УНИВЕРЗИТЕТ У БЕОГРАДУ
МАТЕМАТИЧКИ ФАКУЛТЕТ

Александар Стоиљковић

ДИЗАЈН И ИМПЛЕМЕНТАЦИЈА VPN РЕШЕЊА СА АНАЛИЗОМ БЕЗБЕДНОСТИ И ПЕРФОРМАНСИ

мастер рад

Београд, 2026.

Ментор:

др Александар Картељ
Универзитет у Београду, Математички факултет

Чланови комисије:

др Славко Гајин
Универзитет у Београду, Математички факултет

др Мирко Спасић
Универзитет у Београду, Математички факултет

Датум одбране: _____

Наслов мастер рада: Дизајн и имплементација VPN решења са анализом безбедности и перформанси

Резиме: Овај рад приказује пројектовање, имплементацију и експерименталну евалуацију истраживачког система виртуелне приватне мреже. Систем је реализован у програмском језику Python над Linux TUN интерфејсом и UDP транспортом. Поверљивост и интегритет саобраћаја обезбеђени су алгоритмом AES-256-GCM, док се сесијски кључеви успостављају аутентификованом разменом заснованом на X25519 и функцији HKDF. Реализација обухвата заштиту од поновног слања пакета, ротацију кључева, одржавање везе, аутоматско поновно повезивање, рад са више клијената и графички кориснички интерфејс.

Исправност и безбедносна својства проверени су јединичним, интеграционим и мрежним тестовима. Перформансе су испитане у реалном окружењу у облаку и упоређене са директном везом, нешифрованом варијантом истог система, WireGuard-ом и OpenVPN *Community Edition* системом. Спроведено је шест уравнотежених рандомизованих кругова са укупно 30 мерења по режиму. Резултати показују да безбедносни слој доноси мерљив трошак, нарочито у TCP пропусности, процесорском оптерећењу и губитку UDP пакета, али и да прототип остварује TCP пропусност упоредиву са OpenVPN-ом. Анализа истовремено показује предност WireGuard-а и издваја ограничења корисничке Python имплементације.

Кључне речи: виртуелна приватна мрежа, Linux TUN, UDP, AES-256-GCM, X25519, HKDF, WireGuard, OpenVPN, мрежна безбедност, анализа перформанси

Садржај

Глава 1

Увод

Савремени информациони системи све чешће повезују кориснике и услуге који нису у истој физичкој мрежи. Рад на даљину, инфраструктура у облаку и географски распоређени сервиси захтевају да се саобраћај преноси преко мрежа којима стране у комуникацији не управљају. Виртуелна приватна мрежа (енгл. *Virtual Private Network*, VPN) решава овај проблем успостављањем логичког тунела кроз јавну мрежу. Унутрашњи IP пакети преносе се као садржај спољашњих пакета, а криптографски механизми могу да обезбеде поверљивост, интегритет и аутентичност саобраћаја.

Иако се у пракси користе зрела решења као што су IPsec, OpenVPN и WireGuard, израда мањег система од основних компонената има истраживачку и образовну вредност. Она омогућава да се истовремено размотре рад виртуелног мрежног интерфејса, бинарни формат протокола, успостављање сесијских кључева, заштита од поновљених пакета, управљање везом и цена криптографске обраде. Посебно је значајно питање колико реализација у корисничком простору, написана у језику Python, може да се приближи системима чији је критични пут реализован у компајлираном језику или у језгру оперативног система.

Предмет овог рада је пројектовање, имплементација и евалуација сопственог VPN система за Linux. Систем користи TUN интерфејс за размену IPv4 пакета са мрежним стеком оперативног система и UDP као спољашњи транспорт. Заштита података заснована је на аутентификованом шифровању AES-256-GCM, док се сесијска тајна успоставља аутентификованом ефемерном разменом X25519 и изводи функцијом HKDF. Реализација подржава усмерене кључеве, клизни прозор за заштиту од поновног слања,

ротацију кључева, поруке за одржавање везе, аутоматско поновно повезивање и рад сервера са више клијената.

1.1 Циљеви и истраживачка питања

Основни циљ је да се изгради функционалан и проверљив истраживачки прототип, а затим да се његове безбедносне и перформансне особине испитају у реалном мрежном окружењу. Постављена су следећа истраживачка питања:

1. Колики трошак уноси Python тунел у односу на директну мрежну везу?
2. Колики додатни трошак у истој реализацији потиче од шифровања, аутентификације и управљања сесијским кључевима?
3. Какве перформансе прототип остварује у поређењу са системима WireGuard и OpenVPN *Community Edition*?
4. Која безбедносна својства су остварена, а који ризици остају због истраживачке природе система?

Поређење је организовано тако да раздвоји трошак обраде у Python-у од трошка безбедносног слоја. Зато су, поред директне везе и три шифрована VPN режима, мерења обављена и над нешифрованом варијантом истог Python тунела.

Изворни код, тестови и скрипте за експерименте доступни су у јавном репозиторијуму [13]. Систем је истраживачки прототип и није представљен као замена за независно проверена продукциона VPN решења.

1.2 Организација рада

Мрежне и криптографске основе и системи WireGuard и OpenVPN описани су у глави 2. Архитектура, формат пакета, размена кључева и програмски модули приказани су у глави 3. Поступци провере и безбедносна анализа дати су у глави 4. Поставка, ток и резултати мерења обједињени су у глави 5. Завршна разматрања и правци даљег рада налазе се у глави 6.

Глава 2

Теоријске основе и постојећа VPN решења

2.1 Виртуелне приватне мреже и тунеловање

Тунеловање је поступак енкапсулације пакета једног протокола у пакет другог протокола. У посматраном систему унутрашњи пакет је IPv4 датаграм, док је спољашњи пакет UDP датаграм који се преноси јавном мрежом. На страни пошиљаоца тунел преузима унутрашњи пакет од оперативног система, додаје сопствено протоколско заглавље, по потреби га шифрује и шаље удаљеној страни. На страни примаоца поступак се обавља обрнутим редоследом.

VPN не мора нужно да преноси сав саобраћај уређаја. Код приватног тунела само руте за изабране мреже усмеравају се кроз виртуелни интерфејс. Код потпуног тунела подразумевана рута пролази кроз VPN, уз посебну руту ка јавној адреси самог VPN сервера како не би настала рекурзија. Реализовани систем подржава оба начина рада за IPv4.

2.2 Linux TUN интерфејс

Linux TUN/TAP управљачки програм омогућава програмима у корисничком простору да примају и шаљу мрежне пакете [5]. TUN представља интерфејс трећег слоја и размењује IP пакете, док TAP ради са Ethernet оквирима. Програм отвара уређај `/dev/net/tun`, а затим позивом `ioctl(TUNSETIFF)` региструје интерфејс. У овом раду користе се заставе

(енгл. *flags*) `IFF_TUN` и `IFF_NO_PI`, па сваки прочитани бајтни низ почиње непосредно IP заглављем.

Избор TUN-а поједностављује систем: није потребно обрађивати Ethernet адресе, протокол за разрешавање адреса (енгл. *Address Resolution Protocol*, ARP) нити широкодифузне оквире. Са друге стране, програм мора имати право `CAP_NET_ADMIN` за креирање и подешавање интерфејса и рута. То је важна оперативна и безбедносна последица, јер грешка у процесу који ради са повишеним привилегијама има већи утицај.

2.3 UDP као спољашњи транспорт

UDP обезбеђује датаграмску комуникацију без успостављања транспортне везе [12]. Не гарантује испоруку, редослед нити јединственост пакета. Ове особине одговарају VPN тунелу јер се губици не прикривају додатним слојем поновног слања. Употреба TCP-а као спољашњег транспорта за унутрашњи TCP саобраћај може довести до међусобног утицаја два механизма контроле загушења.

Недостатак UDP-а је то што протокол апликације мора сам да обради промену редоследа, дупликате, проверу доступности друге стране и ограничење величине датаграма. У реализацији су зато уведени 64-битни секвенцијски број, клизни прозор за поновљене пакете и поруке `KEEPALIVE`. Максимална теоријска величина UDP садржаја за IPv4 износи 65.507 бајтова, али практична величина унутрашњег пакета зависи од максималне јединице преноса (енгл. *Maximum Transmission Unit*, MTU) путање.

2.4 Аутентификовано шифровање

Само шифровање није довољно за безбедан канал. Нападач који не може да прочита поруку и даље може да је измени, понови или замени другом. Аутентификовано шифровање са придруженим подацима (енгл. *Authenticated Encryption with Associated Data*, AEAD) истовремено обезбеђује поверљивост садржаја и проверу интегритета садржаја и придружених, нешифрованих података [9].

AES-GCM

Режим Галоа и бројача (енгл. *Galois/Counter Mode*) јесте AEAD режим стандардизован у NIST SP 800-38D [3]. У режиму бројача блоковска шифра AES производи ток који се комбинује са отвореним текстом, док се аутентификациона ознака израчунава над шифрованим текстом и придруженим подацима. У раду се користи AES са кључем од 256 бита и GCM аутентификационом ознаком од 128 бита.

За фиксирани кључ једнократна вредност (енгл. *nonce*) мора бити јединствена [9]. Њена поновна употреба у GCM-у може да открије однос између отворених текстова и угрози аутентификацију. У пројектованом протоколу једнократна вредност има 96 бита и гради се као

$$N = \text{uint32}(\textit{session\_id}) \,\|\, \text{uint64}(\textit{sequence\_number}), \tag{2.1}$$

где $\|$ означава надовезивање у мрежном редоследу бајтова. У оквиру једне сесије секвенцијски број монотоно расте, а смер комуникације користи посебан кључ. Тако је пар кључ–једнократна вредност јединствен док не дође до прекорачења 64-битног бројача.

2.5 HMAC и извођење кључева

HMAC је код за аутентификацију поруке заснован на криптографској хеш функцији и тајном кључу [6]. У поступку размене кључева HMAC-SHA256 аутентификује ефемерне јавне кључеве, случајне вредности и идентификатор сесије помоћу тајне добијене из статичких X25519 кључева.

HKDF је функција за извођење кључева по обрасцу издвајања и проширења (енгл. *extract-then-expand*) [7]. Фаза издвајања претвара улазни материјал у псеудослучајни кључ,

$$PRK = \text{HMAC}(\textit{salt}, \textit{IKM}), \tag{2.2}$$

док фаза проширења изводи потребан број бајтова и везује их за контекст протокола кроз параметар *info*. Контекст раздваја употребе исте улазне тајне. У систему се HKDF-SHA256 користи за сесијску тајну, два усмерена кључа саобраћаја и кључеве по епохама ротације.

2.6 X25519 и ефемерна размена кључева

X25519 је функција за размену кључева над Montgomery обликом криве Curve25519, дефинисана у RFC 7748 [8]. Приватни и јавни улази и излази имају по 32 бајта. Две стране са приватним вредностима a и b и јавним вредностима A и B израчунавају исту дељену тајну:

$$X25519(a, B) = X25519(b, A). \tag{2.3}$$

Статички кључеви омогућавају препознавање познатог учесника, али сами не обезбеђују тајност унапред (енгл. *forward secrecy*). Ако би један дугорочни кључ накнадно био откривен, снимљене старе сесије могле би бити угрожене. Зато свака сесија ствара нов пар ефемерних кључева. Сесијска тајна зависи од ефемерно-ефемерне, ефемерно-статичке, статичко-ефемерне и статичко-статичке X25519 вредности. Откривање само дугорочног кључа после завршетка сесије није довољно за реконструкцију ефемерно-ефемерне тајне.

2.7 Заштита од поновног слања и ротација кључева

Важећи AEAD пакет може бити злонамерно послат више пута. Зато пријемник памти највећи аутентификовани секвенцијски број и битску мапу претходних бројева. Пакет унутар прозора прихвата се само ако његов бит још није постављен. Пакет старији од прозора одбацује се, док пакет са већим бројем помера прозор. Стање се мења тек након успешне провере GCM аутентификационе ознаке, чиме неаутентификовани пакет не може да помери прозор и изазове одбацивање исправних пакета.

Ротација кључева ограничава количину саобраћаја обрађену једним кључем. За интервал од P пакета епоха пакета са бројем s је

$$e = \left\lfloor \frac{s}{P} \right\rfloor. \tag{2.4}$$

Кључ епохе изводи се HKDF-ом из усмерене тајне, идентификатора сесије и броја епохе. Ова ротација није замена за нову размену кључева: ако је активна сесијска тајна откривена, из ње се могу извести све епохе исте сесије.

2.8 Постојећа VPN решења

За поређење са прототипом изабрани су WireGuard и OpenVPN *Community Edition*. Они представљају два различита приступа: WireGuard има мали, строго одређен протокол и обраду пакета у језгру Linux-а, док је OpenVPN дугогодишње, широко подесиво решење у корисничком простору.

WireGuard

WireGuard је пројектован као једноставан и високоперформантан мрежни тунел [2]. На Linux-у је интегрисан у језгро, па се пакети не пребацују за сваки корак између језгра и Python процеса као у систему овог рада. WireGuard користи UDP, размену кључева засновану на обрасцу NoiseIK, Curve25519 за размену кључева, ChaCha20-Poly1305 за AEAD, BLAKE2s за хеширање и HKDF за извођење кључева [1].

Протокол аутоматски обнавља размену и сесијске кључеве на основу времена и броја порука. За сваког учесника одржава стање, ред пакета и дозвољене IP адресе. Дозвољене адресе истовремено имају улогу табеле усмеравања при слању и листе контроле приступа при пријему. WireGuard садржи и механизам потврдних колачића за ограничавање трошка неаутентификованих захтева за размену кључева под оптерећењем.

У односу на прототип, важне разлике су место извршавања, избор AEAD алгоритма, знатно већи прозор за заштиту од понављања и временски засновано поновно успостављање кључева. Зато поређење не изолује само криптографски алгоритам, већ целокупну архитектуру система.

OpenVPN *Community Edition*

OpenVPN је VPN систем у корисничком простору који користи TUN или TAP и може да ради преко UDP-а или TCP-а [11]. За управљачки канал и аутентификацију ослања се на безбедност транспортног слоја (енгл. *Transport Layer Security*, TLS), док канал података може користити различите симетричне алгоритме. У експерименту је употребљен OpenVPN *Community Edition* преко UDP-а са AES-256-GCM, како би избор алгоритма за заштиту података био што ближи прототипу.

OpenVPN има шири скуп могућности од система развијеног у овом раду: сертификате и инфраструктуру јавних кључева, богата правила подешавања, компатибилност са више оперативних система и дугогодишње продукционо искуство. Та флексибилност доноси сложенију конфигурацију и већи кодни обим. Пошто и OpenVPN и прототип обрађују пакете у корисничком простору, њихово поређење је посебно корисно, али се резултат не сме тумачити као директно мерење ефикасности AES-GCM-а: разликују се језик, I/O модел, формат пакета, TLS контролни канал и степен оптимизације.

Глава 3

Пројектовање и имплементација

Систем је организован као скуп малих модула са јасно раздвојеним одговорностима. TUN модул повезује процес са Linux мрежним стеком, транспортни модул управља UDP сокетом, протоколски модул кодира бинарне поруке, а криптографски слој обавља извођење кључева, шифровање и проверу поновљених пакета. `TunnelEngine` повезује ове компоненте у петљу за пренос пакета.

3.1 Архитектура тока података

На страни пошиљаоца језгро усмерава IPv4 пакет ка TUN интерфејсу. Процес чита пакет, формира протоколско заглавље и шифрује садржај. Заглавље је придружени аутентификовани податак, али остаје видљиво ради избора сесије и провере дужине. Добијени датаграм шаље се UDP-ом. Пријемник проверава спољашњи формат, проналази сесију, проверава стање заштите од понављања и GCM ознаку, а затим уписује изворни IP пакет у TUN интерфејс. Поступак се може свести на следеће кораке:

1. при слању, језгро предаје IP пакет TUN интерфејсу.
2. процес додаје протоколско заглавље и шифрује садржај.
3. шифровани датаграм се преноси UDP-ом до друге крајње тачке.
4. пријемник проверава заглавље, редни број и ознаку аутентичности.
5. тек исправно проверен IP пакет уписује се у пријемни TUN интерфејс.

Суштина овог тока је да TUN само размењује IP пакете са оперативним системом, док кориснички процес обавља целокупну протоколску енкапсулацију, заштиту и UDP пренос. Ниједан примљени пакет не враћа се у мрежни стек пре успешне провере.

Исти интерфејси користе се у шифрованом и нешифрованом режиму. То је важно за експеримент: промена режима не мења TUN, UDP, петљу догађаја и поступак мерења, па разлика боље представља цену безбедносног слоја.

3.2 Формат протоколске поруке

Свака порука почиње заглављем од 16 бајтова, кодираним у мрежном редоследу. Сва бројчана поља су неозначени цели бројеви, а њихов распоред приказан је у табели 3.1.

Табела 3.1: Бинарни формат протоколског заглавља

Поље	Бајтови	Намена
`version`	1	верзија протокола, тренутно 1
`message_type`	1	врста поруке: `DATA`, `KEEPALIVE`, `CLIENT_HELLO` или `SERVER_HELLO`
`session_id`	4	идентификатор криптографске сесије
`sequence_number`	8	редни број за једнократну вредност и заштиту од поновљених пакета
`payload_length`	2	дужина садржаја који следи после заглавља

Декодер одбацује непознату верзију, непознат тип, прекратак датаграм и неслагање декларисане и стварне дужине. На тај начин неисправан улаз не стиже до сложеније логике. За поруке `DATA` и `KEEPALIVE` секвенцијски број расте од нуле, док поруке за размену кључева користе нулу пре успостављања канала за пренос података.

Одвојено поље верзије омогућава да пријемник недвосмислено одбаци формат који не разуме, уместо да исти низ бајтова протумачи по правилима друге верзије. Дужина садржаја није неопходна UDP-у, јер датаграм већ чува границу поруке, али представља додатну проверу формата пре криптографске обраде. Она такође улази у додатне аутентификоване податке (енгл. *Additional Authenticated Data*, AAD), па нападач не може да је промени без неуспешне провере GCM аутентификационе ознаке. Максимална вредност поља дужине је 65.535, док је практичан максимум мањи због UDP

и протоколског заглавља. Реализација зато проверава границу од 65.507 бајтова за цео UDP садржај.

Енкапсулација повећава сваки пакет података за 16 бајтова сопственог заглавља и 16 бајтова GCM аутентификационе ознаке, поред спољашњих IP и UDP заглавља. За IPv4 спољашњу путању додатни трошак износи 60 бајтова. Ако је физички MTU 1500 бајтова, унутрашњи MTU мора бити довољно мањи да шифровани датаграм не захтева фрагментацију. У експерименту је MTU подешаван конфигурацијом, али систем не аутоматски открива MTU путање. То остаје једно од ограничења прототипа.

3.3 Аутентификована размена кључева

Обе стране поседују дугорочни X25519 приватни кључ и унапред знају јавни кључ друге стране. Клијент за сваку везу генерише случајни 32-битни идентификатор сесије, ефемерни X25519 пар и 32 случајна бајта. Тело поруке `CLIENT_HELLO` је

$$C = E_C \,\|\, R_C, \tag{3.1}$$

где су E_C и R_C дужине 32 бајта. Статичка дељена тајна је $K_{SS} = X25519(S_C, S_S^{pub})$. Клијент додаје HMAC аутентификациону ознаку

$$T_C = \mathrm{HMAC}_{K_{SS}}(context \| client \| session\_id \| C). \tag{3.2}$$

Сервер најпре проверава ознаку. Тек затим генерише свој ефемерни пар и случајну вредност. Његова ознака покрива идентификатор сесије, целу клијентску поруку и тело серверске поруке:

$$T_S = \mathrm{HMAC}_{K_{SS}}(context \| server \| session\_id \| C \| T_C \| E_S \| R_S). \tag{3.3}$$

Обе HELLO поруке имају садржај од 96 бајтова: 32 бајта јавног ефемерног кључа, 32 случајна бајта и 32 бајта HMAC-SHA256 аутентификационе ознаке. Након узајамне провере стране располажу четирима X25519 вредностима:

$$K_{EE} = X25519(E_C, E_S), \qquad K_{ES} = X25519(E_C, S_S), \tag{3.4}$$

$$K_{SE} = X25519(S_C, E_S), \qquad K_{SS} = X25519(S_C, S_S). \tag{3.5}$$

Хеш целокупне размене користи се као HKDF додатак (енгл. *salt*), а надовезивање четири вредности као улазни кључни материјал. Резултат је

32-бајтна сесијска тајна:

$$K_{session} = \text{HKDF-SHA256}(K_{EE} \| K_{ES} \| K_{SE} \| K_{SS}, \text{SHA256}(\textit{transcript}), \textit{context}). \tag{3.6}$$

Укључивање транскрипта везује кључ за конкретну размену, а ефемерно-ефемерна компонента обезбеђује тајност старих сесија након каснијег откривања дугорочног кључа, под претпоставком да су ефемерни приватни кључеви уништени.

Редослед успостављања сесије је следећи:

1. клијент генерише идентификатор сесије, ефемерни кључ E_C и случајну вредност R_C, а затим шаље `CLIENT_HELLO`.
2. сервер проверава клијентску HMAC ознаку. Ако је исправна, генерише E_S и R_S и шаље `SERVER_HELLO`.
3. клијент проверава серверску HMAC ознаку.
4. обе стране из исте четири X25519 вредности и истог садржаја размене изводе једнаку сесијску тајну, а затим засебне кључеве за сваки смер.

Суштина размене је у томе да статички кључеви потврђују идентитет друге стране, а нови ефемерни кључеви обезбеђују свежу тајну за сваку сесију. Ако било која ознака није исправна, канал за пренос података се не успоставља.

Размена кључева намерно садржи само две мрежне поруке како би успостављање било кратко на путањама са већим кашњењем. Сервер не прихвата поруку података док не успостави стање сесије, а клијент не прелази у повезано стање пре провере серверске аутентификационе ознаке. Пошто статички јавни кључеви морају бити унапред подешени, протокол не решава дистрибуцију поверења. Ова одлука поједностављује истраживачки систем, али код већег броја корисника захтевала би засебан управљачки механизам за издавање, опозив и замену кључева.

3.4 Канал за пренос података

Из сесијске тајне HKDF изводи независне кључеве $K_{C\to S}$ и $K_{S\to C}$. Ово раздвајање спречава да исти пар кључа и једнократне вредности настане

у оба смера. При шифровању се најпре формира заглавље са дужином шифрованог садржаја, а његових 16 бајтова прослеђују се GCM-у као AAD. Промена типа, сесије, секвенцијског броја или дужине зато доводи до неуспешне аутентификације.

Провера поновљених пакета изводи се у две фазе. Брза провера утврђује да ли је број већ виђен или превише стар, али битска мапа се ажурира тек после успешне GCM провере. Подразумевани прозор садржи 64 секвенцијска броја и дозвољава умерену промену редоследа карактеристичну за UDP.

Опциона ротација дели саобраћај на епохе од подесивог броја пакета. Пријемник може да задржи шифре за две последње епохе, што омогућава обраду пакета који су стигли ван редоследа преко границе епохе.

3.5 Одржавање везе и поновно повезивање

Када нема корисничког саобраћаја, страна периодично шаље шифровану поруку `KEEPALIVE`. Сваки успешно аутентификован пакет освежава време последњег пријема. Ако унутар подесивог интервала провере везе нема таквог пакета, преносни механизам пријављује истек времена. Клијентски надзорни слој тада затвара старе ресурсе, обавља нову размену кључева и поново покреће пренос пакета. Одржавање и провера везе морају бити укључени заједно како подешавање не би стварало једностране и неочекивану семантику.

3.6 Сервер са више клијената

Регистар клијената садржи име, додељену адресу у тунелу и статички јавни кључ. Сесија се идентификује паром спољашње UDP адресе и идентификатора сесије, а додатна мапа повезује унутрашњу IPv4 адресу са активном сесијом. Пакет са TUN интерфејса усмерава се клијенту на основу одредишне адресе.

Након дешифровања пакета примљеног од клијента сервер проверава да ли је унутрашња изворна IPv4 адреса једнака адреси додељеној тој сесији. Пакет са лажираном адресом се одбацује. Нова успешно аутентификована размена кључева замењује стару сесију исте тунелске адресе, што омогућава поновно повезивање са друге спољашње адресе.

Сервер чува до 1024 отиска већ обрађених `CLIENT_HELLO` порука и њихове одговоре. Поновљени идентичан захтев добија исти одговор без замене активне сесије. Пошто се чува само 1024 најновија отиска, доласком нових записа најстарији се уклањају. Нападач зато може великим бројем захтева да уклони траг старијег захтева и поново га пошаље, па овај механизам није потпуна заштита од ускраћивања услуге.

3.7 Конфигурација, бележење и графички клијент

JSON конфигурација се учитава у типизиране структуре и строго проверава. Проверавају се TUN адресни опсег у бескласном међудоменском запису (енгл. *Classless Inter-Domain Routing*, CIDR), MTU, идентификатор сесије, адресе, временски интервали, ротација и међусобни односи опција. Непозната поља и неважеће вредности доводе до грешке пре измене мрежног стања.

Догађаји се бележе као структурирани JSON записи, што омогућава да командна линија, графички клијент и експерименталне скрипте читају исте сигнале стања. PyQt клијент управља процесом крајње тачке, приказује фазу повезивања и активни режим AES-256-GCM са ефемерном X25519 разменом кључева. Криптографска логика није дуплирана у графичком клијенту, већ остаје у истим модулима које користи командна линија.

3.8 Границе модула и обрада грешака

Протоколски и криптографски слојеви не мењају мрежну конфигурацију. Они примају и враћају низове бајтова и при неважећем улазу подижу контролисану грешку. TUN и UDP ресурси су издвојени из ове логике, па се могу заменити тестним реализацијама. Оваква подела је омогућила да криптографске инваријанте буду проверене без административних привилегија и без зависности од стања локалних рута.

Петља за пренос разликује очекиване неважеће датаграме од грешака које захтевају прекид сесије. Пакет непознате верзије, погрешне дужине, неважећег аутентификационе ознаке или поновљеног бројача одбацује се без

уписа у TUN. Прекорачење дозвољеног интервала без исправно примљеног пакета, затварање системског дескриптора и прекорачење секвенцијског броја прекидају активну сесију. Клијентски надзор може затим да покрене нову размену кључева, док сервер наставља да опслужује остале клијенте.

Структурирани догађаји садрже тип, време, фазу и опис грешке, али не садрже кључни материјал нити отворени садржај пакета. Такво бележење је важно и за графички клијент и за експерименте: аутоматизација може да разликује успешно повезивање, контролисано одбацивање пакета и неочекиван прекид процеса без анализирања неструктурираног терминалског текста.

Глава 4

Верификација и безбедносна анализа

Провера система обухвата више нивоа: јединичне тестове (енгл. *unit tests*) чисте логике, интеграционе тестове компоненти, UDP комуникацију преко повратног интерфејса (енгл. *loopback*) и привилеговане тестове са стварним Linux мрежним просторима и TUN интерфејсима. Безбедносна анализа полази од нападача који контролише мрежу, али нема приватне кључеве легитимних крајњих тачака.

4.1 Аутоматизовани тестови

Основни скуп садржи 101 тест покренут стандардним Python модулом `unittest`. Тестови протокола проверавају кодирање и декодирање, непознате верзије, неважеће типове и неслагање дужине. Криптографски тестови проверавају усмерене кључеве, промену AAD-а, неважећу GCM аутентификациону ознаку, јединственост једнократне вредности, границе прозора за заштиту од понављања, пакете ван редоследа и прелаз између епоха ротације.

Тестови размене кључева проверавају да клијент и сервер изводе исту сесијску тајну, да измена било ког аутентификованог дела доводи до одбацивања и да се размена може обавити стварном UDP комуникацијом преко повратног интерфејса. Тестови преносног механизма користе контролисане замене за TUN, UDP и `select`, чиме се испитују путање слања, пријема, одржавања везе, откривања прекида и прекорачења бројача.

Посебни тестови покривају регистар више клијената, замену сесије, усмеравање према тунелској адреси и одбацивање пакета са лажираном унутрашњом изворном адресом. Остали тестови обухватају JSON конфигурацију, командну линију, графички клијент, мерење ресурса и прикупљање резултата експеримената.

Тестни скуп је организован у 22 модула. Његова расподела по деловима система приказана је у табели 4.1. Број тестова није сам по себи доказ исправности, али показује да провера није усмерена само на успешан ток шифровања. Више од половине тестова покрива управљање стањем, неважеће улазе, рад са више клијената, мрежну конфигурацију и алате за мерење.

Табела 4.1: Расподела аутоматизованих тестова по областима

Област	Број	Примери проверених својстава
Протокол и криптографија	28	формат поруке, размена кључева, измена шифрованог садржаја, усмерени кључеви, прозор и ротација
Преносни и мрежни слој	19	TUN и UDP, слање и пријем, истек везе, неисправни датаграми и мрежно подешавање
Конфигурација и рад система	33	провера улаза, дозволе кључева, више клијената, лажирање адресе, командна линија и графички клијент
Мерење и обрада резултата	21	рашчлањивање излаза алата, поновљени покушаји, распоред рунди, спајање блокова и ресурси
Укупно	101	22 тестна модула

Посебно су провераване негативне путање. Тестови мењају шифровану поруку и свако аутентификовано поље заглавља, шаљу пакет у погрешном смеру, понављају већ прихваћен број и испитују пакет старији од прозора. Проверавају се и прекорачење дозвољене дужине, непозната верзија, погрешна величина кључа, небезбедне дозволе датотеке и недозвољена адреса клијента. У овим случајевима критеријум успеха није само појава грешке. Проверава се да пакет није уписан у TUN, да неуспела аутентификација није потрошила секвенцијски број и да остале сесије настављају са радом.

4.2 Интеграциони тестови мрежног пута

Јединични тест не може да докаже да правила усмеравања у Linux-у, TUN уређај и UDP процеси раде заједно. Зато су направљене три привилеговане интеграционе скрипте. Прва успоставља тунел типа тачка–тачка (енгл. *point-to-point*) између мрежних простора и проверава протокол контролних порука Интернета (енгл. *Internet Control Message Protocol*, ICMP) кроз стварни TUN у језгру. Друга покреће сервер са више клијената и проверава изолацију адреса и рутирање. Трећа проверава пут који покреће графички клијент.

Мрежни простори обезбеђују поновљиву локалну топологију без зависности од јавног Интернета. Ипак, они не замењују проверу у реалној инфраструктури, па су приватни и потпуни тунел додатно проверени на Azure серверу пре завршне експерименталне кампање.

Први привилеговани сценарио покреће клијента пре сервера да би намерно изазвао неуспелу размену кључева. После покретања сервера проверава успостављање TUN интерфејса и ICMP саобраћај, затим зауставља сервер, чека да клијент открије прекид и поново покреће сервер. Сценарио је успешан тек када се успостави нова сесија и ICMP поново прође кроз тунел. Тиме се у једном извршавању проверавају почетно повезивање, истек провере везе, бележење догађаја и опоравак.

Други сценарио ствара три мрежна простора и две независне основне везе до серверског простора. Сервер учитава два различита јавна кључа и две тунелске адресе. Оба клијента морају засебно да успоставе сесију и размене ICMP пакете са сервером, што проверава регистар клијената, избор сесије и усмеравање према одредишној адреси.

Трећи сценарио не заобилази графички клијент. Он у изолованом мрежном простору покреће PyQt процес, бира профил, повезује се, пропушта ICMP саобраћај и затим прекида везу. Поред мрежног резултата проверава се редослед стања које графички клијент прима из структурираних догађаја. Тако је обухваћен цео пут од корисничке радње до стварног TUN пакета, а не само приказ елемената интерфејса.

Локални мрежни сценарији и Azure провера имају различите улоге. Мрежни простори дају контролисан и поновљив услов за откривање функционалних грешака. Azure провера потврђује да исто решење ради са стварним рутирањем, превођењем мрежних адреса (енгл. *Network Address*

Translation, NAT) и јавном мрежом. Тек после обе врсте провере покренута је завршна кампања перформанси.

4.3 Модел претњи

Предмет заштите су поверљивост и интегритет IPv4 пакета, идентитет подешених учесника, прихватање само свежег саобраћаја, раздвајање клијената и доступност крајњих тачака. Нападач може да чита, мења, убацује, брише, понавља и мења редослед UDP датаграма. Може да шаље произвољне неаутентификоване захтеве серверу. Не претпоставља се да нападач већ контролише крајњу тачку, чита њену меморију или поседује дугорочни приватни кључ. Главне претње, примењене контроле и начини њихове провере приказани су у табели 4.2.

Табела 4.2: Главне претње и примењене контроле

Претња	Контрола	Начин провере
Прислушкивање	AES-256-GCM над унутрашњим пакетом	тест да шифрована порука не открива отворени текст
Измена пакета	GCM аутентификациона ознака и заглавље као AAD	измена сваког поља и очекивано одбацивање
Понављање	64-битни број и клизни прозор	дупликати, стари и ванредни пакети
Лажно представљање учесника	HMAC размене из статичке X25519 тајне	погрешан кључ и измењен садржај размене
Лажирање адресе клијента	веза сесије и додељене IPv4 адресе	пакет са туђом изворном адресом
Губитак учесника	одржавање везе, истек провере и поновно повезивање	симулиран престанак пријема

4.4 Остварена безбедносна својства

Под наведеним претпоставкама, AES-GCM обезбеђује поверљивост и интегритет канала за пренос података, а аутентификација заглавља спречава непримећену промену сесије, типа и бројача. Статички X25519 материјал омогућава узајамну проверу подешених учесника. Ефемерни кључеви и

укључивање K_{EE} у HKDF дају тајност старих сесија у односу на накнадно откривање само статичког кључа. Усмерени кључеви и монотони бројачи обезбеђују јединственост GCM једнократних вредности у предвиђеном веку сесије. Прозор за заштиту од понављања прихвата умерено промењен редослед без прихватања истог аутентификованог пакета два пута.

Ова својства зависе од исправности библиотеке `cryptography`, извора случајности оперативног система, чувања приватних кључева и одсуства компромитације крајње тачке. Тестови показују очекивано понашање реализације, али нису формални доказ протокола.

4.5 Преостали ризици и ограничења

Најважнија ограничења су следећа:

- систем није прошао независну безбедносну ревизију нити формалну верификацију.
- процес ради са повишеним привилегијама потребним за TUN и руте. продукциона варијанта би издвојила подешавање интерфејса и затим одбацила непотребне привилегије.
- размена кључева нема ограничење учесталости нити механизам потврдних колачића, па велики број неважећих HELLO порука може да троши процесорско време.
- чува се само 1024 најновија отиска размене кључева. Велики број нових захтева уклања најстарије записе и тиме поново омогућава обраду раније виђеног захтева.
- случајни 32-битни идентификатор сесије има ненулту вероватноћу судара.
- ротација по броју пакета не обезбеђује сигурност после компромитовања (енгл. *post-compromise security*) унутар исте сесије, јер се све епохе изводе из исте тајне.
- прозор за заштиту од понављања од 64 пакета може одбацити исправан пакет при изузетно великој промени редоследа.

- нису реализовани контрола загушења, откривање MTU-а путање, стратегија фрагментације нити IPv6 пренос пакета.
- спољашње IP адресе, UDP портови, величине и времена пакета остају видљиви посматрачу мреже.

Стога резултат треба посматрати као истраживачку реализацију погодну за анализу протокола и перформанси. За продукциону употребу били би неопходни тестирање насумичним улазима, раздвајање привилегија, ограничење саобраћаја при размени кључева, независна ревизија и редовно ажурирање зависности.

Глава 5

Експерименти

Евалуација је осмишљена тако да одговори на истраживачка питања из главе 1, уз контролу временске варијације јавне мреже и виртуелне машине са дељеним процесорским ресурсима. Сви режими испитани су између истог физичког Linux клијента и истог Azure сервера у региону *West Europe*.

5.1 Поређени режими

Испитано је пет режима:

1. ***Direct*** – директна јавна путања до Azure сервера.
2. ***Plaintext Python*** – исти TUN/UDP и Python пренос пакета без криптографске заштите.
3. ***Encrypted Python*** – комплетан систем са шифрованим каналом за пренос података и аутентификованом X25519 разменом кључева.
4. **WireGuard** – референтни VPN у језгру.
5. **OpenVPN** – OpenVPN *Community Edition* преко UDP-а са AES-256-GCM каналом за пренос података.

Режим *Direct* показује основне особине мрежне путање. Разлика *Plaintext–Direct* укључује цену TUN/UDP/Python обраде, док разлика *Encrypted–Plaintext* боље изолује цену безбедносних механизама. WireGuard и OpenVPN дају практичне референтне тачке, али због различитих архитектура нису контролисани микроекперименти једног алгоритма.

5.2 Окружење

Сервер је био Azure `Standard_B2s` са Ubuntu 24.04 LTS, а клијент физички Linux рачунар. B серија користи модел процесорских кредита: виртуелна машина акумулира кредите испод основног оптерећења и троши их при наглом порасту оптерећења, после чега може бити ограничена на основне перформансе [10]. Ово својство је важно за интерпретацију временских промена CPU-а и UDP губитка. Верзије главних компоненти на обе крајње тачке дате су у табели 5.1.

Табела 5.1: Верзије главних компоненти експерименталног окружења

Компонента	Клијент	Сервер
Linux језгро	5.15.0-139-generic	6.17.0-1022-azure
Python	3.8.10	3.12.3
cryptography	2.8	50.0.0
iperf3	3.7	3.16
WireGuard алати	1.0.20200513	1.0.20210914
OpenVPN	2.4.12	2.6.19

Различите верзије на крајњим тачкама забележене су ради поновљивости и представљају ограничење поређења, али свих пет режима је мерено у истој поставци и у оквиру исте кампање.

5.3 Azure топологија и мрежно подешавање

Клијент је физички Linux рачунар повезан на јавни Интернет преко локалног провајдера. Серверска страна је једна Azure виртуелна машина са јавним мрежним интерфејсом и приватном адресом у виртуелној мрежи. Директни режим користи јавну путању до сервера, док сваки VPN режим користи сопствени тунелски опсег и одговарајућу крајњу тачку на истој машини. Тако се између режима мења логичка путања пакета, али не и физички клијент, сервер, локални провајдер или Azure регион. Јавна адреса и конкретни портови намерно нису наведени у раду, јер нису потребни за разумевање методологије и представљају променљиве податке о постављању система.

На Azure мрежном слоју дозвољен је само саобраћај неопходан за управљање и експерименталне крајње тачке. Linux сервер има укључено

IPv4 прослеђивање како би пакет примљен са TUN интерфејса могао да се испоручи локалној услузи или проследи ка спољашњој мрежи. За потпуни тунел примењен је NAT на излазном интерфејсу. Клијент задржава посебну руту до јавне адресе VPN сервера преко физичког мрежног пролаза. Без ње би и спољашњи UDP пакети тунела били усмерени назад у исти тунел.

Приватни режим усмерава само одређени тунелски опсег преко TUN-а и погодан је за мерење услуга на серверској страни. Потпуни режим мења подразумевану руту клијента и потврђује да се произвољни IPv4 саобраћај може пренети и вратити кроз Azure крајњу тачку. Завршна кампања користи контролисане циљеве за сваки режим, тако да промена режима не мења физички клијент, виртуелну машину или географски регион.

Сви процеси VPN крајњих тачака били су доступни током кампање, али је у датом блоку активно оптерећиван само изабрани режим. То смањује трошак поновног покретања инфраструктуре и ризик од различитог почетног стања. Недостатак је што мерења ресурса целог система садрже мало позадинско оптерећење неактивних процеса, што треба имати у виду при тумачењу апсолутних вредности.

5.4 Припрема и извршавање кампање

Пре завршног мерења проверени су приватни и потпуни тунел, пренос ICMP, TCP и UDP саобраћаја и доступност услуге `iperf3` на серверу. Скрипта завршне кампање затим је проверавала присуство локалних алата `python3`, `ping`, `iperf3`, `ssh` и `scp`, као и могућност покретања удаљеног монитора ресурса. Ове провере су извођене пре уписивања мерних резултата, па грешка у окружењу није могла да буде протумачена као лоша перформанса неког режима.

За сваки од пет режима пре кампање су послата два пробна ICMP захтева и извршен кратак, незабележен TCP пренос. Загревање је потврдило исправност рута и услуге и умањило утицај једнократног успостављања процеса на прво забележено мерење. Његови резултати нису укључени у коначни скуп.

Један експериментални блок извршаван је следећим редоследом:

1. из сачуваног распореда учитавају се број рунде, режим и циљна адреса.

2. проверава се да ли за блок већ постоје и мрежни резултат и запис ресурса. Потпуно завршен блок се не покреће поново.

3. на серверу се покреће монитор који једном у секунди чита процесорско оптерећење и заузеће меморије целог система.

4. изводи се пет група од по 20 ICMP захтева.

5. изводи се пет TCP преноса од по 10 секунди.

6. изводи се пет UDP преноса од по 10 секунди при понуђеној брзини од 20 Mbps. Између преноса оставља се пауза од две секунде.

7. монитор се зауставља, а његов излаз се преузима са сервера и повезује са ознаком истог блока.

Овај редослед је важан за тумачење резултата. Узорци ресурса покривају цео блок, а не само један пренос, док се мрежне метрике чувају за свако од пет понављања. Време успостављања VPN сесије није део измереног времена двосмерног проласка (енгл. *round-trip time*, RTT), јер су крајње тачке биле покренуте пре оптерећења.

5.5 Аутоматизација и поновљивост кампање

Скрипта кампање унапред генерише и чува распоред режима. За сваки блок активира одговарајућу путању, проверава доступност циља, обавља незабележено загревање, покреће пет понављања и снима серверске ресурсе. Напредак се чува после сваког блока, па прекид SSH везе или локалног процеса не захтева понављање већ успешно завршених режима.

Излаз сваког алата претвара се у структурирани JSON. Уз резултат се чувају режим, број рунде, редни број понављања, време, параметри и статус. Неуспешан `iperf3` покушај не брише се: бележи се разлог, а ограничено понављање се чува као посебан догађај. Ова политика је спречила да два пролазна ресетовања контролног канала неприметно буду замењена успешним резултатом.

Сваки блок добија јединствену ознаку састављену од редног броја и режима. Мрежни резултат се најпре уписује у привремену датотеку, а затим атомски премешта на коначно место. Тиме прекид током писања не оставља

датотеку која изгледа као завршено мерење. При спајању резултата програм захтева тачно шест мрежних и шест ресурсних блокова за сваки режим и проверава сагласност мерних параметара унутар режима.

Анализа се извршава над изворним записима мерења, а табеле и PNG графикони генеришу се истом командом. Тиме бројеви у раду не зависе од ручног преписивања. Репозиторијум садржи програм за анализу и скрипту кампање, док су адресе и подешавање постављеног система издвојени из јавне историје.

5.6 Насумични блоковски распоред

Експеримент има шест блокова, односно рунди. Сваки режим се појављује тачно једном у свакој рунди, редослед је насумичан, а исти режим се не јавља непосредно уз себе преко границе рунди. По појављивању режима обављено је пет понављања, што даје

$$6 \text{ рунди} \times 5 \text{ понављања} = 30 \tag{5.1}$$

понављања по режиму и 150 укупно. Кратко незабележено загревање претходило је прикупљању. Распоред је сачуван пре мерења, чиме се избегава накнадни избор повољнијег редоследа.

Укупна кампања садржи 30 блокова. У њима је послато 3.000 ICMP захтева и извршено по 150 TCP и UDP преноса. Само контролисано проточно оптерећење зато траје 3.000 секунди, односно 50 минута, не рачунајући ICMP мерења, паузе, загревање, промену режима и преузимање ресурсних записа. Овај обим је изабран да сваки режим буде мерен више пута у различитим деловима кампање, уместо да се закључак заснива на једном дугом преносу.

5.7 Радно оптерећење и метрике

Сваки блок садржао је пет понављања сваке мрежне провере. Једно понављање кашњења обухватало је 20 ICMP захтева за одјек, док су TCP и UDP преноси трајали по 10 секунди. UDP је у свим режимима имао понуђену брзину од 20 Mbps. TCP и UDP мерења обављена су алатом `iperf3` [4]. На серверу су CPU и заузеће меморије узорковани једном у секунди током целог блока.

Три дела понављања испитују различите особине система. ICMP захтеви мере време двосмерног проласка кратких пакета и показују стабилност кашњења. TCP тест препушта протоколу да повећава количину података до границе коју дозвољавају мрежа и обрада крајњих тачака, па служи за мерење остваривe пропусности. UDP тест намерно нуди истих 20 Mbps у свим режимима. Код њега зато нису најважнији послати битови, већ број датаграма који стигну до пријемника и промена времена њиховог пристизања. Серверско процесорско оптерећење и меморија бележени су упоредо са овим тестовима да би се мрежни резултат повезао са ценом обраде.

Једно понављање одговара једној основној вредности за RTT, TCP или UDP. За сваку од ових метрика и сваки режим постоји 30 вредности. За CPU и меморију основна јединица је просек узорака у једном блоку, па за сваки режим постоји шест блоковских вредности. Ово раздвајање спречава да велики број једносекундних узорака ресурса створи лажан утисак већег броја независних експеримената.

За кашњење су посматрани средња вредност, медијана, 95. перцентил, стандардна девијација и ICMP губитак. За TCP је посматрана пропусност. `iperf3` UDP брзина на страни пошиљаоца представља понуђену, а не успешно примљену брзину, па је ефективна корисна пропусност дефинисана као

$$G_{UDP} = R_{sender}\left(1 - \frac{L}{100}\right), \tag{5.2}$$

где је L проценат изгубљених датаграма. Варијација UDP кашњења представља промену размака између узастопних примљених датаграма коју пријављује `iperf3`. Мања вредност означава равномерније пристизање, али не говори сама по себи колико је датаграма изгубљено. Додатно су анализирани губитак, средњи и 95. перцентил CPU оптерећења и средње и 95. перцентил заузећа меморије.

5.8 Интегритет мерења

Сачувани су изворни резултати у формату JSON, распоред, узорци ресурса, грешке и поновљени догађаји. Режим *Direct* примио је 597 од 600 ICMP одговора, OpenVPN 599 од 600, а преостала три режима 600 од 600. Свих 30 TCP и 30 UDP мерења по режиму има важећи резултат. Два прекида управљачког канала алата `iperf3`, по један у режиму *Plaintext* и

режиму OpenVPN, успешно су поновљена. И неуспели покушаји и поновљени догађаји остали су у изворној збирци резултата.

Провера целовитости није се свела само на број датотека. За сваки режим потврђено је 30 важећих TCP и 30 важећих UDP резултата, очекивани број рунди и постојање одговарајућих ресурсних записа. Појединачни ICMP одговори нису третирани као независни експерименти. Њих двадесет чини једно мерење кашњења, чиме се избегава вештачко увећавање узорка. Слично томе, једносекундни узорци CPU-а и меморије сведени су на једну блоковску вредност. Ниједна удаљена вредност није накнадно уклоњена из скупа.

5.9 Резултати и тумачење

У наставку су приказани агрегатни резултати и одговори на истраживачка питања. Бројеви су заокружени ради читљивости.

Преглед резултата

Главне мрежне метрике приказане су у табели 5.2, а потрошња серверских ресурса у табели 5.3.

Табела 5.2: Мрежни резултати по режимима

Режим	Медијана RTT (ms)	TCP (Mbps)	Корисна UDP (Mbps)	UDP губитак (%)
Direct	33.1	22.91	18.08	0.00
Plaintext Python	35.3	23.42	19.44	2.80
Encrypted Python	33.7	19.62	17.58	12.09
WireGuard	32.7	24.48	19.43	0.00
OpenVPN	33.1	18.49	19.12	4.39

Табела 5.3: Средња потрошња серверских ресурса по режимима

Режим	Процесор (%)	Меморија (MiB)
Direct	0.64	565.08
Plaintext Python	4.92	568.29
Encrypted Python	5.81	573.51
WireGuard	3.18	566.83
OpenVPN	4.11	564.76

Медијане RTT-а налазе се у уском распону од 32.7 до 35.3 ms. Разлике у TCP пропусности, UDP губитку и CPU-у су израженије. *Encrypted Python* има највећи просечан UDP губитак и највеће просечно серверско CPU оптерећење.

Посматрање резултата по рундама показало је да су четири режима имала медијану RTT-а углавном између 32.5 и 34.2 ms. Режим *Plaintext Python* био је између приближно 34.9 и 36.1 ms. Пошто шифровани режим није имао веће кашњење од нешифрованог у истим рундама, разлика се не може приписати само криптографији. На њу утичу и тренутно стање јавне путање и редослед блокова.

TCP мерења су имала широко преклапање режима. WireGuard је имао највећу, а OpenVPN најмању средњу вредност, али су се појединачни резултати директне путање и WireGuard-а простирали од веома ниских до највиших забележених вредности. Зато разлике из табеле представљају опис ове кампање, а не непроменљив поредак система. Засебни RTT и TCP графикони нису задржани јер не додају јасан образац који се не може исказати табелом и овим поређењем.

Трошак преноса пакета у Python-у

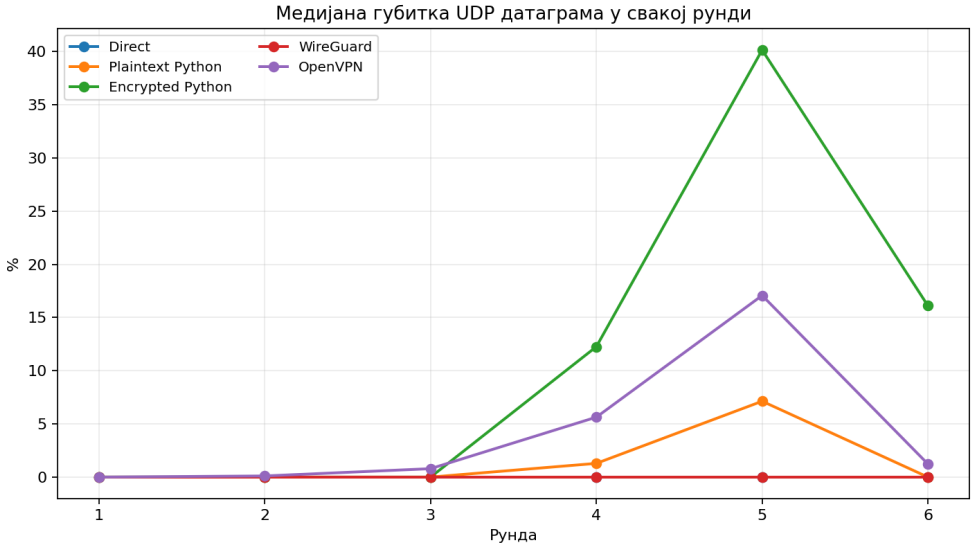
Plaintext Python има 2.2% већу средњу TCP пропусност од режима *Direct*, али ова мала описна разлика не значи да тунел убрзава мрежу. Варијација јавне путање и рунди већа је од очекиване користи било каквог паковања. Јаснији показатељ трошка је серверски CPU: средња вредност расте са 0.64% на 4.92%. Медијана RTT-а је већа за 2.2 ms. Резултат показује да TUN, UDP и петља догађаја у Python-у имају мерљиву цену и без криптографије.

Трошак безбедносног слоја

У односу на *Plaintext Python*, *Encrypted Python* остварује 16.2% мању средњу TCP пропусност, 18.2% већи средњи CPU и око 5.2 MiB више средње меморије. Ефективна UDP корисна пропусност је мања за 9.6%, док је просечан губитак већи за 9.29 процентних поена. Ово поређење најнепосредније одговара на друго истраживачко питање јер су остали делови имплементације исти.

Медијана RTT-а шифрованог режима нижа је од вредности режима *Plaintext*. То се не тумачи као убрзање услед шифровања, већ као последица

временске и мрежне варијације коју шест блокова није потпуно уклонило.



Слика 5.1: Медијана губитка UDP датаграма по рундама

На слици 5.1 приказано је кретање UDP губитка по рундама. Ово је једини графикон задржан у приказу резултата, јер објашњава зашто једна просечна вредност није довољна. Свака тачка представља медијану пет понављања истог режима у једној рунди. У прве три рунде губитак је код свих режима био близу нуле. У четвртој и петој рунди расте код три решења у корисничком простору, а код режима *Encrypted Python* медијана достиже око 40% у петој и остаје око 16% у шестој рунди. Режими *Direct* и WireGuard остају на нули. Ефективна UDP пропусност из табеле прати исти образац, јер се рачуна умањењем понуђених 20 Mbps за измерени губитак.

Поређење са WireGuard-ом и OpenVPN-ом

WireGuard има 19.9% већу средњу TCP пропусност од режима *Encrypted Python*. CPU прототипа је већи за 2.63 процентна поена, односно 82.6% у односу на WireGuard средњу вредност. WireGuard није имао UDP губитак ни у једном од 30 понављања, док је *Encrypted Python* имао просечно 12.09%. Ови резултати су у складу са очекиваном предношћу оптимизованог преноса пакета у језгру, али истовремено обухватају различите AEAD алгоритме и протоколе.

Encrypted Python има 6.1% већу средњу TCP пропусност од OpenVPN-а, али 1.70 процентних поена већи средњи CPU. OpenVPN има знатно мањи UDP губитак, 4.39% према 12.09%, и већу корисну UDP пропусност. Због нестабилности TCP ранга по рундама, разлика у средњој пропусности према OpenVPN-у остаје описни налаз, а не доказ опште предности прототипа.

Блоковски резултати процесора дају додатни контекст просецима из табеле 5.3. Директна путања остаје испод 1%, док сва VPN решења захтевају више обраде. Режим *Plaintext Python* груписан је око 5%. Шифровани режим има сличну средишњу вредност, али и један блок изнад 11%, што повећава његов укупан просек. Овај издвојени блок и раст UDP губитка у каснијим рундама подржавају опрез при приписивању целе разлике само криптографском алгоритму. Засебан CPU графикон је изостављен јер је за главно поређење довољна табела, док је издвојени блок експлицитно описан у тексту.

Кашњење, његова варијација и меморија

RTT 95. перцентил износи 72.84 ms за *Direct*, 51.82 ms за *Plaintext*, 55.86 ms за *Encrypted*, 47.11 ms за WireGuard и 55.84 ms за OpenVPN. Виши 95. перцентил режима *Direct* показује да екстремне вредности јавне путање могу доминирати репом расподеле и да кратка разлика медијана није довољна за закључак о додатном трошку VPN-а.

Средња варијација UDP кашњења је 1.16, 0.85, 1.51, 0.89 и 0.93 ms истим редом режима. Средња серверска меморија је 565.08 MiB за *Direct*, 568.29 MiB за *Plaintext*, 573.51 MiB за *Encrypted*, 566.83 MiB за WireGuard и 564.76 MiB за OpenVPN.

Варијација UDP кашњења је мала у поређењу са трајањем теста и не даје исти поредак режима као UDP губитак. Шифровани Python режим има највећу средњу вредност од 1.51 ms, али се распони режима преклапају. Зато ова метрика није приказана засебном сликом и не користи се као главни доказ разлике у квалитету преноса.

Ни разлика у меморији није довољно велика да оправда засебан графикон. Шифровани режим у просеку користи око 5.2 MiB више од нешифрованог, док су сви режими у распону од приближно 565 до 574 MiB. Пошто је мерена меморија целог система, ове вредности обухватају оперативни систем и позадинске процесе, а не само меморију VPN процеса.

Анализа UDP губитка

Велики просек режима *Encrypted Python* није последица једне екстремне вредности. Медијана губитка је 7.35%, а губитак се јавља у 17 од 30 понављања. OpenVPN има медијану 1.01% и ненулти губитак у 20 понављања, *Plaintext* медијану 0% и губитак у 10 понављања, док *Direct* и WireGuard немају губитак ни у једном понављању.

Код тунела у корисничком простору губитак расте у четвртој и петој рунди, а код режима *Encrypted Python* остаје висок и у шестој. Једно могуће објашњење лежи у начину обраде пакета. Једна синхрона петља преузима по један UDP датаграм, декодира заглавље, проверава сесију и редни број, а у шифрованом режиму затим извршава AES-GCM проверу и дешифровање пре уписа у TUN. Док се тај низ корака извршава, нови датаграми остају у пријемном реду оперативног система. Реализација не увећава подразумевани пријемни међуспремник UDP сокета, па се при дужем приливу бржем од обраде ред може попунити и наредни датаграми могу бити одбачени пре него што их Python процес прочита.

Ово тумачење је сагласно са два запажања: губитак се јавља код решења у корисничком простору, док га директна путања и WireGuard немају, а шифровани Python режим истовремено има највеће процесорско оптерећење и највећи губитак. Ипак, експеримент није бележио број одбацивања у UDP реду нити стање B2s процесорских кредита. Зато се не може искључити утицај ограничења виртуелне машине или пролазне промене јавне мреже. Резултат указује да би наредна провера требало засебно да мери одбацивања у пријемном реду UDP сокета, дужину тог реда и процесорске кредите, а затим да понови тест са већим међуспремником и обрадом више датаграма у једном пролазу петље.

Поновљена кампања

Ради провере да ли је велики UDP губитак шифрованог Python режима стабилно својство имплементације, цела кампања је поновљена у истој основној поставци: на истом физичком клијенту и Azure B2s виртуелној машини, са истих пет режима, шест насумично распоређених рунди и пет понављања по режиму у свакој рунди. Поновљена кампања обухватила је свих 30 планираних мрежних блокова и по 30 UDP и TCP мерења за

сваки режим. Њени резултати анализирани су одвојено и нису спојени са првобитним скупом.

Поређење UDP губитка режима *Encrypted Python* у две кампање приказано је у табели 5.4. У поновљеној кампањи ненулти губитак забележен је у само 2 од 30 мерења, а медијана је била 0% у свих шест рунди.

Табела 5.4: UDP губитак шифрованог Python режима у две кампање

Кампања	Просек (%)	Медијана (%)	Максимум (%)	Ненулта мерења
Првобитна	12.09	7.35	44.96	17/30
Поновљена	0.12	0.00	2.77	2/30

Потпуно понављање зато не подржава закључак да је првобитни скок UDP губитка стабилно или својствено понашање шифроване Python имплементације. Оно не поништава првобитна мерења, већ показује да је забележени временски образац зависио од услова конкретне кампање. Попуњавање пријемног реда UDP сокета остаје могуће објашњење, али није потврђено јер нису непосредно мерени бројачи одбачених датаграма. За утврђивање узрока потребан је посебан дијагностички експеримент који би пратио те бројаче, процесорске кредите виртуелне машине и више понуђених UDP брзина.

Глава 6

Закључак

У раду је пројектован и имплементиран функционалан VPN систем у језику Python за Linux. Систем повезује мрежни стек језгра и кориснички процес преко TUN интерфејса, а IPv4 пакете преноси UDP-ом. Протокол користи верзионисано бинарно заглавље, AES-256-GCM са заглављем као AAD, усмерене кључеве и 64-битне секвенцијске бројеве. Аутентификована ефемерна X25519 размена везује четири дељене тајне и комплетан транскрипт помоћу HKDF-SHA256. Реализовани су прозор за заштиту од понављања, ротација кључева, одржавање и провера везе, поновно повезивање и сервер са више клијената и провером унутрашње изворне адресе.

Исправност је проверавана са 101 аутоматизованим тестом, UDP интеграцијом преко повратног интерфејса (енгл. *loopback*), Linux мрежним просторима и стварним системом постављеним на платформи Azure. Овим су обухваћени и чиста протоколска логика и стварни TUN/рутински путеви.

Експериментална кампања је у шест насумично распоређених блокова прикупила по 30 понављања за *Direct*, *Plaintext Python*, *Encrypted Python*, WireGuard и OpenVPN. Медијане RTT-а свих VPN режима биле су блиске, између 32.7 и 35.3 ms. Укључивање безбедносног слоја у односу на *Plaintext Python* смањило је средњу TCP пропусност за 16.2%, повећало средњи CPU за 18.2% и повећало просечни UDP губитак за 9.29 процентних поена. Прототип је имао 6.1% већу средњу TCP пропусност од OpenVPN-а. WireGuard је остварио највећу средњу TCP пропусност, ниже процесорско оптерећење и нулти UDP губитак.

Потпуно понављање кампање у истој основној поставци није репродуковало велики UDP губитак шифрованог Python режима: просек је износио 0.12%,

медијана 0%, а максимум 2.77%, наспрам 12.09%, 7.35% и 44.96% у првобитној кампањи. Зато се првобитни скок не може сматрати стабилним недостатком имплементације, нити се без непосредног мерења одбацивања у реду сокета може приписати попуњавању UDP пријемног реда. Промене услова јавне мреже и дељених ресурса виртуелне машине утичу на појединачна мерења, па резултати описују посматране кампање и не представљају опште рангирање свих VPN примена.

Рад показује да је Python погодан за јасну и потпуно проверљиву истраживачку реализацију VPN протокола, али и да једноставност развоја има цену у обради пакета. Систем није замена за продукциони WireGuard или OpenVPN: недостају му независна ревизија, раздвајање привилегија, ограничење учесталости размене кључева, откривање MTU-а путање, механизам, IPv6 и зрелија заштита од ускраћивања услуге.

Даљи рад треба прво да профилише пријемну и предајну путању UDP сокета и раздвоји време TUN улаза и излаза, пријема и слања преко UDP сокета и AES-GCM обраде. Затим треба испитати групну или асинхрону обраду пакета и поновити кампању на виртуелној машини фиксних перформанси са више насумично распоређених блокова и више понуђених UDP брзина. Безбедносни правац обухвата механизам потврдних колачића или ограничења учесталости размене, већи идентификатор сесије, периодичну нову размену кључева ради опоравка после компромитације, тестове декодера насумичним улазима и независну ревизију протокола и кода.

Литература

[1] Jason A. Donenfeld. *WireGuard Protocol and Cryptography*. Englesки. 2022. URL: https://www.wireguard.com/protocol/.

[2] Jason A. Donenfeld. *WireGuard: Next Generation Kernel Network Tunnel*. Englesки. Техн. изв. ZX2C4, 2017. URL: https://www.wireguard.com/papers/wireguard.pdf.

[3] Morris Dworkin. *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*. Englesки. NIST Special Publication 800-38D. National Institute of Standards и Technology, Нов. 2007. DOI: 10.6028/NIST.SP.800-38D. URL: https://doi.org/10.6028/NIST.SP.800-38D.

[4] ESnet. *iperf3: A TCP, UDP, and SCTP Network Bandwidth Measurement Tool*. Englesки. URL: https://software.es.net/iperf/.

[5] Maxim Krasnyansky и Florian Thiel. *Universal TUN/TAP Device Driver*. Englesки. The Linux Kernel Documentation. URL: https://docs.kernel.org/networking/tuntap.html.

[6] Hugo Krawczyk, Mihir Bellare и Ran Canetti. *HMAC: Keyed-Hashing for Message Authentication*. Englesки. RFC 2104. Internet Engineering Task Force, Феб. 1997. DOI: 10.17487/RFC2104. URL: https://www.rfc-editor.org/rfc/rfc2104.

[7] Hugo Krawczyk и Pasi Eronen. *HMAC-based Extract-and-Expand Key Derivation Function (HKDF)*. Englesки. RFC 5869. Internet Engineering Task Force, Maj 2010. DOI: 10.17487/RFC5869. URL: https://www.rfc-editor.org/rfc/rfc5869.

[8] Adam Langley, Mike Hamburg и Sean Turner. *Elliptic Curves for Security*. Englesки. RFC 7748. Internet Research Task Force, Jан. 2016. DOI: 10.17487/RFC7748. URL: https://www.rfc-editor.org/rfc/rfc7748.

[9] David A. McGrew. *An Interface and Algorithms for Authenticated Encryption*. Енглески. RFC 5116. Internet Engineering Task Force, Јан. 2008. DOI: 10.17487/RFC5116. URL: https://www.rfc-editor.org/rfc/rfc5116.

[10] Microsoft. *Bsv2-series Sizes – Azure Virtual Machines*. Енглески. 2026. URL: https://learn.microsoft.com/en-us/azure/virtual-machines/sizes/general-purpose/bsv2-series.

[11] OpenVPN Inc. *OpenVPN 2.6 Manual*. Енглески. URL: https://openvpn.net/community-resources/reference-manual-for-openvpn-2-6/.

[12] Jon Postel. *User Datagram Protocol*. Енглески. RFC 768. Internet Engineering Task Force, Авг. 1980. DOI: 10.17487/RFC0768. URL: https://www.rfc-editor.org/rfc/rfc768.

[13] Александар Стоиљковић. *MATF Master VPN*. Енглески. 2026. URL: https://github.com/Aca99BG/matf_master_vpn.

Биографија аутора

Александар Стоиљковић рођен је 24. јула 1999. године. Основне академске студије завршио је на Математичком факултету Универзитета у Београду. Мастер академске студије уписао је на истом факултету, на студијском програму Информатика, број индекса 1043/23.

Тренутно ради у компанији Microsoft као софтверски инжењер, где је претходно обавио и стручну праксу. Раније професионално искуство обухвата рад у настави и послове техничке подршке у Фудбалском клубу Црвена звезда, као и друге послове током студија.