Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

Telekomunikazio Teknologiako Ingeniaritza Gradua

GRADU AMAIERAKO LANA

Ziurtagiri digitalen errebokazio egoeraren balidazio automatikoa sarean CRL erabiliz

Ikaslea: Alkorta Gascon Iker

Zuzendaria (1): Huarte Arrayago, Maider

Zuzendaria (2): Astorga Burgo, Jasone

Ikasturtea: 2021/2022

Data: 2022ko uztailaren 22a

Aurkibidea

IRUDIAK

TAULAK

LABURPENA

Proiektu hau *Internet of Things* gailuen segurtasunean oinarrituta egongo da. Askotan IoT gailuek segurtasun mekanismo urriak dituzte haien baliabideak oso txikiak izaten direlako. Orduan, beste gailu batzuekin komunikatu behar direnean zibererasoak izateko probabilitate handia dago eta gaur egungo IoT gailuen gorakada dela beharrezkoa da arazo honi irtenbidea ematea. Ziurtagiri digitalak izango dira proiektuan erabilitako segurtasun mekanismoa IoT gailu bat eta zerbitzari bat balioztatzeko. Gainera, ziurtagiri hauen errebokazio egoera konprobatuko da fidagarritasuna bermatzeko.

Proiektua software definitutako sareetan (SDN) oinarrituko da, hau dela eta, ziurtagirien konprobazioa switch programagarri batean egingo da IoT gailuaren baliabideak ez agortzeko. Horrela, proiektuaren emaitzak ikusita konprobatu ahal izango da SDN egokia dela IoT inguruneetan lan egiteko.

Hitz gakoak: IoT, SDN, segurtasuna, errebokazio egoera, balioztatu, switch programagarria.

RESUMEN

Este proyecto estará basado en la seguridad de los dispositivos *Internet of Things*. Muchas veces los dispositivos IoT tienen escasos mecanismos de seguridad porque sus recursos son muy reducidos. Entonces, hay una alta probabilidad de ciberataques cuando se tienen que comunicar con otros dispositivos y el aumento de los actuales dispositivos IoT hace necesario dar solución a este problema. Los certificados digitales serán el mecanismo de seguridad utilizado en el proyecto para validar un dispositivo IoT y un servidor. Además, se comprobará el estado de revocación de estos certificados para garantizar su fiabilidad.

El proyecto se basará en redes de software definidas (SDN), por lo que la comprobación de los certificados se realizará en un switch programable para no agotar los recursos del dispositivo IoT. Así, a la vista de los resultados del proyecto se podrá comprobar que SDN es adecuado para trabajar en entornos IoT.

Palabras clave: IoT, SDN, seguridad, estado de revocación, comprobación, switch programable.

ABSTRACT

This project will be based on the safety of the *Internet of Things*. IoT devices often have very small security mechanisms because their resources are very small. So when it comes to communicating with other devices, there is a high probability of cyberattacks, and it is the elevation of today's high-tech IoT that it is necessary to solve this problem. Digital certificates will be the security mechanism used in the project to validate an IoT device and a server. Furthermore, the revocation of these certificates shall be checked to ensure reliability.

The project will be based on Software defined networks (SDN), which means that the verification of certificates will be done in a programmable switch to prevent the resources of the IoT device from being exhausted. Thus, the results of the project can be verified that SDN is suitable for working in IoT environments.

Key words: Iot, SDN, security, revocation state, validation, programmable switch.

Laburduren zerrenda

API Application Programming Interface
CA Certificate Authority
CDP CRL Distribution Point
CPU Central Processing Unit
CRL Certificate Revocation Lists
CSR Certificate Signing Request
DB Database
DH Diffie-Hellman
DDoS Distributed Denial of Service
DTLS Datagram Transport Layer Security
EJBCA Enterprise Java Beans Certificate Authority
ESP Encrypting Security Payload
HTTP Hypertext Transfer Protocol
HTTPS Hypertext Transfer Protocol Secure
IEEE Institute of Electrical and Electronics Engineers
IETF Internet Engineering Task Force
IoT Internet of Things
IP Internet Protocol
MAC Message Authentication Code
OCSP Online Certificate Status Protocol
ONF Open Networking Foundation
RA Registration Authority
RAM Random Access Memory
SDN Software Defined Networking
SSL Secure Sockets Layer
TCP Transmission Control Protocol
TLS Transport Layer Security
UDP User Datagram Protocol
VA Validation Authority
VM Virtual Machine

1. SARRERA

Azken hamar urteotan teknologiaren hazkundea izugarria izan dela eta, komunikazio saretaz hitz egiten denean Internet da jomuga. Mundu osoan milioika dispositibo daude konektatuta Internet sarera. Alde batetik, gailu pertsonalak daude, hau da, pertsonek erabiltzeko sortzen direnak, hala nola, smartphonak, tabletak, erlojuak, telebistak, autoak… Beste aldetik, Industrian erabiltzen diren gailuak daude, adibidez, sentsoreak, alarmak, osasuna kontrolatzeko gailuak, termostato adimentsuak, eta abar. Esan bezala gailu guzti hauek Internet sarera konektatuta daude eta *Internet of Things (IoT)* izenarekin ezagutzen dira.

Beraz, IoT gailuak Internetera konektatuta dauden gailu arruntak dira eta hauek arlo espezifiko baten inguruan informazioa transmititzen dute automatikoki. Enpresetan, etxeetan eta hirietan gero eta IoT gailu gehiago inplementatzen ari dira, gainera, informazio kantitate handia transmititzen dute etengabe eta batzuetan informazioa oso garrantzitsua izan ahal da. Hala nola, oso ezaguna den Amazon enpresako Alexa ahots laguntzailea IoT gailua da baina ez da ahots laguntzailea bakarrik baizik eta, etxe guztiaren gailuetara konektatu ahal da haiek kontrolatuz. Hau da, etxearen argitasuna eta tenperatura kontrolatu ahal ditu, adibidez. Orduan, IoT gailuek Interneterako konexio segurua izan behar dute, hala ere, ez da asko hitz egiten IoT gailuen segurtasunari buruz.

1. Irudia. IoT gailuak

Gaur egun, Internet konexioak seguruak izateko erabiltzen den protokolorik nagusiena *Transport Layer Security* (TLS) da non protokolo honek *Trasnport Control Protoco*l (TCP) gainean inplementatzen den. Aldiz, *User Datagram Protocol* (UDP) gaineko segurtasun protokoloa *Datagram Transport Layer Security* (DTLS) izena du eta honek UDPrekin lan egiten duenez ez ditu helburuko gailuarekin konexioa eraiki behar. Beraz, IoT gailuetan DTLS erabiltzea hauek egokiagoa da informazio asko bidaltzen dutelako eta konexio ezarpena (TCPen egin beharrekoa) egiten ez delako.

DTLS TLSn oinarrituta dago baina aldaera batzuk ditu UDPra egokitzeko. [1] DTLS (eta TLS) bi mailetan banatuta daude. Lehenik, (D)TLS *Handshake* protokoloa. Maila honetan jatorrizko eta helburuko gailuek kriptografia asimetrikoa erabiliz autentikatu ohi dira baina beste metodo batzuk daude autentikazioa egiteko. Proiektu honetan kriptografia asimetrikoa erabiliko da bikoteak autentikatzeko. Bigarrenik, (D)TLS *Record* protokoloa dago. Maila honetan bikoteen artean adostutako gakoa (handshakean adostuta) erabiliko da informazioa enkriptatzeko, horrela, informazioa era seguruan trukatuko da.

Esan denez bikoteak autentikatzeko ziurtagiri digitalak erabiliko dira, eta prozedimendu hau Public Key Infrastructure (PKI) oinarritzen da. Hurrengo ataletan prozedimendua zehazki azalduko da baina idea bat egiteko gailu bakoitzak ziurtagiri bat du bere fidagarritasuna bermatzeko eta gailu batek beste gailu batekin konexioa lortu nahi badu honen ziurtagiria jaso behar du fidagarria dela egiaztatzeko. Horrela, haien artean autentikatzen dira eta komunikazioan erabiliko den gakoa adostu dezakete. Hala ere, ziurtagiriak ere fidagarriak direla egiaztatzen duen entitatea egon behar da. Entitate honi Certificate Autorithy (CA) deritzo eta gailuei ziurtagiriak hornitzen die.

Batzuetan CA batek dituen ziurtagiriak zenbait arrazoigatik fidagarritasuna galdu ahal du eta horren inguruan jakinarazi behar du mundu guztiari ziurtagiriaren errebokazio egoera aldatuz. Bi modu daude jakinarazpen hori egiteko. Alde batetik*, Online Certificate Status Protocol* (OCSP) erabiliz eta beste aldetik*, Certificate Revocation List* (CRL) erabiliz. Dokumentu honetan CRL zerrendarekin lan egingo da non CRLa fidagarriak ez diren ziurtagirien identifikatzaileen zerrenda izango den. Orduan, bezeroak zerbitzariaren ziurtagiria baliozkoa den ala ez jakiteko CAra jo behar du eta ea bertan dagoen egiaztatu behar du.

Beraz, proiektu honetan bi gailuen arteko konexioa segurua izateko DTLS handshakea egingo da eta zerbitzariaren ziurtagiri digitala fidagarria dela konprobatzeko, CRL zerrendan begiratuko da bere errebokazio egoera konprobatuz.

2. TESTUINGURUA

Txostenaren bigarren atal honetan proiektuan erabiliko diren teknologien azalpena egingo da. Alde batetik, segurtasunaren araberako puntuak, hala nola, Public Key Infrastructure eta DTLS eta beste aldetik, Software Defined Network eta P4 azalpenak.

Txosten honetan DTLS handshakeari buruzko aipamen asko egingo dira, orduan, handshake honen analisi zehatza egin da. Analisi hau 13.1. puntuko eranskinean aurkitu daiteke eta proiektua ondo ulertzeko beharrezkoa da handshakearen prozedimendua ondo ulertzea eta jakitea. Hortaz, oso gomendagarria da eranskina irakurtzea.

2.1. PKI (Public Key Infrastructure)

Lan honen oinarria DTLS handshakea denez, handshakean erabiltzen diren segurtasun mekanismoetan sakonduko da eta ez autentikazioa jadanik eginda dagoenean egiten diren informazio-truke mezu arrunten segurtasun mekanismoetan (Record-layer).

(D)TLSk hanshakean erabiltzen dituen segurtasun mekanismoei begira, bezeroak eta zerbitzariak elkar autentikatzen dira kriptografia asimetrikoa edo gako publikodun kriptografia erabiliz. Orduan, lanaren inplementazioa ondo ulertzeko PKI bat zer den eta bere azpiegitura nolakoa den azaldu behar da.

PKI ziurtagiri digitalen atzean dagoen azpiegitura da [2]. PKI datuak zifratzea, dokumentuak sinatzea eta autentikazioa egitea ahalbidetzen duen teknologia da gako publikoen kriptografian eta ziurtagiri digitalen erabileran oinarrituta dagoena. Esan denez, PKI ez da elementu bakarra baizik eta, azpiegitura bat non teknologia elementu asko bere baitan dagoen, hau da, azpiegitura honek beharrezko softwarea, politikak eta prozedurak ditu ziurtagiri digitalak eratzeko, kudeatzeko, gordetzeko, garraiatzeko eta baliogabetzeko.

Dokumentu honek PKI azpiegituraren elementuetan oinarrituko denez, elementu garrantzitsuen azalpena egingo da. Lehenik, hainbat alditan aipatutako ziurtagiri digitalak azalduko dira eta hauek duten formatua, x.509v3 dena. Bigarrenik, ziurtagiriaren jabea benetan bera dela autentikatzen duen ziurtagiri autoritatea (Certificate autority) azalduko da eta azkenik, ziurtagirien baliogabetze edo errebokatze egoera ikusteko CRL (Cetificate Revocation List) aztertuko da.

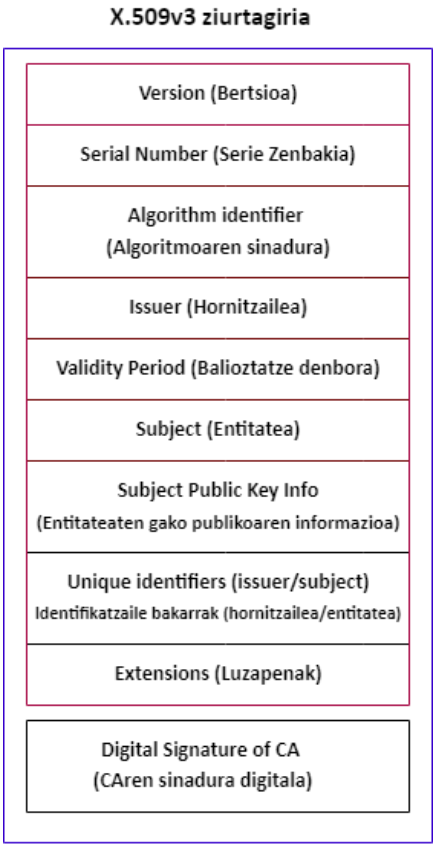
2.2. Ziurtagiri digitalak

Bi entitateen artean komunikazio segurua eduki nahi denean autentikazio eta gako-truke bat egon behar da, gero gako hauek datuak zifratu ahal izateko. PKIren izenean ikusten den bezala, teknologia hau gako publikoetan edo kriptografia asimetrikoan oinarrituta dago. Gako publikodun kriptografian entitate batek gako pare bat dauka, alde batetik, gako pribatua, bakarrik bere jabeak ezagutzen duena eta bestetik, gako publikoa, edonork dakiena. Kriptografia asimetrikoaren ezaugarririk nagusiena entitate baten gako publikoaren bidez zifratu den mezu bat bakarrik deszifratu ahal da entitate horren gako pribatua erabiliz. Beraz, entitate batek beste bateri mezu bat bidali nahi dionean helburuko ekipoaren gako publikoarekin (edonork dakiena) zifratuko du eta mezua helburura heltzean helburuko entitateak bere gako pribatuarekin deszifratuko du.

Entitate batek derrigorrez jakin behar du mezua bidali nahi dion entitatearen gako publikoa, hortaz, mekanismo bat egon behar da gako publikoa era fidagarrian lortzeko. Horretarako, ziurtagiri digitalak daude.

Laburbilduz, ziurtagiri digitala entitate baten gako publiko benetan berarena dela egiaztatzen duen dokumentua da. Baina, kriptografia tresna erabilgarria izateko erabiltzaileek erabat ziur egon behar dira beste entitate fidagarri batekin komunikatzen direla eta ez erasotzaile batekin. Egoera hau ziurtatzeko Ziurtagiri autoritatea (Certificate Authority) deitutako konfiantzazko agentea eratu zen. CAn PKI azpiegituran dauden erabiltzaileen ziurtagiri digitalak daude, hau da, erabiltzaileen nortasunak erregistratuta daude bere gako publikoarekin lotuta. Horrela, erabateko ziurtasuna dago entitate seguru batekin komunikatzen dela. Orain, gehiago sakonduko da Ziurtagiri autoritatean.

Beraz, x.509v3 ziurtagiriak PKI azpiegituran inplementatzen den formatu estandarizatua da [3] eta ziurtagiri honek bere jabearen eta ziurtagiria hornitu duen CAren informazioa du. Ziurtagiri guztiek erabiltzailearen eta hornitzailearen (CAren) izenak, erabiltzailearekin erlazionatutako gako publikoa, balioztatze periodoa, bertsio zenbakia eta serie zenbakiaz osatuta dago. Horrez gain, aukerazko eremu batzuk ere eduki ditzake, hala nola, erabiltzailearen eta CAren identifikatzaile unikoak eta luzapen batzuk. Hau da, x.509v3 ziurtagiriek honako egitura dute:



2. Irudia. x509 ziurtagiri digitalaren egitura

- Bertsioa (Version): Ziurtagiriaren bertsioa adierazten du. Kasu honetan eta gaur egun erabiltzen den hirugarren bertsioa izango dena.
- Serie zenbakia (Serial number): CAk ziurtagirari bati ematen dion Integer positibo bakarra da.
- Algoritmo sinadura (Signature): Eremu honek CAk ziurtagiria sinatzeko erabili duen algoritmoaren identifikatzailea du.
- Hornitzailea (Issuer): Issuer eremuak ziurtagiri digitala sinatu eta hornitu duen CA identifikatzen du. CA identifikatzeko Distinguised Name (DN) izeneko eremua dago (Issuer barruan). DNak CAren izena espezifikatzen du eta atributu batzuk ditu, hala nola, herrialdia, erakundea, izen arrunta…
- Balioztatze denbora (Validity period): CAk ziurtagiri jakin baten egoera informazioa noiz arte gordeko duen denborari balioztatze periodoa deritzo. Eremu honen edukia bi datetan oinarritzen da, alde batetik, noiz baieztatu zen lehen eguna eta beste aldetik, balioztatze denboraren amaierako data.
- Entitatea (Suject): Eremu honetan gako publikoaren jabea den entitatearen identifikatzailea dago non entitatea, erabiltzaile arrunt bat, zerbitzari bat edo edozein ekipo izan daitekeen. Subjectean, Issuer eremuan bezala DNa egon behar da.
- Entitatearen gako publikoaren informazioa (Subject Public Key Info): Eremu hau, bere izena esaten duen bezala, entitatearen gako publikoa darama eta gakoa erabiltzen duen algoritmoaren identifikatzailea du. (RSA, DSA edo Diffie-Hellman).
- Identifikazaile unikoak (Unique Identifiers): Entitatearen eta hornitzailearen identifikatzaile esklusiboak ziurtagirian dauden denboran zehar hauen izenak berrerabiltzeko aukera kudeatzeko balio du.
- Luzapenak: ziurtagiriak pertsonalizatzeko aukera ematen dute, ziurtagirian eremu arbitrarioak gehitzen lagunduz.
- Ziurtagiri autoritateak (CA) egindako sinadura digitala (signatureValue): Sinadura honekin CAk ziurtatzen du ziurtagiri hori bere jabearena dela eta gainera, komunikatzeko entitate segurua dela.

2.2.1. Ziurtagiri autoritatea (Certificate Authority)

Esan denez, CA bat konfiantzazko entitatea da non bertan erabiltzaileen (pertsonak, bezeroak, zerbitzariak) ziurtagiri digitalak gordetzen diren [4]. Erabiltzaileek CAn konfiantza duten bitartean ziurtatu ahal dira eskaintzen dituen ziurtagiria benetakoak, seguruak eta konfiantzakoak direla.

Orduan, entitate batek beste baten gako publikoa behar duenean CAra jotzen du. Entitateak CAren gako publikoa dakienez ziurtatzen da helburuko entitatearen gakoa publikoa fidagarria dela CAren gako pribatuarekin markatuta. Horrela, komunikazioaren konfidentzialtasuna (helburuko entitateak bere gako pribatuarekin bakarrik jatorrizko entitateak gako publikoaren zifratu duen mezua deszifratzen du) eta entitateen autentikazioa (CAk ziurtatzen du helburuko entitatearen gako publiko benetan berarena dela) lortzen da. Esan den, CAren gako publikoa CAen ziurtagirietatik lortzen da non CAen ziurtagiriak arakatzaileetan instalatuta dauden. CAn ziurtagiri digitalak hainbat formatutan gorde daitezke baina, DTLS handshakea azaldu denean esan den x.509v3 formatua da gaur egun nagusiki erabiltzen dena.

CAren funtzionamenduarekin jarraituz, esan den bezala, erabiltzaile, aplikazio, zerbitzari edo entitate batek beste baten ziurtagiria lortzeko konfiantzazko CAra jo behar du. Batzuetan ziurtagiri batzuk ez dira fidagarriak eta CAk ziurtagiri hauek baliogabetu behar ditu. Hainbat arrazoi daude ziurtagiri bat atzera botatzeko, adibidez, ziurtagiriaren gako publikoarekin lotutako gako pribatua arriskuan jarri delako, pertsona batek erakunde batetik irten delako bere ziurtagiria baliogabetu behar da (entitate baten eta CAren asoziazio aldaketa), etb. Kasu hauek gertatzen badira CAk entitatearen ziurtagiria baliogabetu behar du. Beraz, ziurtagiri bat erabili aurretik bere baliogabetasun egoera egiaztatu behar.

2.2.2. Ziurtagirien errebokazioa

Baliogabetasun egoera hau konprobatzeko bi metodo ezberdin daude [5]. Alde batetik, Online Certificate Status Protocol (OCSP) metodoa erabili daiteke OCSP bezero batek ziurtagiriak zein egoera duen galdetzen dio OCSP zerbitzari bati "*OCSP request*" mezu batekin. OCSP zerbitzariak ziurtagiriaren egoerarekin erantzungo dio bezeroari, "*OCSP response*" mezuarekin eta erantzun hau fidagarria izateko beti CAren sinadura izango du. OCSP response mezua hiru erantzun mota izan ditzake hiru ziurtagiri egoera ezberdin daudelako: Baliozkoa, ziurtagiriaren egoera baliogarria da konexio segurua izateko; Baliogabea (Revoked) egoerak ziurtagiria baliogabea dela esan nahi du, beraz, entitate horrekin konektatzea fidagarria ez dela esan diteke; Azkenik, Ezezaguna (Unknown) egoerak OCSP zerbitzariak ziurtagiriaren egoera ez dakiela esan nahi du.

Beste aldetik, Certificate Revocation List (CRL) [3] [6] metodoak CA batek baliogabetu duen ziurtagiri guztien zerrenda bat eskaintzen du eta proiektua CRL zerrenda erabiliz inplementatuko denez gehiago sakondukoa da.

X.509v3 formatua ziurtagirien baliogabetasun metodo bat definitzen du, CRL (Certificate Revocation List) izena duena. CRL, bere izena adierazten duenez, ziurtagiri baliogabeen zerrenda bat da. CA bakoitzak kudeatzen du bere CRL espezifikoa, hau da, periodikoki (ordua, eguna, astea…) ezeztatuta dauden ziurtagiri digitalak zerrendan eguneratzen dira. CRL zerrenda bat berari dagokion CAk edo CRL hornitzaileak sinatzen du eta publikoa den dohaineko errepositorio batean eskuragarri daude. Aipatu behar da ziurtagiri ezeztatu batek bere serie zenbakiarekin identifikatzen dela CRL zerrendan eta entitate batek beste baten ziurtagiria behar duenean bere sinadura eta balio periodoa begiratzeaz gain, jatorrizko entitateak ziurtagiriaren serie zenbakia CRL zerrendan ez dagoela konprobatu behar du.

Ziurtagiri batek CRL zerrenda batean bi egoera izan ditzake:

1. Baliogabea (Revoked) : Ziurtagiria errebokatuta dago eta ez da baliozkoa.
2. Mantendu (Hold): Ziurtagiria denbora batean baliogabea da baina ez da guztiz errebokatua izan. Ziurtagiria baliagarritasuna berriro lor dezake.

OCSP protokoloa CRLrekin konparatzean, OCSP duen abantailarik nagusiena datu gutxiago garraiatzen duela da, hau da, OCSP zeribitzariak bakarrik ziurtagiri espezifiko baten egoera bidaltzen du, aldiz, CRL ziurtagiri guztien errebokazio egoera duen zerrenda bidaltzen du. Orduan, esan ahal da, OCSPk sarea gutxiago asetzen duela CRL baino. Hala ere, ziurtagiri indibidualen egoeren eskaerak OCSP zerbitzaria gainkargatu ahal du trafiko altuko web tokitan.

2.3. DTLS

Datagram Tansport Layer Security (DTLS) [7] [8], User Datagram Protocol (UDP) gaineko garraioa segurua izatea bermatzen duen protokoloa da. DTLS Internet bidezko bezero/zerbitzari komunikazioetan e*avesdropping, tampering (Man In the Middle)* eta *message forgery* erasoak saihesteko diseinatuta dago. Internet gainean bi ekipoen artean konexio seguruak lortzeko erabilitako segurtasun protokolorik erabiliena Transfer Layer Security (TLS) da, non *Transmission Control Protocol* (TCP)n lan egiten duen. DTLS TLS protokoloan oinarrituta dago eta segurtasun-berme berdinak eskaintzen ditu, *order protection/non-replayability* izan ezik.

Gaur egun, DTLS version 1.2 indarrean dagoen bertsioa [RFC 6347] da eta liburutegi ugari daude bertsio honekin lan egiteko aukera ematen dutenak. Hala era, gehitu behar da DTLS 1.3 bertsioa *draft* egoeran dagoela eta oraindik ez dagoela bertsio hau jasaten duen liburutegirik. Aldiz, TLS 1.3 [RFC 8446] bertsioa onartuta dago beraz, espero da denbora gutxi barru DTLS version 1.3 onartuta egotea. Orduan, dokumentu honetan DTLSv1.2 bertsioa jorratuko da.

Esan denez, DTLSk UDPrekin lan egiten du eta UDP, TCP bezala, garraio mailako protokoloa da non IP gainean lan egiten duen datagramak bidaliz ekipo batetik beste batera. UDPk ez du host-to-host konexiorik eratzen, hau da, konexioa behar ez duen protokoloa da. Horregatik, paketeen latentzia baxuago da TCPrekin konparatuz (TCPk host-to-host konexioa egitea behar du) eta gainera, UDPk datu garraio baliabide gutxiago erabiltzen ditu. Hala ere, ezaugarri hauek zenbait desabantaila eratzen dituzte, adibidez, paketeak ordenez kanpo heldu daitezke edo batzuetan paketeak galtzen dira eta bi ekipoen arteko konexiorik ez dagoenez ezin da iturri hostari paketea birbidaltzeko eskaera egin. Beraz, UDP erabiltzea egokia da paketeen helduera fidagarria behar ez duten eta transmisio azkarra behar duten aplikazioetan. Baita ere, UDPk baliabide gutxien beharra izatea eta bere goiburuaren tamaina txikia izateak (8 byte) gailuen prozesaketa baliabideak eta denbora aurreztea ekartzen du.

IoT gailu askok UDP protokoloarekin lan egiten dute, datuak gal daitezkeen egoeretan oso egokiak dira berehala datu berriak heltzen direlako eta zaharrak ordezkatzen dituztelako. Baita ere, konexioa mantendu behar ez duten aplikazioetan DTLS inplementatzea aukerarik onena da.

IoT esparruan segurtasuna beti izan da kontuan hartzeko arazo handia. Gainera, UDPk erasoen aurrean ahultasun gehiago ditu ez duelako host-to-host autentikazio konexioa ezartzen datuak transmititu baino lehen. Lehen adierazi denez, UDP gaineko trafikoaren erasoei aurre egiteko DTLS eratu zen. Kasu askotan, TLS erabiltzea egokiena izaten da bezero/zerbitzari komunikazio seguruak eratzeko baina TLSk datagramekin lan egiteko ezintasuna dela eta, DTLS eratu zen. DTLS TLSren ahalik eta antzekoen izateko diseinatu da, bai segurtasun-asmakizun berriak minimizatzeko, bai kodea eta azpiegitura berrerabilena maximizatzeko. Hau da, DTLSren oinarrizko filosofia TLS gaineko datagrama garraioa eraikitzea da.

Orduan, DTLSk TLS duen estiloa izateko zenbait mekanismo berri gehitu behar dira, UDP gainean lan egiten duelako. UDPk dituen fidagarritasun arazoak bi mailetan eragiten dio TLSri. Alde batetik, TLSk ez du onartzen pakete indibidualen deszifratze independentea, osotasun-egiaztapena sekuentzia-zenbakiaren araberakoa delako. DTLSk arazo hau konpontzen du sekuentzia-zenbaki esplizituak gehituz, beraz, UDP duen paketeen orden arazoen eragina deuseztatzen da. Beste aldetik, TLS handshake mailak hansdhake mezuak fidagarritasunez

bidaltzen direla suposatzen du eta mezu horiek galtzen badira handshakea eten egiten da. DTLSk bigarren arazo hau konpontzen du birtransmisio timer bat inplementatzen. Birtransmisio timer honekin paketeen galera arazoa kentzen da.

Bezero/zerbitzariak DTLS handshake mezuak ordenean bidali behar dituzte eta orden espezifikoan ez badira bidaltzen handshakeak huts egin dezake, hala ere, ez da beharrezkoa. Orduan, orain azaldu diren bi mekanismoekin handshakea fidagarria izatea lortu da.

2.2. SDN ETA P4

2.2.1. SDN sareak

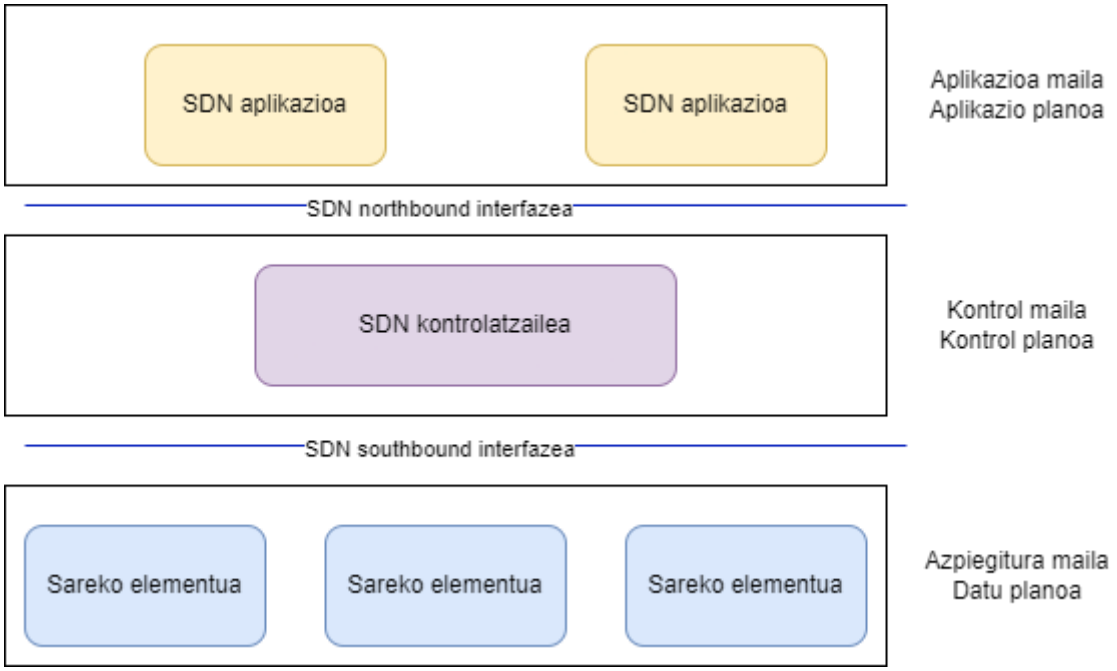
Software definitutako sareak programatzen eta automatizatzen diren sareak dira. [9] Sare hauek abiadura handiagoa, infraestruktura biziagoa eta kostu txikiagoak proposatzen dute sare tradizionalekin konparatuz eta gainera, dinamismo handia duten aplikazioei erantzun eta baliabide hobeagoak eskaini ahal diete.

Sare tradizionaletan hardware dispositibo bakoitzak (router, switch…) kontroleko funtzioak eta kommutazio funtzioak egiten ditu [22], beraz, dispositiboak era indibidualean kudeatu behar dira. Aldiz, SDN sareen oinarria hardware gailuetatik kontrol funtzioa banatzea da. Horrela, zentralizatuta dagoen software aplikazio batetik sare guztiaren dispositibo bakoitza kontrolatu eta administratu ahal dira era dinamiko. Hau da, sare administratzaile batek kontsola zentralizatu batetik sare guztiaren trafikoa kudeatu ahal du kommutadoreak banan banan konfiguratu ordez.

Sare baten kontrol funtzioak software zentralizatu batean kudeatzeak, controller izena duena, hainbat ezaugarri ekartzen ditu, hala nola, programagarria den infraestruktura, kudeaketa zentralizatua, sarearen abstrakzioa (kontrol planoa eta datu planoa banatua), malgutasuna eta denbora errealean aktualizazioak egin ahal dira, sarea automatizatu ahal da eta gainera, Open Source dira (fabrikatzailearen neutraltasuna lortzen da). [10]

2.2.1.1. SDN arkitektura

Esan denez, SDN saretan datu planoa eta kontrol planoa banatzen dira eta kontroleko funtzioak zentralizatuta dagoen software batean, Controller-ean, kudeatzen dira non Controller-a sarearen ikuspegi osoa duen. Orduan, SDN sareak sare baliabideak eta sarean dagoen trafiko fluxuak kontrolatu ahal izateko interfaze batzuk eskaini behar ditu. SDN konponente basikoak hauek dira:

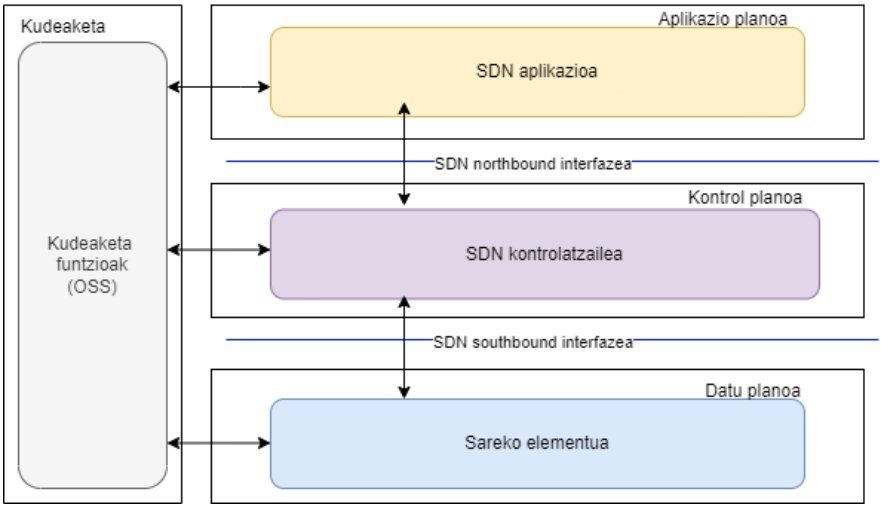


3. Irudia. SDN arkitektura basikoa

Hirugarren irudian ikusi ahal denez, hiru maila ezberdin daude: Azpiegitura maila, kontrol maila eta aplikazio maila.

Azpiegitura mailak (datu planoa) sare elementuak ulertzen eta ezagutzen ditu. Elementu hauek kontrol mailari (kontrol planoa) haien gaitasuna adierazten diote Southbound interfazearen bidez. Gainera, southbound interfazea erabiliz kontrol mailak sare elementuak programatu ahal izango ditu. SDN aplikazioak aplikazio mailan egongo dira (aplikazio planoa) eta haien sare eskakizunak komunikatuko diote kontrol planoari Northbound interfazea (NBI) erabiliz. Beraz, SDN Controller-a aplikazioen eskakizunak itzuliko ditu eta maila baxuko kontrola gauzatuko du datu planoko sare elementuetan.

Kudeaketa funtzio tradizional asko aplikazio-kontrol plano interfazeak alde batera utzi arren zenbait kudeaketa funtzio beharrezkoak dira, beraz, SDN arkitekturari kudeatzaile funtzioa gehitzen zaio. Kudeatzaile honek hiru mailetako zenbait funtzio kudeatzen ditu, adibidez, datu planoan sare elementuen hasierako konfigurazioa egiten du, SDN bidez kontrolatzen diren zatiak esleitzen ditu eta haien SDN krontrolerra konfiguratzen ditu. Kontrol planoan, kudeatzaileak SDN aplikazioari emandako kontrol-esparrua definitzen duten politikak konfiguratu behar ditu eta gainera, sistemaren errendimenduaren jarraipena egiten du. Azkenik, aplikazio planoan kudeatzaileak normalean kontratuak eta zerbitzu-maila hitzarmenak (SLA) kudeatzen ditu. Plano guztietan, kudeatzaileak segurtasuna konfiguratzen du banatutako funtzioak modu seguruan komunikatzeko. 2. Irudian ikusi ahal da SDN konponente guztiak kudeatzailea gehitzen zaionean.



4. Irudia. SDN konponenteak kudeatzailearekin

Orain, datu, kontrol eta aplikazio planoan gehiago sakonduko da.

2.2.1.1.1. Datu planoa

Datu planoak sareko elementu bat edo gehiagoz osatuta dago, eta horietako bakoitzak trafikoa birbidaltzeko edo trafikoa prozesatzeko baliabide multzo bat dauka. Gainera, datu planoak bezeroen trafikoarekin zuzenean tratatzen duten baliabideak biltzen ditu, birtualizazioa, konektibitatea, segurtasuna, erabilgarritasuna eta kalitate egokiak bermatzeko beharrezko baliabideekin batera.

2.2.1.1.2. Kontrol Planoa

Kontrol planoan SDN Controller-ak daude. SDN Controller bakoitzak datu planoan dituen sare elementuek duten baliabide multzoen kontrol esklusiboak ditu.

SDN Controller batek onartzen dituen aplikazioen eskakizunak bete behar ditu beti, aplikazio bakoitza beste guztiengandik isolatuz. Funtzio hau betetzeko, SDN Controller batek bere parekoa den beste SDN kontrolatzaile batekin komunikatu ahal da, edo SDN controller batek bere menpe dauden beste Controller batzuk izan ahal ditu edo SDN ez diren inguruneekin komunikatu ahal da, beharrezkoa balitz.

SDN Controller-ak haien artean komunikatzeko Westbound edo Eastbound interfazeak erabiltzen dute, horrela, haien artean sare bat kontrolatu ahal dute non kontrolatzaile bakoitza sarearen domeinu bat kudeatzen duen.

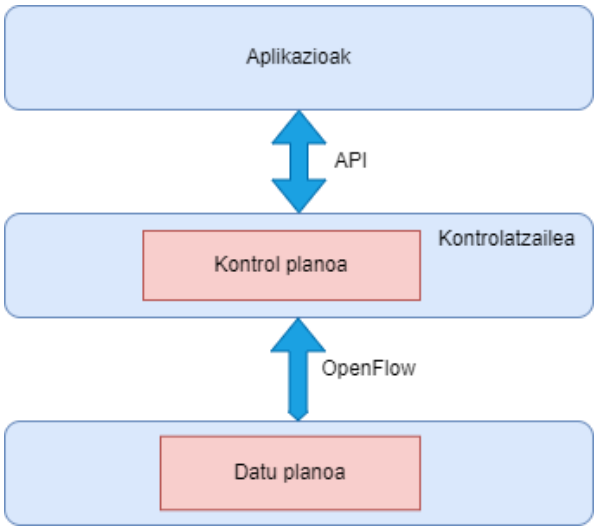
Gainera, SDN kontrolatzaile baten funtzio arrunta baina funtsezkoa ez den bat feedback-begizta batean kontrol-elementu gisa jardutea da, horrela, sareko gertaerei erantzun ahal du hutsegitetik berreskuratzeko, baliabideen esleipenak berriro optimizatzeko edo bestelakoak.

2.2.1.1.3. Aplikazio planoa

Aplikazio planoak aplikazio bat edo gehiago ditu, eta horietako bakoitzak SDN kontrolagailu batek edo gehiagok erakusten dituen baliabide multzo baten kontrol esklusiboa du.

2.2.1.1.4. OpenFlow

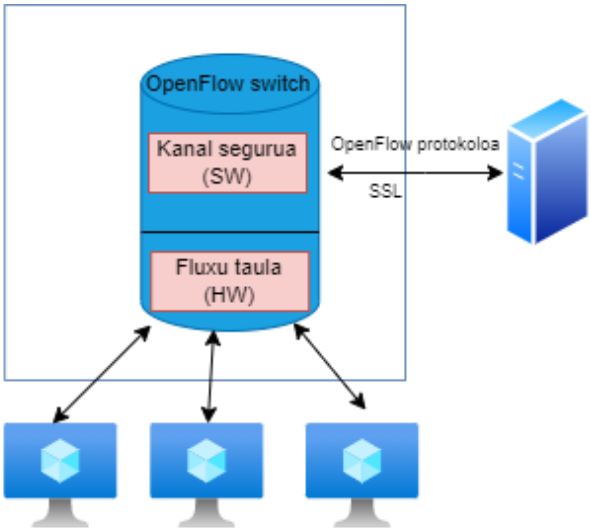
Esan denez, kontrol planoa eta datu planoa komunikatzeko Southbound interfazea erabiltzen dute. Bi plano hauek komunikatzeko erabiltzen den protokolorik esanguratsuena OpenFlow protokoloa da. Ikusi 5. Irudia.



5. Irudia. SDN eskema OpenFlow protokoloa erabilita

OpenFlow protokoloak [11] switchak programatzeko aukera ematu du, horrela, software zerbitzari baten paketeak switch sare batean zer bide hartu behar duten erabakitzen duen programa sortu ahal da. Gainera, OpenFlow-ek fluxu-taulak programatzeko protokolo ireki bat eskaintzen du kommutadore ezberdinetan eta sare administratzaile batek trafikoa beste fluxu batzuetan banatu dezake.

OpenFlow hobeto ezagutzeko OpenFlow switch bat ikusiko da 4. Irudian.



6. Irudia. OpenFlow switch

6. Irudian ikusten denez, OpenFlow switch batek hiru atalez osatuta dago. Lehenik, Fluxu taula (Flow Table) aurkitzen da eta honen fluxu-sarrera bakoitza ekintza batekin lotzen da switch-ari fluxua nola prozesatu behar duen esateko. Bigarrenik, kanal segurua (Secure Channel) switch-a eta Controller-a (datu planoa kontrolatzen du) konektatzen duen kanala da eta komandoak eta paketeak bidaltzea ahalbidetzen du haien artean. Komunikazio segurua izateko SSL edo TLS konexioak erabili egiten dira. Azkenik, kontrolatzailearen eta switcharen artean komandoak eta paketeak bidaltzeko irekia eta estandarra den OpenFlow protokoloa erabiliko da.

2.2.2. P4 lengoaia

P4 [12] lengoaia birbidaltze-elementu programagarri baten datu planoaren bidez paketeak nola prozesatzen diren adierazteko hizkuntza da, hala nola, router bat, software edo hardware switch bat, sare txartel bat edo edonolako sare tresna. P4-ren izena lengoaia azaldu zen P4 Language Consortium-en lehenengo paper-etik eratorria da “Programming Protocol-independent Packet Processors ”.

P4 lengoaia datu planoak definitzeko tresna bezala definitu zen, hau da, P4 programa bat ezin da kontrol planoa deskribatzeko erabili. Hala ere, P4 programek datu planoa eta kontrol planoa komunikatzeko interfazeak definitu ahal dute partzialki. Orduan, P4 programek pakete prozesadore batek pakete baten gainean exekutatzen duen funtzioak definitzeko, ulertzeko eta manipulatzeko gai da pakete hori jaso, birbidali, prozesatu edo deuseztatzen den arte. Gainera, P4 lengoaia SDN sareen abantailak mantentzen ditu, kontrol planoa eta datu planoa bananduta daudelako eta kontrola zentralizatua delako, adibidez.

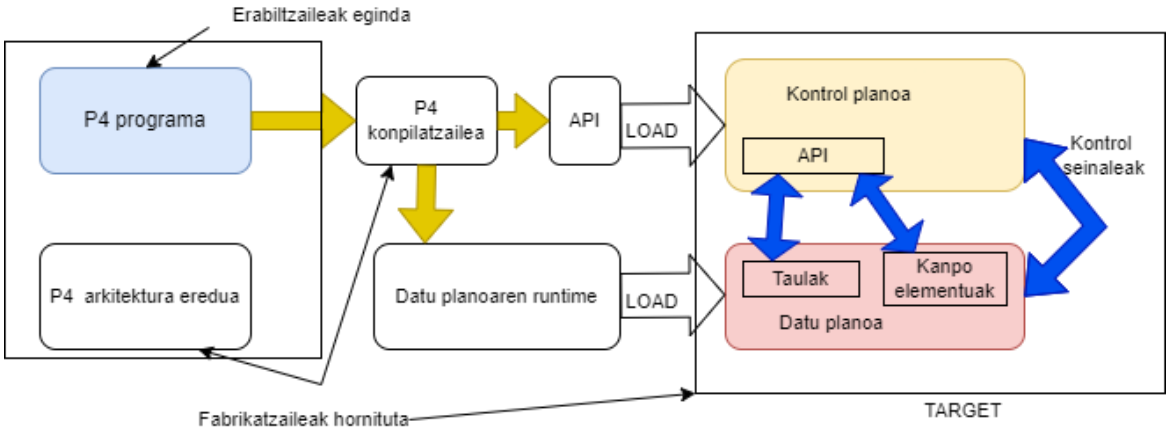
P4 lengoiaren bertsioei begira, dokumentua egiten ari den dataren arabera, bi bertsio daude: P4-14 et P4-16. P4-16 P4 hizkuntzaren egungo berrikuspena da eta bertsio honi buruz hitz egingo da dokumentu honetan. P4-16 P4-14rekin konparatzen denean P4-16k sintaxi zehatzagoa du P4 programa baten konportamendua definitzeko. Gainera, hizkuntza-ezaugarri ugari ezabatu dira P4-14 bertsiotik eta liburutegietara eraman dira, besteak beste, kontagailuak, checksum-unitateak, neurgailuak, etab. Horrela, P4 lengoaia hizkuntza konplexu bat izatetik oinarrizko

hizkuntza nahiko itxi bat izatera pasa da P4 gehiena idazteko beharrezkoak eta oinarrizkoak diren egiturak liburutegietan daudela.

2.2.2.1. P4 kontzeptuak

Dokumentu hau eta nola P4 lengoaia nola aplikatzen den ondo ulertzeko, lehenik, P4ren oinarrizko bi kontzeptu azaldu behar dira, hala nola, P4 target-a eta P4 arkitektura.

- P4 Target: P4 programa bat exekutatzeko gai den pakete-prozesamendu sistema da, hardware edo software izan daitekeena.
- P4 arkitektura: P4n programagarriak diren blokeak (adibidez, analizatzailea, sarrera-kontrol-fluxua, etab.) eta haien datu-planoko interfazeak identifikatzen ditu. P4 arkitektura programaren eta target-aren arteko kontratu bezala har daiteke.



7. Irudia. P4 target

7. irudian ikusi daiteke P4 erabiliz target bat programatzerakoan lan-fluxu tipikoa. Target fabrikatzaileek hardware edo softwarearen inplementazio framework-a, arkitektura definizioa eta target horretarako P4 konpiladore bat eskaini behar dituzte. P4 programatzaileek arkitektura espezifiko baterako programak idazten dituzte non programa target-ean P4n programagarri diren osagaien multzo bat definitzen duen. P4 programa bat konpilatzean bi artefaktu sorrarazten ditu. Alde batetik, sarrerako programan deskribatzen den logika inplementatzen duen datu plano konfigurazioa, eta beste aldetik, kontrol planotik datu planoen objektuen egoera kudeatzeko API bat sortzen du. Laburbilduz, konpilatutako P4 programa bat eta kontrol planoa programa horrekin elkar eragiteko API bat.

3. IRISMENA ETA HELBURUAK

Txostenaren atal honetan proiektuaren helburuak aurkeztuko dira. Lehenik, helburu nagusia azalduko da eta ondoren, hau lortzeko aurrera eraman behar diren bigarren mailako helburuak adieraziko dira.

3.1. HELBURU NAGUSIA

Proiektu honen oinarrizko helburua IoT gailu baten, bezeroa izango dena, eta zerbitzari baten arteko konexio segurua ezartzea izango da non DTLS protokoloa erabiliko den eta zerbitzariaren ziurtagiri digitalaren errebokazio egoera era automatikoan konprobatuko da CRLa erabiliz. Konexioa segurua izatea lortzeko zerbitzariaren ziurtagiria fidagarria dela konprobatuko da dagokion CAren CRLan begiratuz, hau da, zerbitzariaren ziurtagiriaren identifikatzailea CRLan badago ez da bien arteko konexioa burutuko, aldiz, identifikatzailea ez badago bien arteko konexioa egingo da datuak trukatu ahal izateko. Gainera, CRLaren jaitsiera eta ziurtagiriaren konprobazioa P4 switch batean egingo da, horrela, ez da IoT gailuaren baliabide konputazionalik beharko eta IoT gailua ez da trafikoz gainkargatu.

3.1.1. Sistemaren diseinu orokorra

Proiektuaren helburu nagusia lortzeko, lehenik, sistema osoaren diseinua orokorra pentsatu behar da. Horretarako, inplementazioan erabiliko diren teknologien aukeraketa egin behar da, hala nola, zelako sare mota, segurtasunarekin erlazionatutako zer teknologia erabili eta maketaren arkitektura ere pentsatu behar da. Gero, inplementazioan zehar teknologia hauek non eta noiz inplementatuko diren jakin behar da.

3.1.2. Modulu bakoitzaren diseinua

Sistemaren diseinu orokorra egin eta gero, modulu zehatz bakoitzaren diseinua egin behar da. Lehenik, maketaren arkitektura sortuko da bezeroa, zerbitzaria, CA, P4 switcha eta kontrolatzailea izango dituena. Bigarrenik, CA sortuko da eta honek bezeroari eta zerbitzariari ziurtagiri digitalekin hornituko die. Gainera, CAk CRL zerrenda sortuko du. Behin bikoteek ziurtagiriak edukita P4 switcharen eta kontrolatzailearen programa idatziko da (CRLa deskargatu eta konprobatzen duena). Azkenik, DTLS handshakea egingo da bezero eta zerbitzariaren artean.

3.1.3. Modulu bakoitza inplementatzeko beharrezko tresnak

Hurrengo helburua, modulu bakoitza inplementatzeko behar diren tresnak bilatu beharko dira, hau da, modulu espezifiko bat aurrera eramateko dauden tresnen analisia eta aukeraketa egin behar da. Adibidez, CA eta CRLak sortzeko zein liburutegi hartuko diren edo DTLS handshakea inplementatzeko liburutegia aukeratu behar da. Gainera, kontuan hartu behar dira inplementaziorako behar diren teknologien hardware eskakizunak.

3.1.4. Sistemaren elementuen arteko integrazioa eta balidazioa

Behin modulu bakoitza indibidualki eginda hauen arteko integrazioa egin behar da, hau da, maketa nagusian beharrezko konexioak sortuko dira. CA, bezeroa, zerbitzaria, programak eta beharrezkoak diren entitateak maketan kokatuta DTLS inplementazioarekin hasi ahalko da. Gehitu beharra dago bi egoera ezberdin egongo direla. Lehenengo egoeran, CRL deskarga eta ziurtagirien konprobazioa bezeroak egingo du eta bigarren egoeran, switchak egingo du. Horrela, bi egoeren arteko konparazioa burutu daiteke.

3.1.5. Balidazio funtzionala eta errendimenduaren analisia

Frogak egin aurretik, lehenik, eraikitako sistemaren balidazio funtzionala egin behar da, hau da, egin den inplementazioa guztiz ondo funtzionatzen duela egiaztatu behar da. Esandako balidazioa egiaztatu ondoren, errendimendu analisiarekin hasi daiteke. Beraz, lehen esandako bi egora desberdinetan frogak egingo dira, hala nola, errendimendu frogak, denbora frogak, eta abar. Froga horien emaitzak edukita bi egoeren konparazioa egin eta amaitzeko, konklusioak egingo litzateke.

4. ONURAK

Txostenaren atal honetan proiektua arrera ateratzean sortzen diren onurak azalduko dira. Hiru onura mota deskribatuko dira; lehenik, onura teknikoak, ondoren, onura ekonomikoak eta atal honekin amaitzeko, onura sozialak.

4.1. Onura teknikoak

Proiektuaren onura teknikoei dagokionez, proiektu honek onura tekniko handiak lortzen dute IoT komunikazioen esparruan, bereziki, Internetera konektatzeko segurtasunean. Hurrengoko puntuak dira proiektuan sortutako onura teknikoak:

1. **Segurtasuna IoT gailuetan:** IoT gailuak askotan ez dute segurtasun handirik eskaintzen. Honen zergatia zenbait arrazoi ditu, hala nola, gailuak oso txikiak direlako eta beraz, haien ahalmen konputazionala baxua delako, haien baliabideak agortzea nahi ez delako edo konfiguratzeko zailtasuna interfaze itxiak dituztelako. Arazo hauek kontuan hartuz, IoT gailuek askotan ez dute beste gailuen ziurtagirien errebokazio egoera konprobatzen ezta CRL erabiliz ezta OCSP erabiliz.
   Orduan, proiektu honek errebokazio egoera CRL bidez konprobatzeko soluzioa ematen du switch programagarri bat erabiliz. Horrela, bi gailuen konexioa segurua izango da eta gainera, ez da IoT gailuaren baliabiderik agortuko CRL konprobaketa switchak egingo duelako.

2. **Amaierako sarearen hedapena eta konfigurazioa errazagoa:** Proiektuan switch programarri bat erabiliko da, honen ondorioz, sarearen konfigurazioa edonoiz aldatu daiteke proiektuaren helburura egokitzeko. Laburbilduz, switcharen datu planoa programatzaileak nahi duenean aldatu dezake, adibidez, proiektu honetan CRL bidezko errebokazio egoeraren konprobazioa egingo da, hala ere, switch programeriaren kontrolatzailearen programa erraz aldatu da OCSP bidezko konprobazioa egiteko.

4.2. Onura ekonomikoak

Problema bati irtenbide bat ematen denean onura ekonomikoak sortzen dira. Honako hauek izan daitezke proiektu honen onura ekonomiak:

1. **Segurtasun ezaren ondoriozko galerak:** Gailuak komunikatzen direnean gaur egun, segurtasuna oinarrizkoa da eta komunikazioetan segurtasuna inplementatzen denean edonolako erasoak egon daitezke. Eraso hauek gertatzen direnean beti egongo dira galera ekonomikoak eta enpresek diru asko galdu dezakete haien informazioa berreskuratzeko. Orduan, IoT gailuetan segurtasuna sartzea, DTLS eta CRL mekanismo bezalakoak erabiliz eraso asko ekidin ditzakete eta honekin batera diruen galera.
2. **SDN erabilera:** SDN motako sareak erabiltzeagatik asko aurreztu daiteke sarean hedapenean eta geroko konfigurazioetan. Sareko elementuak progrmagarriak izatean eta kontrolatzaile batek kudeatzen dituenez denbora eta diru aurrezpena lortzen da. Gainera, SDN softwarrean oinarrituta ez dagoenez ez dira gailu gehigarririk erosi behar.

4.3. Onura sozialak

Onura sozialei begira, gizarteak onura hauek lortu ahal du proiektu honen soluzioan:

1. **Erabiltzaile datuak**: Txosten honetan hainbat alditan esan denez, IoT gailuen erabilera esponentzialki igotzen ari da. Enpresa askok IoT gailuak erabiltzen dituzte eta egoera askotan, dispositiboek transmititzen dituzten datuak erabiltzaile arruntenak dira eta hauek datu pertsonalak izaten dira. Erabiltzaileek beti izan behar dute komunikazio segurua, beraz, IoT gailuen eta zerbitzarien arteko transmisioan segurtasuna gehitzeko onura sozial bat dela esan daiteke. Hau da, erasotzaile batek pertsona bate datuak lapurtzen baditu aukera izan dezake, adibidez, erabiltzailea non bizi den jakiteko edo bankuko txartel zenbakia lapurtzeko. Orduan, segurtasun mekanismoen bidez erasotzaileek datu pertsonal horiek lapurtzeko probabilitatea asko jaitsi egiten da.

5. ARTEAREN EGOERA

Txostenaren atal honetan proiektuaren arazoari irtenbidea ematen dion beste antzeko proiektua komentatu egingo da. Kasu honetan, *DTLShps: SDN-Based DTLS Handshake Protocol Simplification for IoT* [13] proiektua azalduko da eta nola SDN erabiliz DTLS handshakea inplementatu den zerbitzari eta IoT gailu baten artean.

Aipatutako proiektuaren helburua IoT gailuen baliabideak ez agortzeko SDN sareak erabiltzea da segurtasun mekanismoak inplementatzeko. DTLS protokoloak baliabide konputazionalak behar ditu eta askotan, IoT gailuek ezin dituzte karga horiek jasan, beraz, proiektuaren autoreek SDN sareak erabiltzea aukeratu dute irtenbide modura.

Proiektuan, lehenik, SDN kontrolatzailea erabiltzen dute gako simetrikoa era dinamikoan sortzeko. Gero, gailuen ziurtagiri digitalen egiaztapena beste kontrolatzaile indartsu batean egiten da eta azkenik, kontrolatzaileak DTLS zerbitzaria ordezkatzen du Hello Client DTLS mezuetan cookiearen hartu-emana egiteko.

Proiektu honen errendimendu ebaluaketan ikusi ahal da ez direla bakarrik IoT gailuaren baliabide konputazionalak eta energia-kontsumoa era eraginkorrean murrizten, baizik eta, DTLS handshakearen iraupena ere jaisten dela SDN kontrolatzaileak erabiliz.

Bi proiektuetan antzekotasun ugari ikusi daitezke. Alde batetik, proiektuen helburuak oso antzekoak dira, hau da, IoT gailuen baliabideak ez agortzeko SDN sareak inplementatzea DTLS handshakea egitean. Albaina, prozedimendua nahiko desberdina da. Proiektu honetan kontrolatzaile bat bakarrik erabiltzen da eta gainera, CRL zerrenda deskargatzeko eta ziurtagirien errebokazio egoera konprobatzeko erabiltzen da. Aldiz, beste proiektuan kontrolatzaile bat baino gehiago erabiltzen dira. Alde batetik, DTLS zerbitzaria ordezkatzeko eta gako simetrikoa sortzeko eta beste aldetik, beste kontrolatzaile bat ziurtagirien egiaztapena egiteko.

Bi proiektuen prozedimenduak desberdinak izan arren, bien ondorioak bat datoz. SDN sareak IoT inguruneetan erabiltzea irtenbide egokia direla gailu hauen errendimendu hoberena ateratzeko, horrela, IoT gailuek oinarrizko datuak bidaltzeaz bakarrik arduratzen dira eta aldiz, kontrolatzaileek kudeaketaz, segurtasunaz edo beste prozeduretaz arduratzen dira.

Beste aldetik, hainbat proiektu daude IoT inguruneetan SDN sareak erabiltzen dituztenak. Hala nola, *Managing AAA in NFV/SDN-enabled IoT scenarios* proiektuak [25] SDN erabiliz IoT inguruneetan gailuen autentikazioa, baimena, kontabilitatea eta kanalen babesaren funtzioak kudeatzeko frameworka proposatzen du. Switch programagarriak autentikazioa, baimen erabakiak eta sarbide kontrolaz arduratuko dira eta kanal babesa DTLS tunelak eratuz inplementatuko da.

Ikusi ahal denez, zenbait proiektu daude SDN sareak inplementatze dituztenak IoT inguruneetan. Orduan, esan daiteke SDN sareek gero eta gehiago erabiltzen hasi direla eta hurrengo urteetan gehiago kontsideratu egingo direla sareak inplementatzeko orduan.

6. ALTERNATIBEN ANALISIA

Dokumentuaren atal honetan proiektua aurrera eramateko alternatibak azalduko dira eta aukera hauen azterketa eta ikasketa eginez inplementaziorako egokiagoak izango direnak aukeratuko dira.

Alternatiben aukeraketa egiteko, lehenik, problematika zein den azalduko da. Problematika azaldu ondoren, aukera bakoitzaren azterketa egingo da eta gero, hautaketa irizpideak zeintzuk izango diren adieraziko dira. Amaitzeko, hautaketa irizpideak kontuan hartuta inplementazioa egiteko aukera hoberena aukeratuko da.

Irizpideen ebaluazioa egiteko, irizpide bakoitzak ponderazio bat izango du garrantziaren arabera. Irizpide bakoitzaren poderazioaren batura, logikoki, %100 izango da. Alternatiba bakoitza, irizpide bakoitzeko, 1-5 puntuaketa batekin ebaluatuko da eta azkenik, guztizko puntuak zenbatuko dira eta puntu gehien duen alternatiba aukeratuko da.

Kasu honetan, hiru problematika identifikatu dira:

1. Sarea programatzeko mekanismoa
2. DTLS inplementaziorako liburutegiak
3. CA inplementatzeko mekanismoa

6.1. Sarea programatzeko mekanismoa

Lehen aipatu den bezala, proiektua SDN sare batean inplementatuko da eta SDN sare bateko gailuak programatzeko mekanismo bat baino gehiago daude. Bi alternatiba aurkeztuko dira, alde batetik, Openflow mekanismoa erabiliz eta beste aldetik, P4 lengoaia erabiliz.

Openflow eta P4 lengoaia zer diren testuinguruan azaldu dira, Openflow 2.2.1.1.4 puntuan eta P4 2.2.2 puntuan. Analisiak egiteko haien laburpen txiki bat egingo da, ondoren, irizpideak azalduko dira.

6.1.1. Openflow

OpenFlow switchak programatzeko aukera ematen duen protokoloa da [14], horrela, sare baten datu fluxua kontrolatu daiteke. Gainera, OpenFlowek fluxu-taulak programatzeko protokolo ireki bat eskaintzen du kommutadore ezberdinetan eta sare administratzaile batek trafikoa beste fluxu batzuetan banatu dezake.

Ikertzaileek bideratze protokolo berriak probatu ditzakete, segurtasun ereduak, helbideratze-eskemak eta baita alternatibak ere. Hala ere, OpenFlowek ez du benetan switcharen portaera kontrolatzen; taulak betetzeko modua ematen du. Gainera, gailuak Openflow protokoloa erabiltzeko espezializatuta egon behar dira, beraz, gailu hauen kostua asko igotzen da.

6.1.2. P4

P4 switch programagarriak programatzeko lengoaia da. Openflow ez bezala, P4k switcharen portaera programatzeko aukera ematen du, eta gainera, paketeak nola prozesatu behar diren programatu daiteke. P4 erabiliz switchak paketeen goiburuak ezagutu ditzake eta horren arabera pakete horrekin zer egin behar duen programatu daiteke.

Beraz, P4 lengoaiaren bidez hainbat aukera daude sare bateko trafikoa kontrolatzeko. Gainera, badago P4 programa bat non Openflow protokoloa simulatzen duen, orduan, esan daiteke P4k Openflow baino askoz aukera gehiago ematen duela.

6.1.3. Hautaketa

Hautaketa egiteko honako irizpideak hartuko dira kontuan:

1. Eskalagarritasuna: Eskalagarritasun gehiago ematen duen mekanismoa hobeto kontsideratuko da.
2. Kostua: Kostuari dagokionez, switch programagarrien arabera hitz egiten da. Hau da, Openflow edo P4 erabiltzeko switchen kostua kontuan hartuko da eta kostu baxuena behar duena hobeto ebaluatuko da.
3. Erabilera gaitasuna: Erabilera modu kopurua ebaluatuko da, hau da, erabilera gehien duen mekanismoa hobeto kontsideratuko da.

Behin irizpideak edukita, progarmatzeko mekanismoen ebaluaketa egingo da:

	P4	Openflow
Eskalagarritasuna (%30)	9	7
Kostua (%20)	7	6
Erabilera gaitasuna (%50)	10	5
TOTALA	9,1	5,8

1. Taula. P4 vs Openflow

Taulan ikusten denez, P4 askoz aukera hobea da Openflow protokoloarekin konparatuz. P4k alderdi askotan erabili daiteke, hau da, erabilera anitz ditu sareko fluxua manipulatzeko, horregatik, proiektu honetan P4 lengoaia erabiliko da switch programagarria programatzeko.

6.2. DTLS handshake inplementaziotarako liburutegiak

Proiektuak DTLS handshakea inplementatuko duenez beharrezkoa da prozesua simulatzen duen fitxategiak eta liburutegiak izatea. Interneten DTLS erabiltzeko errepositorio asko aurkitu daitezke eta bi errepositorio analizatuko dira egokiena aukeratzeko asmoarekin.

Bi errepositorio horiek Eclipse/Californium eta dtls/python izango dira. Lehenik, bi alternatiba hauen azterketa egingo da eta ondoren, hautaketa.

6.2.1. Eclipse/Californium

Eclipse Californium [15] IoT zerbitzuetarako Java inplementazioa da eta eskalagarritasuna eta erabilgarritasuna dira Californiumen helburuak, beraz, IoT gailuen segurtasunaz arduratzen diren liburutegiak eskaintzen ditu. Californiumek DTLS 1.0 eta DTLS 1.2 bertsioekin lan egiteko aukera ematen du gailuei segurtasunez hornitzeko. Errepositorio honetako programak eta liburutegiak Java lengoaiaz idatzita daude eta *https://github.com/eclipse/californium* Github errepositorian deskargatu daitezke. Gainera, liburutegia erabiltzeko Eclipse IDEa behar da eta honek, hardware baliabide asko kontsumitzen ditu.

Californium erabiltzeko beharrezkoak hauek dira:

- Eclipse
- Maven Integration for Eclipse
- UTF-8 workspace text file encoding
- Java openjdk-7

6.2.2. Python3/dtls

https://github.com/mobius-software-ltd/pyton3-dtls errepositorioan aurkitu daitezkeen programak DTLS 1.2. bidezko komunikazio seguruak simulatzen dituzte [16]. Bere izenak esaten duen bezala Python3 lengoaian idatzita daude eta DTLS protokoloan espezializatuta dago, horregatik, sinplea da proiektu honetan liburutegi honekin lan egitea.

Liburutegi honek behar dituen beharrekoak hauek dira:

- OpenSSL 1.1.1. edo bertsio altuagoko liburutegia.
- Python 3.6 edo altugoa.

6.2.3. Hautaketa

Behin alternatibak ikusita hautaketa irizpideak azalduko dira:

1. Proiektura egokitzeko erraztasuna: Liburutegia proiektura ondo egokitu behar da, beraz, DTLS protokoloan gehien espezializatua dagoen liburutegia hobeto baloratu egingo da.
2. Dependentziak: Dependentzia gutxien duen liburutegia proiektuan inplementatzeko hobeto kontsideratuko da.
3. Hardware eskakizunak: Hardware baliabide gehien behar duen liburutegia txarrago baloratuko da.

Hautaketa irizpideak jakinda alternatiben ebaluaketa egingo da:

	Californium	Python/DTLS
Proiektura egokitzeko erraztasuna: (%30)	7	9
Dependentziak (%40)	5	9
Hardware eskakizunak (%30)	5	8
TOTALA	5,6	8,7

2. Taula. Californium vs Python3/DTLS

Taularen emaitzetan oinarrituz proiektuan DTLS protokoloa inplementatzeko liburutegia Python/DTLS da. Liburutegi honek ez ditu dependentzia askorik behar eta gainera, DTLS protokoloan espezializatuta dago. Beraz, askoz egokiagoa da Californium liburutegiarekin konparatuz.

6.3. CA inplementatzeko mekanismoa

Proiektuan oinarrizkoa den gailua ziurtagiri autoritate da gailuei ziurtagiri digitalez hornitzen duelako eta CRL zerrenda sortzen duelako. Hainbat metodo daude CA inplementatzeko eta kasu honetan, bi mekanismoen artean konparatuko da; alde batetik, Entreprise Java Bean Certificate Authority (EJBCA) erabiliz. Eta beste aldetik, open Secure Socket Layer(SSL) erabiliz.

6.3.1. EJBCA

EJBCA PKI Ziurtagiri Autoritate software da Java lengoaiaren bidez programatuta dagoena. [17] EJBCAk ziurtagiri digitalak hornitu, kudeatu eta baliozkotu egiten ditu, hau da, PKI azpiegitura guzti bat eraikitzeko erabili dezake. Gainera, EJBCA dohainekoa den softwarea da eta edozein PKI jasaten ditu, hala nola, IoT eta Intustria IoT. EJBCA OCSP erantzunak eta CRL zerrendak sortu ditzake ziurtagirien errebokazio egoera konprobatzeko.

EJBCA instalatzeko zenbait pausu eman behar dira, lehenik, dependentziak instalatu behar dira, gero, EJBCA deskargatu behar da. Eta azkenik, Mariadb datu base plataforma instalatu behar da. Instalatu ondoren, Wildfly 10 zerbitzaria eta deskargatu den datu basea konektatu behar dira.

EJBCA dependentzia batzuk behar ditu bere funtzioa egin ahal izateko. Honako hauek dira dependentziak:

- Java openjdk-8
- Mariadb datu basea
- Wildfly zerbitzaria
- Hardware: 1 GB RAM

6.3.2. OpenSSL

OpenSSL protokoloa TLS eta DTLS protokoloetarako kode irekiko segurtasun tresna sendoa da. [18] Protokoloaren inplementazioa liburutegi kriptografiko indartsu batean oinarritzen da, era autonomoa ere erabil daitekeena. OpenSSL CAk, x.509 ziurtagiri digitalak, CRL zerrendak, giltzak, SSL/TLS bezero eta zerbitzariak eta hainbat segurtasun mekanismo gehiago sor ditzake, hau da, PKI azpiegitura oso bat eraiki dezake bakarrik Linux terminalean komandoak exekutatuz. OpenSSL instalatzeko bakarrik *sudo apt-get install openssl* komandoa exekutatu behar da Ubuntuko terminalean.

OpenSSL erabibiltzeko behar diren dependentziak hauek dira:

- Libssl eta libcrypto liburutegiak TLS oinarriak edukitzeko.

6.3.3. Hautaketa

Behin bi alternatibak azalduta proiektuan egokien funtzionatuko duena aukeratu behar da. Horretarako, lehen esan bezala, hautaketa irizpideak deskribatu behar dira. Ondoren, hautaketa irizpideak kontuan hartuz bi alternatibak ebaluatuko eta aukeratuko dira. Honako hauek dira kasu honetako irizpideak:

1. Dependentziak: Dependentzia gutxien duen alternatiba hobeto ebaluatuko da.
2. Hardware eskakizunak: Hardware eskakizun gehien behar duen alternatiba izango da txarrago ebaluatuko dena.

3. Instalazio erraztasuna: Instalatzeko errazagoa den softwarea puntuazio altuagoa izango du.
4. Komunitatearen laguntza: Alternatiba bakoitzaren inguruko Interneten dagoen komunitatearen laguntza baloratuko da. Komunitatean laguntza gehiago duen alternatiba hobeto baloratuko da.

Irizpideak edukita, alternatiben ebaluazioa egingo da:

	EJBCA	OpenSSL
Dependentziak (%40)	5	9
Hardware eskakizunak (%30)	5	9
Instalatzeko erraztasuna (%20)	7	9
Komunitatearen laguntza (%10)	4	7
TOTALA	5,3	8,8

3. Taula. EJBCA vs OpenSSL

Taulan ikusi ahal denez, OpenSSL aukerarik hoberena da PKI azpiegitura sortzeko. Laburbilduz, OpenSSL bakarrik Linux terminal bat behar du bere lana egiteko, beraz, proiektuaren CA, ziurtagiriak eta CRLa sortzeko alternatibarik egokiena da.

7. ARRISKUEN ANALISIA

Dokumentuaren atal honetan proiektuaren garapenaren zehar egon ahal diren arriskuak aztertuko dira. Lehenik, aipatutako arriskuak identifikatu egin behar dira. Identifikatu eta gero, hauek agertzeko probabilitate-eragin matrizea egingo da, hau da, arriskua agertzeko probabilitatea kalkulatuko da eta arriskua agertzen bada proiektuan zer eragin izango duen definituko da. Azkenik, arrisku bakoitza saihesteko hartuko diren irtenbideak azalduko dira.

7.1. Arriskuak

Hasteko, proiektuan gertatu daitezkeen lau arrisku identifikatu dira eta arrisku hauek sailkatzeko probabilitate-egoera matrizea egingo da. Hauek dira, identifikatu diren lau arriskuak:

1. **Sarearen arkitektura txarto diseinatzea. (A1)**

   Lehen arriskua sarearen arkitektura txarto diseinatzea da. Arkitektura txarto diseinatzen bada hainbat arazo sor daitezke, hala nola, proiektuaren ebazpena asko konplikatzea edo denbora galtzea arkitektura berriro diseinatzeko. Beraz, arrisku hau edonola ekiditea beharrezkoa da proiektua arrakastatsua izateko.

2. **Plangintza atzerapenak.(A2)**

   Edozein proiektutan gertatu ahal den arriskuetako bat plangintza atzerapenak dira eta beti kontsideratu behar dira. Plangintza txarto eramaten bada eta atazak atzeratu egiten badira probabilitate handia dago proiektuaren bukaerako epera ez heltzeko.

3. **Proiektuaren helburua ez lortzea. (A3)**

   Arriskurik larriena eta eragin handia izango lukeena proiektuaren helburua ez lortzea izango litzateke. Honek proiektuaren porrota suposatuko luke eta beharrezkoa da hau ez gertatzea. Eragin oso handia izan arren probabilitatea baxua da honako arrisku mota gertatzeko.

4. **Informazioaren galera. (A4)**

   Bere izena esaten duen moduan proiektuan informazioaren galera gertatu daiteke, hau da, proiektua egiten den bitartean amaitutako atazak galtzea. Adibidez, programa bat amaituta dagoenean hau galtzea. Beraz, berriro hasieratik hasi beharko litzateke.

Behin arriskuak identifikatuta, lehen aipatu denez, probabilitate-eragin matrizea azalduko da:

		ERAGINA				
		10%	30%	50%	70%	90%
PROBABILITATEA	10%					A4
	30%				A1	A3
	50%			A2		
	70%					
	90%					

4. Taula. Probalitate-eragin matrizea

Taulan ikusten denez, arrisku gehien sortzen duen arazoa proiektuaren helburua ez lortzea da (A3). Hurrengoak, sarearen arkitektura txarto diseinatzea (A1) eta plangintza atzerapenak (A2) dira. Azkenik, eragin oso handia duena baina oso probabilitate txikia proiektuaren informazioaren galera (A4) da.

7.2. Kontigentzia neurriak

Orain, arriskuak identifikatuta eta probabilitate-eragin matrizea deskribatuta arrisku hauek saihesteko irtenbideak azalduko dira:

1. **Sarearen arkitektura txarto diseinatzea**
   Proiektuan sarearen arkitektura txarto diseinatzea gertatu daiteke. Honek oso eragin handia suposatuko litzateke proiektu osoaren egitura aldatuko lukeelako. Gainera, arkitektura txarra edukitzeak proiektuaren konplexutasuna asko handituko luke.
   Arrisku hau saihesteko modurik onena ikasleak zuzendariekin bilerak edukitzea izango litzateke, hau da, behin ikasleak sare arkitektura diseinatuta zuzendariekin geratu behar du haiek sare ontzat emateko.

2. **Plangintza atzerapenak**
   Plangintza atzerapen arazoak ekiditeko gomendagarria izango litzateke lan-plan on bat diseinatzea eta haren epeak betetzea. Hala ere, zaila da epeak egunean eramatea, beraz, ideia ona izango litzateke lan-planaren azken data proiektua entregatu baino pare bat aste lehenago jartzea. Horrela, edozein arazo agertzen da denbora nahiko egongo litzateke hau konpontzeko eta epea pasatu barik.

3. **Proiektuaren helburua ez lortzea.**
   Arriskua saihesteko modua sinplea da. Hilabetero zuzendariekin bilera bat edukitzea oso gomendagarria izango lirateke, horrela, zuzendariek ikaslearen aurrerapenak ikusi ahal dituzte eta edozein arazo ikusten badute ikasleak arazoa konpontzeko denbora izango du.

4. Informazioaren galera.

Proiektuaren informazio galera latza izango litzateke proiektua egiten den bitartean. Adibidez, proiektuaren dokumentazioa galtzea edo programa informatikoak galtzea eragin oso handia izango luke guztia hasieratik hasi egin behar delako.

Hala ere, probabilitate baxua dago hau gertatzeko. Gaur eguneko, ordenagailuak nahiko fidagarriak dira, hala ere, ekintza gehiago egin ahal dira e, hala nola, ataza bat amaitzen denean disko gogor batean gorde edo proiektu guztia hodeian gorde daiteke. Horrela, arazoa agertzen bada proiektuaren backupak egongo lirateke puntu horretatik jarraitzeko.

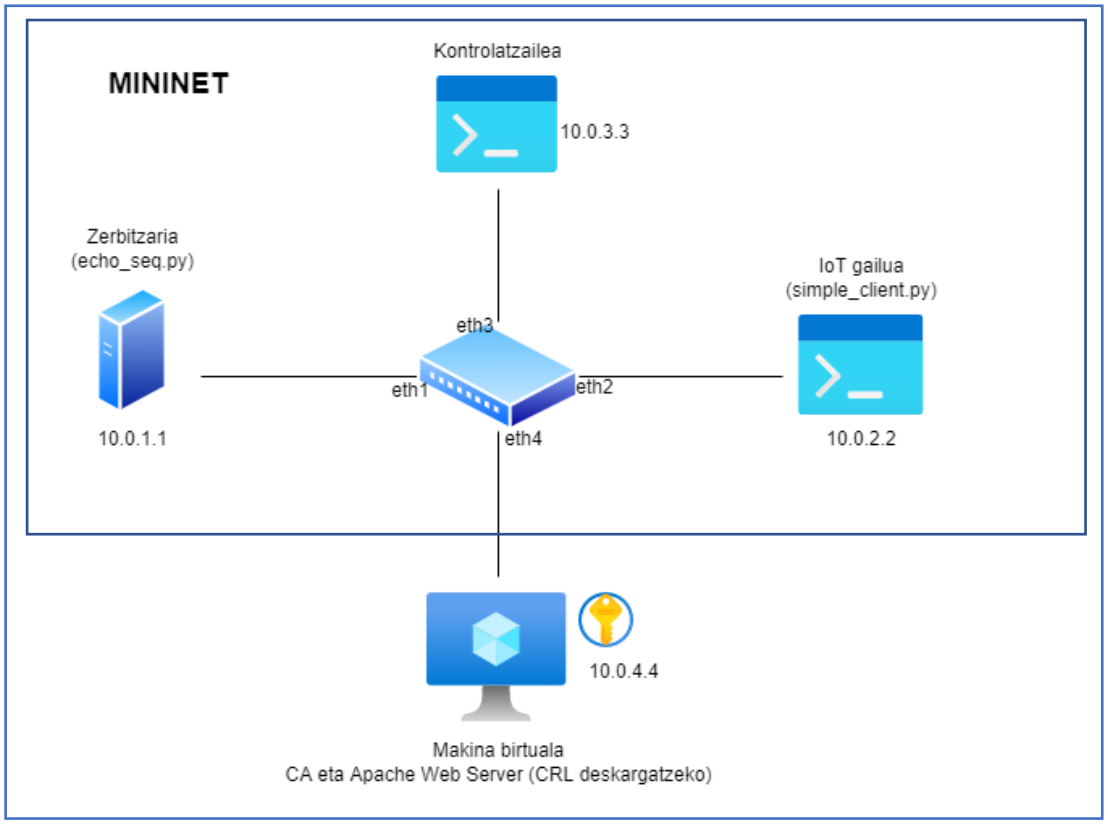
8. EBAZPENA

Txostenaren atal honetan proiektua aurrera egiteko eman diren pausuen ebazpena azalduko da. Azalpena bitan banatuko da; alde batetik, sarearen diseinua eta sarean parte hartzen duten elementuak nola inplementatu diren azalpena egingo da. Eta beste aldetik, DTLS handshakea maketan nola egingo den deskribatuko da.

8.1. Sareren arkitektura

Sarearen arkitekturari dagokionez, maketa sei elementurekin osatuta dago. 8. irudian elementu guztiak ikusi daitezke. Lehenik, IoT gailua dago non DTLS handshakea hasiko duen, hau da, bezeroa. Bigarren elementua, bezeroak konektatu nahi den zerbitzaria izango da. Sarearen erdian dagoen elementua, switch programagarria da, paketeak manipulatzeko eta nahi duen bezala saretik bideratzeko. Laugarrena, kontrolatzaile da eta honek switcha kontrolatuko du. Lau elementu hauek Mininet sare birtual baten barnean egongo dira eta Mininet SDN inguruneetan sare hedapenak eta testak egitea ahalbidetzen duenez, alternatiben analisian erabaki denez, aukera oso egokia da.

Hurrengo bi elementuak, CA (Ziurtagiri autoritatea) eta honen CRLa gordeko duen zerbitzaria izango dira. Kasu honetan, bi elementuak makina birtual batean egongo dira, hau da, ez dira hedatuko Mininet sarean, orduan, makina birtualaren eta Mininet sarearen arteko konexioa sortuko da DTLS prozedimendu osoa egitea lortzeko. Hurrengo irudian, sarearen arkitektura ikusi ahal da:



8. Irudia. Sare arkitektura

Sarearen arkitekturaren azalpena egiteko, lehenik, makina birtualean dauden elementuak argituko dira (CA, CRL biltegi zerbitzaria). Ondoren, Mininet sarean egongo diren elementuen deskribapena eta inplementazioa azalduko da (IoT gailua, zerbitzaria eta switcha) .

8.1.1. Makina birtualaren elementuak

Aurreko atalean esan denez, makina birtualean Mininet sarean dauden IoT gailuaren eta zerbitzariaren ziurtagiri digitalak hornituko dituen CA egongo da eta CA horrek CRL bat izango du, hau da, makina birtualean Apache web zerbitzari bat iraunkorki exekutatzen arituko da eta Apache zerbitzarian CA horren CRL zerrenda argitaratuta egongo da. Horrela, beste gailuek web orrialde horretara jo dezakete CRLa deskargatzeko ziurtagirien errebokazio egoera konprobatzeko.

Makina birtualaren sare konfigurazioa hurrengo taulan ikusi daiteke:

GAILUA	IP HELBIDEA	MAC HELBIDEA	SWITCH PORTUA
CA eta Apache zerbitzaria	10.0.4.4	08:00:00:00:01:11	root-eth0

5. Taula. Makin birtualaren sere konfigurazioa

Gehitu behar da, DTLS hanshake prozesuan switch programagarriaren kontrolatzaileak Apache web zerbitzariari Hypertext Transfer Protocol (HTTP) eskaera bat bidaliko dio CRLa eskuratzeko asmoz. Apache zerbitzariak eskaera honi erantzungo dio CRL zerrendaren informazioarekin non zerrenda .crl.pem luzapena izango duen.

Txostenaren hurrengo atalean, inplementazioan erabili diren CAren, gailuen ziurtagiri digitalen eta CRLaren sorkuntza azalduko da. Gainera, CAk hornitzen duen ziurtagiri digital baten errebokaketa ikusiko da.

8.1.1.1. CA eragiketak

Alternatiben analisian aipatu da, CA, ziurtagiriak eta CRLa OpenSSL segurtasun tresnaren bidez eratuko direla. Beraz, openssl erabiliz honako pausuak eman dira inplementazioan erabili diren segurtasun elementuak sortzeko [19]:

1. **CAren eraketa.**
    a. **CAren giltza pribatua eratu**: CAren giltza pribatua eratzeko (cakey.pem) AES 256-bit kriptografia erabili da 4096 bitekin giltza oso segurua izateko. Gainera, cakey.pem fitxategia makina birtualeko karpeta seguru batean biltegiratu da non bakarrik makina birtualeko erabiltzailea irakurtzeko aukera izango duen. Hurrengo komandoa exekutatuz CAren giltza pribatua eratzen da:

```
openssl genrsa -aes256 -out private/cakey.pem 4096
```

    b. **CAren ziurtagiria eratu**: Sortu den CAren giltza pribatuarekin CAren ziurtagiria eratuko da. Ziurtagiriak eratzeko openssl.cnf konfigurazio fitxategia erabiliko da. Fitxategi honetan ziurtagiri digitalak eratzeko konfigurazioa egongo da, beharrezko luzapenekin eta hemen CAren informazioa gordeko da, horrela, CAk beste ziurtagiri batzuk sinatu ahal izango (Eranskinetan). Kasu honetan, v3_ca

luzapena erabili da. Hurrengo komandoan ikusi ahal da ziurtagiria x509 motakoa dela eta 7300 egunetan erabilgarri egongo dela. Gainera, sha256 hash funtzio kriptografikoa erabili da sinatze algoritmo bezala. CAren izena CAroot izango da. Hurrengo komandoa exekutatuz CAren ziurtagiria eratzen da:

```
openssl req -config openssl.cnf -key private/ca.key.pem -new -x509 -days 7300 -sha256
-extensions v3_ca -out certs/ca.cert.pem
```

2. **CA bidez IoT gailuari eta zerbitzariari ziurtagiri digitalez hornitu**.
   Kasu honetan, IoT gailuaren eta zerbitzariaren ziurtagiri digitalak eratzeko prozedura berdina da, beraz, bakarrik zerbitzariaren ziurtagiri digitalaren eraketa azalduko da. [20]

   a. **Giltza eratu:** Zerbitzariaren giltza pribatua eratzeko kasu honetan ere, 4096 bitekoa izango da *eta server-1.key.pem* izena izango du. Hurrengo komandoa exekutatuz zerbitzariaren giltza pribatua eratzen da:

```
openssl genrsa -out server-1.key.pem 4096
```

   b. **Certificate Signing Request (CSR) eratu zerbitzariaren giltza erabiliz**: CSR fitxategian ziurtagiriaren informazioa egongo da, hala nola, herrialdea, autonomia-erkidegoa, herria, korporazioa eta garrantzitsuena dena izen arrunta (*Common Name*). Common Name ziurtagiriaren identifikatzailea da. Behean dagoen komandoa exekutatuz, esan den informazioa idazteko eskatzen da eta inplementazioa egiteko idatzi den informazioa honakoa da:
      i. Country: Spain
      ii. Region: Basque Country
      iii. City: Sopela
      iv. Company: Iker Alkorta
      v. Common Name: server-1

   Hurrengo komandoa exekutatuz zerbitzariaren CSRa eratzen da:

```
openssl req -config openssl.cnf -key certs/server-1.key.pem -new -sha256
-out certs/server.csr.pem
```

   Ziurtagiria eratu: Lehen aipatutako openssl.cnf konfigurazio fitxategiarekin zerbitzariaren ziurtagiria eratuko da, CAk sinatuta. Ikusi ahal denez, server_cert luzapena gehitzen da , urte bateko baliogarritasuna du eta sha256 kriptografia erabiltzen da ziurtagiria sortzeko. Hurrengo komandoa zerbitzariaren ziurtagiria eratzen da:

```
openssl ca -config openssl.cnf  -extensions server_cert -days 375 -notext -md sha256
-in crts/server.csr.pem -out certs/server-1.crt
```

3. **CRL zerrenda eratu.**
   CA eta zerbitzariaren ziurtagiriak sortuta, hurrengo pausua, CRL zerrenda eratzea da ziurtagirien errebokazio egoera konprobatzeko. CRLa eratzen denean, ez da egongo errebokaturiko ziurtagiririk, beraz, *No Revoked Certificates* esaldia adieraziko da. [21]

   Hurrengo komandoa exekutatuz CRL zerrenda eratzen da:

```
openssl ca -config /openssl.cnf  -gencrl -out crl/CAroot.crl.pem
```

4. **Ziurtagiri bat errebokatu.**
   Azkenik, ziurtagiria errobokatzea geratzen da. Ziurtagiri bat errebokatzen denean CRL zerrenda eguneratu egingo da ziurtagiriaren serie zenbakia jarriz, horrela, bezero batek ziurtagiri baten serie zenbakiarekin bere errebokazio egoera konprobatu ahal du zerrendan begiratuz. Honako komando hau exekutatu behar da ziurtagiri bat errebokatzeko:

```
openssl ca -config openssl.cnf  -revoke certs/server-1.crt
```

8.1.1.2. Apache Web zerbitzaria

Lehen esan denez, CAk sortu duen CRLa Apache Web Zerbitzaria erabiliz argitaratuko da. Orduan, Apacheren konfigurazio fitxategietan CRL fitxategia (CAroot.crl.pem) dagoen karpetara apuntatzeko konfiguratu behar da. Kasu honetan, Apache Web Zerbitzariaren konfigurazioa asko ez aldatzeko CRL fitxategia karpetaz aldatu da, hau da, CAroot.crl.pem fitxategia */var/www/SRC* izeneko karpetan biltegiratu egin da, beraz, */etc/apache2(sites-enabled/000-default.conf* fitxategian honako lerroa idatzi da:

```
DocumentRoot /var/www/SRC
```

Orduan, CRLa web orrialde batean argitaratuta egongo da eta inplementazioan, switch programagarriaren kontrolatzailea web orrialde honetara joko du CRLa deskargatzeko. Ondoren, errebokazio egoeraren konprobazioa egingo du zerbitzariaren ziurtagiriaren serie zenbakia erabiliz. Orrialde honen URLa *http://10.0.4.4:80/CAroot.crl.pem* izango da.

8.1.2. Mininet sarea

Lehen esan denez, Mininet SDN hedapenak egitea ahalbidetzen duen sare emulazio sistema da eta IoT gailua, zerbitzaria, switcha eta kontrolatzailea Mininet sare horretan simulatuko dira. Sare birtual honetako gailuen konfigurazioa hurrengo taulan adieraziko da:

GAILUA	IP HELBIDEA	MAC HELBIDEA	SWITCH PORTUA
Zerbitzaria	10.0.1.1	08:00:00:00:01:11	eth1
IoT gailua	10.0.2.2	08:00:00:00:02:22	eth2
Kontrolatzailea	10.0.3.3	08:00:00:00:03:33	eth3

6. Taula. Mininet sareko gailuen sare konfigurazioa

Gehitu behar da, switcharen laugarren portua makina birtualean egongo diren CA eta CRL zerrenda dagoen web zerbitzarira konektatuta egongo dela.

Mininet sareko gailuekin hasi aurretik, DTLS hanshakea eta datu transmisioa inplementatzeko beharrezkoak diren liburutegiak eta programak aipatu behar dira.

8.1.2.1. DTLS hanshakea inplementatzeko beharrekoak

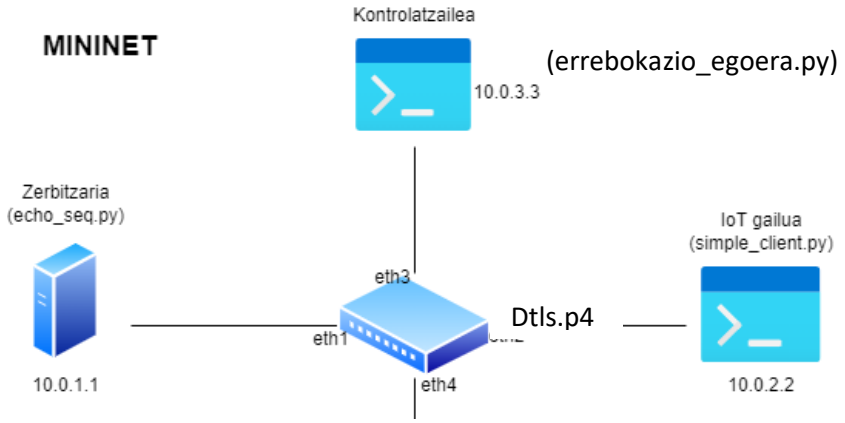
Proiektuaren soluzioa aurrera eramateko DTLS hanshakea simulatzen duen programa multzoa behar da. Proiektu honetan inplementazioa *https://github.com/mobius-software-ltd/pyton3-dtls* GitHub errepositorioa erabili da. Errepositorio honetako programak Python lengoaiaz programatuta daude non Python 3.6 edo altuago dauden bertsioekin erabili ahal den.

Errepositorio honetan dauden programekin lan egin ahal izateko, alternatiben analisian aukeratu den crypthography liburutegia erabili da (Python 3.7). Liburutegi honek segurtasun aukera asko eskaintzen ditu, adibidez, ziurtagirien balidazioa eta ziurtagirien errebokazio egoeraren konprobazioa. Beste aldetik, DTLS hanshakea simulatzen duten programek OpenSSL Python liburutegia ere behar dute funtzionatzeko beraz instalatu behar da.

Hainbat alditan aipatu da DTLS handshakea IoT gailuaren eta zerbitzariaren artean egingo dela eta ziurtagiriaren errebokazio egoera switchean konprobatuko dela. Beraz, lau programa dira prozedura honetan parte hartu dutenak: IoT gailuko programa, zerbitzariaren programa, switch programagarriaren P4 programa eta kontrolatzailearen programa. Programa hauek hurrengo ataletan azalduko dira baian GitHub errepositorioa eskaintzen duten programak hauek dira:

1. Simple_client.py: IoT gailua simulatuko du eta DTLS hanshakea hasi egingo du.
2. Echo_seq.py: Zerbitzaria simulatuko du.
3. Errebokazio_egoera.py: Switcharen kontrolatzailean exekutatuko den programa izango da ziurtagirien errebokazio egoera konprobatzeko.
4. Dlts.p4: Switch programagarrian egongo den programa non pakete zehatz batzuk manipulatuko dituen.

Hurrengo irudian Minineten kokatutako gailu bakoitzen programak non exekutatuko diren ikusi ahal da:

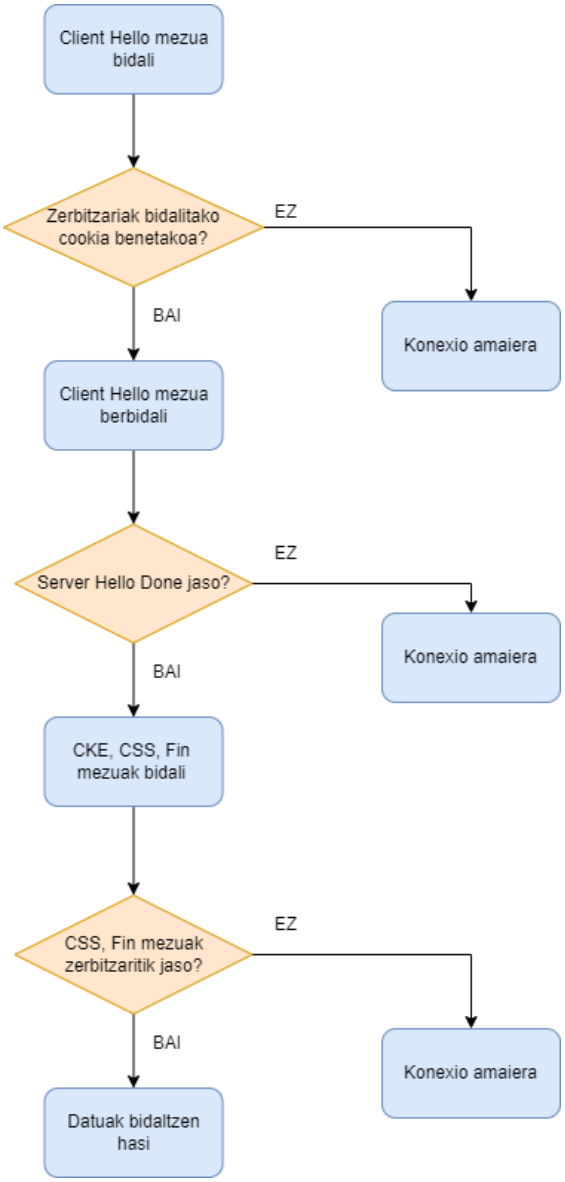


9. Irudia. Mininet sarearen gailuen kokapena

8.1.2.2. IoT gailua (Bezeroa)

Proiektuaren sarreran esan denez, bezeroa eta DTLS konexioa hasiko duen gailua IoT dispositibo bat izango da, hala nola, sentsore bat edo alarma bat. IoT gailuak erabiltzailearen edo ingurunearen informazioa bidaltzeko helburua du eta kasu honetan, Mininet sarean dagoen zerbitzariari bidaliko dio. Transmititutako informazioa modu seguruan bidaltzeko bikoteak autentikatu eta giltza bat adostu behar dute DTLS handshakea erabiliz, gainera, IoT gailua switch programagarriari zuzenean konektatuta egongo da.

IoT gailua simulatzen duen programa, aurreko atalean azaldu den simple_client.py oinarrituta dago. Programa honek DTLS konexioa hasiko du zerbitzariarekin, hau da, 10.0.1.1 IP helbidea duen gailuarekin. Beraz, simple_client.py Client Hello mezuarekin hasiko da, ondoren, zerbitzariak bidali dion cookiearen konprobaketa egingo du DoS arazoa ekiditzeko. Cookiea egokia bada berriro Client Hello mezua transmitituko du, cookiea gehituz. Hurrengo pausuak, zerbitzariaren artean eta switch programagarriaren artean izango dira (ziurtagiriaren fidagarritasun eta CRL konprobazioak). Switchak egiaztapenak egin eta gero, bezeroari jakinaraziko dio eta bezeroak komunikazioan erabiliko den giltza adostuko du zerbitzariarekin. Giltza adostuta egonda, bezeroak informazioa bidaltzen hasi daiteke. Hurrengo diagrama da programaren portaera azaltzen duena:

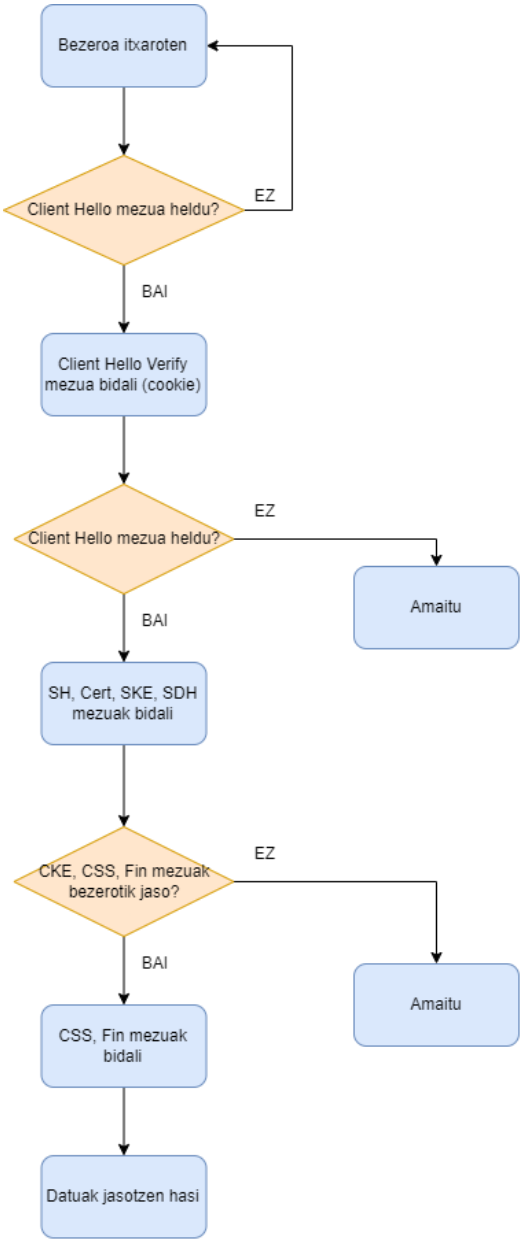


10. Irudia. simple_client.py programaren diagrama

8.1.2.3. *Zerbitzaria*

Mininet sarean egongo den zerbitzaria DTLS handshakean parte hartuko duena izango da. Laburbilduz, bezeroak (IoT gailua) datuak bidali nahi dituen gailua izango da, hau da, IoT gailua ahots laguntzaile bat bada *Alexa* adibidez, zerbitzaria Amazon konpainiako zerbitzariak izango dira.

Zerbitzariak, lehenik, Client Hello mezua hartzen duenean, Client Hello Verify mezua bidali beharko dio, lehen esan bezala, DoS arazoak ekiditzeko. Ondoren, bezeroari bere ziurtagiri digitala bidaliko dio eta switchak ziurtagiri konprobazioa egin eta gero bikoteak komunikazioan zehar erabiliko duten giltza adostuko dute. Honako hau da echo_seq.py programaren diagrama:



11. Irudia. echo_seq.py programaren diagrama

8.1.2.4. Switch programagarria

Switch programagarriari dagokionez, bere datu planoa P4 lengoaian idatzitako programa batekin konfiguratu beharko da, hau da, switchak jakin behar du Server Hello (SH), Certificates (Cert), Server Key Exchange (SKE) eta Server Hello Done (SHD) hanshake mezua noiz heltzen zaion eta mezu hau kontrolatzaileari bidali beharko dio honek ziurtagirien errebokazio egoera konprobatzeko. Laburbilduz, DTLS handshake mezua switchera heltzean datu planoa programatu behar da eth3 (kontrolatzailera lotura) portutik mezua birbidaltzeko eta kontrolatzaileak konprobazioa egin ondoren, berriz switchera bidaliko dio, honek eth2 portutik bezerora garraiatzeko (ziurtagiria errebokatuta ez badago).

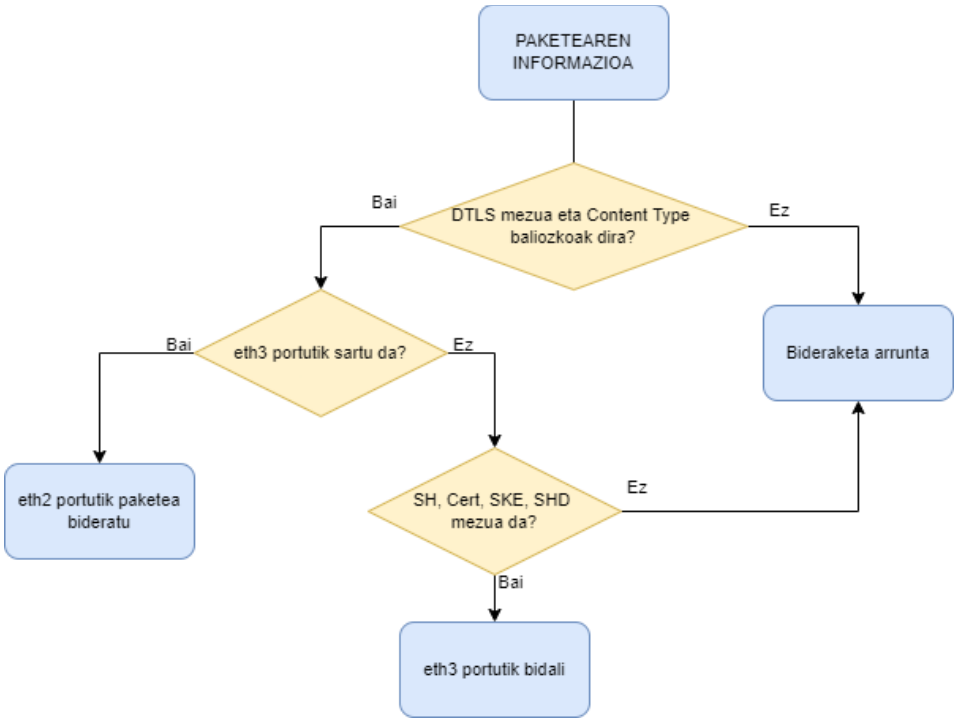
P4 eta switcherako tresnak, lehen Mininet instalatzeko erabili den GitHub errepositoriak eskaintzen ditu. Beraz, ez da instalazio gehiagorik egin behar.

Switcharen konfigurazioari begira, V1Model elementuen deskribapen indibiduala ikusikoa da, hau da, parser, Ingress match-action pipeline, egress match-action pipeline eta despaser elementuak.

1. **Parser:** Switchera paketea heltzen denean parserrak paketearen goiburuak atera beharko ditu zein DTLS mezu heldu diren jakiteko. Oso garrantzitsua da switchak zein DTLS mezu heldu diren zehaztea ea zein interfazetik bideratu behar duen jakiteko. Lehen esan bezala, switchak lau DTLS handshake mezu antzeman behar ditu: Server Hello, Certificates, Server Key Exhange eta Server Hello Done.
   Mezu hauek antzemateko parserrak lehenik, DTLS gainean lan egiten duten protokoloen goiburuak aterako ditu, hau da, Ethernet, IP eta UDP goiburuak. Ondoren, DTLS mezura heltzean switchak bi prozesu egingo ditu. Alde batetik, DTLS mezu osoa aterako du eta beste aldetik, DTLS handshake mezu desberdinetan Handshake Type eremua aterako du. Horrela, switchak zein DTLS handshake mezu mota den jakingo du eta horren arabera, dagokion funtzioa exekutatuko du. Hanshake Type eremua 8 bitez osatuta dago eta hauek dira (kode hamaseitarrez):
   - Server Hello: 0x02
   - Certificates: 0x0b
   - Server Key Exhange: 0x0c
   - Server Hello Done: 0x0e
2. **Ingress match-action pipeline:** Paketeko goiburuak eta eremuak atera eta gorde ondoren, hauen informazioa ikusi behar da ea intereseko paketea den zehazteko. Orduan, bi egoera mota egon daitezke:
   1. **Intereseko paketea heltzea:** DTLS paketea heltzen bada eta SH, Cert, SKE, SHD eremuak dituen paketea bada switchak programatutako algoritmoa exekutatuko du. Kasu honetan, bi azpiegoera egon daitezke.
      a. **Paketea switcharen hirugarren portutik sartzea:** Pakete mota hau switcharen hirugarren portutik sartzen bada switchak automatikoki jakingo du paketea eth2 portutik bideratu beharko duela. Hau da, kontrolatzaileak CRL konprobazioa egin du eta ziurtagiria errebokatuta ez dagoela zehaztu du, beraz, paketea bezeroari helaraziko dio. Eth3 portutik sartu eta eth2 portutik atera. (Kontrolatzailea-switch-IoT gailua)

   b. **Paketea switcharen lehenengo portutik sartzea:** Kasu honetan paketea switcharen eth1 portutik heltzen bada eta SH, cert, SKE edo SHD eremuak badira switchak paketeak eth3 portutik bidaliko ditu. Laburbilduz, kontrolatzailera garraiatuko ditu honek CRL konprobazioa egiteko.
2. **Intereseko paketea ez denean:** Intereseko paketea heltzen ez denean, adibidez, TCP paketea, HTTP paketea (CRL zerrenda deskargatzen denean) edo gainerako DTLS mezuak heltzen direnean, switchak portaera arrunta izango du eta paketa helburuko gailura bideratuko du.

Hurrengo irudian Ingress match-action pipeline algoritmoaren egoera diagrama ikusi daiteke:



12. Irudia. Ingress match-action pipeline algoritmoaren diagrama

3. **Egress match-action pipeline:** Kasu honetan, egress match-action pipelinek ez du ezer berezirik egingo.
4. **Despaser:** Despaserrak bere ohiko funtzioa egingo du, modifikatu egin diren goiburuei bere payload determinatua itsatsiko die, eta horrela, paketea osatuko da helburura bideratzeko asmoarekin.

8.1.2.5. Kontrolatzailea

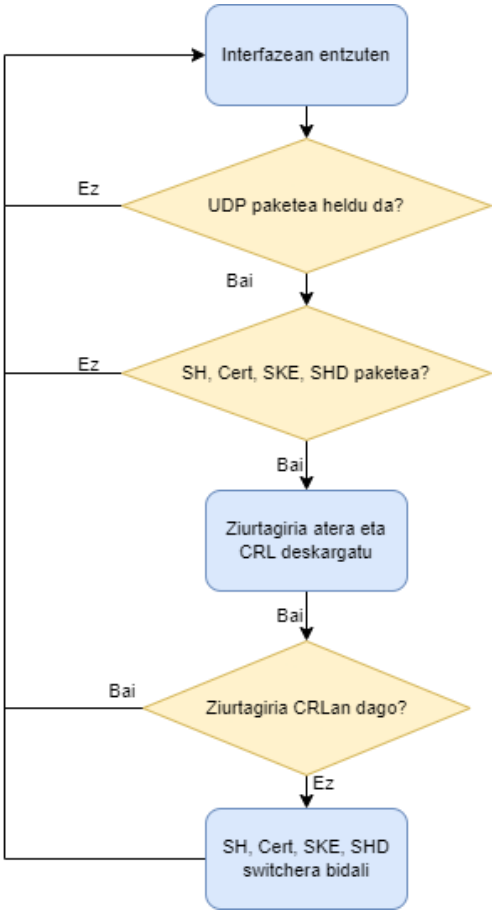
Azkenik, Mininet sarean dagoen azken gailua switch programagarriaren kontrolatzailea izango da. Kontrolatzaileak zerbitzariaren ziurtagiriaren errebokazio egoera konprobatuko du CRL zerrenda makina birtualean (10.0.4.4) dagoen Apache Web Zerbitzaritik. Kontrolatzailea Python lengoaian idatzitako programa bat da non *cryptography* eta *scapy* liburutegiak dira liburutegi garrantsitzuenak.

Alde batetik, cryptography liburutegiak ziurtagirien baliagarritasuna eta ziurtagiria ea CRL zerrendan dagoen konprobazioa egiten duten funtzioak eskaintzen ditu. Beste aldetik, scapy liburutegiak paketeetatik eremuak eta bitak ateratzen laguntzen du, hau da, DTLS paketetik zerbitzariaren ziurtagiria ateratzen lagunduko du.

Kontrolatzaileak hainbat funtzio ditu prozesua aurrera eramateko eta honako hauek dira:

1. Paketeen zain itxaron: Kontrolatzailea dagokion portutik itxaroten egongo da switchak paketeak bidali arte.
2. Paketeen helduera: Paketea kontrolatzailera heltzean paketearen goiburuak aterako dira eta paketea UDP gainean heldu bada ziurtagiri konprobazioa egingo da.
3. Ziurtagirien ateratzea: DTLS eremuetara heltzean zerbitzariaren ateratzea egingo da, horrela, bere serie zenbakiarekin ea CRL zerrendan dagoen konprobatu ahal da.
4. CRLaren deskarga: Behin zerbitzariaren ziurtagiria edukita kontrolatzaileak 10.0.4.4 makinan dagoen web zerbitzarira konektatuko da eta CRL deskargatu egingo du. CRLa deskargatzean bakarrik ordu bat egongo da kontrolatzailearen cachean, beraz, ordu bat pasatzerakoan ezabatu egingo da eta berriz deskargatu beharko da.
5. Ziurtagiria CRLan bilatu: Hurrengo pausua, ziurtagiria CRLan badagoen bilatzea da. Horretarako, cryptography Python liburutegia eskaintzen duen funtzioa erabiliko da. Zerbitzariak duen ziurtagiriaren serie zenbakia hartuko da eta CRL zerrendan badago ziurtagiria errebokatuta dago eta prozesua amaituko da. Berriz, ziurtagiraren serie zenbakia CRLan ez badago prozesua jarraituko du.
6. Paketeak berbidali: Esan den moduan, ziurtagiria CRLan ez badago kontrolatzaileak paketeak bere interfazetik switchera bidaliko dio gero honek IoT gailura bidaltzeko, hau da, SH, Cert, SKE, SHD paketeak.

Hurrengo irudian, kontrolatzailek exekutatzen duen programaren portaera azaltzen duen diagrama adieraziko da:



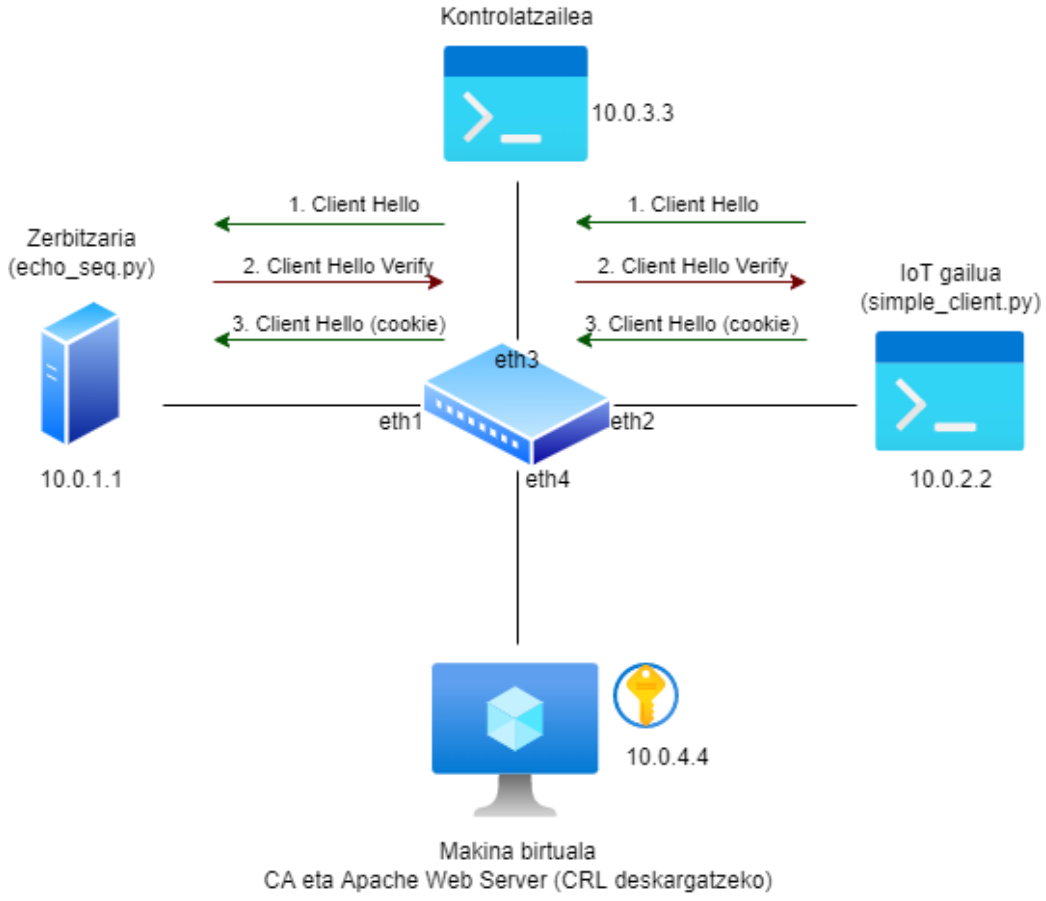
13. Irudia. Kontrolatzaile programaren diagrama

8.2. DTLS handshakea maketan

Txostenaren puntu honetan sarearen maketan DTLS handshakea nola egingo den azalduko da. Hau da, gailu bakoitzak handshakearen zer pausu egingo dituen hobeto ikusiko da eta irudiak erabiliko dira inplementazioa argiago ulertzeko.

8.2.1. Client Hello mezuak

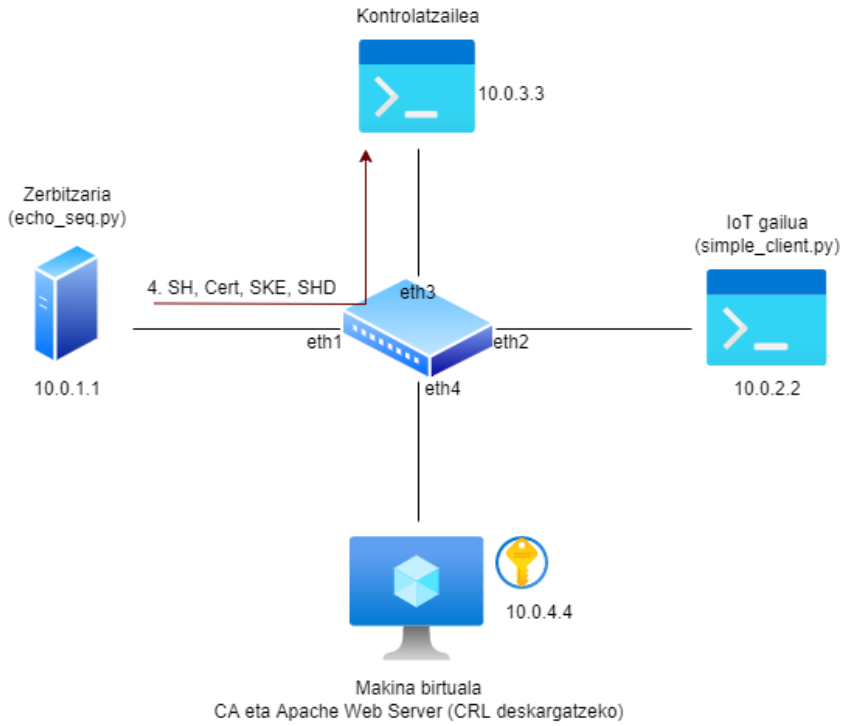
DTLS handshakearekin hasteko lehenik IoT gailuak Client Hello mezuarekin agurtuko dio. DTLS handshakea denez eta UDP gainean lan egiten duenez zerbitzariak Client Hello Verify mezuarekin erantzungo dio cookiea bidaltzen. Ondoren, bezeroak cookiea birbidaliko dio zerbitzariari, horrela, zerbitzariak DoS arazoak ekiditen ditu. X. irudian Client Hello mezuen prozedura ikusi ahal da proiektu honen maketan inplementatuta.



14. Irudia. Client hello pausua

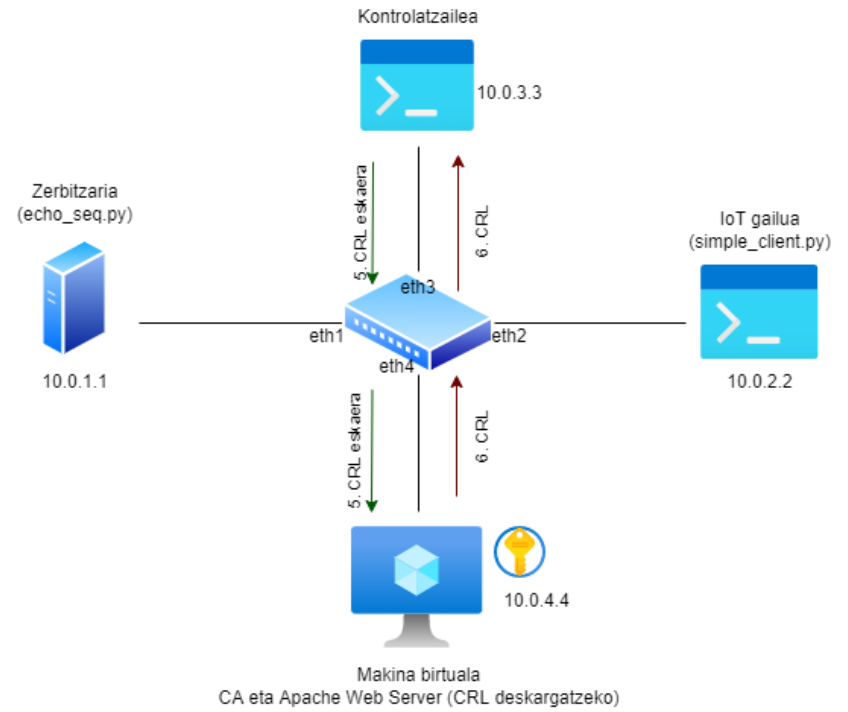
8.2.2. SH, Cert, SKE, SHD mezuak

Txosten honetan hainbat alditan esan denez, garrantzia handien duten mezuak SH, Cert, SKE eta SHD dira manipulatu egingo diren paketeetan daudelako. Mezu hauek zerbitzariak bidaliko ditu bezeroari baina bidaiaren erdian switchak hartuko ditu bere kontrolatzailera bideratzeko. Kontrolatzaileak informazioa duenean ziurtagiria aterako du eta CRLa deskargatuko du makina birtualetik. Ondoren, ziutagiria ea CRLan dagoen begiratuko du eta ez badago paketa berriz IoT gailura birbidaliko da. Hurrengo irudietan hainbat alditan azaldutako prozesua ikusiko da:



15. Irudia. zerbitzari-kontrolatzaile pausua

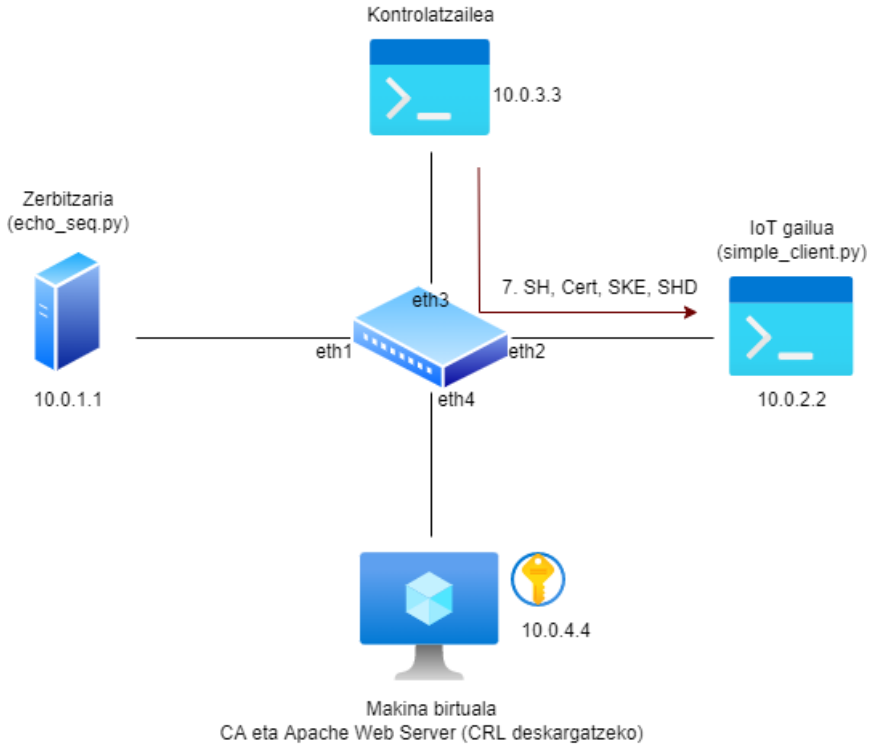
X irudian ikusten denez, zerbitzariak paketea bidaltzen du eta switchera heltzean honek P4n programatu den algoritmoa exekutatzen du bere kontrolatzailera bideratzeko, hau da, eth3 portutik bidaltzen du.



16. Irudia. CRL deskarga

Paketea kontrolatzailera heltzean honek ziurtagiri digitala ateratzen du, ondoren, HTTP protokoa erabiliz makina birtualean dagoen Apache Web zerbitarian dagoen CRLaren eskaera egiten du. Honek, CRLa duen fitxategiarekin erantzungo dio. Azkenik, kontrolatzaileak ziurtagiriaren errebokazio egoera konprobatko du CRLan begiratzen (X. irudia)

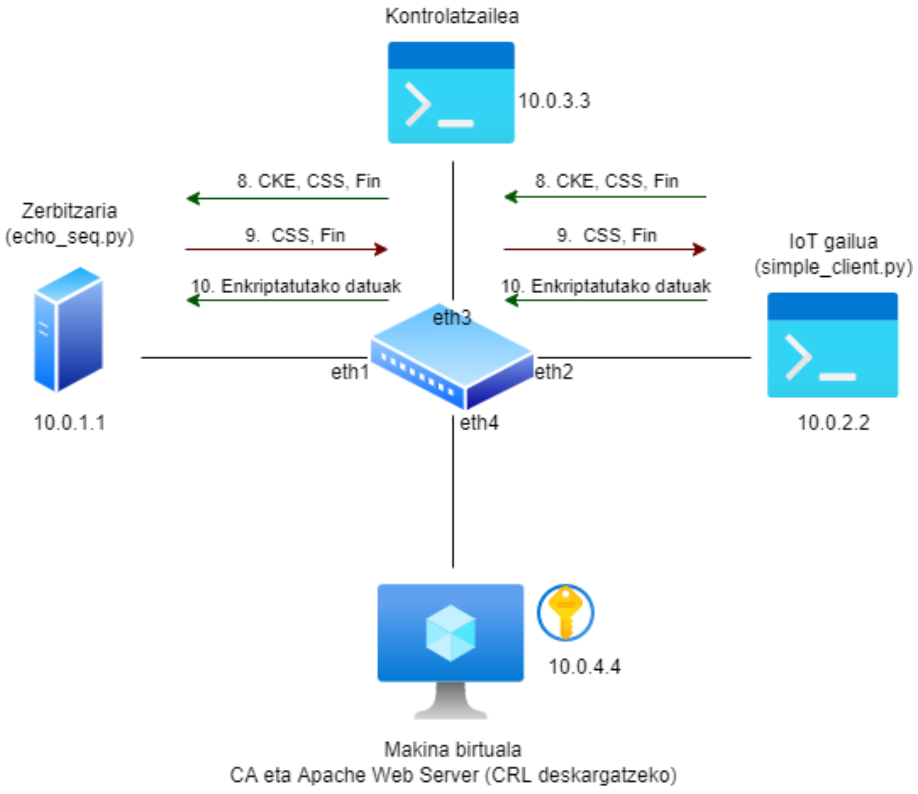
Azkenik, beheko irudian ikusi ahal denez, ziurtagiri digitalaren serie zenbakia CRLan ez badago kontrolatzaileak SH, Cert, SKE eta SHD mezua switchera bideratzen du eta aureko ataletan esan den bezala, paketa switcharen eth3 portutik heltzen bada automatikoki eth2 portutik irtengo da.



17. Irudia. Kontrolatzaile-bezeroa

8.2.3. CKE, CSS, Fin

Prozedimenduren azkeneko fasea, Client Key Exchange mezua eta amaierako mezuak trukatzean datza. Mezu hauek trukatzen amaitzean bezeroa eta zerbitzaria jadanik adostu dute komunikazioan erabiliko duten giltza eta bezeroak informazioa bidaltzen hasi daiteke era seguruan.



18. Irudia. CKE, CSS, Fin pausua

9. ERRENDIMENDU ANALISIA

Txostenaren atal honetan inplementazioan eman diren emaitzak aztertuko dira. Emaitzen analisia egiteko egoera desberdinak aukeratu dira eta haien azterketa indibiduala egingo da. Indibidualki analizatu ondoren, egoeren arteko konparazioa burutuko da errendimendu onena duen egoera aurkitzeko.

9.1. Errendimendu analisirako metodologia

Atal honetan errendimendua analizatzeko erabiliko den metodologia argituko da. Hala nola, zein egoerak egongo diren, zein parametro neurtuko diren eta zelan neurtuko diren parametro horiek.

Egoerei dagokionez, bi egoera nagusi dira bereizi direnak. Alde batetik, CRL deskarga eta konprobazioa kontrolatzaileak egiten duenean eta beste aldetik, bezeroak egiten duenean. Hala ere, proiektu honetan prozedimenduan kontrolatzailea dagoenean gehiago sakonduko da. Gehitu behar da, egoera bakoitzak zenbait azpi egoera izango dItela, hau da, egoera bakoitzean proba ezberdinak egingo dira haien arteko konparaketa egiteko. Hurrengo eskeman hobeto ulertuko dira zein egoera eta azpi egoera egongo diren analisian.

Egoerak

1. **Egoera**: CRL deskarga eta ziurtagirien errebokazio egoeraren konprobazioa switch programagarriaren kontrolatzailean egingo da (proiektuan inplementatu den soluzioa).
2. **Egoera**: CRL deskarga eta ziurtagirien errebokazio egoeraren konprobazioa IoT gailuan (bezeroan) egingo da.

Azpi egoerak

1. **Azpi egoera:** DTLS mezua (SH, Cert, SKE, SHD) fragmentatuta badago, hau da, UDP datagrama bat baino gehiagotan zatituta badago.
2. **Azpi egoera:** DTLS mezua UDP datagrama bakar batean enkapsulatuta badago.
3. **Azpi egoera**: CRL zerrendan bakarrik bi ziurtagiri errebokatuta badaude.
4. **Azpi egoera:** CRL zerrendan berrogeita bederatzi ziurtagiri errebokatuta badaude.
5. **Azpi egoera:** CRL zerrenda web zerbitzaritik deskargatu behar bada.
6. **Azpi egoera:** CRL zerrenda cachean badago, hau da, ez da web zerbitzaritik deskargatu behar.

Ikusten denez, hainbat azpi egoera daude baina analisi hau egiteko oinarrizkoak izango direnak lehenengoa, hirugarrena eta bostgarrena izango dira, hau da, DTLS mezua fragmentatuta egongo da, CRLan bi ziurtagiri egongo dira errebokatuta eta gainera, kontrolatzaileak CRL zerrenda web zerbitzaritik deskargatu behar izango du.

Emaitzekin hasi aurretik, emaitzen analisia egiteko hiru denbora kontuan hartuko dira, hau da, hiru denbora hauek izango dira Key Preformance Indicator-ak (PKI):

- DTLS handshake osoa egiteko beharrezko denbora.
- CRL deskargatzeko denbora.

- Kontrolatzaileak prozesuan parte hartzen duen denbora (paketea hartu + CRLa lortu + errebokazio konprobazioa + paketea birbidali).

9.2. Emaitzak

Esan denez, emaitzen analisi atalean lehenengo egoeran sakonduko da non errebokazio egoeraren konprobazioa kontrolatzailean egingo den, CRL bat erabiliz. Beraz, hurrengo azpi ataletan egoera honen azpi egoeren emaitzak ikusi eta analizatuko dira.

9.2.1. Fragmentazioa, CRL tamaina txikia eta deskargatu beharra - Kontrolatzailean

Kasu honetan, lehen aipatu denez, DTLS mezua UDP datagrama bat baino gehiagotan helduko da kontrolatzailera zerbitzariaren ziurtagiriaren tamaina handia delako, hau da, fragmentatuta helduko da. Gainera, kontrolatzaileak CRLa deskargatu behar du 10.0.4.4 IP helbidea duen web zerbitzaritik. Deskargatu behar duen CRLa tamaina txikia izango du; bakarrik bi errebokatutako ziurtagiri egongo dira. Hurrengo taulan, emaitzak ikus daitezke:

	Batez besteko denbora (ms)	Bariantza (ms)	Desbideratze estandarra (ms)
CRL deskarga	14,935	1,381	3,716
Kontrolatzailearen prozesatze denbora	24,568	0,138	0,124
Handshake guztia	180,960	2,402	49,008

7. Taula. Lehenengo egoeraren emaitzak

9.2.2. Fragmentaziorik ez, CRL tamaina txikia eta deskargatu beharra - Kontrolatzailean

Puntu honetan, aurreko atalean egin den proba bezalakoa da baina DTLS mezua kontrolatzailera fragmentuta heldu beharrean datagrama bakar batean heltzen da. CRL deskarga denbora berdina izango da baina kontrolatzailearen prozesatze denborak eta handshake guztiaren denborak ezberdinak izango dira.

	Batez besteko denbora (ms)	Bariantza (ms)	Desbideratze estandarra (ms)
CRL deskarga	14,935	1,381	3,716
Kontrolatzailearen prozesatze denbora	22,439	3,584	9,986
Handshake guztia	57,574	2,546	5,988

8. Taula.Bigarren egoeraren emaitzak

Taulan ikus daiteke, bataz besteko denborak askoz txikiagoak direla mezua fragmentatuta heltzen den egoeran baino. Logikoa denez, switch programagarriak eta kontrolatzaileak prozesaketa gutxiagoa egin behar dute UDP datagrama bakar bat prozesatu behar dutelako. Lehenengo egoeran, aldiz, DTLS mezua hiru UDP datagrametan zatituta heltzen da, beraz, gailuek karga konputazional handiagoa behar dute prozedimendu osoa egiteko. Gainera, kontrolatzaileak prozesuarekin hasteko hiru datagramak heldu arte itxaron behar du eta honek, denbora gehigarria ematen du.

9.2.3. Fragmentazioa, CRL tamaina txikia, CRLa cachean - Kontrolatzailean

Hirugarren kasu honetan, CRL zerrenda ez da web zerbitzaritik deskargatu behar, hau da, CRLa konputagailuaren cachean dago. Beraz, emaitzei begira, denbora txikiagoa lortzea itxaroten da deskarga prozedimendu osoa deuseztatzen delako. Hurrengo taulan, besteetan ez bezala, CRL deskarga denbora ez da kontuan hartuko.

	Batez besteko denbora (ms)	Bariantza (ms)	Desbideratze estandarra (ms)
Kontrolatzailearen prozesatze denbora	15,731	2,361	3,652
Handshake guztia	148,095	1,230	35,070

9. Taula. Hirugarren egoeraren emaitzak

Emaitza honetan, X. (lehenengo) taularekin konparatuz denbora baxuagoak lortu dira. Lehen esan denez, itxaroten zen denbora baxuagoak izatea CRL zerrendaren deskarga egiten ez delako, beraz, CRLa cachean izatea abantaila izan daiteke denborari dagokionez. Hala ere, ez da oso gomendagarria CRLa cachean denbora asko egotea CRLan segurtasun gaurkotzeak egon daitezkeelako eta cachean dagoen CRLa zaharkitua egon daitekeelako.

9.2.4. Fragmentazioa, CRL tamaina handia, CRLa deskargatu beharra - Kontrolatzailean

Kontrolatzailea prozesuan zehar egongo den azken proba honetan, DTLS mezua fragmentatuta egongo da eta CRLa zerbitzaritik deskargatu behar da. Aldatzen den alderdi CRL izango da. Kasu honetan, CA erabiliz berrogeita hamar ziurtagiri sortu eta berrogeita bederatzi errebokatu dira. Orduan, CRL zerrendan berrogeita bederatzi lerro gehiago egongo dira errebokatutako ziurtagirien serie zenbakiekin. CRLak hainbat ziurtagiri errebokatuta izatean kontrolatzaileak lerro guztien konprobazioa egin behar du zerbitzariaren ziurtagiria ez dagoela egiaztatzeko, beraz, kontrolatzailearen prozesaketa denbora handiagoa izango da eta honekin batera handshake denbora ere handituko da. Gainera, CRLaren deskarga denbora altuagoa izango da.

	Batez besteko denbora (ms)	Bariantza (ms)	Desbideratze estandarra (ms)
CRL deskarga	20,417	3,542	5,598
Kontrolatzailearen prozesatze denbora	143,098	0,648	25,462
Handshake guztia	282,473	1,479	38,456

10. Taula. Laugarren egoeraren emaitzak

9.2.5. Fragmentazioa, CRL tamaina txikia eta deskargatu beharra – Bezeroan

Azkeneko proba, lehen azaldu den bigarren egoeran egingo da, hau da, CRL deskarga eta errebokazio egoeraren konprobazioa switcharen kontrolatzailean egin beharrean bezeroan egingo da, kasu honetan, IoT gailuan. Itxaroten diren emaitzak baxuak izango dira kontrolatzailea parte hartzen ez duelako, hala ere, proiektu honen helburuetako bat IoT

gailuaren baliabide gutxien erabiltzea izan da. Beraz, denbora baxuagoak lortu arren egokiagoa da kontrolatzailea erabiltzea prozedimendua egiteko. CRL deskarga denbora aurreko ataleko berdina izango da jausi kopurua berdina delako.

	Batez besteko denbora (ms)	Bariantza (ms)	Desbideratze estandarra (ms)
CRL deskarga	14,935	1,381	3,716
Prozesatze denbora	29,253	1,267	35,599
Handshake guztia	46,315	0,107	4,4348

11. Taula. Bostgarren egoeraren emaitzak

Esan denez, handshakea egiteko behar den denbora askoz baxuagoa da maketaren gailu bat deuseztatzen delako.

Hala ere, gehitu behar da, proiektu hoenetan ez dela benetako IoT gailu bat erabiltzen. Hau da, Mininet sarean dagoen IoT gailua ordenagailu arrunt bat da baliabideak dituena, beraz, ezin da aurreko taulako emaitzetan oinarritu. Benetako IoT gailua izango balitz, baliabide murriztuekin, denboren balioak handituko lirateke eta asko nabarituko litzateke kontrolatzailearen erabilera.

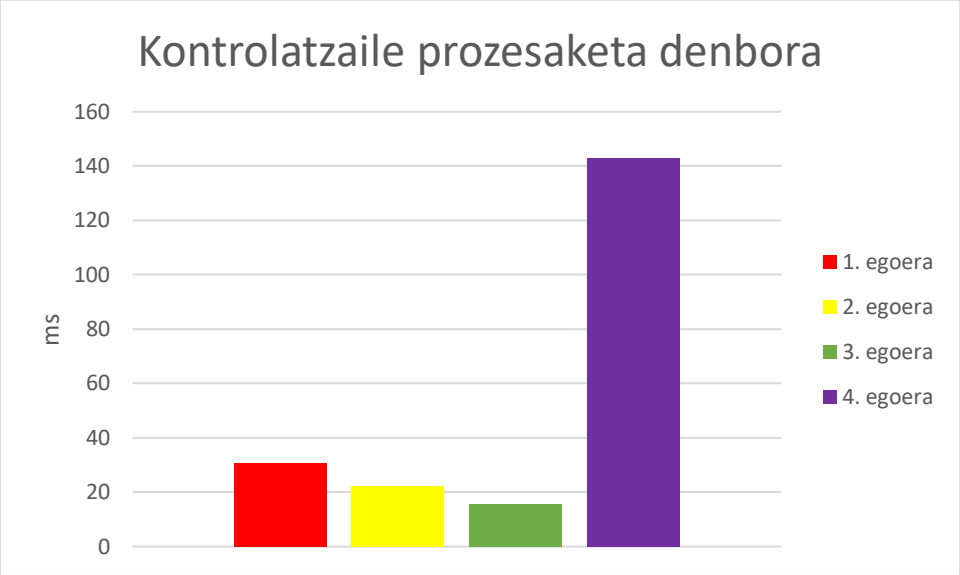
9.3. Emaitzen konparaketa

Atal honetan, lortutako emaitzen konparaketa egingo da. Lehenik, kontrolatzailea erabili den egoera bakoitzean behar izan den CRL deskargatzeko eta errebokazio egoera konprobatzeko prozesatze denboren konparazioa egingo da. Ondoren, kasu honetan ere, kontrolatzailea erabili diren DTLS handshake osoaren denboraren konparaketa egingo da. Azkenik, bi egoera ezberdinak konparatuko dira, hau da, inplementazioan kontrolatzaileak parte hartzen duenean ela ez.

9.3.1. Kontrolatzailearen prozesatze denboraren konparazioa

Hurrengo grafikoan ikusi ahal da, lau kasuen kontrolatzailearen bataz besteko prozesatze denbora. Egoerei dagokionez, lehen azaldu den ordenean ikusi daiteke:

1. Egoera: Datagrama fragmentatuta, CRL deskargatu beharra eta CRLan bakarrik bi ziurtagiri.
2. Egoera: Datagrama bakarra, CRL deskargatu beharra eta CRLan bakarrik bi ziurtagiri.
3. Egoera: Datagrama fragmentatuta, CRLa cachean eta CRLan bakarrik bi ziurtagiri.
4. Egoera: Datagrama fragmentatuta, CRLa cachean eta CRLan berrogeita bederatzi ziurtagiri.



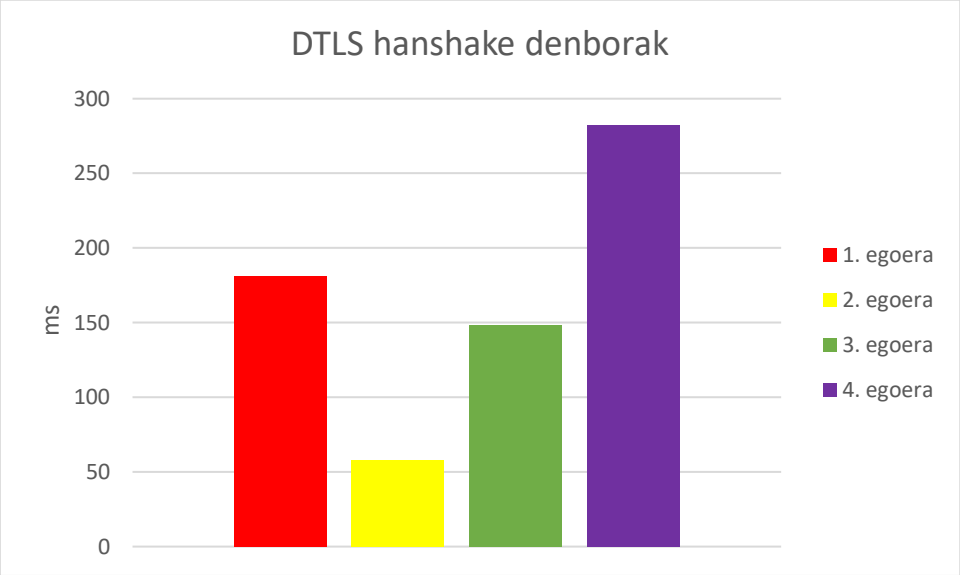
19. Irudia. Kontrolatzaile prozesaketa denboraren grafikoa

Grafikoan ikusi daitekeenez, kontrolatzailearen prozesatze denbora ez da asko aldatzen lehenengo hiru egoeren artean, hau da, ziurtagiria fragmentatuta badago ala ez, ez da nabaritzen. Eta berdina gertatzen da CRLa deskargatzen bada ala ez. Grafikoan ikusten den kasurik nabariena laugarren egoera da non CRLan berrogeita bederatzi ziurtagiri errebokatuta dauden. Kasu honetan, kontrolatzaileak CRL zerrenda guztia konprobatu behar du, beraz, grafikoan adierazten den bezala prozesatze denbora askoz handiago izango da.

9.3.2. DTLS handshake osoaren denboraren konparazioa

Hurrengo kasuko grafikoan egoera bakoitzeko DTLS handshake denbora totalak adierazten dira. Egoerei begira, aurreko atalekoak bezala dira:

1. Egoera: Datagrama fragmentatuta, CRL deskargatu beharra eta CRLan bakarrik bi ziurtagiri.
2. Egoera: Datagrama bakarra, CRL deskargatu beharra eta CRLan bakarrik bi ziurtagiri.
3. Egoera: Datagrama fragmentatuta, CRLa cachean eta CRLan bakarrik bi ziurtagiri.
4. Egoera: Datagrama fragmentatuta, CRLa cachean eta CRLan berrogeita bederatzi ziurtagiri.



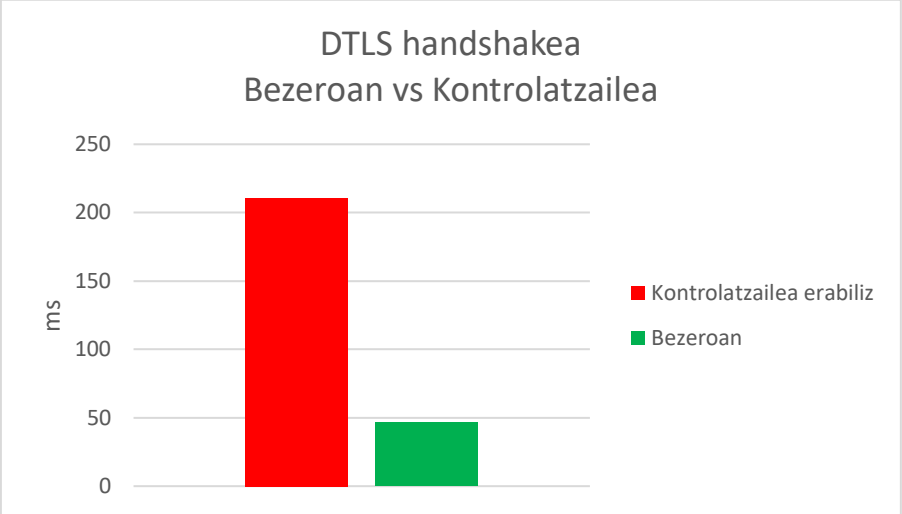
20. Irudia. DTLS handshake denborak grafikoa

Grafikoan irudikatzen denez, lehenengo grafikoaren ondorio berdina atera daiteke. Kontrolatzailearen prozesatze denboraren ondorioz, DTLS handshakearen denbora igotzen da. Orduan, laugarren egoera denbora gehien behar duena da kontrolatzaileak berrogeita bederatzi ziurtagirien artean bilaketa egin behar duelako.

9.3.3. Inplementazioan kontrolatzailea erabilita ala ez erabilita

Azkenik, hurrengo grafikoan DTLS handshake guztiaren denbora ikusten da baina CRL deskarga eta errabokazio egoera bi gailu ezberdinetan eginda. Kasu honetan, bi egoeratan DTLS mezuko informazioa fragmentatuta dago, CRL deskarga egin behar da eta CRLa bi ziurtagiri ditu.

1. Egoera: Kontrolatzailea erabilita. Switchak DTLS mezua hartu eta bere kontrolatzaileak CRL zerrenda deskargatzen du zerbitzariren ziurtagiriaren errebokazio egoera konprobatzeko.
2. Egoera: Prozesu berdina baina kontrolatzailean egin beharrean IoT gailuan egiten da, hau da, switch programagarria portaera arrunta du.



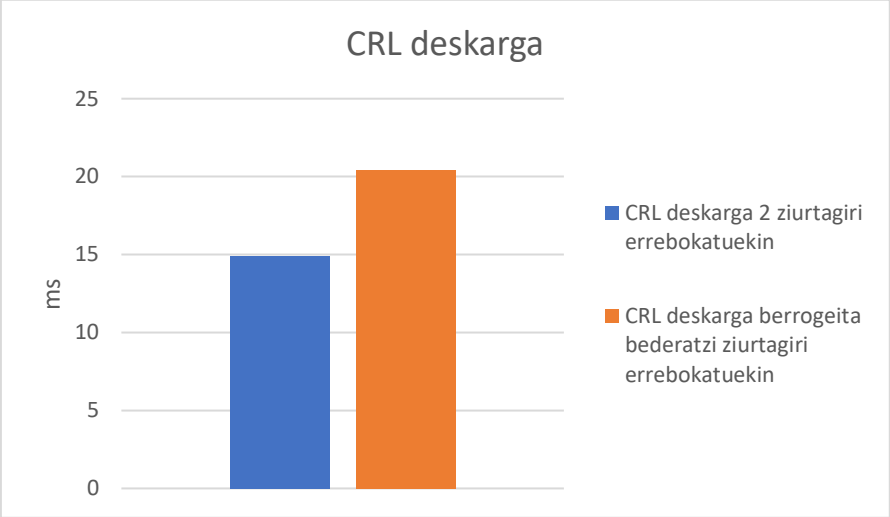
21. Irudia. DTLS handshake denbora (bezeroan vs kontrolatzailean) grafikoa

Ikusten denez, kontrolatzailea erabiliz askoz denbora gehiago behar da DTLS handshakea egiteko, zeren eta, salto gehiago egin behar dira prozesuan zehar eta paketeen manipulazio gehiago egin behar dira. Hala ere, lehen esan bezala, proiektu hau egin da IoT gailuaren baliabideak aurrezteko, beraz, denbora ez da oso garrantzitsua kasu honetan. Horrela, IoT gailua bakarrik datuak bidaltzen arduratzen da eta ez errebokazio egoerak konprobatzeaz.

Lehen esan bezala, emaitza hauek ez dira egiazkoak bezeroa IoT gailua ez delako. Benetako IoT gailua balitz DTLS handshake denbora handiagoa izango litzateke eta egokiagoa izango litzateke ziurtagiri errebokazio egoera kontrolatzailean egitea.

9.3.4. CRLaren deskarga

Esan denez, kontrolatzaileak CRLaren deskarga egin behar du 10.0.4.4. IP helbidea duen gailuaren Apache Web zerbitzaritik. Hala ere, bi egoera egon dira CRLaren deskargari dagokionez. Alde batetik, CRLak bi ziurtagiri bakarrik errebokatuta dituenean eta beste aldetik, berrogeita bederatzi ziurtagiri errobokatuta dituenean. Logikoa denez, CRL batek pisu handiagoa du bestea baino eta horregatik, CRL deskarga denbora aldatzen da. Hurrengo grafikoan bi denbora horien arteko konparaketa ikusi ahal da:



22. Irudia. CRL deskarga denbora

22. irudian ikusi ahal denez, CRLan berrogeita bederatzi ziurtagiri errebokatuta badaude kontrolatzaileak denbora gehiago itxaron behar du Apache zerbitzaritik CRLa jasotzeko. Hala ere, bi denboren ezberdintasuna ez da oso altua.

10. LAN PLANA

Atal honetan proiektua arrakastaz aurrera eramateko jarraitu beharreko plangintza azalduko da.

10.1. Lan taldea

Proiektuaren beharrak kontuan hartuz, lan-taldea bi zuzendari eta proiektua osatu duen ikaslea dira.

Honela identifikatuko dira taldekideak atal honetan:

Rola	Identifikatzailea	Izena
Zuzendaria I	H1	Jasone Astorga
Zuzendaria II	H2	Maider Huarte
Ikaslea	H3	Iker Alkorta

12. Taula. Proiektuaren lan taldea

10.2. Faseak eta zereginak

Atal honetan, proiektua aurrera eramateko egin den planifikazioa azalduko da, hau da, lana paketeak eta pakete bakoitzaren zereginak adieraziko dira. Gehitu behar da, proiektua 2021ko azaroaren 1an hasten dela eta 2022ko uztailaren 22an amaitzen dela.

LP1. Sistemaren beharren analisia

Proiektua sistemaren beharren analisiarekin hasiko da. Honetan, zuzendarien eta ikaslearen arteko lehenengo hartu-emanak egingo dira, haien arteko bilerak eginez, proiektuaren helburuak, betekizunak eta nondik norakoak zehaztu eta adosteko.

- Iraupena: 4 aste

A1.1. Espizifikazioen definizioa

Ataza honetan proiektuaren eskakizunak definituko dira. Hau da, zuzendarien eta ikaslearen artean proiektuak izango dituen helburua eta ezaugarriak komentatuko dira. Ondoren, adostutakoarekin, softwarearen espezifikazioak definituko dira, horiek zehatzak eta egonkorrak izan beharko direlarik.

- Iraupena: 2 aste

A1.2. Alternatiben analisia

Behin espezifikazioak izanda, ikasleak alternatiben analisiak jorratuko ditu. Espezifikazioetan komentatu diren teknologiak inplementatzeko dauden alternatibak aztertuko dira eta aukera horien artean egokiena aukeratuko da, hala nola, segurtasun teknologiak, DTLS handshakea inplementatzeko softwarea edo SDN sarea inplementatzeko aukerak.

- Iraupena: 2 aste

LP2. Arkitekturaren diseinua eta ingurunearen instalazioa

Espezifikazioak idatzita eta erabiliko diren teknologiak aukeratuta, proiektuaren arkitektura-diseinua egingo da. Ataza honetan, sarearen arkitektura-diseinua sortzeaz gain, hau inplementatzeko behar den beharrezko ingurunea sortuko da.

- Iraupena: 8 aste

A2.1. Arkitekturaren diseinua

Ataza honetan, proiektu osoaren oinarrizko arkitektura eraikiko da erabiliko diren gailu eta teknologia guztiak kontuan hartuz. Hau da, switch programagarria, IoT gailua, zerbitzaria CA eta CRL hornitzailea nola konektatuko diren diseinatuko da.

- Iraupena: 3 aste

A2.2. Ingurunearen instalazioa

Arkitekturaren diseinua egin ostean, hau inplementatzeko ingurunea instalatuko da, hala nola, SDN sarea eraikiko den ingurunea (Mininet) eta makina birtuala ere konfiguratuko da proiektua arrera eramateko.

- Iraupena: 3 aste

LP3. Sarearen inplementazioa eta integrazioa

Lan pakete honetan SDN sarearen inplementazioa instalatu den ingurunean sortuko da eta makina birtualarekin konektatuko da. Ondoren, inplementazioan parte hartuko duten gailuen konfigurazioa eta programak egingo dira DTLS handshake segurua lortzeko.

- Iraupena: 18 aste

A3.1. Mininet sarearen inplementazioa

Ataza honetan Mininet sarearen inplementazioa egingo da, hau da, SDN sarea sortuko da non hiru gailu izango dituen. Alde batetik, IoT gailua eta konektatu nahi den zerbitzaria egongo dira eta beste aldetik, hautetara konektatuta egongo den switch programarria egongo da kontrolatzaile batek kontrolatuko duena.

- Iraupena: 3 aste

A3.2. Gailu bakoitzaren konfigurazioa

Behin gailu guztiak sarean sartuta daudela eta ingurunean ondo egokituta daudela DTLS handshakea egitea lortzeko gailu bakoitzaren konfigurazioak eta programak egin behar dira. Lehenik, CA sortu behar da eta honek ziurtagiriak hornituko ditu IoT gailuari eta zerbitzariari. Honez gain, CRL zerrenda sortuko du eta Apache Web zerbitzarian argitartuko du. Ondoren, Mininet sarean dauden IoT gailua eta zerbitzarietan DTLS hanshakea egitea ahalbidetzen duten programak instalatu behar dira. Azkenik, switcharen datu planoa programatu behar da honek

zerbitzariaren ziurtagiria bere kontrolatzailera bidaltzeko. Kontrolatzailearen programa egingo da zerbitzariaren errebokazio egoera konprobatzeko CRL bidez.

- Iraupena: 13 aste

A3.3. Aplikazioaren exekuzioa

Sare guztia konektatuta dagoenean eta gailuak konfiguratuta daudenean aplikazioa exekutatuko da ea DTLS handshakea ondo funtzionatzen duen.

- Iraupena: 2 aste

LP4. Balioztatzea eta probak

Azkenik, programa osoa eta modulu guztiak era sinkronizatu batean lan egiten dutela konprobatzeko balioztatze fasea egingo da. Balioztatze fasea bi azpi ataza izango ditu:

- Iraupena: 4 aste

A4.1. Sistemaren balioztatze eta egonkortasun-probak

Ataza honetan sistema osoaren balioztatze eta egonkortasun-probak egingo dira. Hau da, programa osoa era egokian funtzionatzen duela konprobatuko da, bere funtzionalitate guztiak probatzearen bitartez, hala nola, DTLS handshakea era egokian egiten den eta kontrolatzaileak CRL bitartez ziurtagirien errebokazio egoera ondo konprobatzen den.

- Iraupena: 2 aste

A4.2. Errendimendu-probak

Balioztatzea egin eta gero, errendimendu probak egingo dira. Sistemaren egoera ezberdinak hartuko dira eta aplikazioak nolako erantzunak izango dituen analizatuko dira. Adibidez, DTLS handshake denbora hartuko da, CRLa deskargatzeko denbora eta kontrolatzailearen prozesatze denbora (ziurtagiria errebokatuta dagoen konprobatzeko).

- Iraupena: 2 aste

LP5. Proiektuaren kudeaketa

Azkenik, proiektuaren kudeaketa proiektu osoaren zehar egingo da. Epeak kontuan hartuz, fase eta ataza bakoitzaren iraupenak zehaztu egingo dira proiektuaren arrakasta lortzeko.

- Iraupena: 38 aste

A5.1. Dokumentazioa

Aipatu beharra dago dokumentazioa proiektu osoan zehar egin beharko dela, epeak betez, zuzendariek eskatzen duten dokumentuak entregatuz.

- Iraupena: 38 aste

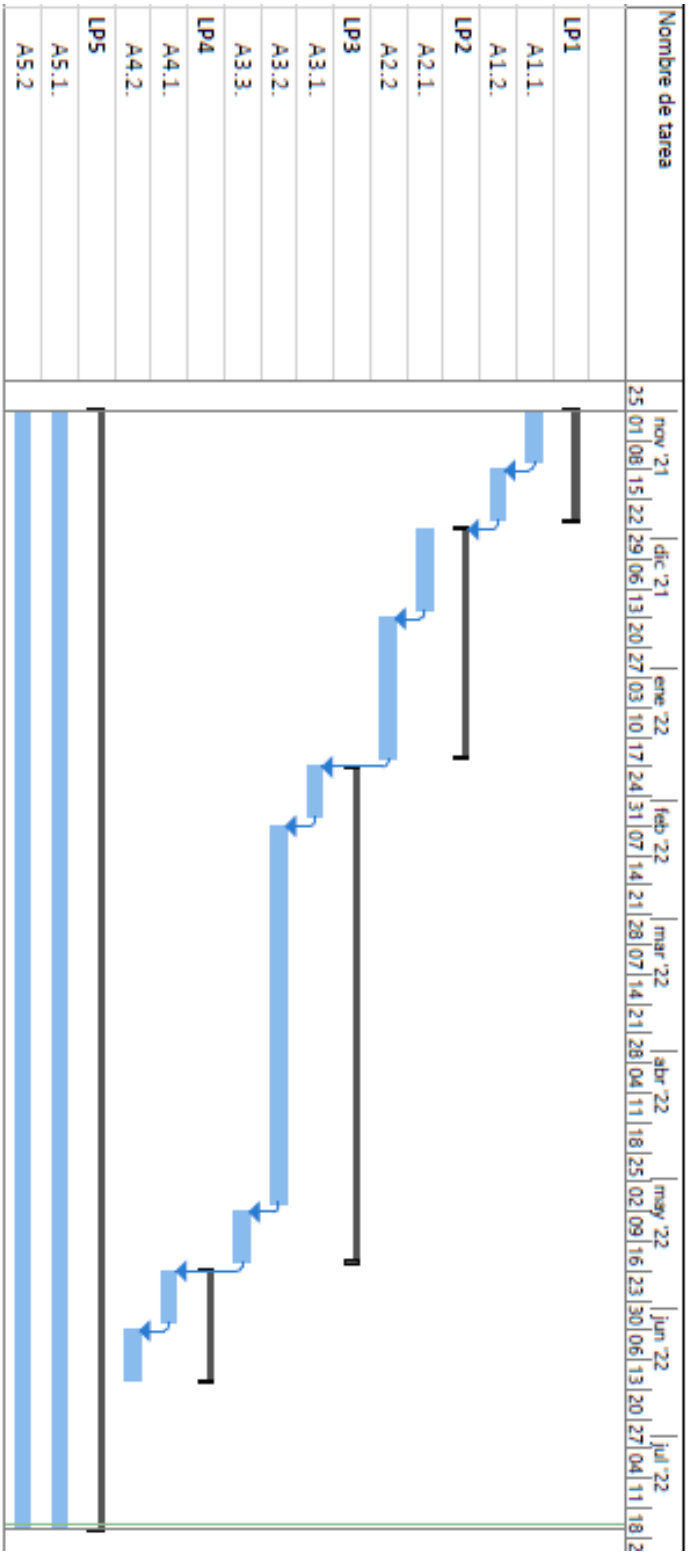
A5.2. Proiektuaren kontrola

Azpi ataza hau ere proiektu osoan zehar egingo da eta zuzendariek eta ikaslea bilerak izango dituzte proiektuaren egoera komentatzeko. Hau da, proiektuan aurrera egiteko arazoei irtenbideak aurkitzeko edo zenbait aldaketa egiteko.

- Iraupena: 38 aste

10.3. Gantt diagrama

Hurrengo irudian planifikazioaren Gantt diagrama ikusi daiteke:



23. Irudia. Gantt diagrama

11. ALDERDI EKONOMIKOAK

Txostenaren atal honetan proiektuari lotutako alderdi ekonomikoa azalduko da. Proiektuan zehar behar izan diren baliabideen kostuen jorraketa egingo da.

11.2. Giza baliabideak

Puntu honetan proiektuan zehar parte hartu duten pertsonen kostua ikusiko da. Esan denez, hiru pertsona parte hartu dute, bi zuzendari eta ikasle bat.

BARNE ORDUAK				
Langileak	Orduko kostua (€/h)	Orduak (h)	Pertsona kopurua	Guztira (€)
Zuzendariak	35	60	2	4.200
Ikaslea	15	300	1	4.500
			TOTALA	8.700

13. Taula. Barne orduak

11.2. Amortizazioak

Atal honetan proiektuan egon diren amortizazioak deskribatuko dira, hau da, proiektua aurrera eramateko erabili diren baliabideak baina proiektua amaitu ondoren, erabiltzen jarraitu daitekeenak.

AMORTIZAZIOAK				
Baliabideak	Kostua (€)	Orduak (h)	Bizitza erabilgarria (h)	Guztira (€)
Ordenagailua	400	300	10.000	12
Microsoft Office Lizentzia	20	150	356	8,427
			TOTALA	20,427

14. Taula. Amortizazioak

11.3. Gastuak

Aurrekontuarekin jarraitzeko proiektuak izan dituen gastuak kalkulatuko dira.

Kontzeptua	Kostua (€)
Internet konexioa	200
Elektrizitatea	50
TOTALA	250

15. Taula. Gastuak

11.4. TOTALA

Amaitzeko, aurrekontuaren kostu totala kalkulatuko da, hau da, giza baliabideak, amortizazioak eta gastuak gehituz geratuko den balioa. Gainera, kostu ez-zuzenak eta ezusteak %10koak izango dira. 16. Taulan ikusten denez, proiektua gauzatzeko behar izan den diru totala **10.764,513 €** da.

Laburpena	Guztira (€)
Barne orduak	8.700
Amortizazioak	20,427
Gastuak	250
Subtotal 1	8.970,427
Kostu ez-zuzenak (%10)	897,043
Ezusteak (%10)	897,043
TOTALA	10.764,513

16. Taula. Aurrekontu totala

12. ONDORIOAK

Txostena amaitzeko proiektuan zer ondorio atera diren azalduko dira. Esan daiteke proiektu honen helburuak era zuzenean lortu egin direla. IoT gailu baten eta zerbitzari baten DTLS handshakea egitea lortu da switch programagarri bat erabiliz ziurtagirien errebokazio egoerak konprobatzeko.

Lehenik, IoT gailu baten eta zerbitzari baten DTLS handshakea era eraginkorrean inplementatu egin da. Horretarako, CA bat eratu da eta honek ziurtagiri digitalak hornitu dizkie bezero eta zerbitzariei. Gainera, CAk berrogeita hamar ziurtagiri gehiago eratu ditu errendimendu analisia errealagoa izateko. Beraz, esan daiteke proiektua gauzatzeko PKI bat eratu dela. Horrez gain, CAk berrogeita bederatzi ziurtagiri errebokatu ditu eta CRL zerrenda bat eratu du ziurtagirien errebokazio egoera konprobatu ahal izateko.

Bigarrenik, SDN sare bat inplementatu da non switch programagarri bat izan den honen oinarria. Switch programagarria era egokian programatu izan da DTLS handshake mezu zehatzak ikusteko eta ondoren, kontrolatzaileak mezu hauek manipulatzeko. SDN kontrolatzaileak CRL zerrenda deskargatzea lortu du eta zerbitzariaren errebokazio egoera konprobatzea lortu du. Horrela, proiektuaren helburura iritsi da:

IoT gailu baten eta zerbitzari baten komunikazio segurua lortzea IoT gailuaren baliabideak agortu gabe.

Gainera, proiektu honek SDN sareak IoT ingurune batean erabiltzeak aukera oso ona dela frogatzen du eta sare mota hauek gero eta gehiago erabili behar direla iradokitzen du.

13. ERREFERENTZIAK

[1] User Datagram Protocol: What role does UDP play in Cellular IoT?

https://www.hologram.io/blog/udp

[2] What is a public-key infrastructure (PKI)?

https://www.entrust.com/es/resources/certificate-solutions/learn/what-is-pki#:~:text=La%20PKI%20o%20Public%20Key,propietario%20cifre%2C%20firme%20y%20autentique.

[3] Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) (RFC 5280). D. Cooper, S Santesson, S.Farrell, S.Boeyen, R. ousley, T.Polk.

https://datatracker.ietf.org/doc/html/rfc5280

[4] EHUko Telekomunikazioaren Ingeniaritza Teknikoko Graduko Informazio Sistemen Arkitektura Liburua.

[5] Beginners guide on PKI, Certificates, Extensions, CA, CRL and OCSP

https://www.golinuxcloud.com/tutorial-pki-certificates-authority-ocsp/

[6] certificate revocation list (CRL)

https://www.techtarget.com/searchsecurity/definition/Certificate-Revocation-List

[7] Datagram Transport Layer Security Version 1.2 (RFC 5246). E. Rescola, N. Modadugu.

https://www.rfc-editor.org/rfc/rfc6347

[8] The Transport Layer Security (TLS) Protocol Version 1.2 (RFC 5249)

https://www.rfc-editor.org/rfc/rfc5246

[9] SDN architecture by Open Networking fundation

https://opennetworking.org/wp-content/uploads/2013/02/TR_SDN_ARCH_1.0_06062014.pdf

[10] Redes definidas por Software (Universidad Complutense de Madrid)

https://informatica.ucm.es/data/cont/media/www/pag-103596/transparencias/redes-por-software-SDN.pdf

[11] OpenFlow: Enabling Innovation in Campus Networks

http://ccr.sigcomm.org/online/files/p69-v38n2n-mckeown.pdf

[12] P4 Language Specification (The P4 Language Consortium)

https://p4.org/p4-spec/docs/P4-16-v1.0.0-spec.pdf

[13] *DTLShps: SDN-Based DTLS Handshake Protocol Simplification for IoT*

https://ieeexplore.ieee.org/document/8962334

[14] Clarifying the differences between P4 and OpenFlow

https://opennetworking.org/news-and-events/blog/clarifying-the-differences-between-p4-and-openflow/

[15] Californium (Cf) - CoAP for Java

https://github.com/eclipse/californium

[16] Datagram Transport Layer Security for Python

https://github.com/mobius-software-ltd/pyton3-dtls

[17] EJBCA product documentation

https://www.ejbca.org/documentation/

[18] OpenSSL cryptography and SSL/TLS toolkit

https://github.com/openssl/openssl

[19] Create Certificate Authority

https://jamielinux.com/docs/openssl-certificate-authority/create-the-root-pair.html

[20] OpenSSL create client certificate & server certificate with example

https://www.golinuxcloud.com/openssl-create-client-server-certificate/

[21] Revoke certificate and generate CRL OpenSSL

https://www.golinuxcloud.com/revoke-certificate-generate-crl-openssl/

[22] Software-Defined Networking

https://www.cisco.com/c/en/us/solutions/software-defined-networking/overview.html

[23] ¿Qué ocurre durante un protocolo de enlace TLS? | Protocolo de enlace SSL

https://www.cloudflare.com/es-es/learning/ssl/what-happens-in-a-tls-handshake/

[24] Karn, P. and W. Simpson, "Photuris: Session-Key Management Protocol", RFC 2522, March 1999.

https://datatracker.ietf.org/doc/rfc2522/

[25] Managing AAA in NFV/SDN-enabled IoT scenarios

https://ieeexplore.ieee.org/document/8534551

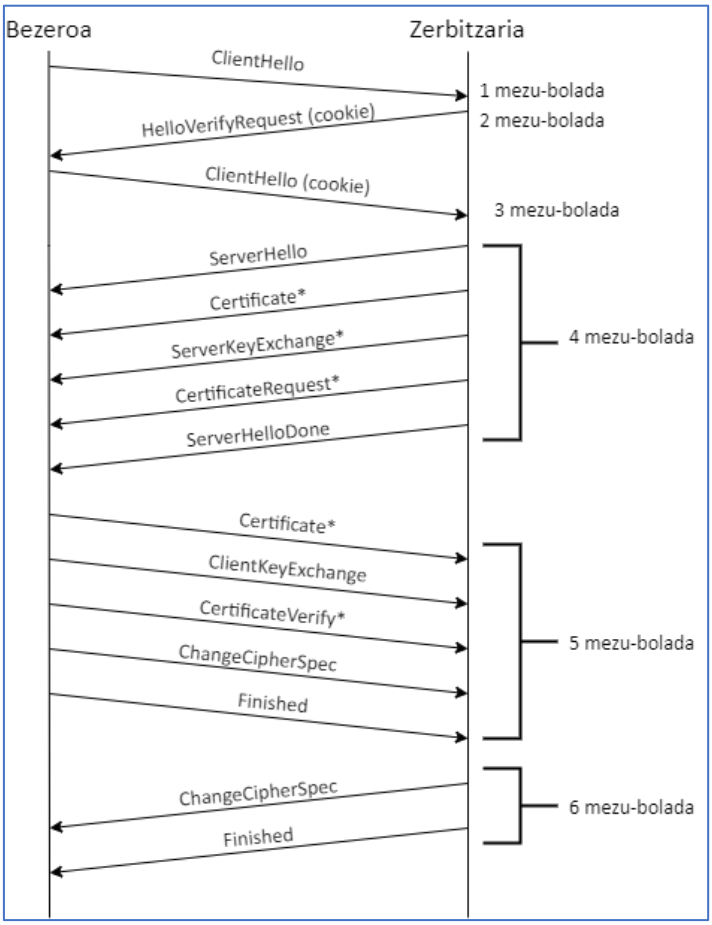
14. ERANSKINAK

14.1. DTLS HANDSHAKE

DTLS TLS-k erabiltzen dituen handshake mezu eta fluxu berdinak ditu, baina hiru aldaketa nagusiekin [8]:

1. *Denial-of-service* erasoak prebenitzeko egoera gabeko cookiea gehitu da.
2. Handshake goiburuan aldaketak mezuen galera, berrantolaketa eta DTLS mezuen zatiketa kudeatzeko (IP zatiketa saihesteko).
3. Mezu galerak prebenitzeko birtransmisio timeraren gehikuntza.

Handshakea bezeroaren eta zerbitzariaren arteko sesioaren segurtasun parametroak negoziatzeko diseinatu da, hau da, bezeroa eta zerbitzaria komunikatzen hasten direnean protokoloaren bertsioa adosten dute, algoritmo kriptografikoa aukeratzen dute, autentifikatzen dira eta sekretuak elkarbanatzeko gako publiko zifratze teknikak erabiltzen dituzte. Hurrengo irudian ikusiko dira DTLS handshake pausu guztiak [23]:



24. Irudia. DTLS handshakea

* ikurra mezua aukerazkoa dela adierazten du. Aldiz, ikurra ez badu mezua derrigorrezkoa da.

Orain, handshake mezuak deskribatuko dira:

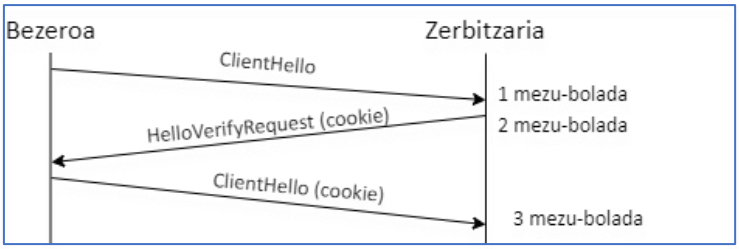
14.1.1. Client Hello fasea (1., 2., 3. mezu boladak)

Fase honetan, DTLSn gertatu ahal den DoS erasoari aurre egiteko mekanismoa inplementatzen da. DoS erasoa ekiditzeko, lehen aipatu den egoera gabeko cookiearen erabilera gehitzen da. Lehenik, bezeroak *ClientHello* mezua bidaltzen dio zerbitzariari eta honek *HelloVerifyRequest* mezuarekin erantzun behar dio. *HelloVerifyRequest* mezuak egoera gabeko cookiea garraiatzen du Photuris [24] teknika erabiliz. Client hello fasea amaitzeko, bezeroak *ClientHello* mezua birtransmititu behar dio zerbitzariari baina kasu honetan, zerbitzariak bidali dion cookiea gehituta. Azkenik, zerbitzariak cookie hori egiaztatuko du eta baliozkoa bada handshakearekin jarraituko du, ez bada handshakea amaituko da.

Beraz, mekanismo honek bezeroa (edo erasotzailea) cookiea jasotzera behartzen du, eta horrek zaildu egiten ditu IP helbide faltsuak dituztezten DoS erasoak. Hala ere, prozesu hau ez du inolako defentsarik eskaintzen baliozko IP helbideetatik muntatutako DoS erasoen aurka.

Cookiearen egitura: Cookie = HMAC(Secret, Client-IP, Client-Parameters)

Gainera, *ClientHello* mezuak cookieaz gain ausazko balio bat (gero erabiliko dena) eta jasaten dituen aukera kriptografikoak, konpresio metodoak eta luzapen informazioa garraiatzen ditu. Horrela, zerbitzariak bezeroaren ezaugarriak ezagutuko ditu.



25. Irudia. Client Hello fasea

14.1.2. Server Hello fasea (4. mezu bolada)

Bigarren fase honetan, laugarren mezu boladan, zerbitzariak bost mezu bidaliko dizkio bezeroari, batzuetan mezuak bananduta baina gehienetan bost mezuak pakete bakar batean garraiatzen dira.

Lehenik, zerbitzariak *ServerHello* mezua bidaliko du *ClientHello* mezuari erantzuteko eta honen barnean bezeroak esan dion enkriptazio algoritmo-multzoetatik bat aukeratuko du eta zein aukeratu duen adieraziko dio bezeroari. Horrez gain, *ServerHello* mezuan ausazko balio bat garraiatuko da, bezeroan gertatu den bezala. Zerbitzariak algoritmo multzo horretatik ezin badu inolako algoritmo bat aurkitu *handshake failure alert* mezuarekin erantzungo du.

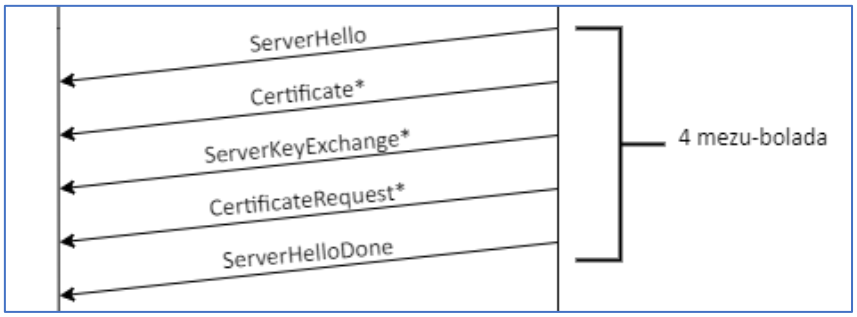
Esan behar da, TLS/DTLSek hiru autentikazio mekanismo onartzen dituela: Pre Shared Key (PSK), Raw Shared Key(RSA) eta ziurtagiriak. Lan honetan autentikazio mekanismo bezala ziurtagiri digitalak erabiliko direnez, honetan arreta jarriko da. Beraz, *ServerHello* mezua bidali ondoren, zerbitzariak *Certificate* mezua bidaliko du. Mezu hau derrigorrez bidali behar da autentikaziorako adostutako autentikazio metodoa ziurtagiri digitalak direnean. Beraz, mezu honek zerbitzariaren ziurtagiri-katea helarazten dio bezeroari, horrela bezeroak jakingo du zerbitzaria autentikoa dela. Ziurtagiri katea zerrenda bat bezala ulertu daiteke eta zerrendaren

lehenengo ziurtagiria beti zerbitzariaren ziurtagiria izango da. Zerrendaren hurrengo ziurtagiriak zerbitzariarena ziurtatzeko balio dute. Zerbitzariek bidaltzen dituzten ziurtagiriek bi arau bete behar dituzte; Alde batetik, ziurtagiria beti x.509v3 motakoa izan behar da (beste mota bateko izan ahal da baina aurretik adostu behar da) eta beste aldetik, amaierako entitatearen ziurtagiriaren gako publikoa bateragarria izan behar da aukeratutako gako-truke algoritmoarekin.

Certificate mezua eta gero, *ServerKeyExchange* bidaliko litzateke, baina zerbitzariak mezu hau bakarrik bidaliko du Certificate mezuak (bidaltzen bada) bezeroak *premaster* sekretu bat trukatzeko datu nahikorik ez duenean. *ServerKeyExchange* mezuek gako-truke metodoen menpekoa da, hau da, gako-truke metodo espezifiko batzuetan mezu hau bidaltzea ilegala da. Mezu honek *premaster* gakoaren trukea ahalbidetzeko bidaliko da, hau da, zerbitzariak lehen bezeroak *ClientHello* mezuan bidali dion ausazko balioa, zerbitzariak bidalitako ausazko balioa eta gehienetan Diffie-Hellman parametroak hartu eta bere gako pribatuarekin enkriptatuko ditu. Gero, bezeroak zerbitzariaren gako publikoarekin (ziurtagirian lortu duena) mezu honen edukia desenkriptatzean egiaztatuko du zerbitzaria autentikoa dela.

CertificateRequest mezuari dagokionez, zerbitzariak bezeroaren ziurtagiria nahi duenean bidaltzen den aukerazko mezua da. Mezu hau bidaltzen bada bezeroa derrigorrez *Certificate* mezua bidali behar dio zerbitzariari, lehen azaldutako egitura berdinarekin.

Fase hau amaitzeko, zerbitzariak *ServerHelloDone* mezua bidaltzen dio bezeroari ServerHello mezua amaitu dela zehazteko, hau da, zerbitzariak bukatu du bere gako-truke fasea. Orduan, zerbitzariak bezeroaren erantzunari itxarongo dio.



26. Irudia. Server Hello fasea

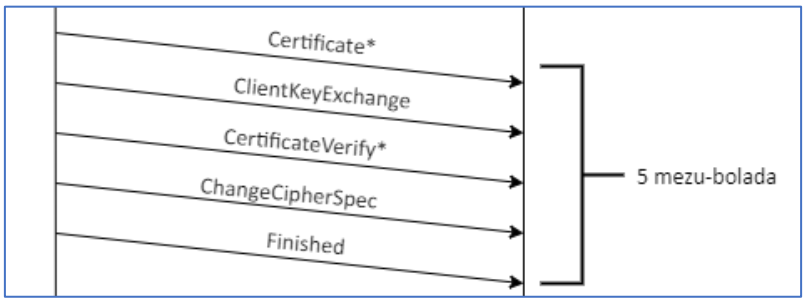
14.1.3. Bezeroaren erantzun fasea (5. mezu-bolada)

Bezeroaren erantzun fasea bezeroak *ServeHelloDone* bidalitako mezua jasotzean hasten da. Fase honen lehen mezua *Certificate* mezua izango da baina lehen esan denez, bakarrik bidaliko da zerbitzariak *CertifateRequest* eskatzen badio. Bezeroaren *Certificate* mezu honek zerbitzariaren *Certificate* mezuaren formatu berdina izango du eta zerbitzariak handshakearen azkeneko fasean egiaztatuko du.

Ondoren, bezeroak eta zerbitzariak Hello fasetan adostu zuten gako-truke metodoaren arabera bezeroak *premaster* sekretua, ausazko byte balioa dena, zerbitzariaren gako publikoarekin

enkriptatu egingo du (ziurtagirian lortu duena). Sekretu enkriptatua *ClientKeyExchange* mezuaz baliatuz bidaltzen da zerbitzarira. Gero, zerbitzariak mezu hau bere gako pribatua erabiliz desenkriptatuko du.

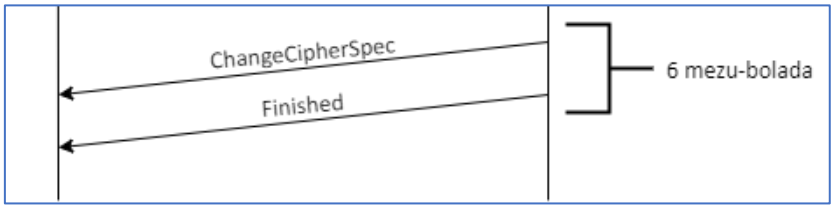
Azkenik, *ChangeCipherSpec* eta *Finished* mezuak bidaliko dira. Alde batetik, *ChangeCipherSpec* mezuak adierazten du hurrengo datuak enkriptatuta egongo direla negoziatu berri diren algoritmoak, sekretuak eta gakoak erabiliz. Beraz, *Finished* mezua gakoak emandako segurtasunarekin eta algoritmoekin bidaliko da bezeroa bere handshake mezu-txanda amaitu dela jakinaraziz.



27. Irudia. Bezeroaren erantzun fasea

14.1.4. Zerbitzariaren agur fasea (6. mezu bolada)

DTLS handshakearekin amaitzeko, zerbitzariak bezeroari informatzen dio handshake prosezua bukatu duela. Horretarako, bezeroak egin duen bezala, *ChangeCipherSpec* mezua bidaltzen du eta negoziatu diren gakoak erabiliz Finished mezua enkriptatzen du.



28. Irudia. Zerbitzari agur fasea

Azkeneko fasea amaitzean esan daiteke DTLS handshake prozesua guztiz bukatu dela. Laburbilduz, prozesu hau erabiliz bezeroa eta zerbitzaria elkar autentifikatu dira eta segurtasun tresnak sortu eta elkarbanatu dituzte sesio seguru bat izateko, hau da, sortu duten gakoarekin bi ekipoen arteko komunikazio segurua izatea lortu dute.