https://www.blake2.net/blake2.pdf

[10] *Mosh: An interactive Remote Shell for Mobile Clients*. [online]. 2012 [cit. 2022-11-23]. Dostupné z: https://mosh.org/mosh-paper.pdf

[11] LAUTER, Kristin a Anton MITYAGIN. Security Analysis of KEA Authenticated Key Exchange Protocol. In: YUNG, Moti, Yevgeniy DODIS, Aggelos KIAYIAS a Tal MALKIN, ed. *Public Key Cryptography - PKC 2006* [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, 2006, s. 378-394 [cit. 2022-11-27]. Lecture Notes in Computer Science. ISBN 978-3-540-33851-2. Dostupné z: doi:10.1007/11745853_25

[12] *TAI64, TAI64N and TAI64NA*. [online]. [cit. 2022-11-27]. Dostupné z: https://cr.yp.to/libtai/tai64.html

[13] DOBRAUNIG, Christoph, Florian MENDEL a Martin SCHLÄFFER. Differential Cryptanalysis of SipHash. In: JOUX, Antoine a Amr YOUSSEF, ed. *Selected Areas in Cryptography -- SAC 2014* [online]. Cham: Springer International Publishing, 2014, 2014-11-29, s. 165-182 [cit. 2022-12-4]. Lecture Notes in Computer Science. ISBN 978-3-319-13050-7. Dostupné z: doi:10.1007/978-3-319-13051-4_10

[14] RIVEST, R. L., A. SHAMIR a L. ADLEMAN. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM* [online]. 1978, **21**(2), 120-126 [cit. 2022-12-04]. ISSN 0001-0782. Dostupné z: doi:10.1145/359340.359342

[15] *Biclique Cryptanalysis of the Full AES.* [online]. [cit. 2022-12-04]. Dostupné z: https://web.archive.org/web/20160306104007/http://research.microsoft.com/en-us/projects/cryptanalysis/aesbc.pdf
[16] *More Modular Exponential Diffie-Hellman groups for Internet Key Exchange.* [online]. 2003 [cit. 2022-12-08]. Dostupné z: https://www.ietf.org/rfc/rfc3526.txt
[17] MERKLE, Ralph C. Secure communications over insecure channels. *Communications of the ACM* [online]. 1978, **21**(4), 294-299 [cit. 2022-12-12]. ISSN 0001-0782. Dostupné z: doi:10.1145/359460.359473
[18] *ECP Groups for IKE and IKEv2.* [online]. 2007 [cit. 2022-12-09]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc4753
[19] *Standards for Efficient Cryptography.* [online]. 2009 [cit. 2022-12-10]. Dostupné z: http://www.secg.org/sec1-v2.pdf
[20] *Digital Signature Standard (DSS).* [online]. 2013 [cit. 2022-12-10]. Dostupné z: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf
[21] *Authenticating remote peers and clients.* [online]. [cit. 2022-12-11]. Dostupné z: https://docs.fortinet.com/document/fortigate/6.0.0/handbook/530590/authenticating-remote-peers-and-clients
[22] *OpenSSL.* [online]. 2021 [cit. 2022-12-11]. Dostupné z: https://www.openssl.org
[23] *Extending the Salsa20 nonce.* [online]. 2011 [cit. 2022-12-10]. Dostupné z: https://cr.yp.to/snuffle/xsalsa-20110204.pdf
[24] [online]. [cit. 2023-05-22]. Dostupné z: https://iperf.fr
[25] *Https://wiki.mikrotik.com/wiki/Manual:IP/Fasttrack* [online]. [cit. 2023-05-20]. Dostupné z: https://wiki.mikrotik.com/wiki/Manual:IP/Fasttrack
[26] KIVINEN, T. a M. KOJO. *More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)* [online]. [cit. 2023-05-22]. Dostupné z: doi:DOI 10.17487/RFC3526
[27] PENARD, Wouter a Tim van WERKHOVEN. *On the Secure Hash Algorithm family* [online]. [cit. 2023-05-23]. Dostupné z: https://web.archive.org/web/20160330153520/https://www.staff.science.uu.nl/~werkh108/docs/study/Y5_07_08/infocry/project/Cryp08.pdf

SEZNAM SYMBOLŮ A ZKRATEK

Zkratky:

FEKT	Fakulta elektrotechniky a komunikačních technologií
VUT	Vysoké učení technické v Brně
AEAD	Authenticated Encryption with Associated Data
AKE	Authenticated Key Exchange
AES	Advanced Encryption Standard
DH	Diffie-Hellman
DTLS	Datagram Transport Layer Security
ECDH	Elliptic-curve Diffie-Hellman
ECDSA	Elliptic-curve Digital Signature Algorithm
HKDF	Hash-based Key Derivation Function
ICMP	Internet Control Message Protocol
IKEv2	Internet Key Exchange version 2
IP	Internet Protocol
KCI	Key Compromise Impersonation
KEA+C	Key Exchange Algorithm with key confirmation
MAC	Message Authentication Code
MB	Megabyte
MHz	Megahertz
NAT	Network Address Translation
PSK	Pre-shared Key
RAM	Random Access Memory
RSA	Rivest-Shamir-Adleman cryptosystem
RTT	Round Trip Time
SSH	Secure Shell
SSL	Secure Socket Layer
TAI	International Atomic Time
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
UTF-8	Unicode Transformation Format
VPN	Virtual Private Network