Maisterintutkielma

Tietojenkäsittelytieteen maisteriohjelma

DAGLAP: protokollasto pilvipalvelimen sijainnin todentamiseen

Sampo Hippeläinen

7.4.2022

Matemaattis-luonnontieteellinen tiedekunta

Helsingin yliopisto

Ohjaaja(t)

Prof. Valtteri Niemi

Tarkastaja(t)

Prof. Valtteri Niemi

Yhteystiedot

PL 68 (Pietari Kalmin katu 5)
00014 Helsingin yliopisto

Sähkopostiosoite: info@cs.helsinki.fi
URL: http://www.cs.helsinki.fi/

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Study programme
Matemaattis-luonnontieteellinen tiedekunta	Tietojenkäsittelytieteen maisteriohjelma

Tekijä — Författare — Author

Sampo Hippeläinen

Työn nimi — Arbetets titel — Title

DAGLAP: protokollasto pilvipalvelimen sijainnin todentamiseen

Ohjaajat — Handledare — Supervisors

Prof. Valtteri Niemi

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Maisterintutkielma	7.4.2022	56 sivua

Tiivistelmä — Referat — Abstract

Pilvipalveluiden yleistyminen on tuonut mukanaan uusia ongelmia, joista yksi liittyy maantieteelliseen sijaintiin. Nykyaikaisissa palveluissa on usein sijaintikohtaista sisältöä ja mahdollisesti jopa tietoa, jonka ei pidä päätyä tietyn alueen ulkopuolelle. Palveluntarjoajalla voi kuitenkin olla syynsä siirtää sovelluspalvelin muualle. Pilvipalvelussa ajettavan sovelluksen tulisi saada tietää, missä sitä ajetaan ja minne sen tiedot on tallennettu.

Tämän työn aiheena on ratkaista tämä ongelma uudella tavalla käyttämällä hyväksi palvelinkeskukseen pysyvästi sijoitettua apulaitetta, joka tarjoaa sijaintitietoa samassa sisäverkossa oleville palvelimille. Työssä kehitetään suunnittelutieteen menetelmin uusi protokollasto, jolla sovellus voi tarkistaa sijaintinsa.

Tuloksena syntynyt protokollasto rakentuu toimivaksi todettujen kryptografisten protokollien varaan. Palvelinkeskuksen sijaintitietoja tarjoavaan laitteeseen muodostetaan turvallinen yhteys, jonka yhteydessä laitteen aitous varmistetaan. Tietojen sijainti varmistetaan tarkistamalla, että tietopalvelimella on tosiaan tiedot saatavilla. Läheisyys todetaan mittaamalla kierrosaikoja ja asettamalla niille enimmäisraja.

Työn aikana protokollaston ja sitä käyttävän ratkaisun todetaan ratkaisevan ongelman ja täyttävän sille asetetut tarkemmat vaatimukset aiemmin esitettyjä ratkaisuja paremmin. Protokollasto osoitetaan toteuttamiskelpoiseksi toteuttamalla siitä prototyyppi. Tarkempien yksityiskohtien suhteen on kuitenkin lisää tutkittavaa.

ACM Computing Classification System (CCS)
Security and privacy → Network security → Security protocols
Security and privacy → Database and storage security → Information accountability and usage control

Avainsanat — Nyckelord — Keywords

tietoturva, yksityisyys, sijainti, pilvipalvelut

Säilytyspaikka — Förvaringsställe — Where deposited

Helsingin yliopiston kirjasto

Muita tietoja — övriga uppgifter — Additional information

Ohjelmistojärjestelmien opintosuunta

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Study programme
Faculty of Science	Master's Programme in Computer Science

Tekijä — Författare — Author

Sampo Hippeläinen

Työn nimi — Arbetets titel — Title

DAGLAP: protokollasto pilvipalvelimen sijainnin todentamiseen

Ohjaajat — Handledare — Supervisors

Prof. Valtteri Niemi

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Master's thesis	April 7, 2022	56 pages

Tiivistelmä — Referat — Abstract

One of the problems with the modern widespread use of cloud services pertains to geographical location. Modern services often employ location-dependent content, in some cases even data that should not end up outside a certain geographical region. A cloud service provider may however have reasons to move services to other locations. An application running in a cloud environment should have a way to verify the location of both it and its data.

This thesis describes a new solution to this problem by employing a permanently deployed hardware device which provides geolocation data to other computers in the same local network. A protocol suite for applications to check their geolocation is developed using the methodology of design science research.

The protocol suite thus created uses many tried-and-true cryptographic protocols. A secure connection is established between an application server and the geolocation device, during which the authenticity of the device is verified. The location of data is ensured by checking that a storage server indeed has access to the data. Geographical proximity is checked by measuring round-trip times and setting limits for them.

The new solution, with the protocol suite and hardware, is shown to solve the problem and fulfill strict requirements. It improves on the results presented in earlier work. A prototype is implemented, showing that the protocol suite can be feasible both in theory and practice. Details will however require further research.

ACM Computing Classification System (CCS)
Security and privacy → Network security → Security protocols
Security and privacy → Database and storage security → Information accountability and usage control

Avainsanat — Nyckelord — Keywords

computer security, privacy, geolocation, cloud services

Säilytyspaikka — Förvaringsställe — Where deposited

Helsinki University Library

Muita tietoja — övriga uppgifter — Additional information

Software Systems study track

Sisällys

1 Johdanto

Tämän vuosisadan aikana pilvipalvelut ovat nousseet erittäin suureen ja merkittävään asemaan muun muassa erilaisten verkkopalveluiden toteuttamisessa. Lähes kaikki verkkosivuja ja -palveluita suoritetaan nykyään pilvipalvelimilla. Tämä murros on johtanut uusiin ja kiinnostaviin ongelmiin pilvipalvelualalla, etenkin jos kyseessä on hajautettu eli useammassa paikassa samaan aikaan ajettava palvelu. Hajautettujen pilvipalveluiden ongelmia ovat muun muassa palveluiden saaman kuorman tasaaminen sekä tietojen saattaminen ajan tasalle palvelinkeskusten välillä.

Yksi ongelma liittyy myös juuri sijainteihin. Pilvipalveluissa voi olla joskus hyvinkin tärkeää se, missä fyysisessä paikassa niitä ajetaan ja missä niiden tiedot sijaitsevat. Palvelimien käsittelemä tieto voi esimerkiksi olla sijaintikohtaista. Tätäkin tärkeämpää ovat tapaukset, joissa tieto ei saa päätyä tietyn alueen ulkopuolelle, ja sovelluspalvelimen siirtyminen muualle kuin käyttäjän haluamaan sijaintiin voi johtaa pahimmassa tapauksessa tietojen vuotamiseen. Jos kyseessä ovat yksityiset tai muuten arkaluontoiset tiedot, tämän seuraukset voivat olla tuhoisia.

Ongelman ydin on se, kuinka sovelluspalvelin voi merkittää sijaintinsa ja sen avulla estää palvelun toiminnan tietyn alueen ulkopuolella. Tähän tarkoitukseen on suunniteltu erilaisia ratkaisuja, joita tullaan myöhemmin tarkastelemaan tarkemmin. Tämän työn tarkoituksena on kehittää uutta ehdotusta, palvelinkeskukseen pysyvästi asennettavaa sijaintitietoja tarjoavaa laitetta, ja tarkalleen ottaen kehittää sille protokollasto, jonka avulla sovelluspalvelimet voivat varmistua itsensä ja tietojensa sijainnista.

Palvelinkeskuksia on ympäri maailmaa ja niissä ajetaan lukematon määrä erilaisia palveluita. Palvelinkeskuksen omistajalla voi olla useita eri syitä siirtää palvelu toiseen maantieteelliseen sijaintiin, kuten kapasiteetin ja kuorman tasaaminen, korkean saatavuuden takaaminen esimerkiksi huollon yhteydessä tai onnettomuuden (kuten palvelinkeskukseen tapahtuvan tulipalon) sattuessa, palvelun siirtäminen kustannustehokkaimpaan sijaintiin, tai kenties jopa jokin hämärämpi tarkoitus.

Sijaintitiedolla on väliä, koska data ei osaa kertoa sijainnistaan, mutta sillä voi silti olla seurauksia. Tiedot saattavat esimerkiksi olla laittomia tietyn valtion sisäpuolella tai ulkopuolella. Tiedot voivat olla yksityisiä tai arkaluonteisia, jolloin niiden päätyminen tietyn valtion ulkopuolelle voi olla merkittävä riski. Esimerkki tästä on Ruotsissa tapahtunut poliittinen skandaali, jossa valtion kuljetushallituksen yksityiset ja arkaluonteiset tiedot, jotka sisälsivät jopa valtionsalaisuuksia, vuotivat vuodesta 2015 valtion ulkopuolelle, koska hallinnoijat eivät olleet tarpeeksi huolellisia tiedon tallennussijainnista [7].

Aiheella on myös tieteellistä merkitystä. Jos fyysistä laitetta ja sille tässä työssä suunniteltua ohjelmistoratkaisua hyödyntävä ratkaisu voidaan osoittaa käytännölliseksi, se voi täysin tai osittain kelvata myös laajemminkin erilaisiin tarkoituksiin.

Tutkimus liittyy siis luotettavan sijaintitiedon tarjoamiseen palvelinkeskuksessa ajettavalle sovellukselle. Tutkimusasetelmana on siten tutkia, miten fyysisen laitteen ja sitä käyttävä sovelluspalvelimen ohjelmisto tulee toteuttaa ja samalla selvittää kyseisen yhdistelmän mahdollisuuksia luotettavan sijaintitiedon tarjoamiseen turvallisesti.

Kyseistä asetelmaa lähdetään tutkimaan suunnittelutieteen (*design science*) avulla. Työn tuloksena on sovelluspalvelimen ja fyysisen laitteen välinen ohjelmistoprotokollasto, jota arvioidaan tieteellisin menetelmin. Ennen suunnitteluvaihetta lopulliselle ratkaisulle määritellään tarkat vaatimukset, jotka ratkaisun on täytettävä.

Tutkimukseen kuuluu fyysisen laitteen olemus, sen sijaintia tarjoavan protokollaston suunnittelu ja se, kuinka sovelluspalvelimen tulee toimia sijaintinsa ja tietojensa sijainnin tarkistamiseksi. Tässä työssä ei paneuduta tarkemmin kyseisen fyysisen laitteen laitteistoon tai siihen, kuinka laitteen tai palvelimen ohjelmisto tullaan tarkemmin toteuttamaan, vaan ainoastaan sen teoreettiseen puoleen eli protokolliin. Työn aikana suunnitelluista protokollista toteutetaan alkeellinen prototyyppi, jota käytetään arvioinnin apuvälineenä. Tietoa ei voi paikantaa, joten sen sijaan keskitytään tietoa säilövän tietopalvelimien paikannukseen ja siihen, etteivät tiedot voi vuotaa selkomuodossa sallitun alueensa ulkopuolelle. Työn tarkoituksena ei siis ole esimerkiksi varmistaa, ettei palvelinkeskuksen omistaja eli palveluntarjoaja voi luoda varmuuskopioita tiedosta muihin sijainteihin, kunhan palveluntarjoajan tekemät varmuuskopiot eivät voi vuotaa tietoa selkomuodossa. Mahdollisia tutkimukseen liittyviä rajoituksia tullaan käsittelemään tarkemmin pohdintaluvussa.

Työ koostuu useasta eri luvusta, ja tutkielmassa edetään seuraavasti. Johdantoa seuraavassa toisessa luvussa käsitellään aiheen taustaa, johon kuuluu tarkempi perehtyminen ongelmaan luonteeseen, aiemmassa tutkimuksessa esitettyjen ratkaisujen läpikäynti ja analyysi, tarkempi kuvaus aiemmin mainituista fyysisistä laitteista, *sijainnintodennuslaitteista*, sekä myöhemmin työssä käytettäviä määritelmiä. Kolmannessa luvussa kuvataan käytettävät tutkimusmenetelmät, ongelma ja sen rajaus sekä tarkat vaatimukset.

Suunniteltu protokollasto esitellään neljännessä luvussa, jossa myös käsitellään sen suunnittelutyötä. Luvussa kuvataan myös lopullinen ratkaisu, jossa fyysistä laitetta ja protokollastoa käyttävä yhdistelmä ratkaisee aiemmin esitetyn ongelman. Luvussa käsitellään lyhyesti myös prototyyppiä, tarkalleen ottaen sen tarkoitusta ja luonnetta. Viidennessä luvussa arvioidaan kehitetty ratkaisu aiemmin esitettyjen vaatimusten pohjalta sekä vertaillaan sitä aiemmin esitettyihin ratkaisuihin. Prototyyppiä käytetään joidenkin vaatimusten arviointiin käytetyn datan keräämiseen.

Työn kuudes luku on pohdintaluku, jossa pohditaan ratkaisun toteuttamiseen käytännössä liittyviä ongelmia ja sen mahdollisia rajoitteita. Viimeisessä luvussa tehdään lopulliset johtopäätökset, ehdotetaan jatkotutkimusaiheita sekä suoritetaan yhteenveto työstä ja sen aikaansaannoksista.

2 Tausta

Tässä luvussa käsitellään aiheen teoreettista taustaa ja käsitteitä sekä perehdytään tarkemmin ongelmaan, jonka ratkaisemiseksi tämä tutkimus on tehty.

2.1 Pilvipalvelimen paikannus

Perimmäisenä ongelmana on tarjota pilvipalvelun käyttäjälle keino selvittää, missä maantieteellisessä sijainnissa hänen tietonsa ja niitä käsittelevä palvelin on. Tätä ongelmaa, kuten muitakin yleisesti tietojenkäsittelytieteeseen liittyviä ongelmia, voi lähestyä usealla eri tavalla, mutta hankaloittavana tekijänä on vaatimus sijaintitiedon luotettavuudesta. Käyttäjän tulee voida tietää tietonsa sijainti varmuudella niin, ettei esimerkiksi palvelinkeskuksen omistaja tai ulkoinen toimija pysty väärentämään sijaintipyyntöön kuuluvaa tietoa ja näin väärentämään sijaintitulosta.

Sijaintitiedon tarkkuus voi olla myös usein tärkeä tekijä. Karkea tieto, kuten maanosa, maa tai tarkemmin ottaen lainsäädännöllinen alue voi olla riittävä suureen osaan tarkoituksista, mutta tarkemmalla sijaintitiedolla on omat käyttötarkoituksensa. Lainsäädäntö voi olla tietyissä tapauksissa tarkoin alueellista, maiden rajat voivat siirtyä, ja joskus palvelinkeskus voi olla niin lähellä sellaista rajaa, jossa tieto siitä, kummalla puolen rajaa ollaan, voi olla elintärkeä. Jos sijaintitieto on tarpeeksi tarkka yksilöimään palvelinkeskuksen, jotka ovat harvoin yhtä korttelia suurempia, tarkkuuden voi kokea riittäväksi lähes kaikkiin tarkoituksiin. Sekä palvelimen että tiedon sijainti tulee varmistaa, sillä tiedon käsittelyyn usein liittyy samanlaisia rajoituksia kuin sen tallentamiseenkin.

Myös käytännön tekijöillä on merkitystä. Luotettavuuden lisäksi ratkaisun tulisi toimia myös suuremmillakin palvelinkeskuksilla. Jos ratkaisun tulee olettaa päätyvän laajempaan käyttöön, skaalautuvuuden merkitystä ei tule unohtaa. Korkeat asennus- tai käyttökustannukset voivat toimia pelotteena käyttöönottoa vastaan. Tiedon siirron sen omistajan suostumuksella tulisi olla mahdollisimman helppoa, mutta tämän tärkeys on edellä mainittuja tekijöitä vähäisempi. Näiden kaikkien tekijöiden vaikutusta on hankalampi arvioida ilman tarkempaa tutkimusta, joten niitä arvioidaan tässä työssä vain teoreettisella tasolla.

2.2 Naiivit ratkaisut

On aihetta käsitellä joitain paikannusongelmaan liittyviä ratkaisuja, jotka voidaan syystä tai toisesta hylätä suoraan ilman tarkempaa tutkimusta, koska ne eivät täytä edes aiemmassa alaluvussa asetettuja vaatimuksia (joita tullaan myöhemmin tarkentamaan). Näiden käsittely osoittaa, ettei ongelmaa voi ratkaista triviaalisti.

1. **Keskitetty paikkatietopalvelin**: palvelinkeskuksen omistaja ottaisi käyttöön keskitetyn palvelimen, joka tarjoaa muille lähiverkossa oleville palvelimille ja mahdollisesti myös ulkoverkkoon keinon pyytää sijaintitietoa. Sitä, miten kyseinen paikkatietopalvelin tietää sijaintinsa, ei tarkoin määritellä eivätkä asiakasohjelmat tiedä tiedon lähdettä.

   Asiakas ei voi luottaa sijaintitietoon, koska hänellä ei ole syytä olettaa sijaintitiedon olevan luotettava. Palvelinkeskuksen omistaja voi helposti saada paikkatietopalvelimen antamaan asiakasohjelmille väärää tietoa, kun taas ulkopuoliselle hyökkääjälle paikkatietopalvelimesta tulee hyökkäyksen kohde. Tietoturvan osalta kyseinen keskitetty palvelin on yksittäinen haavoittuvuuspiste.

2. **Satelliittipaikannusmoduuli**: jokaiseen palvelimeen tai palvelinhyllyyn asennettaisiin satelliittipaikannusmoduuli, joka tarjoaa esimerkiksi GPS-, Glonass-, BeiDou- tai Galileo-navigointijärjestelmän tarjoamia tietoja. Asiakasohjelma voi sitten laitteistotukensa kautta kysyä moduulilta sijaintitietoa ja välittää pyynnön jollain erikseen sovitulla tavalla ulkoverkkoon.

   Satelliittipaikannukseen perustuvia radiosignaaleja voi häiritä ja näin estää moduulin toiminta. GPS-tiedon väärentäminen on osoitettu mahdolliseksi käytännössä [20]. Sekä palvelinkeskuksen omistaja että ulkopuolinen hyökkääjä voi tietyissä tapauksissa pienilläkin resursseilla näin sekoittaa moduulin toiminnan. Moduulin asennuskustannukset ovat mittavat, sillä olemassa oleville palvelimelle pitää tehdä laitteistomuutoksia.

3. **IP-osoitteeseen perustuva sijaintitieto**: palvelinkeskuksessa ajettava asiakasohjelma voisi selvittää julkisen IP-osoitteensa luotettavasti, tai ulkoisesti selvittää sovelluspalvelimensa IP-osoitteen, ja sitten käyttää IP-osoitteeseen perustuvaa sijaintitietoa (GeoIP) selvittääkseen sijaintinsa.

GeoIP:n sijaintitieto ei ole tarkka ja sen tarkkuus rajoittuu usein vain valtion tai maan tarkkuudelle [31]. Tulokset perustuvat ulkoisiin tietokantoihin, joiden ajantasaisuus on usein kyseenalaista. Vastikään varatut IP-osoitteet voivat aiheuttaa ongelmia. Ulkoisen IP-osoitteen saaminen luotettavasti on oma ongelmansa ja vaatii joko sisäpuolelta lisätietoa palvelinkeskuksen ja sen yritysverkon rakenteesta, tai ulkopuolelta julkisen IP-osoitteen, joita palvelinkeskuksen omistajalla ei ole syytä julkistaa.

2.3 Aiempi tutkimus

Pilvipalvelimien paikannus on noussut tärkeäksi tutkimusaiheeksi vasta viime vuosikymmenellä ja ongelmaan on kehitetty useita eri ratkaisuja, joista löytyy lukuisia julkaisuja. Tässä alaluvussa käydään joitain niistä läpi, selitetään niiden toimintaperiaatteet lyhyesti ja arvioidaan ne aiempien vaatimusten perusteella. Joitain aiemmista ratkaisuista tullaan myöhemmin käyttämään vertailuun työn pohdintavaiheissa.

Yksi ensimmäisistä tähän liittyvistä töistä on Albeshrin, Boydin ja Nieton GeoProof vuodelta 2012 [2]. Kyseessä on kryptografinen protokolla, jossa sijainti varmistetaan arvioimalla etäisyys tarkistajan ja palvelinkeskuksen välillä viivetietoon perustuen. Protokolla ei vaadi luottamusta palvelinkeskuksen omistajaan, mutta vaatii luotetun tarkastajan (”third-party auditor”), joka osaa varmistaa tiedon sijainnin varmistuslaitteen puolesta. Varmistuslaite on palvelinkeskukseen kytketty laite, jossa on satelliittipaikannusantenni, ja jonka oletetaan olevan sellainen, ettei sen sisäiseen toimintaan voi vaikuttaa palvelinkeskuksen omistaja tai ulkoinen tekijä.

Tiedon sijainti ja saatavuus varmistetaan käyttämällä POS (proof of storage) -protokollaa, tarkalleen ottaen Juelsin ja Kaliskin POR (proof of retrievability) -protokollaa [18]. Tämä yhdistetään kryptografiseen haasteprotokollaan, jossa haasteeseen vastaaja, tiedon omistaja eli jokin palvelinkeskuksen palvelin, on vastattava haasteeseen tietyn ajan sisällä.

Laitteen sijainnin perustuminen GPS-tietoon mahdollistaa sijainnin häirinnän ja väärentämisen. Artikkeli mainitsee asian, muttei kiinnitä siihen juurikaan huomiota ja ehdottaa kuin ohi mennen triangulaatiota luotettuun laitteeseen. GeoProof todistaa vain tiedon, muttei palvelimien eikä myöskään tiedosta tehtyjen kopioiden sijaintia [28]. Yleisesti ottaen GeoProof on kuitenkin tarkoitukseen erittäin lupaava protokolla ja hyvä vertailukohde, josta löytää yhtymäkohtia myöhemmin esitettäviin sijainnintodennuslaitteisiin.

Samat tekijät julkaisivat kaksi vuotta myöhemmin parannellun version protokollasta, joka parantaa sijainnin tarkkuutta suorittamalla erillisen mittausvaiheen ja näin arvioimalla haasteen laskenta-aikaa, joka voidaan vähentää lopullisesta viipeestä [1]. Aiemmin mainittuihin epäkohtiin parannuksilla ei kuitenkaan ole vaikutusta.

Toinen lupaava protokolla on Paladin, Aslamin ja Gehrmannin vuonna 2014 esittelemä protokolla, jota nimitetään tästä lähtien TGADP:ksi (artikkeli ei nimeä protokollaa, joten tämä nimi perustuu artikkelin otsikkoon). Protokolla perustuu tiedon salaamiseen TPM:ään (tietoturvapiiriin) tallennettavilla avaimilla [27]. Palvelinkeskuksen osana on luotettu laite, johon salausavain on salattu niin, että sen purkaminen onnistuu vain samassa sijainnissa (”sinetöinti”). Kun käyttäjä lähettää tietoa palvelimelle, niiden mukana lähetetään tieto sallituista sijainneista, ja tiedot säilötään vain sen mukaan asiakkaan sallimille palvelimille.

Sijaintitieto TGADP:ssa perustuu GPS-paikannukseen, jonka väärentämisestä on jo aiemmin puhuttu, eikä siihen voi siten luottaa. Artikkelissa mainitaan tämä mahdollisuus, mutta oletetaan GPS-tiedon olevan luotettava, eli ero on tietoturvamalleissa. Menetelmä myös perustuu keskitettyyn tietokantaan, jolla sijainnista erotetaan tarvittavat sijaintitiedot, kuten hallinnolliset alueet. Tämäkin toimii mahdollisena haavoittuvuutena.

Tiedon tallennusvaiheessa palvelinkeskuksen omistajaan tulee olla luottaminen, koska järjestelmä lähettää tiedon ja sen salausavaimen niille palvelimille, joiden se kokee olevan oikeassa sijainnissa. Palvelimien siirtäminen muualle estetään TPM:n avulla, koska siirtyminen eri sijaintiin johtaa eri TPM-tilaan, jolla sinetöityä salausavainta ei voi purkaa. Tämä kuitenkin vaatii, että kyseisissä palvelinkoneissa on TPM-moduuli, johon avaimen voi turvallisesti säilöä.

Näiden kahden erittäin lupaavan protokollan ja vertailukohteen lisäksi on ehdotettu myös muita protokollia, jotka eivät vaikuta täyttävän kaikkia vaatimuksia.

Noman ja Adams esitteli vuonna 2012 DLAS-protokollan [25]. Kahta vuotta myöhemmin samat tekijät julkaisivat artikkelin eräästä tarkemmasta toteutuksesta, laitteistopohjaisesta DLAS:sta (HDLAS) [26]. Menetelmä perustuu luotettuihin laitteisiin, jotka käyttävät TPM:ää, sekä PDP (*provable data possession*) -protokolliin, joilla tietoa säilövä palvelin voi todistaa säilövänsä jonkin asiakkaan sille aiemman lähettämän tiedon.

DLAS vaatii edes jonkinasteista luottamusta palvelinkeskuksen omistajaan, johon ei kuitenkaan voi vaatimusten perusteella välttämättä luottaa. Protokollassa tiedostopalvelin lähettää satelliittisijaintinsa luotetulle laitteelle. Artikkeli ehdottaa TPM:n käyttöä GPS-sijainnin todentamiseen, mutta tämä suojaa sitä vain muutoksilta lähetyksen aikana. Palvelin voisi väärentää sijaintitietonsa jo ennen niiden lähettämistä. DLAS:n sijaintitieto ei myöskään perustu minkäänlaiseen ulkoiseen tietoon muuten kuin GPS:n osalta ja näin sitä voidaan virtualisoida [10].

Gondrae ja Peterson esittelivät vuonna 2013 CDBG (constraint-based data geolocation) -protokollan [11]. Protokollan periaate on melko samanlainen GeoProofin kanssa: tieto jaetaan tietynkokoisiin lohkoihin ja tiedon omistaja, jonka oletetaan olevan palvelinkeskuksen ulkopuolella, lähettää luotettujen ”maamerkkien” kautta haasteen palvelinkeskukselle. Maamerkit voivat sitten viiveensä perusteella arvioida tiedon sijainnin käyttämällä hyväkseen tilastollista mallia.

CDBG:n tarkkuus riippuu suuresti maamerkkien määrästä ja sijainnista, eivätkä artikkelin tulokset olleet koskaan kovinkaan tarkkoja. Keskihajonta oli suuri, usein satoja kilometrejä. Palvelinkeskuksen omistaja voi vaikuttaa sijaintiarvion tarkkuuteen vastaamalla vain osaan haasteista. Maamerkkien tarjonta jäisi luotetun kolmannen osapuolen vastuulle, mikä tekee ratkaisusta käytännössä hankalan. Luottamussuhde kyseisiin kolmansiin osapuoliin tulisi muodostaa erikseen, ja järjestelmään jäisi heidän palveluilleen jatkuva riippuvuus, jotta sijaintitieto voitaisiin varmistaa. Tiedon sijaintia ei myöskään taata, vaan ainoastaan sovelluspalvelimen.

Eskandarin, de Oliveiran ja Crispon VLOC-protokolla vuodelta 2014 perustuu paljolti samaan ratkaisuun, jossa luotettujen palvelimien avulla arvioidaan palvelinkeskuksen sijainti trianguloimalla [10]. Toisin kuin CDBG:n tapauksessa, VLOC ei käytä erityisiä maamerkkejä, vaan sen arviot perustuvat olemassa oleviin verkkosivuihin ja niiden tunnettuihin sijainteihin. Tämä lähestymistapa on arvatenkin vähemmän luotettava, sillä muita verkkosivuja käytetään ikään kuin ponnahduslautoina, ja jos näiden omistajat tekevät järjestelyihinsä muutoksia, mittaukset voivat menettää merkityksensä.

Sama artikkeli kuvaa tarkemmin menetelmää, jolla etäisyys arvioidaan viipeen pohjalta [10]. Palvelinkeskuksessa oleva laite lähettää useaan eri aiemmin valittuun verkko-osoitteeseen viestin ja mittaa viipeen perusteella etäisyyden. Ennen mittauksia viipeen arviointipolynomi

on alustettava tilanteen mukaiseksi iteratiivisella prosessilla. VLOC:n tarkkuus on käytännössä melko alhainen (parhain arvio on 150 kilometrin säteellä), se ei takaa tiedon sijaintia ja sen riippuvuus maamerkkeinä käytettäviin verkko-osoitteisiin on kyseenalainen.

Jaiswalin ja Kumarin vuonna 2015 kuvaama IGOD-protokolla on hyvin samankaltainen CBDG:n kanssa ja kärsii samoista ongelmista. Sekin perustuu erikseen olemassa oleviin maamerkkipalvelimiin, joiden mittaamien viipeiden perusteella voidaan arvioida sijainti [16]. IGOD:n määritelmään on erikseen määritetty korkeintaan 10 mailin virheraja sijainnille, jota ei kuitenkaan aina noudateta. Näin menetelmän tarkkuus on parempi, muttei silti riittävän hyvä ainakaan kaikissa tapauksissa.

Usean tutkijan yhteisvoimin vuonna 2015 julkaistu SecLoc toimii hieman eri tavalla, ja pohjautuu siihen, että käyttäjä valitsee ennalta palveluntarjoajan luettelolta sijainnin osalta sopivat palvelimet [22]. Tietoja siirrettäessä prosessi tulisi aloittaa kokonaan alusta, tosin artikkelissa esitellään myöhemmin myös versio, joka ei tätä vaadi. Ongelmana on kuitenkin se, että SecLoc on suunniteltu sen oletuksen varaan, että palveluntarjoajaan voi luottaa edes jonkinasteisesti, kuten että esimerkiksi tietyt verkon solmut eivät vaihda paikkaa. Tiedot on salattu palvelimien puolella, mutta mikään ei estä palvelimien tietojen siirtämistä muualle ja palveluntarjoajan viestien välitystä keskuksesta toiseen. Näin protokollan olettama tietoturvamalli ei toimi, jos palveluntarjoajaan ei ole luottamista.

Vuonna 2015 julkaistu NIST:n artikkeli, jossa toteutettiin prototyyppinä luotettavan sijaintitiedon saaminen pilvipalvelussa, perustuu Trusted Computingin ja Intelin Trusted Execution Technologyn varaan. Artikkelissa kuvaillaan tapa luoda niin sanottuja luotettuja laskentajoukkoja, joille annetaan luontivaiheessa sijaintitiedot [3]. Sitä, mistä sijainti saadaan, ei tarkoin määritellä, vaan ainoastaan, että se saadaan jostain ulkopuolisesta lähteestä.

Kyseinen ratkaisu varmistaa vain palvelinten, ei niiden käsittelemän tiedon, sijainnin. Se myös perustuu suljetun lähdekoodin ohjelmistopinoon, jonka turvallisuutta ei voi ulkoisesti varmistaa [28].

2.4 Etäisyydenrajausprotokollat

Etäisyydenrajausprotokolla (*distance-bounding protocol*) on protokolla, jonka tarkoituksena on selvittää etäisyys kahden laitteen välillä. Etäisyydenrajoitusprotokollat nojaavat käytännössä siihen olettamukseen, että kahden laitteen välinen etäisyys voidaan päätellä mittaamalla niiden välinen kierrosaika (*round-trip time*). Ajan oletetaan riippuvan vain etäisyydestä, ei esimerkiksi suunnasta tai muista tekijöistä. Näin kierrosajasta voidaan johtaa etäisyysraja kertomalla se viestin kulkunopeudella, yleensä valonnopeudella. Tulos pitää vielä jakaa kahdella, koska kierrosaika mittaa viivettä viestin kulussa edestakaisin, kun taas etäisyys halutaan selvittää vain yhteen suuntaan. Käytännön toteutuksissa on erittäin tärkeää, että kaikki käsittelyviiveet ovat mahdollisimman alhaisia, sillä jo yhden mikrosekunnin viive suuntaansa kasvattaa etäisyysrajaa noin 300 metrillä.

Lähes kaikki protokollat perustuvat tästä syystä erilaisiin langattomiin teknologioihin, joissa kaksi isäntäkonetta keskustelevat suoraan keskenään. Esimerkkinä tästä on RFID:hen perustuva etäisyydenrajaus [12]. Jos oletus on, ettei sovelluspalvelimille tule asentaa lisälaitteistoa ratkaisun toteuttamiseksi, nämä vaihtoehdot eivät kelpaa.

2.5 Sijainnintodennuslaitteet

Sijainnintodennuslaite (*location assurance device*, LAD) on fyysinen palvelinkeskukseen asennettava sen sijainnin todentava laite. Idea muistuttaa aiemmin käsitellyssä GeoProof-artikkelissa kuvailtua varmistuslaitetta, mutta laitteella on tätäkin enemmän vastuuta. Käsite on syntynyt itse ideaa myöhemmin: jälkimmäinen esiteltiin ensimmäistä kertaa vuonna 2017 julkaistussa artikkelissa alun perin nimellä *site anchor* [14]. Nokia Bell Labs, joka on johtanut sijainnintodennuslaitteisiin liittyvää suunnittelu- ja kehitystyötä, haki sille patenttia, joka myönnettiin vuonna 2019 [15].

Sijainnintodennuslaitteen perusajatukseen kuuluu kolme tärkeää kohtaa. Ensimmäinen on, että kyseinen laite asennetaan pysyvästi palvelinkeskukseen ja kytketään siellä lähiverkossa oleviin palvelimiin jollakin tavalla. Asennus tapahtuu fyysisesti ja niin, että laitteeseen kajoaminen sen käyttöönoton jälkeen esimerkiksi aiheuttaa rakennuksessa hälytyksen tai on mahdotonta ilman että paikannuslaite hajoaa tai vaurioituu pysyvästi. Laitteen toimintaan vaikuttaminen tulee estää hyökkääjiltä, jopa palvelinkeskuksen omistajalta, asennuksen

jälkeen. Laite voidaan esimerkiksi pultata kiinni rakenteisiin tai jopa muurata sisään uusiin palvelinkeskuksiin.

Toinen kohta on, että laitteeseen ei pidä voida hyökätä verkon kautta, ja näin sekä fyysinen että ohjelmallinen hyökkääminen laitetta vastaan tehdään mahdottomaksi. Hyökkäyspinta-alan vähentämiseksi laitteen tulisi olla mahdollisimman yksinkertainen ja tehdä vain sen, mitä sen täytyy. Tämä vähentää myös laitteen kustannuksia. Jos hyökkääminen on tehty mahdottomaksi, toimivaan sijainnintodennuslaitteeseen voi nyt luottaa omassa palvelinkeskuksessaan. Todennuslaitteen ohjelmisto on ulkopuolisesti tarkastettu ja säilötty *lukumuistille* (*read-only memory*), eikä siten tue päivityksiä tai muita muutoksia.

Kolmas kohta on, että sijainnintodennuslaitteen tiedot ovat esimääriteltyjä, eikä laite luota ulkopuolelta saatavaan sijaintitietoon, kuten satelliittipaikannuksen avulla haettuun sijaintiin. Sijaintitiedot säilötään pysyvästi laitteen lukumuistiin sen käyttöönottovaiheessa, ja menettelyn tarkistaa ulkopuolinen tarkastaja.

Käytännössä sijainnintodennuslaitteille olisi myös lisävaatimuksia, kuten alhaiset valmistuskustannukset ja siten alhainen hinta (jotta laite on tarvittaessa korvattavissa, esimerkiksi jos sijainti muuttuu tai laite hajoaa murtoyrityksen yhteydessä), korkea luotettavuus ja käyttöikä sekä mahdollisesti myös keino havaita tunkeutumis- tai häirintäyritykset. Jälkimmäiseen voi kuulua liiketunnistin tai varajärjestelmänä satelliittipaikannus, mutta tällöin sijainnin väärentämistä esimerkiksi satelliittisignaaleja väärentämällä ja sen vaikutuksia pitää silloin pyrkiä ehkäisemään. Luotettavin tieto laitteessa on sellainen, johon ei voi jälkeenpäin vaikuttaa.

Pilvipalvelimella ajettava sovellus voisi siten palvelinkeskuksessa, johon on asennettu sijainnintodennuslaite, kysyä lähiverkossa tavoitettavissa olevalta sijainnintodennuslaitteelta sijaintiaan. Näin tapahtuisi sovelluksen käynnistyessä sekä sitä ajettaessa säännöllisin tai satunnaisin väliajoin.

Tähänkin ratkaisuun liittyy kuitenkin ongelmia, vaikka oletettaisiin ettei sovelluksenkaan toimintaan kajota suoraan. Välissä oleva palvelinkeskuksen omistama välipalvelin tai reititin voisi muun muassa

1. muuttaa sijainnintodennuslaitteen lähettämiä tietoja

2. tekeytyä sijainnintodennuslaitteeksi ja lähettää väärennettyjä sijaintitietoja
3. toistaa aiempia sijainnintodennuslaitteen lähettämiä aitoja, ja mahdollisesti vanhentuneita, tietoja
4. ohjata sijaintipyynnön toisaalla sijaitsevalle sijainnintodennuslaitteelle ja sieltä takaisin.

Tämän lisäksi ratkaisu ei varmista muuta kuin sovelluspalvelimen sijainnin, ei esimerkiksi sitä, mihin tiedot tallennetaan. Sijainnintodennuslaite voi myös alusta alkaenkin sijaita muualla, jos sijainnintodennuslaitteen ja palvelimen välistä etäisyyttä ei tarkisteta millään tavalla. Palveluntarjoaja voi esimerkiksi asentaa vain yhteen palvelinkeskukseen sijainnintodennuslaitteen ja sitten ohjata kaikki sijaintipyynnöt sille.

Jotta nämä ongelmat saadaan ratkaistua, tulee sijainnintodennuslaitteelle ja sen käyttöön kehittää protokolla tai protokollasto, jonka avulla sijainnintodennuslaite otetaan käyttöön ja jonka avulla sovellus voi varmistua sekä omasta sijainnistaan että tietojensa sijainnista. Tämän työn tarkoituksena on juuri kehittää tämänkaltainen protokollasto, joka täyttää myöhemmin tarkemmin määritellyt vaatimukset.

2.6 Määritelmät

Seuraavia määritelmiä käytetään työssä tästä eteenpäin. Lisää käsitteitä määritellään ehdotettavan ratkaisun kuvauksessa.

- *Palvelinkeskus*, jolla on pysyvä fyysinen sijainti, koostuu yhdestä tai useasta eri palvelimesta.
- *Tietopalvelin* on palvelinkeskuksen palvelin, joka säilöö käyttäjien tietoja.
- *Sovelluspalvelin* on palvelinkeskuksen palvelin, joka ajaa käyttäjän sovellusta.
- *Välipalvelin* on palvelinkeskuksen palvelin, joka ei ole tietopalvelin tai sovelluspalvelin.
- *Pilvialusta* on yhdestä tai useasta palvelinkeskuksesta koostuva alusta, jolla pilvipalveluita ajetaan.
- *Palveluntarjoaja* on palvelinkeskuksen tai pilvialustan omistaja ja/tai hallinnoija. Omistaja ja hallinnoija voivat olla eri taho, jos jälkimmäinen vuokraa kapasiteettia ensimmäiseltä yhden tai useamman välikäden kautta.

- *Käyttäjä* on taho, joka ostaa palveluntarjoajalta kapasiteettia pilvialustasta pilvisovelluksen ajamiseen ja sen käyttämän tiedon säilömiseen.

3 Tutkimusmenetelmät

Tässä luvussa käydään läpi tutkimusmenetelmä, kuvaillaan myös tutkimuksen toteutusta, sekä määritellään ratkaisulta halutut vaatimukset ja tietoturvamalli, jonka puitteissa ratkaisun pitää olla turvallinen.

3.1 Suunnittelutiede

Tämän tutkimuksen muodoksi on valittu suunnittelutiede (*design science*, *design science research*, DSR). Se on tutkimusmuoto, jossa kehitetään jokin käytännön luomus (artefakti) ja arvioidaan sitä tieteellisin tavoin. Suunnittelutieteen käyttö tietotekniikan ja tietojenkäsittelytieteen aloilla on jatkuvasti yleistymässä.

Suunnittelutiede on sopiva tutkimusmenetelmä tälle aiheelle, koska tarkoituksena on ratkaista ongelma, joita aiempien tutkimuksen ehdottamat ratkaisut eivät tyydyttävissä määrin ratkaise. Suunnittelemalla ja arvioimalla oma protokollasto sijainnintodennuslaitteen yhteyteen saadaan aikaan uusi ratkaisu, jota voidaan arvioida sekä myöskin vertailla aiempiin ratkaisuihin.

Luomuksen suunnittelu on suunnittelutieteessä pohjimmiltaan iteratiivinen prosessi. Aiemman tietämyksen ja vaatimusten pohjalta voidaan kehittää luomus, jonka kehitystyötä jatketaan, kunnes saadaan vaatimukset täyttävä tuotos. Myöhemmät tutkimukset voivat asettaa eri vaatimuksia ja punnita työssä aikaan saatuja tuloksia uudessa valossa, ja näin kehittää alan tietämystä yleisesti.

Suunnittelutieteessä on Hevnerin ym. artikkelin mukaan seitsemän tärkeää ohjenuoraa [13]:

1. **Luomuksen suunnittelu** (*Design as an Artifact*): suunnittelutieteen tuloksena on oltava jokin luomus, tarkemmin joko konstruktio, malli, menetelmä tai toteutus.
2. **Ongelman merkittävyys** (*Problem Relevance*): suunnittelutieteen tarkoituksena on kehittää tärkeisiin ja oleellisiin ongelmiin ratkaisuja teknologiaa soveltaen.
3. **Suunnittelutyön arviointi** (*Design Evaluation*): luomuksen käytännöllisyys, laatu ja tehokkuus täytyy arvioida ja osoittaa arviointimenetelmiä oikeaoppisesti käyttäen.

4. **Tieteellinen panos** (*Research Contributions*): suunnittelutiedettä käyttävän tutkimuksen tulee johtaa selkeisiin ja todistettaviin tuloksiin joko luomuksen, siihen liittyvän tietämyksen tai menetelmien alalla.

5. **Tutkimuksen järjestelmällisyys** (*Research Rigor*): sekä luomuksen suunnitteluun että sen arviointiin on käytettävä tieteellisesti päteviä, järjestelmällisiä menetelmiä.

6. **Suunnittelu on löytämistä** (*Design as a Search Process)*: luomuksen luominen vaatii olemassa olevien menetelmien löytämistä ja niiden soveltamista ongelmaympäristöön.

7. **Tulosten viestiminen** (*Communication of Research*): suunnittelutiedettä käyttävän tutkimuksen on esitettävä asiansa niin, että sekä teknologiaan perehtyneet että päätöksiä tekevät pystyvät hyötymään sen esittämästä tiedosta.

Suunnittelutiede on pohjimmiltaan kuusivaiheinen prosessi [30]. Prosessin kuusi vaihetta ovat ongelman tunnistaminen ja sen perusteleminen, ratkaisun ja sen tavoitteiden määrittely, suunnittelu ja kehitys, luomuksen havainnollistaminen, sen arviointi sekä aiheen ja tulosten viestintä.

Tässä työssä noudatetaan suunnittelutieteen ohjenuoria sekä kuusivaiheista prosessia. Ongelman tunnistaminen ja perusteleminen tapahtui luvuissa 1–2, ratkaisun ja sen tavoitteiden määrittely luvuissa 2–3, suunnittelu ja kehitys luvussa 4, havainnollistaminen ja arviointi luvussa 5 ja viestintää tapahtuu luvuissa 4–7 sekä yleisesti koko tämän työn aikana.

Ohjenuorien noudattamiseen on myös kiinnitetty huomiota. Sekä suunnittelu- että arviointiprosessi tulee käyttämään järjestelmällisiä menetelmiä, ja jälkimmäisen suhteen on otettu esimerkkiä Hevnerin ym. artikkelissa esitetyistä arviointimenetelmistä.

3.2 Ongelma ja toteutus

Tässä työssä luomus on protokollasto, jolla palvelinkeskuksessa sijaitseva sovelluspalvelin voi keskustella sijainnintodennuslaitteen kanssa. Sovelluspalvelimen pitää pystyä protokollien avulla saamaan sijainnintodennuslaitteelta luotettava sijaintitieto, koska sovelluksen tietoja ei saa päätyä selkomuodossa esimerkiksi tietyn alueen ulkopuolelle.

Protokollaston suunnittelu tapahtui iteratiivisella prosessilla suunnittelutieteen määritelmän mukaisesti. Uusi protokollasto käyttää aiemmin kehitettyjä ja toistaiseksi turvalliseksi todet-

tuja kryptografisia protokollia ja menetelmiä. Tähän tarkoitukseen koottiin myös lähdeaineistoa, joita käsitellään suunnitteluluvussa tarkemmin. Protokollaston ensi version kehityksen jälkeen sitä verrattiin vaatimuksiin ja kehitettiin jatkoversioita, kunnes syntyi lopullinen luomus joka täyttää kaikki annetut vaatimukset.

3.3 Rajaus

On tärkeää ennen varsinaisiin vaatimuksiin paneutumista tehdä selväksi, miten aihetta on rajattu. Seuraavat olettamukset tehdään, sillä kyseiset asiat eivät ole tutkittavina.

Tämä työ keskittyy vain sijainnintodennuslaitteen yhteydessä käytettävään ohjelmistoon, eli siihen, miten palvelinkeskuksella ajettava sovellus voi todentaa protokollien kautta oman ja tietonsa sijainnin. Tutkimuksessa oletetaan, että sijainnintodennuslaite on, kuten aiemmin käsitelty, turvallinen eikä sen sisäiseen toimintaan tai sijaintiin voi vaikuttaa edes palvelinkeskuksen omistaja. Tämä vaatii, että sijainnintodennuslaite on asennettu pysyvästi, esimerkiksi pulttaamalla tai muuraamalla sen palvelinkeskukseen kiinni, niin ettei sitä voi poistaa tai siihen murtautua sitä vahingoittamatta.

Tämän vaikutuksia ei käsitellä tutkimuksessa, eikä myöskään sitä, minkälainen laitteisto sijainnintodennuslaitteella on. Joitain olettamuksia sen suhteen tehdään, kuten se, että laitteisto kykenee suorittamaan kryptografisia protokollia ja että se voi muodostaa verkkoyhteyksiä paikallisessa verkossa oleviin muihin laitteisiin. Laitteistovalinnoilla on vaikutusta ratkaisun skaalautuvuuteen, joten protokollista ja ohjelmistopuolesta pyritään tekemään mahdollisimman kevyt eli vähän laitteistovaroja käyttävä, ei kuitenkaan turvallisuuden eikä toiminnallisuuden kustannuksella.

Sijaintitietojen oletetaan tulevan lukumuistista, johon ei voi enää kirjoittaa, ja minkäänlaista satelliittipaikannusvaihtoehtoa ei oleteta olevan. Niiden tarkempaa muotoa ei myöskään tässä tutkimukesssa määritellä. Tiedot voivat esimerkiksi sisältää koordinaatteja, maan eli valtion tai hallintopiirin. Ainoa olettamus tässä tutkimuksessa on, että sijaintitiedoissa on jokin sovellukselle sopiva tieto, jonka perusteella sovellus voi päätellä, onko sovelluspalvelimen ja/tai tietopalvelimen nykyinen sijainti sille hyväksyttävä vai ei.

Protokollasto suunnitellaan tapaukseen, jossa sovellusta ajetaan palvelinkeskuksessa, ei sen ulkopuolella. Jos sovellus ei saa haluamaansa sijaintitietoa, se voi yksinkertaisesti aiheuttaa

virheen ja lopettaa toimintansa. Protokollan sovelluspalvelimella ajettavan osion turvallisuus oletetaan (tästä tarkemmin tietoturvamallissa). Jos palvelinkeskuksen ulkopuolella oleva asiakas haluaa varmistaa, että hänen sovelluksensa ja tietonsa ovat oikeassa sijainnissa, hän voi toteuttaa sovellukseensa tavan pyytää sijainnin tarkistusta manuaalisesti ja käyttää kyseistä toiminnallisuutta. Sovelluksen tulee muutenkin aika ajoin tarkistaa sijaintinsa uudelleen, ei vain käynnistyksen tai alustuksen yhteydessä.

Protokollaston varsinaista toteutusta ohjelmallisella tasolla ei käsitellä tarkemmin, eikä siten esimerkiksi turvallisuutta sivukanavahyökkäyksiä vastaan.

3.4 Tietoturvamalli

Tässä alaluvussa on lyhyt kuvaus ratkaisun tietoturvamallista.

Oletetaan, että palveluntarjoaja CSP tarjoaa pilvialustaa IP, jonka eräs palvelinkeskus DC sisältää palvelintensa joukossa sovelluspalvelimen AS, tietopalvelimen DSS ja mahdollisesti yhden tai useamman välipalvelimen. Pilvialusta vuokraa pilvipalvelusta palvelintilaa käyttäjille, jotka haluavat ajaa sovelluksiaan palvelimilla. Käyttäjä U on vuokrannut palveluntarjoajalta tilaa ja valinnut palvelinkeskuksensa DC sijaintirajoitteidensa mukaisesti, mutta haluaa olla varma, että sekä hänen sovelluksensa että hänen tietonsa pysyvät kyseisessä sijainnissa.

Palveluntarjoaja saattaa hajauttaa tietopalvelimen ja sen säilömät tiedot useaan eri sijaintiin tiedon säilyvyyden varmistamiseksi. On täysin mahdollista, että tietopalvelimesta luodaan useampia varmuuskopioita, joista osa saatetaan siirtää myös eri sijaintiin. Palveluntarjoaja CSP saattaa myös yrittää siirtää sovellusta muihin palvelinkeskuksiin kuin käyttäjän valitsemaan esimerkiksi tapahtuvan huollon tai vian takia.

Palveluntarjoajaan CSP voi luottaa vain silloin, kun ulkoinen tarkastaja eli auditoija on paikalla, eli sijainnintodennuslaitteen käyttöönoton aikana. Kun käyttäjä vuokraa palvelinkeskuksesta palvelinkapasiteettia, sijainnintodennuslaitteen oletetaan olevan jo asennettu: palvelinkeskuksessa on sijainnintodennuslaite LAD. Käyttäjä U ei siis missään vaiheessa luota palveluntarjoajaan CSP tässä tietoturvamallissa. Sijainnintodennuslaitteen tietojen muodon oletetaan olevan käyttäjälle jo erikseen tiedossa esimerkiksi avoimen standardin kautta. Myös sijainnintodennuslaitteen aitouden oletetaan olevan tarkistettavissa

jonkin muun kuin palveluntarjoajan hallinnoiman rakenteen kautta. Sijainnintodennuslaitteen julkinen avain on esimerkiksi saatavilla luotettavasta lähteestä.

Sijainnintodennuslaitteen LAD sisäisen toiminnan oletetaan olevan luotettavaa, eli laitteen oletetaan antavan luotettavaa tietoa. Sijaintitiedot ovat esimerkiksi lukumuistissa, jolloin palveluntarjoaja tai ulkoinen hyökkääjä ei pysty tietoja muuttamaan itse laitteen sisällä. Tiedoilla ei ole ulkoista lähdettä, kuten satelliittipaikannusta, jonka tietoja voisi väärentää.

Mahdollisia hyökkäyksiä varten palvelinkeskuksella oletetaan olevan yksi tai useampi järjestelmänvalvoja A, jolla on täydet oikeudet kaikkiin palvelinkeskuksen varsinaisiin palvelinkoneisiin sekä reitityslaitteistoon, muttei sijainnintodennuslaitteeseen, jolla ei ole hallintakäyttöliittymää tai -rajapintaa. Järjestelmänvalvoja voi toimia palveluntarjoajan, omien tai ulkoisen hyökkääjän eduksi, ja hänen oletetaan pääsevän palvelimille ja muille laitteille (paitsi sijainnintodennuslaitteelle) pääkäyttäjän oikeuksin. Fyysinen pääsy on mahdollista, mutta myöhemmin käsiteltävän luotetun ajoympäristön oletetaan toimivan mustana laatikkona, johon ei voi murtautua. A pystyisi esimerkiksi pakottamaan sovelluspalvelimen AS ajamaan sovelluksensa esimerkiksi A:n hallinnoimalla virtuaalikoneella, joka sijaitsee palvelinkeskuksen DC ulkopuolella. Palvelinkeskuksen omistaja voi esimerkiksi käskeä A:n siirtämään sovelluksen ja sen tiedot eri palvelinkeskukseen ja näin pyrkiä vuotamaan käyttäjän U tietoja.

Sovelluspalvelimella oletetaan olevan TPM-moduuli tai muu vastaava ratkaisu, joka pystyy säilömään kryptografisia avaimia turvallisesti, vaikka hyökkääjä saisi laitteeseen pääkäyttäjän oikeudet. Siinä oletetaan myös olevan jonkinlainen luotettu ”musta laatikko” eli suoritusympäristö, jossa voidaan ajaa luotettua koodia ilman että järjestelmänvalvoja pystyy vaikuttamaan sen toimintaan pääkäyttäjänkään oikeuksilla. Protokollia ajavan koodin oletetaan siis olevan luotettava. Jos järjestelmänvalvoja pystyisi vaikuttamaan täysin ohjelman kulkuun, ei voitaisi antaa minkäänlaisia takuita esimerkiksi siitä, ettei jonkinlainen tiedon salaamiseen käytetty avain vuoda jo alustusvaiheessa.

Palvelinkeskuksen omistajan eli palveluntarjoajan CSP oletetaan pyrkivän tarjoamaan mahdollisimman keskeytymätöntä palvelua käyttäjilleen, jotta käyttäjät pysyvät heidän asiakkainaan. Hän ei siis halua palvelunsa estyvän, mutta hän voi kuitenkin yrittää siirtää sovelluksen tai sen tiedot muualle. Ulkoisten hyökkääjien tai palvelinkeskuksen omistajan

tahallaan suorittamaa palvelunestohyökkäystä ei oteta tässä tietoturvamallissa huomioon, mutta sen mahdollisia vaikutuksia tutkitaan pohdintaluvussa.

3.5 Vaatimukset

Aiempien lukujen perusteella voidaan muotoilla sijainnintodennuslaitteen yhteydessä käytettävälle protokollastolle tarkemmat vaatimukset, joiden perusteella sen toteutusta arvioidaan. Näin tullaan siihen tulokseen, että sijainnintodennuslaitteen ohjelmistopuolen tulee täyttää seuraavat vaatimukset:

1. Sovelluspalvelimen, tietopalvelimen ja sijainnintodennuslaitteen tulee olla todistettavasti toisiaan lähellä. Tämä voidaan jakaa viiteen alavaatimukseen:
   (a) Sovelluspalvelimen ja sijainnintodennuslaitteen välille tulee muodostaa luotettava, salattu ja turvallinen yhteys, jonka sisältämiä sanomia välipalvelin tai reititin ei voi muuttaa.
   (b) Tietopalvelimen, joka säilöö asiakkaan tietoja, ja sijainnintodennuslaitteen välille tulee muodostaa luotettava, salattu ja turvallinen yhteys, jonka sisältämiä sanomia välipalvelin tai reititin ei voi muuttaa.
   (c) Sijainnintodennuslaitteen aitous tulee todentaa.
   (d) Protokollien tulee varmistaa, ettei toinen palvelin voi matkia sijainnintodennuslaitetta edes tallentamalla viestejä ja toistamalla niitä.
   (e) Sijainnintodennuslaitteen ja sovellus- sekä tietopalvelimen välinen fyysinen etäisyys on rajattava.
2. Ratkaisun tulee huomata palvelimen siirtäminen ensimmäisen tarkistuksen jälkeen.
3. Sovellus- ja tietopalvelimen siirto muualle tulee huomata, mutta olla toteutettavissa helposti mikäli siirto tapahtuu palvelinkeskuksen sisällä tai käyttäjä on antanut suostumuksensa palvelimen siirtoon.
4. Tiedon päätyminen muihin sijainteihin selkomuodossa on estettävä.
5. Ratkaisun tulee olla skaalautuva ja laitteistovaroja ei saa käyttää turhaan etenkään sijainnintodennuslaitteen puolella.

Vaatimus 1 käsittää kaikki yksittäiseen sijainnin tarkistukseen kohdistuvat vaatimukset. Kuten aiemmin on käsitelty, palvelinkeskuksenkaan omistajan ei pidä pystyä väärentämään

sijainnintodennuslaitteen antamia tietoja, joko suoraan tai epäsuorasti. Epäsuoran väärentämisen ehkäisemiseksi on myös varmistettava, ettei sovelluspalvelimen lähettämiä sanomia voi väärentää, jonka vaatimus 1.a määrittelee. Yhteyden pitää olla salattu eli protokollaston tulee tarjota luottamuksellisuus. Jos sanomia ei salattaisi, palveluntarjoaja voi esimerkiksi saada tietoa, jolla hyökätä sovelluksen sijaintitarkistusta vastaan.

Vaatimus 1.b on sama kuin 1.a, mutta käsittelee tietopalvelinta eikä sovelluspalvelinta. Tietoa säilövän palvelimen tulee olla samassa keskuksessa kuin sovellustakin ajavan, koska salattuunkin tietoon voi päteä esimerkiksi lainsäädännön osalta erinäisiä rajoituksia. Tietopalvelimen siirto muualle voi myös osoittaa palveluntarjoajan pyrkivän siirtämään tietoja muualle kuin missä asiakas haluaa niiden sijaitsevan.

Vaatimuksen 1.c tarkoituksena on estää palvelinkeskusta virtualisoimasta sovelluspalvelinta ja näin simuloimalla sijainnintodennuslaitetta tai asentamasta omaa sijainnintodennuslaitetta, joka tarjoaa palvelimille väärennettyä tietoa. Tietoturvamallissa on oletettu, että käyttäjä voi määrittää sovelluspalvelimen niin, että se pystyy tunnistamaan aidon sijainnintodennuslaitteen jostain ulkoisesta lähteestä saadun luotettavan tiedon pohjalta. Vaatimuksen 1.d tarkoitus on samanlainen, mutta sen on myös tarkoitus estää niin sanottu toistohyökkäys, jossa välipalvelin nauhoittaa aiemman sijaintipyynnön ja toistaa sen sellaisenaan takaisin.

Vaatimus 1.e käsittää tilanteet, jossa palveluntarjoaja ohjaa sovellus- tai tietopalvelinten sijaintipyynnön esimerkiksi toisessa palvelinkeskuksessa, kenties jopa toisessa valtiossa, sijaitsevaan sijainnintodennuslaitteeseen. Protokollien on varmistettava, ettei näin voi tapahtua. Pelkkä sijainnintodennuslaitteen todennus (vaatimus 1.c) ei auta, koska sovelluspalvelin voi esimerkiksi saada aluksi yhteyden oikeaan laitteeseen palvelinkeskuksessa, mutta sovelluspalvelinta voidaan myöhemmin siirtää niin, että sen yhteys alkuperäiseen sijainnintodennuslaitteen jää voimaan.

Vaatimukset 2–5 liittyvät enemmän käytännön toteutukseen. Vaatimus 2 tulee ottaa huomioon protokollien ja niiden käytön suunnittelussa, koska jos sijainti tarkistetaan vain kerran esimerkiksi palvelimen käynnistyessä, palveluntarjoaja voi siirtää palvelimen toiseen sijaintiin käynnistyksen jälkeen. Tämän takia sijainti on tarkistettava uudelleen, luultavammin satunnaisesti ja niin, ettei palveluntarjoaja voi ennalta arvata, milloin sovellus haluaa tarkistaa sijaintinsa.

Vaatimus 3 liittyy siirrettävyyteen. Käyttäjän tulee tarvittaessa pystyä siirtämään sovelluksensa muualle, jos hän siihen itse suostuu. Sovellus- tai tietopalvelimen siirtäminen palvelinkeskuksen sisällä voinee tapahtua myös ilman käyttäjän suostumusta, sillä sijainti ei muutu merkittävästi. Siirtäminen kauemmaksi tulee kuitenkin havaita.

Vaatimus 4 liittyy tietopalvelinvaatimukseen 1.b, mutta siihen lasketaan myös tapaukset, joissa palveluntarjoaja luo kopion tietopalvelimilla olevista tiedoista. Tietojen pääseminen selkomuotoisena muualle kuin minne käyttäjä niitä vaatii pitäisi estää edellä esitellyn tietoturvamallin puitteissa.

Vaatimus 5 on näistä vähiten tärkein ja sen käsittely jää melko pintapuoliselle tasolle, sillä sen käytännön vaikutuksia tulee arvioida tarkemmin varsinaisessa toteutusvaiheessa. Tämä tutkimus keskittyy protokollaston suunnitteluun, ja tähän tutkimukseen ei kuulu esimerkiksi tapaustutkimus siitä, miten ratkaisu toimii oikeassa palvelinkeskuksessa. Protokollien suunnittelussa otetaan kuitenkin laitteistovaatimukset jonkinasteisesti huomioon, sillä sijainnintodennuslaitteen kustannukset tulisi pitää mahdollisimman matalina.

Nämä vaatimukset otetaan huomioon sekä protokollaston suunnittelussa että arvioinnissa.

4 Suunnittelu

Tässä luvussa kuvataan suunnitteluprosessin eteneminen ja sen lopputuloksena syntynyt ratkaisu ja protokollasto, jota kutsutaan DAGLAP-protokollastoksi. DAGLAP-protokollaston ja sitä käyttävän ratkaisun suunnittelu tapahtui iteratiivisella prosessilla.

4.1 Vaaditut alemman tason protokollat

Protokollasto käyttää osina valmiita, aiemmin suunniteltuja kryptografisia protokollia ja menetelmiä. DAGLAP vaatii kaikki neljä seuraavaa:

1. Symmetrinen salausprotokolla, joka takaa viestien luottamuksellisuuden (salakuuntelija ei pysty ymmärtämään viestin sisältöä) lisäksi myös aitouden (aktiivinenkaan salakuuntelija ei pysty väärentämään viestin lähettäjää) ja eheyden (aktiivinenkaan salakuuntelija ei pysty muuttamaan viestin sisältöä). Kyseinen salausprotokolla vaatii symmetrisen avaimen K_{AB} (osapuolet A ja B) ja salaa sillä sanoman, jossa on yksi tai useampi osa, esimerkiksi $\{X, Y\}_{K_{AB}}$ (sanomassa asiat X ja Y). Sanoman voi purkaa vain samalla avaimella.

2. Allekirjoitusprotokolla, jonka varaan voidaan rakentaa julkisten avainten hallintajärjestelmä (PKI). Allekirjoitusprotokollalla on avainpari (SK_A, SK'_A), joista ensimmäinen on julkinen avain ja jälkimmäinen yksityinen eli salassa pidettävä. Sanoman X voi allekirjoittaa yksityisellä avaimella niin, että syntyy allekirjoitus $S = \{X\}_{SK'_A}$. Allekirjoituksen ja sanoman aitouden voi tarkistaa julkisella avaimella SK_A.

3. Todennettu avaintenvaihtoprotokolla, jolla kaksi osapuolta A ja B voivat päätyä yhteiseen symmetriseen salausavaimeen niin, että kumpikin osapuoli vaikuttaa lopputulokseen, mutta ilman, että aktiivinenkaan salakuuntelija voi saada tietoonsa kyseistä avainta. Todennus riittää yksisuuntaisesti, eli niin, että A:n aitous todennetaan B:lle.

4. Hallussapitotodistusprotokolla (proof of data possession protocol). Protokollan tarkoituksena on todistaa, että jollain palvelimella tai koneella on hallussa juuri se tieto, joka sille on aiemmin tallennettu. Todistaminen tapahtuu niin sanotuilla haasteilla, joita tiedon omistaja voi haastettavalle esittää. Haasteet on suunniteltu niin, ettei oikeaa vastausta voi käytännössä arvata ilman kyseistä tietoa.

4.2 Käytetyt teknologiat

Yllä oleviin protokollavaatimuksiin valittiin seuraavat protokollat, mutta ne voidaan myös vaihtaa toisiin samat vaatimukset täyttäviin protokolliin, sillä DAGLAP ei vaadi tietyn protokollan käyttöä.

4.2.1 Symmetrinen salaus

Yleensä symmetrinen salain ei takaa tiedolle kuin vain luottamuksellisuuden, mutta myös aitous ja eheys ovat pakollisia ominaisuuksia DAGLAP:n yhteydessä käytettävälle symmetriselle salaukselle.

Pelkän luottamuksellisuuden tapauksessa AES olisi luonteva valinta. Se on tunnettu lohkosalain, joka esiteltiin vuonna 2001 [24]. Sen käyttö on hyvin laajaa, eikä sitä ole onnistuttu tähän asti murtamaan. Moni AES:n yleisistä lohkotiloista, kuten CBC tai CTR, eivät kuitenkaan suoraan kelpaa juuri sen takia, että ne eivät suoraan takaa aitoutta tai eheyttä. Niiden yhteydessä pitäisi käyttää esimerkiksi HMAC:ia tai muuta viestintodennuskoodia. Toinen vaihtoehto AES:n sijasta olisi käyttää jonosalainta kuten Salsa20 [4].

Tämän takia valittiin AES-GCM. GCM on lohkotila, joka vaatii lohkosalaimen, mutta joka takaa luottamuksellisuuden lisäksi myös aitouden ja eheyden [23]. GCM:n etuna on myös toteutettavuus alhaisillakin laitteistoresursseilla. Mahdollisena vaihtoehtona mainittakoon myös ChaCha20-Poly1305 [21].

Aiemmin käytetyssä symmetrisen salausprotokollan määritelmässä avain K_{AB} vastaa AES-avaimen lisäksi myös alustusvektoria (IV).

4.2.2 Allekirjoitus

Allekirjoitus- ja asymmetriseen salaukseen sopivat algoritmit perustuvat usein samoihin rakenteisiin. Tähän tarkoitukseen valitaan elliptisiä käyriä hyödyntävä algoritmi, tässä tapauksessa ECDSA [17].

4.2.3 Avaintenvaihto

Avaintenvaihto toteutetaan käytännössä Diffie–Hellman-avaintenvaihdolla tai siihen perustuvalla menetelmällä. Diffie–Hellman-avaintenvaihto voidaan toteuttaa kokonaislukujen päälle (kuin RSA) tai elliptisten käyrien päälle (jolloin sitä kutsutaan ECDH:ksi). Tässä tapauksessa käytetään jälkimmäistä. Todennus toteutetaan niin, että todentava osapuoli A allekirjoittaa oman sanomansa B:lle.

On tärkeää huomata, ettei kumpikaan näistä ole kvanttiturvallinen, koska kokonaislukujen tekijöihin jakaminen onnistuu kvanttitietokoneilla polynomisessa ajassa [35], ja tätä voi soveltaa myös elliptisiin käyriin. Pohdintaluvussa käsitellään tätä tarkemmin.

4.2.4 Hallussapitotodistus

Juelsin ja Kaliskin ehdottamalla protokollalla tiedoston koko kasvaa tietyllä prosentilla, mutta tiedostoon tehnyt muutokset voidaan havaita [18]. Menetelmä ei kuitenkaan tue rajatonta määrää haasteita [19]. Erwayn, Küpçün, Papamanthoun ja Tamassian ehdottama menetelmä on hyvä vaihtoehto, [9] samoin kuin Kaanichen, El Moustainen ja Laurentin ehdottaman menetelmä [19]. Kumpi tahansa niistä kelpaa toteutukseen.

4.2.5 TPM

Protokollaston kuvauksessa oletetaan, että sovelluspalvelimella on käytössä TPM (Trusted Platform Module). Kyseessä on usein laitteistotasolla toteutettava moduuli, joka toteuttaa erilaisia kryptografisia ominaisuuksia. Kuvattava protokollasto käyttää erästä TPM:n toiminnallisuutta, *sinetöintiä* (*sealing*), jolla tietoa voi salata niin, että se voidaan purkaa vain samoilla *PCR-parametreilla*, joilla tieto on myös sinetöity [37]. PCR-parametrit ovat taas yksinkertaisesti tiivisteitä, joille ainoa sallittu operaatio on tiivisteen *jatkaminen* (*extend*), eli uuden tiivisteen muodostaminen aiemmasta tiivisteestä ja uudesta sisällöstä. Esimerkkinä PCR-parametreista ovat PCR_0 – PCR_7, jotka alustetaan laitteistokohtaisin tiedoin niin, että muutokset laitteistoon muuttavat myös kyseisten parametrien alkuarvoa. TPM-vaatimusta käsitellään myöhemmin pohdintaluvussa.

4.3 DAGLAP-protokollasto

DAGLAP (Device-Assisted Geographic Location Assurance Protocol) on protokolla tai tarkemmin protokollasto. Sen avulla sovelluspalvelin voi todentaa oman ja tietopalvelimensa sijainnin hyödyntämällä palvelinkeskuksessa olevaa sijainnintodennuslaitetta.

Protokolla vaatii, että sovelluspalvelimelle on määritelty jokin enimmäiskierrosaika T_{max} (missä kierrosaika vastaa aikaa, joka kestää viestin kulkemisessa laitteelta toiselle ja takaisin). Tämän lisäksi on toinen aikaparametri, T_{seek}, joka on sallittu aikavälys tietopalvelimen tapauksessa (esimerkiksi tietojen hakemiseen massamuistista). Erikseen on määritettävä myös N_{to}, yhteysyritysten lukumäärä, ja N_c, luotavien haasteiden lukumäärä (tietopalvelimen sijaintia tarkistettaessa). Näiden määritelmiä käsitellään tarkemmin parametrialaluvussa.

4.3.1 Alustus

Kun tiedot tallennetaan ensimmäistä kertaa tietopalvelimelle, suoritetaan alustus. Alustuksen osana sovelluspalvelin, joka ohjaa tietojen siirtämistä tietopalvelimelle, toimii seuraavasti:

1. Luo tietojen salaukseen jollakin symmetrisellä salausalgoritmilla (jota ei tässä tarkemmin määritellä) käytettävä avain K_{data}. Avain on luotava turvallista satunnaislukugeneraattoria käyttäen.

2. Pyydä lähimmältä sijainnintodennuslaitteelta sijaintitietoja (tarkemmat yksityiskohdat alaluvussa 4.3.2).

3. Muodosta yllä saaduista sijainnintodennuslaitteen sijaintitiedoista jollakin tavalla yksi tai useampi TPM:n PCR-parametri (kuten esimerkiksi PCR_{15}, joka on yleensä aluksi vapaana).

4. Sinetöi (TPM:llä) ja tallenna K_{data}. Sinetöinnin yhteydessä on käytettävä (ainakin) edellisessä vaiheessa luotuja PCR-parametreja, joita ilman avainta ei voi siis purkaa.

5. Salaa kaikki tiedot avaimella K_{data} ennen siirtämistä tietopalvelimelle.

6. Luo salatusta tiedosta hallussapitotodistusprotokollan avulla aineisto, jonka perusteella tietopalvelinta voidaan myöhemmin haastaa todistamaan salatun tiedon hallussapito. Aineiston muoto ja sisältö riippuu valitusta protokollasta.

4.3.2 Sovelluspalvelimen sijainnin tarkistaminen

Sovelluspalvelin A haluaa tarkistaa oman sijaintinsa. Se toimii seuraavasti:

1. Avaa yhteys sijainnintodennuslaitteeseen L.
2. Suorita todennettu avaintenvaihto, ja varmista, että L:n lähettämä sanoma on allekirjoitettu oikein. Allekirjoituksen aitous todetaan allekirjoitusjärjestelmän päälle rakennetulla julkisten avainten hallintajärjestelmällä. Avaintenvaihdon yhteydessä lähetettävä julkinen allekirjoitusavain on allekirjoitettu jonkin luotettavan tahon yksityisellä avaimella. Kyseisen tahon luotettavuus perustuu ulkoisesti saatavilla olevaan luotettavaan tietoon. Jos avaintenvaihto onnistuu, kaikki sanomat turvataan tästedes saadulla avaimella K_{AL}.
3. Lähetä L:lle pyyntö, jossa pyydetään sijaintitietoa. Sijainnintodennuslaitteen sijaintitietojen muotoa ei tarkemmin määritellä, mutta kyseessä voi olla esimerkiksi lukumuistissa sijaitseva avainarvotietokanta. Käynnistä ajastin viestiä lähetettäessä.
4. Vastaanota L:n vastaus, jossa on sijaintitiedot. Pysäytä ajastin, jonka mittaama aika on A:n ja L:n välinen kierrosaika. Jos kierrosaika $> T_{max}$, aikakatkaisu.
5. Muodosta L:n antamista tiedoista TPM:n PCR-parametrit samalla tavalla kuin alustuksessakin niin, että parametrit ovat samat jos ja vain jos olennaiset sijaintitiedot eivät ole muuttuneet.
6. Jos parametrit ovat samat, avaa K_{data} TPM:stä ja pura tiedot sillä.

Jos kierrosaika on liian suuri, tapahtuu aikakatkaisu. Sovelluspalvelin voi silloin yrittää uudelleen, mutta yhteys pitää sulkea ja koko prosessi aloittaa alusta, mahdollisesti ensimmäistä vaihetta lukuun ottamatta. Uusi istuntoavain on luotava. Jos yritysten määrä ylittää tietyn rajan (eli N_{to} yritystä ylitetty), voidaan todeta, että sijainnintodennuslaite on liian kaukana, joten sijaintia ei voi luotettavasti varmistaa.

Avaintenvaihdon tuloksena saatua symmetristä avainta käytetään yhteyden turvaamiseen. Vaihtoehtoisesti symmetriseen salaukseen voi myös käyttää eri avaimia eri suuntiin. Tällöin avaintenvaihto suoritetaan esimerkiksi samaan aikaan kahdesti.

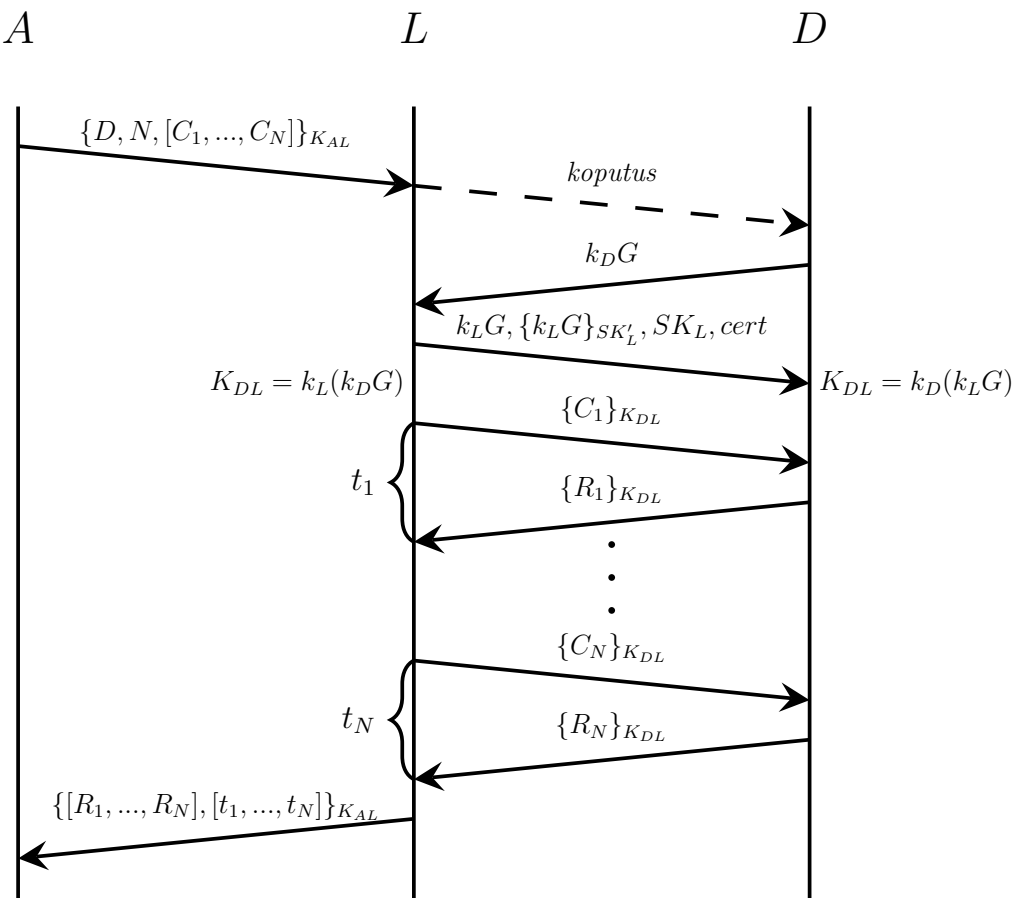
4.3.3 Tietopalvelimen sijainnin tarkistaminen

Sovelluspalvelin A haluaa tarkistaa tietopalvelunsa D sijainnin. A tietää palvelimen D, jolle sen tiedot on tallennettu. Jos A ja D ovat sama laite, tarkistusta ei tarvitse tehdä, muuten se toimii seuraavasti (ks. myös kuva 4.1):

1. A muodostaa uuden yhteyden sijainnintodennuslaitteeseen L, tai käyttää sovelluspalvelimen sijainnin tarkastamiseen aiemmin avattua yhteyttä.
2. A luo alustuksen vaiheessa 6 luodun aineiston perusteella N_c satunnaista hallussapitotodistusprotokollan mukaista haastetta $\{C_i \mid i \in [1..N_c]\}$, joihin D:n on vastattava.
3. $A \rightarrow L : \{D, N_c, [C_1, ..., C_{N_c}]\}_{K_{AL}}$
4. L koputtaa D:lle, eli lähettää sanoman, jolla pyytää avaamaan yhteyden.
5. D avaa yhteyden samaan tapaan kuin A alaluvussa 4.3.2. Istuntoavain on K_{DL}.
6. Jokaiselle haasteelle $\{C_i \mid i \in [1..N_c]\}$:
   (a) $L \rightarrow D : \{C_i\}_{K_{DL}}$. L aloittaa ajan mittauksen.
   (b) $D \rightarrow L : \{R_i\}_{K_{DL}}$ (R_i = vastaus haasteeseen C_i hallussapitotodistusprotokollan mukaisesti). L lopettaa ajan mittauksen ja laskee kierrosajan t_i.
7. $L \rightarrow A : \{[R_1, ..., R_{N_c}], [t_1, ..., t_{N_c}]\}_{K_{AL}}$
8. A varmistaa, että kaikki vastaukset R_i ovat oikein haasteille C_i.
9. A tarkistaa kierrosajat t_i. Oletus on, että kaikki haasteet ja vastaukset ovat tulleet samasta koneesta. Suurinta aikaa verrataan rajaan $T_{max} + T_{seek}$. Jos aika on pienempi tai yhtä suuri, A hyväksyy tarkistuksen tulokset.

4.4 Ratkaisu

Palvelinkeskukseen asennetaan pysyvästi sijainnintodennuslaite, ja kolmannen osapuolen tarkastaja tarkastaa sekä laitteen asennuksen että käyttöönoton. Laitteelle kirjoitetaan lukumuistiin sijaintitiedot, joita ei voi enää kirjoittamisen jälkeen muuttaa. Laite asennetaan palvelinkeskukseen niin, ettei siihen voi fyysisesti kajota. Kajoamisyritys johtaa laitteen tuhoutumiseen, vaurioitumiseen tai hälytyksen laukeamiseen. Tarkastuksena tehtävänä on varmistaa, että laite on aito sijainnintodennuslaite, asennettu pysyvästi keskukseen ja että



Kuva 4.1: Havainnollistava kaavio tietopalvelimen sijainnin tarkistamisprosessista. A on sovelluspalvelin, L on sijainnintodennuslaite ja D on tietopalvelin.

sille syötetyt sijaintitiedot eivät ole väärennettyjä.

Kun asiakas ensimmäistä kertaa vuokraa palvelintilaa ja haluaa tallentaa tietonsa palvelinkeskukseen, hänen sovelluspalvelimensa, jonne hänen pilvisovelluksensa asennetaan, suorittaa DAGLAP-protokollan mukaisen alustusvaiheen. Tiedoille luodaan näin salausavain, jolla tiedot salataan ennen niiden siirtämistä ja pysyvää tallentamista tietopalvelimelle. Alustuksen yhteydessä voidaan myös arvioida verkon kapasiteetti ja kierrosajat sijainnintodennuslaitteen ja sovelluspalvelimen välillä, mutta asiakkaan tulisi määritellä jokin enimmäisraja, koska muuten palvelinkeskuksen omistaja voi jo alusta alkaen välittää sijaintipyynnön kaukana sijaitsevalle sijainnintodennuslaitteelle.

Käyttöönoton jälkeen sovelluspalvelin pystyy nyt suorittamaan sovellustansa. Palvelimen uudelleenkäynnistyksen yhteydessä sovelluspalvelimen on tarkistettava sekä oma sijaintinsa että tietopalvelimensa sijainti DAGLAP:n mukaisesti. Jos sijaintitieto ei täsmää tai

luotettavaa sijaintitietoa ei kierrosajan ylittymisen takia ole saatavilla, sovelluspalvelimen on lopetettava toimintansa kunnes sijaintitarkistus taas onnistuu.

Palvelinkeskuksen omistajan ei pidä pystyä siirtämään palvelinta käynnistämisen jälkeen. Tämän takia on välttämätöntä, että sovelluspalvelin tarkistaa sijaintinsa uudelleen satunnaisin väliajoin. Uudelleentarkistus voi tapahtua esimerkiksi kerran päivässä satunnaiseen aikaan. Tarkistettava on sekä sovellus- että tietopalvelimen sijainti. Jos jompikumpi tieto ei täsmää, voidaan sallia tietty äärellinen määrä kertoja yrittää uudelleen, mutta lopulta sovelluspalvelimen on jälleen lopetettava toimintansa, koska tietojen sijaintiturvallisuutta ei voida taata. Käytännön toteutukseen tietojen salausavaimen käsittelyn suhteen ei tässä paneuduta.

4.5 Parametrit

Yhteysyritysten määrän N_{to} tulisi olla melko pieni luku, esimerkiksi 5, jolloin yritys yritetään muodostaa korkeintaan viisi kertaa. Jos aikakatkaisu tapahtuu kaikilla viidellä kerralla, yhteyden muodostus epäonnistuu. Liian suuren luvun käyttäminen hidastaa epäonnistunutta sijaintitarkistusta ja voi mahdollisesti kasvattaa teoreettista todennäköisyyttä, että hyökkääjä voi väärentää sijainnintodennuslaitteen vain osalle pyynnöistä, mutta toisaalta liian pieni arvo voi johtaa satunnaisiin epäonnistumisiin sijaintia tarkistaessa.

Haasteiden lukumäärä N_c määräytyy käytetyn hallussapitotodistusprotokollan perusteella. Erwayn ym. todistusprotokollassa todennäköisyys, että muutettu tietolohko havaitaan on $1 - (1 - f)^{N_c}$, missä f on muutettujen lohkojen osuus tiedostosta [9]. Jos esimerkiksi oletetaan, että tietopalvelin, jolla ei oikeasti ole kyseistä tietoa, saa oikein vain 0,1 % lohkoista ($f = \frac{999}{1000}$, mikä jo erittäin epätodennäköistä keskikokoisillakin lohkoko'oilla, koska palvelimen tulisi arvata oikein kaikki lohkon bitit), silloin esimerkiksi neljällä haasteella todennäköisyys oikeiden vastausten saamiseen on yksi biljoonasta (10^{-12}). Jos halutaan todennäköisyyden olevan alle 10^{-50} tässä tapauksessa, haasteiden lukumäärän tulisi olla vähintään 17. Yhden haasteen vaatima kaista on $O(l)$, missä l on lohkon koko [19]. Täten koko protokollan kaistavaativuus on $O(nl)$, missä n on haasteiden lukumäärä ja l on lohkon koko. Haasteiden määrä voitaisiin valita satunnaisesti joka kerta ennen niiden lähettämistä niin, että määrä on vähintään 17, kun taas ylärajaksi asetetaan jokin käytännöllinen arvo.

Kaanichen ym. menetelmässä yksittäinen haaste vaatii vain vakiomäärän kaistaa [19]. Näin sen kaistavaativuus yhteensä on $O(n)$, kun lähetetään n haastetta. Todennäköisyys saada oikea vastaus sattumanvaraisesti lohkokoolla l (bittiä) on 2^{-l}. Suuremmat lohkokoot hidastavat kuitenkin laskutoimituksia merkittävästi jokseenkin lineaarisella suhteella. Jos pyritään samaan todennäköisyyteen eli 1^{-50} ja lohkokooksi valitaan esimerkiksi 4096 bittiä, jo yksi tarkistus riittää, mutta useammastakin tarkistuksesta voi olla hyötyä muun kuin vain yhden lohkon tarkistamiseen. N_c voitaisiin kenties valita satunnaisesti lukuväliltä $[2, 8]$.

Aikaparametreja DAGLAP-protokollastossa on kaksi: T_{max}, enimmäiskierrosaika sovelluspalvelimen ja sijainnintodennuslaitteen välillä, ja T_{seek}, sallittu välys tietopalvelimen osalta niin, että enimmäiskierrosaika tietopalvelimen ja sijainnintodennuslaitteen välillä on $T_{max} + T_{seek}$. Ajanjakson T_{max} voi tulkita vastaavan enimmäisetäisyyttä d_{max} niin, että $d_{max} = T_{max} * c$, missä c on valonnopeus.

Käytännössä kuitenkaan lähiverkossakaan yksinkertainen kaikuviestikään ei kulje valonnopeudella. Verkkoviipeet jaetaan usein neljään luokkaan [33]: lähetysviipeeseen (kesto lähettää viesti väliaineeseen bitti kerrallaan), etenemisviipeeseen (kuinka kauan kestää, että ensimmäinen bitti kulkee siirtovälineellä lähettäjältä vastaanottajalle), käsittelyviipeeseen (kuinka kauan kestää käsitellä paketti verkkoprotokollissa) ja jonotusviipeeseen (kuinka kauan paketti joutuu odottamaan lähettämistä tai käsittelyä). Sovelluspalvelin ei voi tietää palvelinverkon topologiaa (palvelinkeskuksen omistaja voi helposti piilottaa kyseiset tiedot), ja välissä olevat lähipalvelimet ja reitittimet voivat vaikuttaa huomattavasti kierrosaikaan. Kierrosaikojen ja viipeiden suhteita otetaan myöhemmin esille pohdintaluvussa.

T_{seek} määräytyy tietopalvelimen massamuistin hakuajan perusteella. Järjestely olettaa, että sovellus- ja tietopalvelimen välillä on jotakuinkin samansuuruinen verkkoviive sijainnintodennuslaitteeseen, ja että lisäaika on suunniteltu pelkästään sitä varten, että tietopalvelin ehtii hakemaan tietonsa massamuistista ja laskemaan vastauksen esitettyihin haasteisiin.

Aikarajoille T_{max} ja T_{seek} ei tässä esitetä tarkkoja määritelmiä, vaan niiden arvot määräytyvät tilanteen mukaan. Niiden määritelmät vaikuttavat ainoastaan sovelluspalvelimen puolella, joten sovelluspalvelin voi tarvittaessa muuttaa niiden arvoja.

4.6 Prototyyppi

Kierrosaikojen ja protokollaston suorituskykyn arvioimiseksi (sijainnintodennuslaitteen puolelta) siitä luotiin myös prototyyppi. Se simuloi ainoastaan sovelluspalvelimen ja sijainnintodennuslaitteen välistä kanssakäyntiä, eikä sen osana toteutettu tietopalvelimen sijainnin tarkistusta tai hallussapitotodistusprotokollaa.

Prototyypissä yhteydet käyttävät UDP-protokollaa ja DTLS-salausprotokollastoa, ja siinä on toteutettu alaluvun 4.3.2 mukainen sovelluspalvelimen sijainnin tarkistaminen. Toteutus on tehty C:llä Unix-pohjaisille käyttöjärjestelmille ja sen kryptografiset osat käyttävät OpenSSL-kirjastoa.

Sekä TLS:ssä että DTLS:ssä avaintenvaihto tapahtuu usein juuri todennetulla Diffie–Hellman-avaintenvaihdolla, kuten DAGLAP:ssakin on esitetty. Toteutuksessa DTLS:lle valittiin versio 1.2 sekä kättelyyn ECDH:ta ja salaukseen AES-GCM:ää käyttävä `TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384`.

Prototyypissä sijaintitietojen muoto on kokonaislukutaulukossa, eli sijainnintodennuslaitteella on neljä kokonaislukua, joista sovelluspalvelin voi pyytää mitä tahansa niistä kokonaisluvulla $[0, 3]$ (≥ 4 antaa luvun 0).

5 Arviointi

Tässä luvussa arvioidaan ylempänä esitelty DAGLAP-protokollasto ja sitä käyttävä ratkaisu. Arviointi tapahtuu eri menetelmillä, pääosin staattisella analyysillä sekä skenaarioilla. Prototyypille suoritetaan kokeita, joilla tarkistetaan prototyyppitoteutuksen suorituskyky. Staattinen analyysi, skenaariot ja valvotut kokeet ovat arviointimenetelmiä, jotka mainitaan Hevnerin ym. artikkelissa [13]. Peffersin ym. vuonna 2012 julkaistun artikkelin jaottelun mukaisesti tässä työssä käytettyjä arviointimenetelmiä on kolme [29]: looginen argumentti, prototyyppi ja kuvaava skenaario.

5.1 Turvallisuus

Tässä alaluvussa tutkitaan protokollastoa ja sitä käyttävää ratkaisua aiemmin esitellyn tietoturvamallin näkökulmasta. Tarkoituksena on osoittaa, että protokollasto täyttää vaatimuksen 1 ja kaikki sen viidestä osavaatimuksesta.

5.1.1 Yhteyden turvallisuus

Vaatimukset 1.a ja 1.b liittyvät palvelimen ja sijainnintodennuslaitteen välisen yhteyden turvallisuuteen. Yhteyden alustus on kuvattu DAGLAP-protokollassa seuraavasti:

1. Yhteyden avaa aina sovelluspalvelin.
2. Sovelluspalvelin ja sijainnintodennuslaite suorittavat avaintenvaihdon, jolloin saadaan yksi avain tai kaksi avainta (yksi kuhunkin suuntaan).
3. Sovelluspalvelin varmistaa, että sijainnintodennuslaitteen sanomat avaintenvaihdon yhteydessä ovat allekirjoitettuja oikealla avaimella.

Molempiin osapuoliin on luottamus: sovelluspalvelimeen on luottamus perustuen oletukseen, että siinä on luotettu ajoympäristö, ja sijainnintodennuslaitteen sisäiseen toimintaan on täysi luotettavuus. Avaintenvaihdon avulla saadaan luotua symmetrinen avain (tai kaksi avainta). Vaikka edes toinen osapuoli pyrkisi luomaan niin sanotun *heikon avaimen* valitussa symmetrisessä salauksessa, sijainnintodennuslaitteen satunnaisuus tekee tästä käytännössä mahdotonta, sillä avaintenvaihdossa tuloksena saatu avain riippuu molempien osapuolien

tekemistä valinnoista. AES-salauksessa ei ole heikkoja avaimia, ja GCM:ssä todennäköisyys heikon avaimen luominen satunnaisesti on mitättömän pieni [34].

Oletetaan nyt, että hyökkääjänä on täysin palveluntarjoajan tai jopa ulkoisen hyökkääjän hallinnoima välipalvelin, joka pystyy lukemaan ja muokkaamaan kaikkea sovelluspalvelimen ja sijainnintodennuslaitteen välillä. Varmenteen tarkistettuaan sovelluspalvelin voi olla varma sijainnintodennuslaitteen aitoudesta. Diffie–Hellman-avaintenvaihdon yksi pääominaisuuksista on se, ettei välissä oleva hyökkääjä pysty päättelemään salausavainta, vaikka hän lukisi kaikki viestit. Avaintenvaihto on todennettu ainakin sijainnintodennuslaitteelta sovelluspalvelimelle, joten hyökkääjä ei voi muokata viestejä siihen suuntaan.

Hyökkääjä pystyisi kuitenkin muuttamaan viestejä toiseen suuntaan. Tällöin kuitenkin avaintenvaihdon tuloksena saadut avaimet eivät yksinkertaisesti täsmää eikä yhteys voi toimia. Näin aktiivinen hyökkääjä voi saada aikaan vain palvelunestohyökkäyksen eikä murtamaan yhteyden turvallisuutta.

Sama pätee myös vaatimukseen 1.b. Vaikka tietopalvelin yrittäisi saada aikaan heikon avaimen, avaintenvaihdossa on ydinominaisuutena se, että kumpikin osapuoli vaikuttaa osaltaan lopputulokseen. Jos sijainnintodennuslaite toimii oikein, niin kuin sopii olettaa, saman avaimen toistuminen usealle eri istunnolle on käytännössä mahdotonta, samoin kuin heikon avaimen saaminen (koska todennäköisyys on häviävän pieni). Tietopalvelin ei voi siis yksin tehdä yhteydestä sijainnintodennuslaitteeseen heikkoa.

Näin vaatimukset 1.a ja 1.b ovat täytetty, sillä yhteys on todettu turvalliseksi sekä sovellus- että tietopalvelimelta sijainnintodennuslaitteeseen.

5.1.2 Sijainnintodennuslaitteen aitous

Vaatimus 1.c koskee sijainnintodennuslaitteen aitoutta, eli sitä, että sovelluspalvelimen on tosiaan varmistuttava siitä, että se keskustelee oikean sijainnintodennuslaitteen kanssa.

DAGLAP-protokollastossa esitetään, että sovelluspalvelin selvittää lähimmän sijainnintodennuslaitteen silloin, kun sijainti halutaan tarkistaa. Tämä voidaan tarkemmin toteuttaa parillakin eri tavalla. Sovelluspalvelimeen voidaan esimerkiksi käyttöönottovaiheessa määritellä pysyvästi se sijainnintodennuslaite, joka on sillä hetkellä lähimpänä. Tähän ja

toiseen ratkaisuun, jossa lähin sijainnintodennuslaite pitää selvittää, liittyy kuitenkin sama olennainen ongelma: sovelluspalvelimen on nimenomaan löydettävä aito eikä esimerkiksi palveluntarjoajan hallinnoima virtuaalinen sijainnintodennuslaite, joka välittää väärää tietoa, tai toisesta sijainnista tunnelilla yhdistetty sijainnintodennuslaite.

Sijainnintodennuslaitteen yksilöivän tiedon (kuten IP-osoitteen) lisäksi laitteella on myös avainpari, jonka yksityisellä avaimella allekirjoitetaan avaintenvaihdon sanomat. Avainparin julkinen avain allekirjoitetaan jollakin luotetulla avainparilla niin, että sovelluspalvelin voi tarkistaa avaintenvaihdossa käytetyn allekirjoituksen aitouden ja luotettavuuden. Tätä varten on otettava käyttöön jonkinlainen julkisten avainten hallintajärjestelmä (PKI), jossa luotettu taho – esimerkiksi se kolmannen osapuolen tarkastaja joka tarkasti sijainnintodennuslaitteen käyttöönoton – allekirjoittaa myös laitteen julkisen avaimen. Vastaavanlaista varmennejärjestelmää käytetään myös internetin yhteydessä HTTPS-liikenteen turvaamiseen.

Tietoturvamallissa on mainittu, että sovelluspalvelimen hallinnoijalla voi olla jokin ulkoinen luotettu kanava, jolla sijainnintodennuslaitteeseen liittyvää tietoa voidaan saada. Laitteen yksilöllinen julkinen avain ja sen varmenneketju voidaan saada myös esimerkiksi tätä kautta, jolloin palveluntarjoajalla ei ole minkäänlaista mahdollisuutta vaikuttaa avaimen hankintaprosessiin.

5.1.3 Toistohyökkäys

Vaatimus 1.d koskee tapausta, jossa sijainnintodennuslaitteeksi tekeytyvä osapuoli toistaa aiemmin tältä tallennettuja sanomia ja toistaa niitä takaisin sovelluspalvelimelle, joka pyrkii todentamaan sijaintiaan. Jos tämä on mahdollista, palveluntarjoaja voi helposti määritellä välipalvelimen toimimaan kyseisellä tavalla ja siten väärentämään sovelluspalvelimen saamaa sijaintitietoa.

Toistohyökkäys ei ole mahdollinen, koska sovelluspalvelimen ja sijainnintodennuslaitteen väliselle yhteydelle luodaan aina uusi satunnainen salausavain. Jos välipalvelin tallentaa ja myöhemmin toistaa liikennettä, tallennettu salattu viestintä ei ole pätevää AES-GCM-tietovirtaa uudella avaimella, ja toistetut sanomat hylätään.

5.1.4 Läheisyys

Sovellus- ja tietopalvelimien läheisyys sijainnintodennuslaitteeseen pyritään todentamaan mittaamalla kierrosaikaa. Kierrosajan käyttäminen vastaa eräällä tavalla etäisyydenrajausprotokollaa. Sovelluspalvelimen ja sijainnintodennuslaitteen väliseksi kierrosajaksi oletetaan aika, joka on korkeintaan T_{max}. Liian pitkä kierrosaika johtaa aikakatkaisuun. Palveluntarjoaja ei voi keinotekoisesti lyhentää kierrosaikaa, koska sovelluspalvelin tietää varmasti sijainnintodennuslaitteen vastanneen (ks. luku 5.1.2) eikä sen viestejä voi toistaa (ks. luku 5.1.3). Palveluntarjoaja voi ainoastaan keinotekoisesti pidentää kierrosaikaa, joka taas voi johtaa aikakatkaisuun.

Jos palveluntarjoaja yrittää esimerkiksi siirtää sovelluksen ja sen tiedot toiseen palvelinkeskukseen mutta yhä ohjata pyynnöt alkuperäisessä palvelinkeskuksessa sijaitsevalla tietolaitteelle, verkkoviive kasvaa niin suureksi ettei vastauksen saapuminen tarpeeksi ajoissa ole mitenkään mahdollista.

Tietopalvelimelle sallitaan myös välys T_{seek}, joka kuvaa sitä ajanjaksoa, jonka aikana tietopalvelin hakee sovelluksen tiedot massamuistista ja laskee vastauksen haasteeseen.

Aikarajoihin luottaminen voi kuitenkaan johtaa hyökkäyksiin. Palveluntarjoaja voi esimerkiksi pyrkiä siirtämään sovellus- ja tietopalvelimen muualle, ja siirron yhteydessä tarjota paljon parempaa laitteistoa, jolla viipeet ovat alhaisemmat kuin aiemmalla laitteistolla. Toinen mahdollisuus on, että palveluntarjoaja voi jo alusta alkaen kasvattaa viipeitä ja näin saada myös sovelluksen nostamaan aikarajoja.

Parametrien arvojen määrittely vaatii lisätutkimuksia. Esimerkiksi viipeitä palvelinkeskuksen lähiverkon sisällä tulisi mitata, samoin kuin massamuistin hakuaikoja ja laskennan yhteydessä myös suorittimen laskenta-aikaa.

5.2 Muut vaatimukset

Vaatimus 1 on nyt arvioitu edellisessä alaluvussa, joten tässä alaluvussa käsitellään, missä määrin ratkaisu täyttää vaatimukset 2–5.

5.2.1 Vaatimus 2

Kuten vaatimusluvussa mainittiin, vaatimus 2 liittyy siihen, että palveluntarjoaja voi siirtää sovelluksen eri sijaintiin sen käynnistymisen jälkeen. Tämän takia sijainti on tarkistettava uudelleen sovellusta ajettaessakin. Käynnistyksen yhteydessä tarkistus tulee suorittaa, koska se on ainoa tapa saada tiedon salausavain, jota ilman sovellus ei pääse tietoihinsa.

Ratkaisussa mainitaan, että sijainnin tarkistus suoritetaan uudelleen satunnaisin väliajoin, esimerkiksi kerran päivässä. Tämä on suurimpaan osaan tapauksista riittävän usein, mutta esimerkiksi virtuaalikoneiden tapauksessa sijainti voi muuttua hyvinkin nopeasti. Sijainnin tarkistus vaatii kuitenkin jonkin verran laskenta-aikaa ja siksi sen suorittaminen liian usein on myöskin vahingollista.

Otetaan esimerkiksi skenaario, jossa palveluntarjoaja pyörittää sovellusta juuri virtualisoimalla, eli sovelluspalvelin on oikeasti virtuaalipalvelin jollakin palvelinkeskuksessa sijaitsevalla virtuaalikoneiden isäntälaitteella. Kyseinen järjestely on hyvin yleinen, sillä monet pilvipalvelut vuokraavat asiakkailleen ainoastaan virtuaalikoneaikaa eikä varsinaista fyysistä palvelinkonetta. Palveluntarjoaja voi esimerkiksi kehittää keinon, jolla virtuaalikoneesta voi luoda täydellisen järjestelmäkuvan, siirtää kuvan eri fyysiseen koneeseen ja jatkaa sovelluksen suorittamista näin ikään kuin samalla virtuaalikoneella eri sijainnissa.

Jotta virtuaalikoneen siirtäminen näin ei ole mahdollista, sijaintitarkistuksen ei tule tapahtua säännöllisin väliajoin, vaan satunnaisemmin. Vaihtoehtoisesti sovellus voisi arpoa satunnaisen ajanjakson, jonka ajaksi sijaintitietoa pidetään ajantasaisena. Satunnaisesti tapahtuvilla sijainnintarkistuksilla voidaan estää palveluntarjoajan yritykset väärentää sijaintitieto, koska tarjoaja ei voi etukäteen tietää seuraavan sijainnintarkistuksen ajankohtaa. Kun tarkistus on jo alkanut, palvelimen siirron voi olettaa olevan niin hidas, että sen yritys aiheuttaisi aikakatkaisun.

Sijainnintarkistuksen ajankohdan määrittäminen ei liity suoranaisesti DAGLAP-protokollastoon, vaan olennaisesti sen toteutukseen. Pohdintaluvussa käsitellään tarkemmin sijainnintarkistuksen ajoittamista.

5.2.2 Vaatimus 3

Vaatimuksessa 3 on oikeastaan kaksi osaa, joita on paras tutkia erikseen.

Ensimmäinen niistä on, että sovellus- tai tietopalvelimen siirto tulee huomata, ja ratkaisun tulee estää sovellusta toimimasta, jos sovellus- tai tietopalvelin ei ole enää hyväksyttävässä sijainnissa. Vaatimus 2 liittyy tähän olennaisesti. Jos sijainti tarkistetaan toistuvasti uudelleen, sijainnin muutos pitäisi olla helppo huomata. Siirto voi tapahtua kolmella eri tavalla:

1. Sovellus siirretään toiselle palvelimelle jollakin tavalla niin, että se jatkaa toimintaansa käynnistymättä uudelleen, ja sen lähin sijainnintodennuslaite muuttuu. Tällöin sijaintitiedot myös muuttuvat ja tiedon salausavainta ei onnistuta enää purkamaan, koska TPM:n sinetöinti sallii avaimen purkamisen vain, jos PCR-arvot ovat samat kuin sinetöidessäkin.
2. Sovellus siirretään toiselle palvelimelle, mutta sen sijaintipyyntö ohjataan alkuperäiselle laitteelle. Tätä tapausta on käsitelty aiemmin alaluvussa 5.1: kierrosaika ylittää rajan ja tapahtuu aikakatkaisu.
3. Vain tietopalvelin siirretään, mutta sovelluspalvelin pysyy siellä missä ennenkin. Sovelluspalvelin huomaa tietopalvelimen ja sijainnintodennuslaitteen välisen viipeen kasvaneen ja ylittävän rajan.

Näin ollen voidaan todeta, että sovellus- tai tietopalvelimen siirto on huomattavissa.

Toinen vaatimuksen osa taas käsittelee tilannetta, jossa asiakas on suostunut siihen, että sovellus- ja/tai tietopalvelin siirretään toiseen sijaintiin. Tällaisessa tapauksessa kyseessä on käytännön ongelma, jossa sovellus pitää pystyä määrittelemään uudelleen niin, että sijainnin vaihto onnistuu.

Sovelluksen uudelleenmäärittely vaatii siirron yhteydessä kaksi muutosta: ensiksi, kierrosaika on mitattava uudelleen niin, että kierrosaikaraja on sopiva uudessa sijainnissa. Kierrosajan mittausta käsitellään tarkemmin pohdintaluvussa. Toiseksi tiedon salausavain on sinetöitävä uudelleen uusien sijaintitietojen mukaisesti. Aiemmin on oletettu, että sovellus pystyy suorittamaan DAGLAP:iin liittyvän koodin luotetussa ajoympäristössä. Samassa ympäristössä asiakas voi purkaa aiemmin sinetöidyn salausavaimen ja sinetöidä sen niin kuin alustusvaiheessa uudelleen uusista sijaintitiedoista johdetuilla PCR-parametreilla.

Asiakkaalla on siis oltava aktiivinen rooli palvelimen siirrossa. Järjestelmä on määritettävä uudelleen samalla tavalla kuin se on määritelty alustuksessakin, mutta sinetöidyn salausavaimen avulla tiedot voidaan purkaa ja salata uudelleen uudella avaimella. Näin ollen asiakkaan suostumuksella tietojen siirtäminen onnistuu. Järjestelmänvalvojan ei tule voida tekeytyä asiakkaaksi. Tämän estämiseksi on käytettävä samaa teknistä ratkaisua kuin alustuksessakin, eikä tässä työssä paneuduta ratkaisun teknisiin yksityiskohtiin.

5.2.3 Vaatimus 4

Vaatimus 4 kuuluu, että tiedon päätyminen muihin sijainteihin selkomuodossa on estettävä. Kaikki tietopalvelimelle tallennettava tieto salataan salausavaimella, joka luodaan tiedon tallennuksen yhteydessä. Näin tietopalvelin ei saa tietää selkokielistä tietoa lainkaan, sillä tieto kulkee selkokielisenä ainoastaan sovelluspalvelimen sisällä. Tiedot eivät paljastu, vaikka palveluntarjoaja esimerkiksi loisi täyden kopion tietopalvelimesta eri sijainteihin. Näin tiedon vuotaminen muihin sijainteihin selkomuodossa estyy.

Itse avaimen käsittelyssä on tärkeää, että sovelluspalvelin ei mitenkään pysty saamaan tiedon salausavainta, jos sijaintitiedot eivät täsmää. Toteutus käyttää tähän TPM:n sinetöintitoimintoa, joka estää salausavaimen purkamisen, jos PCR-parametrien arvot eivät täsmää alustuksessa määriteltyihin. Jos salausavain vuotaa, tiedon voi sen avulla purkaa. TPM:n käyttö takaa, ettei edes palveluntarjoaja saa kyseistä avainta selville ulkoisesti.

Selkomuotoisen tiedon ja sen salausavaimen turvallinen käsittely sovelluspalvelimella asettaa sovellukselle ja sen ajoympäristölle turvallisuusvaatimuksia. Avainta ei tule esimerkiksi säilyttää välimuistissa, johon järjestelmänvalvojalla on pääsy. Jos välimuistia käytetään, avaimen käyttöikää on rajattava vaatimuksen 2 täyttämiseksi. Tarkemmat yksityiskohdat sovelluksen sisäisestä turvallisuudesta on kuitenkin rajattu tämän työn ulkopuolelle.

5.2.4 Vaatimus 5

Vaatimus 5 liittyy protokollaston suorituskykyyn ja laitteistovaatimuksiin. Lähes kaikki protokollassa laskentaan liittyvät toimet ovat yhteydessä kryptografisiin ominaisuuksiin. Sijainnintodennuslaite ei tarkista varmenteensa luotettavuutta, joten sen pitää vain onnistua hoitamaan salaus sekä symmetrisellä että asymmetrisellä protokollalla. Symmetriseksi protokollaksi valittu AES-GCM on hyvä vaihtoehto, sillä se pystyy salaamaan tietoa nopeasti vähäisilläkin laitteistoresursseilla. Asymmetrisen protokollan osalta ECC:n on tutkimuksissa

osoitettu olevan RSA:ta keskimäärin suorituskykyisempi [36], samoin kuin ECDSA- eli elliptisiin käyriin perustuvien allekirjoitustenkin [8].

Tietopalvelimen pitää pystyä vastaamaan sovelluksen valitsemiin haasteisiin. Tämä vaatii käytännössä jonkin verran laskentatehoa, joka riippuu myös lähetettävien haasteiden lukumäärästä. Jos hallussapitotodistuksen protokollana käytetään esimerkiksi Kaanichen ym. menetelmää, laskentakustannukset ovat alhaiset [19].

Protokollasto vaatii myös sekä sijainnintodennuslaitteelta että sovelluspalvelimelta luotettavan ja tarkan monotonisen kellon. Tämä on käytännössä helposti toteutettavissa, sillä monessa käyttöjärjestelmässä tulee monotoninen kello mukana.

Sijainnintodennuslaite voi käyttää tietojensa hakemiseen lukumuistista erilaisia tietorakenteita. Kyseessä voi olla, kuten aiemmin mainittiin, avainarvotietokanta, jossa avaimet ja arvot ovat merkkijonoja, mutta toisaalta myös yksinkertainen taulukko lukuja voi riittää, esimerkiksi jos sijaintitiedot ovat alkeellisia (esimerkiksi pelkkä maa).

5.3 Vertailu

Tässä alaluvussa verrataan ratkaisua, joka käyttää sijainnintodennuslaitteita ja DAGLAP-protokollastoa, aiemmin esitettyihin menetelmiin, joita käsiteltiin alaluvussa 2.3. Päävertailukohteita ovat GeoProof ja TGADP, joiden todettiin olevan käsitellyistä ratkaisuista kaikkein lupaavimmat.

Vertailemme ensin GeoProofia ja DAGLAP:ia. Kumpikin niistä käyttää viipeitä etäisyyksien tarkistamiseen [2]. Siten kenties suurin ero niiden välillä on sijaintitiedon lähde. GeoProof nojaa satelliittitietoon, jonka häirintä tai väärentäminen on mahdollista, kuten on aiemmin mainittu. DAGLAP sen sijaan luottaa palvelinkeskukseen pysyvästi asennettuun sijainnintodennuslaitteeseen, jonka sijainnin voi olettaa olevan muuttumaton.

DAGLAP todistaa sekä sovelluksen käyttämiä tietoja säilövän tietopalvelinten että itse sovelluspalvelimen sijainnin, mutta GeoProof todistaa ainoastaan tiedon sijainnin. Tiedon sijainnin tarkistaminen voi johtaa myös siihen, ettei GeoProof pysty jäljittämään varmuuskopioiden sijaintia. DAGLAP:lla on tosin sama rajoite. Tiedon salaaminen estää molemmissa

menetelmissä tiedon selkokielisen vuotamisen muualle. GeoProofissa salaaminen tapahtuu hallussapitoprotokollan kautta ja DAGLAP:ssa erikseen.

GeoProofin määrittely tapahtuu korkeammalla tasolla, eikä siinä kiinnitetä huomiota siihen, miten sovelluspalvelin (tai GeoProofin tapauksessa sijaintia tarkastava taho) voi tietää sijainnintodennuslaitteen (tai tarkistajan) olevan aito. Artikkelissa ei myöskään kuvailla, miten sanomat kuljetetaan, mutta GeoProofin kanssa voi käyttää esimerkiksi samanlaista salausjärjestelmää kuin DAGLAP:ssakin, mikä takaisi luottamuksellisuuden, eheyden ja aitouden.

Vertailemme seuraavaksi TGADP:ia ja DAGLAP:ia. Niille yhteistä on TPM:n käyttö. TGADP käyttää TPM:n sinetöintitoimintoa tiedon salausavaimen säilömiseen turvallisesti niin, että salausavaimen voi purkaa vain, jos sijaintitiedot täsmäävät alustusvaiheen tietoihin [27]. TGADP vaatii erikseen TPM:n kautta toteuttavaa ohjelmiston etätodistamista (*remote attestation*), joka suoritetaan tietopalvelimille ja sijaintitietoa tarjoavalle laitteelle ennen kuin sen tietoihin voidaan luottaa. Salausavaimen sinetöinti takaa, että tiedot voi purkaa vain asiakkaan sallimissa sijainneissa.

TGADP:ssa sijaintitietoja tarjoava laite luottaa satelliittipaikannukseen, jonka antamat tiedot voidaan väärentää. Epäluotettava palveluntarjoaja voi alustusvaiheessa ohjata tiedot väärään paikkaan, kun taas DAGLAP:ssa sovellus tietää, mitä sijaintitietoja se haluaa. Näin DAGLAP ei vaadi yhtä paljoa luottamusta palveluntarjoajaan kuin TGADP. Toinen merkittävä TGADP:n ongelma on se, että sijaintitiedot perustuvaan keskitettyyn tietokantaan, jolla satelliittipaikannuksen tiedot muutetaan sovelluksille sopivaan muotoon. Tämä tietokanta toimii houkuttelevana hyökkäyskohteena ja palveluntarjoajakin voi sen avulla helposti väärentää kaikki sijaintitiedot halutessaan.

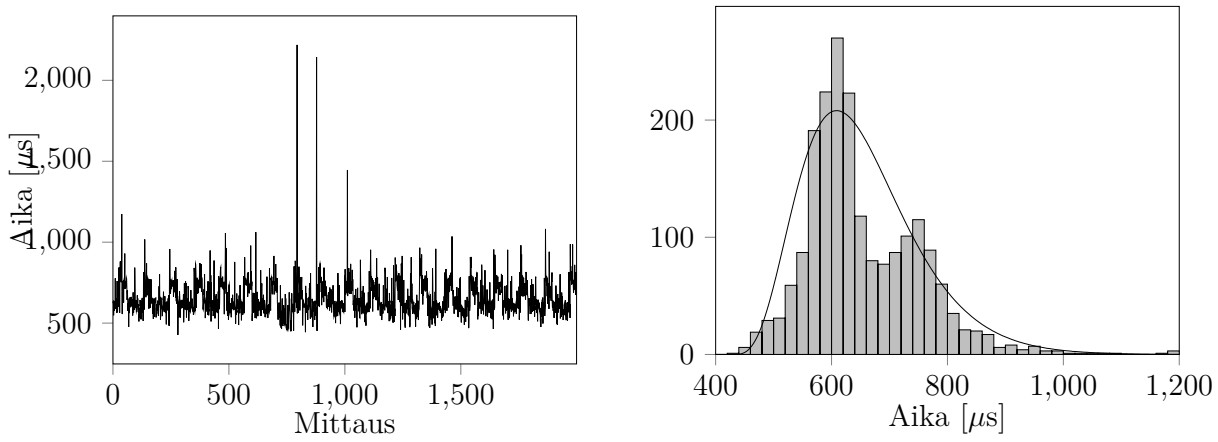
Vertailu osoittaa, että sijainnintodennuslaitteen ja DAGLAP:n yhdistelmä pystyy täyttämään aiemmin asetetut vaatimukset paremmin kuin aiemmin ehdotetut ratkaisut.

5.4 Kokeet

Kierrosajan mittausta varten ja vaatimuksen 5 arvioimiseksi on hyödyllistä myös suorittaa kokeita. Niihin käytetään alaluvussa 4.6 mainittua prototyyppiä. Prototyypin palvelinsovellusta ajettiin Raspberry Pi 3 -tietokoneella, joka oli samassa lähiverkossa kuin sovelluspalve-

limena toiminut tietokone. Molemmat laitteet ovat yhdistetty kotikäyttöön suunniteltuun reitittimeen verkkokaapeleilla. Kaapelien yhteispituus sovelluspalvelimelta sijainnintodennuslaitteelle reitittimen kautta oli noin 50 metriä. Kokeissa mitattiin sekä kierrosaikaa että toteutuksen vaatimaa suoritinaikaa.

Kierrosajan mittaaminen toteutettiin niin, että asiakaskone avasi sijainnintodennuslaitteena toimineelle palvelimelle yhteyden ja kysyi neljäkymmentä kertaa sijaintitietoa (jokaisen kyselysanoman ja vastaussanoman koko 24 tavua). Tietojen kyselyiden toteutus varmistaa, ettei palvelin voi tietää hyväksyttävää vastausta ennen kuin pyyntö oli luettu. Kierrosaika mitattiin jokaisen neljänkymmenen kyselyn yhteydessä. Yhteys avattiin ja suljettiin yhteensä viisikymmentä kertaa, jolloin saatiin kaikkiaan kaksituhatta kierrosaikanäytettä. Kierrosaikoihin sovitettiin myös gamma-jakaumaa, jonka on havaittu esittävän hyvin verkkoviipeitä [5]. Kierrosaikajakauma näkyy kuvassa 5.1.

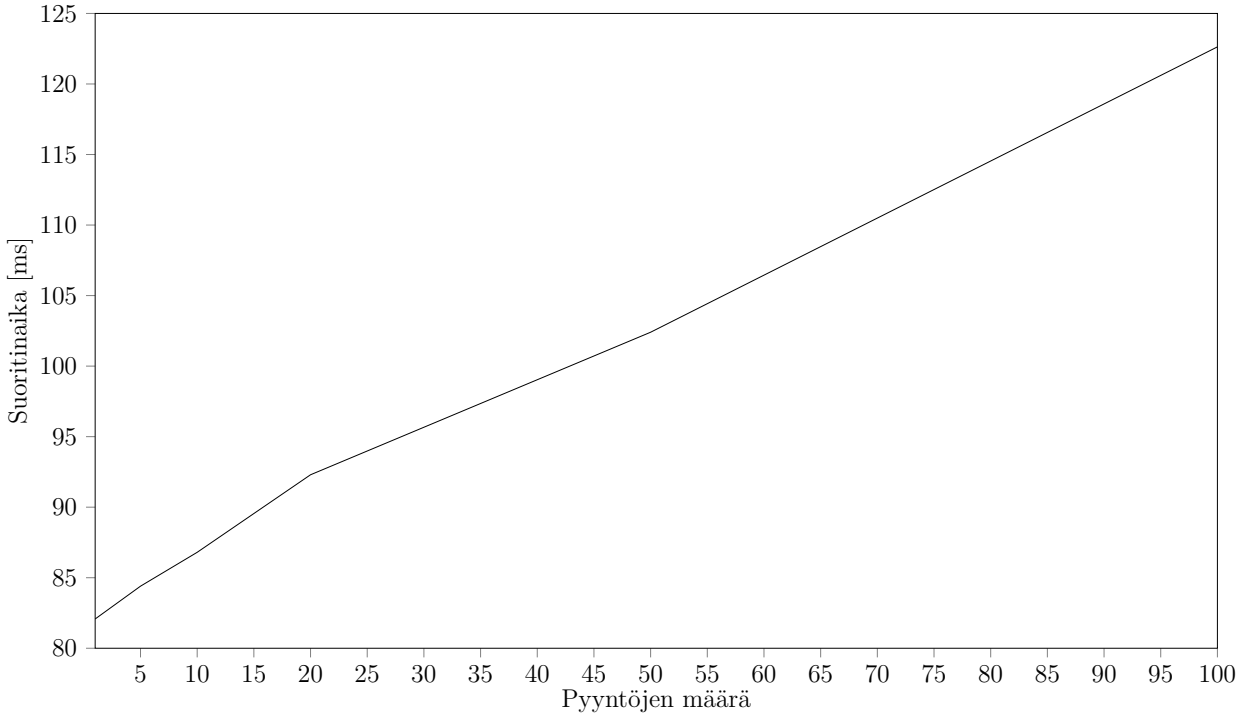


(a) Kierrosajat

(b) Jakauma

Kuva 5.1: Mitatut kierrosajat, niiden jakauma ja sovitettu vakiolla siirretty gamma-jakauma ($c = 426\ \mu s, \alpha \approx 5.11\ \mu s, \beta \approx 0.0225\ \mu s$).

Kaavion ja sovitetun gamma-jakauman perusteella voidaan päätellä, että tällä laitteistolla kierrosaika on keskimäärin noin 630 μs. Pienin mitattu kierrosaika oli 427 μs. Ainoastaan yhdeksän kahdesta tuhannesta kierrosajasta oli yli 1000 μs, ja vain kaksi oli yli 2000 μs. Syy näin korkeihin aikoihin saattoi olla esimerkiksi asiakaskoneelle (josta mittaukset tehtiin) tai palvelimelle tapahtuneet keskeytykset. Tulokset osoittavat, että kierrosaika noudattaa gamma-jakaumaa, ja että sen perusteella voidaan melko tarkasti mallintaa kierrosaikaa. Jos rajaksi asetetaan esimerkiksi 1 $ms = 1000\ \mu s$, todennäköisyys, että kaikki viisi yritystä johtavat aikakatkaisuun, on yksi 520 miljardista. Jos palveluntarjoaja nyt ohjaa pyynnöt muualle niin, että uudelleenohjauksen takia kaikki kierrosajat kasvavat esimerkiksi 400 mikrosekunnilla, tarkistus menisi läpi vain 86,9 % todennäköisyydellä. Jos etäisyys on



Kuva 5.2: Palvelinohjelman vaatima suoritusaika yhteyden avaamisesta sen sulkemiseen, kun sijaintipyyntöjä lähetettiin yhteyden aikana tietty määrä.

pidempi ja kierrosajat kasvavatkin esimerkiksi 550 mikrosekunnilla, tarkistus menee läpi vain kerran noin tuhannesta. Tätä käsitellään tarkemmin pohdintaluvussa.

Suorituskyvyn arvioimiseksi mitattiin palvelimen vaatimaa suoritinaikaa pyyntöjen määrän funktiona. Tämä mittaus toistettiin kymmenesti ja tulosten keskiarvo otettiin talteen. Mittaustulokset näkyvät kuvassa 5.2. Mittaukset osoittavat, että suoritinaika riippuu lineaarisesti pyyntöjen määrästä, ja että suoritinaika koostuu vakiotermistä (yhteyden avaamiseen ja sulkemiseen käytetty aika) sekä lineaarisesti pyyntöjen määrästä riippuvasta termistä. Tietopalvelimen sijainnin tarkistuksen voi myös olettaa toimivan yhtä nopeasti. Haasteiden lähettämisen yhteydessä sijainnintodennuslaite ei tee monimutkaisempia laskutoimituksia, joten suoritinaikojen pitäisi noudattaa samaa jakaumaa.

Laitteiston ja tulosten huomioon ottaen se pystyisi käsittelemään joitain kymmeniä pyyntöjä sekunnissa, mikä voi muodostua ongelmaksi jos suuri määrä sovelluspalvelimia käynnistyy samanaikaisesti. Tehokkaampi ja suunnatumpi laitteisto sekä etenkin laitteistotuki salaukselle kuitenkin luultavasti nopeuttaisivat huomattavasti käsittelyä, joten protokollaa itsessään voi pitää melko kevyenä.

6 Pohdinta

Tässä luvussa pohditaan aiemmin esitettyjä tuloksia tutkimusalan viitekehyksessä ja käydään läpi tulosten merkittävyyttä. Luvussa käsitellään myös ratkaisun rajoitteita sekä pohditaan, miten sen toteuttaminen onnistuisi käytännössä.

Tutkimuksen tarkoituksena oli kehittää keino, jolla pilvipalvelinkeskuksessa ajettava sovellus voi tarkistaa oman ja tietojensa sijainnin luotettavalla tavalla niin, että se voi esimerkiksi estää toimintansa tiettyjen sijaintien ulkopuolella ja estää tietojen pääsy tietyn alueen ulkopuolelle. Työn tuloksena on luomus, joka täyttää vaatimukset aiempia ratkaisuja paremmin, kuten aiempi arviointiluku osoittaa. DAGLAP-protokollasto soveltuu hyvin sijainnintodennuslaitteen ohjelmistopuolen toteuttamiseen, mutta sen soveltamisala on todennäköisesti laajempikin. Samaa protokollaa voisi mahdollisesti käyttää sijaintitietojen sijasta muun palvelinkeskuksille yksilöllisen tiedon säilömiseen.

Työssä noudatettiin suunnittelutieteen seitsemää ohjenuoraa. Tuloksena syntyi luomus, joka on sijainnintodennuslaitteesta ja protokollastosta koostuva menetelmä, ja joka ratkaisee merkittäväksi osoitetun ongelman. Suunniteltu ratkaisu arvioitiin oikeaoppisin menetelmin ja siten sillä on tieteellinen panos. Tutkimus suoritettiin järjestelmällisiä menetelmiä käyttäen. Protokollaston sekä sen suunnittelun osana käytettiin aiempia sekä alemman tason toimivaksi todettuja ratkaisuja, kuten salausprotokollia. Suunnittelu- ja arviointiluvut viestivät työn tuloksia.

Tutkimuksen validiteetti riippuu paljolti siinä käytettyjen menetelmien validiteetista. Suunnittelutieteen on osoitettu olevan toimiva tutkimusmenetelmä tietojenkäsittelytieteen alalla. Ulkoinen validiteetti on osoitettu jo aiemmin, kun on käsitelty tutkimuksen panosta alalle yleisesti. Sisäinen validiteetti on osoitettu todistamalla ratkaisun täyttävän vaatimukset, jotka taas on muotoiltu niin, että vaatimukset täyttämällä ratkaisee samalla myös ongelman.

6.1 Käytännön järjestelyt

Tässä luvussa käydään läpi asioita, jotka liittyvät protokollaston toteutukseen käytännössä.

DAGLAP-protokollaston sijainnintodennuslaitteen aitouden todennus nojaa lähes täysin allekirjoituksiin ja varmenteisiin. Sovelluspalvelin tarkistaa sijainnintodennuslaitteen aitouden tarkistamalla sen lähettämän varmenteen, joka todentaa avaintenvaihdon allekirjoitukseen käytetyn avainparin julkisen avaimen aitouden. Allekirjoitusten yhteydessä kuitenkin herää kysymys siitä, kuinka varmenteen aitous tarkistetaan. Käytännössä tähän tarkoitukseen tarvitaan jonkinlainen julkisten avainten hallintajärjestelmä. Toisin sanottuna vastuu varmenteen allekirjoituksesta annetaan luotetulle taholle, jonka luotettavuuden todistavan varmenteen on allekirjoittanut toinen luotettava taho, ja niin edelleen niin sanottuun juurivarmenteeseen asti.

Vastaavanlaista järjestelmää käytetään jo verkossa käytetyn tiedon salaamiseen, sillä HTTPS-protokolla nojaa varmenteineen omaan julkisten avainten hallintarakenteeseensa. Täten tämä ratkaisu on osoitettu sinänsä toimivaksi ja skaalautuvaksi. DAGLAP-protokollan yhteydessä käytetty vastaavanlainen rakenne voi olla sama kuin HTTPS-järjestelmässä, tai se voidaan perustaa erikseen. On tärkeää varmistaa, ettei palvelinkeskuksen omistaja voi allekirjoittaa sijainnintodennuslaitteelle luotettavaa varmennetta, koska muuten palveluntarjoaja voi väärentää sijainnintodennuslaitteen antamat tiedot. Käytännössä varmenteen voi allekirjoittaa esimerkiksi kolmannen osapuolen tarkastaja.

Luottamus kolmannen osapuolen tarkastajaan vaaditaan myös sijainnintodennuslaitteen asennuksen yhteydessä. Tällöin herää kuitenkin kysymys siitä, ketä voidaan pitää luotettavana tahona. Yksi vaihtoehto on sijainnintodennuslaitteen valmistaja. Toinen vaihtoehto on tarkastaja, joka on sijainnintodennuslaitteen valmistajasta erillinen yritys. Tärkeintä on estää palveluntarjoajan tai ulkoisen toimijan, mahdollisesti jopa valtion, sekaantuminen asiaan, koska muuten sijainnintodennuslaitteisiin ei voi olla luottamista. Kolmas vaihtoehto on, että jos sijainnintodennuslaite toteutetaan esimerkiksi asiakaskohtaisesti älykortilla, asiakasta tai tämän valtuuttamaa tahoa voidaan pitää luotettavana.

Vaihtoehto keskitetylle julkisten avainten hallintajärjestelmälle on niin sanottu hajautettu luottamusverkko, jota käytetään muun muassa PGP-järjestelmässä [6]. Keskitetty ratkaisu

lienee kuitenkin tässä tapauksessa luontevampi, koska ei ole selvää, mihin osapuoleen olisi luottamista hajautetulla mallilla tässä tapauksessa.

Huomiota tulee kiinnittää myös protokollaston yhteydessä käytettävien parametrien valintaan. Protokollastossa on neljä parametria: N_{to}, uudelleenyritysten määrä aikakatkaisun tapahtuessa, N_c, haasteiden määrä, T_{max}, sovelluspalvelimen ja sijainnintodennuslaitteen välisen yhteyden aikakatkaisuraja ja T_{seek}, tietopalvelimelle sallittu tiedonhakuvälys.

Yhteysyrityksien määrän tulisi, kuten aiemmin mainittu, olla melko alhainen, koska liian monen yrityksen salliminen mahdollistaa tilanteen, jossa palveluntarjoaja voi pyrkiä väärentämään sijainnin ohjaamalla vain murto-osan pyynnöistä sijainnintodennuslaitteelle. Liian monen yrityksen salliminen hidastaa myös prosessia siinä tapauksessa, että palvelin sijaitsee liian kaukana sijainnintodennuslaitteesta. Toisaalta liian pieni arvo voi vaikuttaa merkittävästi järjestelmän luotettavuuteen, jolloin hetkellinen häiriö voi johtaa sovelluksen toiminnan lamaantumisen. Sopivan arvon määrittäminen sallittujen yritysten lukumäärälle vaatinee lisätutkimuksia. Haasteiden määrä taas riippuu käytetystä haastealgoritmista ja tiedon määrästä.

Kierrosaikaan liittyvä rajaparametri T_{max} on valittava tarkoin. Sen arvon päättämisessä on otettava huomioon esimerkiksi reitittimien ja välipalvelimien aiheuttamat viipeet. Vaikka yksi millisekunti voi vaikuttaa moniin yhteyksiin järkevältä arvolta, se vastaa valonnopeudella noin 150 kilometriä (koska kierrosaika on kaksisuuntainen). Kierrosaikaan luottamisessa on omat ongelmansa, joita käsitellään tarkemmin rajoitteita käsittelevässä alaluvussa.

T_{seek} on tietopalvelimelle sallittu välys. Lisäajan tarkoituksena on antaa tietopalvelimen hakea haasteeseen vastaamisen tarvitut tiedot esimerkiksi kiintolevyiltä, joiden hakuviipeitä on käsitelty tarkasti muun muassa GeoProofin artikkelissa [2]. Tämän arvon määrittämiseksi tulisi mitata palvelinkeskuksessa käytettyjen levyjen tai muiden massamuistilaitteiden suorituskykyä. Hakuaikaviipeiden määrittäminen voi olla mahdollista esimerkiksi suorittamalla hakukyselyitä oikeassa järjestyksessä, joten kalibrointi on mahdollisesti automatisoitavissa. Tarkempien yksityiskohtien ja tällaisen mittauksen luotettavuuden määrittäminen vaatii lisätutkimuksia.

Protokollasto käyttää eri kryptografisia protokollia. DAGLAP vaatii symmetrisen salausprotokollan, asymmetrisen salausprotokollan (avaintenvaihtoon ja todennukseen), allekir-

joitusprotokollan (varmenteisiin) ja hallussapitotodistusprotokollan (käsitelty aiemmin). Symmetriseksi salausprotokollaksi valittiin AES-GCM, mutta ChaCha20-Poly1305 [21] on toinen vaihtoehto. Jälkimmäistä pidetään mahdollisesti jopa AES-GCM:ää suorituskykyisempänä toteuttaa vähätehoisilla laitteilla.

Sekä valittu allekirjoitusprotokolla että avaintenvaihtoprotokolla nojaavat elliptisiin käyriin. Molemmat niistä voidaan murtaa, jos elliptisten käyrien diskreetille logaritmille keksitään nopea ratkaisu, ja tällainen saattaa tulevaisuudessa olla mahdollista. Kvanttitietokoneille on suunniteltu niin sanottu Shorin algoritmi, jolla pystyy jakamaan kokonaisluvun tekijöihinsä polynomisessa ajassa [35]. RSA:n turvallisuus perustuu tekijöiden jakamisen vaikeuteen, joten Shorin algoritmi murtaa sen. Algoritmia voi myös soveltaa ratkaisemaan diskreetin logaritmin, johon DSA:n (allekirjoitusprotokollan) ja perinteisen Diffie–Hellman-avaintenvaihdon turvallisuus perustuu. Elliptisiin käyriin nojaava kryptografia perustuu myös diskreettiin logaritmiin, tosin elliptisillä käyrillä, ja tämäkin on Shorin algoritmilla nopeasti ratkaistavissa [32]. Tämän takia on aihetta myös harkita sekä allekirjoitus- että avaintenvaihtoprotokollan vaihtamista sellaisiin, jotka olisivat turvallisia myös kvanttitietokoneita vastaan.

DAGLAP:n toteutus voisi kenties TLS:n tapaan tukea useita eri algoritmeja. Sovelluspalvelin ja sijainnintodennuslaite voisivat yhteyttä muodostaessa neuvotella, mitä protokollia käytetään. Tällöin on kuitenkin pyrittävä välttämään TLS:ssä usein tapahtunut tilanne, jossa algoritmi vanhenee ja siitä tulee epäturvallinen. Sijainnintodennuslaitteen tulisi vaatia päivityksiä mahdollisimman harvoin, koska päivitysjärjestelyt vaativat luotetun tarkastajan ja sijainnintodennuslaitteen valmistajan tukea.

DAGLAP myös edellyttää, että sovelluspalvelimille on asennettu TPM, koska siihen säilötään turvallisesti tiedon salaamiseen käytetty salausavain. TPM:n sinetöinnin avulla avaimeen pääsy voidaan estää, jos sijaintitiedot eivät täsmää. Jos TPM:ää ei ole käytössä, salausavaimen päätyminen palveluntarjoajan käsiin voi olla mahdollista, jolloin tämä voi purkaa tiedot ja siirtää ne sen jälkeen toiseen sijaintiin. TPM ei kuitenkaan ole ainakaan teoriassa välttämätön, mikäli avaimen salassapito onnistuu muulla tavalla.

6.2 Rajoitteita

Tässä luvussa käydään läpi ratkaisun rajoitteita. Niiden yhteydessä käsitellään rajoitteiden vaikutuksia sekä mahdollisia ratkaisuja rajoitteiden välttämiseksi.

Ensimmäinen rajoitus on se, että tätä ratkaisua vastaan voi hyökätä siirtämällä sijainnintodennuslaitetta. Yhtenä oletuksena oli, ettei sijainnintodennuslaitteeseen voi fyysisesti kajota rikkomatta tai vahingoittamatta sitä. Tämän oletetaan vaikuttavan myös niin, ettei sijainnintodennuslaitetta voi siirtää siirtämättä sen ympäristöä. On kuitenkin jossain tapauksissa mahdollista, että koko palvelinkeskus tai sen osa, jossa sijainnintodennuslaite sijaitsee, voidaan siirtää toiseen sijaintiin ja näin väärentää sijaintitiedot. Näin voi esimerkiksi tapahtua, jos palvelinkeskus on rakennettu modulaarisista yksiköistä, kuten konteista.

Tämän estäminen vaatii lisälaitteistoa sijainnintodennuslaitteeseen. Siihen pitäisi esimerkiksi asentaa satelliittipaikannus – jonka antamat tiedot voidaan toisaalta väärentää – tai kenties jonkinlaisia antureita, jotka pystyisivät havaitsemaan liikkeen ja merkitsemään laitteen tiedot epäluotettaviksi. Ongelmana on taas luotettavuus: luonnonilmiöt voivat johtaa mekanismin laukeamiseen vahingossa ja laitevian vaara kasvaa. Toisaalta sijainnintodennuslaitteen asennuksen yhteydessä tarkastaja voisi kieltäytyä hyväksymästä laitteen asentamista ympäristöön, jossa sitä voisi liikutella.

Toinen rajoitus liittyy tiedon sijainnin tarkistamiseen. Tiedolla tai datalla ei tunnetusti ole sijaintia eikä se voi kertoa sijaintiaan, joten DAGLAP pyrkii paikantamaan vain tietopalvelimen ja salaamaan tiedot niin, etteivät ne voi selkomuodossa päästä halutun alueen ulkopuolelle. Tietopalvelimien sijainnin tarkistamiseen käytetty menetelmä ei kuitenkaan ole täydellinen. Hallussapitotodistukset takaavat vain, että kyseisellä palvelimella on pääsy alkuperäiseen tietoon, eikä sitä, että tieto on tallennettu kyseiselle palvelimelle. Välipalvelin voi siis tekeytyä tietopalvelimeksi.

Edellä kuvattua ongelmaa lievittää se, että välipalvelin ei voi vastata itse haasteisiin. Näin ollen välipalvelimena toimiminen nostaa haasteiden kierrosaikaa, jonka sovelluspalvelin voi huomata saadessaan kierrosaikamittaukset sijainnintodennuslaitteelta. Toisaalta kierrosajan luotettavuuden liittyy omat ongelmansa, joista kerrotaan lisää kolmatta rajoitusta käsitellessä. Välipalvelin tai vain osaa tiedoista säilövä tietopalvelin voi kyetä vastaamaan

haasteisiin, jotka liittyvät vaan kyseiseen osaan tiedoista. Tämän takia on erityisen tärkeää, että sovelluspalvelimen luomat haasteet ovat satunnaisia.

Kolmas rajoitus liittyy DAGLAP:n olennaiseen osaan eli etäisyyden rajaamiseen kierrosaikoihin nojautuen. Sovellus- ja tietopalvelimen sekä sijainnintodennuslaitteen välinen läheisyys todetaan pelkästään kierrosaikojen eli viipeiden perusteella. Pohjimmaisena oletuksena on, että etäisyys kahden laitteen välillä vaikuttaa lineaarisesti niiden välisiin verkkoviipeisiin. Tiedot verkossa eivät kuitenkaan liiku valonnopeudella tai edes samalla nopeudella koko ajan, sillä johdoissa käytetyt väliaineet vaihtelevat, ja suuri osa viipeestä on odottamista – yleensä reitittimen tai palvelimen jonossa.

Kierrosaikaan luottamiseen liittyy useita ongelmia. Eräs niistä on, että kierrosajan jakauma on gamma-jakauma, jolla on äärettömän pitkä "häntä". Tämä tarkoittaa sitä, että suurin osa kierrosajoista on melko lähellä pienintä mahdollista arvoa, mutta niillä ei ole ylärajaa ja sattumanvaraiset tekijät voivat tehdä kierrosajasta teoriassa mielivaltaisen pitkän. DAGLAP-protokollastoon on suunniteltu uudelleenyritykset juuri sen takia, että hetkelliset häiriöt voivat vaikuttaa kierrosaikamittauksiin huomattavasti.

Tietopalvelimen kohdalla DAGLAP määrittelee, että sovelluspalvelimen on tarkistettava korkein kierrosaika tietopalvelimen ja sijainnintodennuslaitteen välillä ja varmistettava, että se alittaa tietyn rajan. Tämä on turvallisin ratkaisu, sillä se varmistaa, ettei yksikään oikea vastaus voinut tulla muualta kuin tietopalvelimelta – eli siltä palvelimelta, jonka oletetaan vastanneen muihinkin haasteisiin. Ratkaisu on kuitenkin hyvin altis häiriöille. Jos tietopalvelin saa esimerkiksi toisen pyynnön juuri samaan aikaan ja yksi haasteiden vastauksista viivästyy, tämä voi johtaa tulosten hylkäämiseen. Siksi tulisikin harkita suurimman mahdollisen kierrosajan sijasta esimerkiksi kehittyneempää tilastollista mallia. Kierrosajan jakauman tutkiminen tällaisessa ympäristössä vaatii lisätutkimuksia.

Kierrosaikarajan asettamiseen liittyy olennaisesti väärien positiivisten ja negatiivisten tasapainotteluongelma. Vääräksi positiiviseksi lasketaan tapaukset, joissa kierrosaika alittaa rajan, vaikka sijainnintodennuslaite ja sovelluspalvelin sijaitsevat esimerkiksi eri palvelinkeskuksissa. Väärä negatiivinen on taas tapaus, jossa kierrosaika ylittää rajan, vaikka laitteet ovat lähellä toisiaan. Alempi kierrosaikaraja laskee väärien positiivisten määrää, mutta nostaa väärien negatiivisten määrää. Jälkimmäisen vaikutuksia voi vähentää jossain määrin kasvattamalla yritysten lukumäärää, mutta tämä kasvattaa väärien positiivisten

määrää ja hidastaa sijainnintarkistusprosessia. Jos väärät positiiviset ovat vahingollisempia kuin väärät negatiiviset kuten tietoturvassa yleensä, kierrosaikaraja on asetettava melko alhaiseksi.

Toinen mahdollinen kierrosaikoihin liittyvä ongelma on niiden muuttuvaisuus. Jos palveluntarjoaja muuttaa palvelinkeskuksen verkkolaitteiston kokoonpanoa, kierrosajat voivat muuttua. Yleensä parempaan laitteistoon vaihtamisen voisi kuvitella laskevan kierrosaikoja. Kuvitellaan tapaus, jossa asiakas on saanut sovelluksensa palvelinkeskukseen, jonka laitteistoa ollaan uusimassa vain muutaman kuukauden päässä käyttöönotosta. Jos keskiviive sijainnintodennuslaitteen ja sovelluspalvelimen välillä on silloin esimerkiksi 1500 μs, palveluntarjoaja voisi siirtää sovelluksensa muualle ja uudella laitteistolla saada aikaan uudeksi kierrosajaksi 1500 μs, vaikka etäisyys on nyt paljon suurempi. Etenemisviipeen kasvu on siis saatu kompensoitua laskemalla muita viipeitä.

Tällaista hyökkäystä vastaan on vaikea keksiä minkäänlaista puolustuskeinoa, sillä sovelluspalvelin ei tiedä verkkonsa topologiasta yhtään mitään, eikä se myöskään voi tietää. Edes työkalu, kuten `traceroute`, jota käytetään usein verkon ongelmien ratkaisuun ja joka luettelee välissä olevat palvelimet, ei auta, koska se perustuu IP-pakettien elinaikaan eli hyppyrajoitukseen, jonka pystyy helposti väärentämään. Yksi ratkaisu voisi olla kolmannen osapuolen tarkastajat, mutta tämä johtaa muihin käytännön ongelmiin. Vaihtoehtona on myös langattomaan teknologiaan perustuvan etäisyydenrajausprotokollan käyttö, mutta se vaatii lisälaitteistoa sekä sovelluspalvelimiin että sijainnintodennuslaitteisiin.

DAGLAP-protokolla ei sellaisenaan myöskään kestä palvelunestohyökkäyksiä. Sijainnintodennuslaitetta vastaan voidaan suorittaa palvelunestohyökkäys, jossa palveluntarjoajan tai ulkoisen hyökkääjän omistama palvelin avaa laitteeseen mahdollisimman monta yhteyttä ja pyrkii estämään muita palvelimia selvittämästä sijaintiaan. Palveluntarjoaja tuskin suorittaa tällaista hyökkäystä, koska se on palveluntarjoajan etujen vastaista. Sovellus, joka ei pysty tarkistamaan sijaintiaan, ei suostu toimimaan, mikä taas laskee asiakkaan luottamusta palveluntarjoajaan. Ulkoiselle hyökkääjälle tilanne saattaa kuitenkin olla toisenlainen. Jos hyökkääjä on esimerkiksi varannut samasta palvelinkeskuksesta tilaa, hän voi estää kilpailijansa palvelua toimimasta ja näin yrittää epäsuorasti vaikuttaa sen toimintaan niin, että kilpailija saadaan siirtämään palvelunsa toiseen palvelinkeskukseen.

Palvelunestohyökkäyksen ei välttämättä tarvitse kohdistua sijainnintodennuslaitteeseen. Jo kuorman lisääminen palvelinkeskuksen sisäverkkoon voi aiheuttaa häiriöitä, sillä lisäkuorma kasvattaa reitittimien jonotusviivettä ja siten epäsuorasti myös kaikkia kierrosaikoja. Kuorman lisääminen sisäverkkoon ei välttämättä vaadi edes toimia verkon sisällä toimivalta palvelimelta, sillä se voi olla mahdollista ulkopuoleltakin.

Palvelinkeskuksen sisällä välipalvelinta tai reititintä hallinnoiva taho voi myös halutessaan estää yhteyksien muodostamisen sekoittamalla sovellus- tai tietopalvelimen lähettämiä avaintenvaihtosanomia. Tämä estää yhteyksien muodostamisen, koska salausavaimia ei saada täsmäämään. Vain ulkoiset hyökkääjät voivat hyötyä palvelunestohyökkäyksistä. Niiden estäminen on vaikeaa kierrosaikoihin nojautuvan etäisyydenrajauksen takia.

7 Johtopäätökset

Tämän työn tavoitteena on ollut kehittää ratkaisu, jonka avulla pilvipalveluissa ajettavat sovellusohjelmat voivat varmistua palvelimiensa ja tietojensa sijainnista. Työn aikana on perehdytty tämän ongelman eri muotojen ratkaisemista varten kehitettyihin menetelmiin ja osoitettu niiden olevan omalta osaltaan päteviä, muttei kykeneviä ratkaisemaan ongelmaa tyydyttävällä tavalla. Toimivalle menetelmälle on asetettu yksityiskohtaiset vaatimukset, ja uusi menetelmä on kehitetty suunnittelutiedettä käyttämällä. Kuten pohdintaluvussa on aiemmin käsitelty, suunnittelutieteen periaatteet on täytetty suunnittelutyön aikana.

Työn tuloksena on syntynyt ratkaisuna yhdistelmä, jossa laitteistona toimii sijainnintodennuslaitteet ja ohjelmistona sitä varten eritoten suunniteltu DAGLAP-protokollasto. Kyseinen protokollasto ja sen vaatima laitteisto on kuvattu läpikotaisesti ja arvioinnin kautta on osoitettu, että ratkaisu täyttää kaikki aiemmin esitetyt vaatimukset. Tämän lisäksi suoritettu vertailutyö osoittaa, että kyseinen menetelmä ratkaisee ongelman tehokkaammin kuin aiemmin alalla esitetyt menetelmät.

Tämä tulos osoittaa, että alkuperäinen ongelma on ratkaistavissa tavalla, jonka toteuttaminen käytännössä on täysin mahdollista. Suunniteltu menetelmä pystyy paikantamaan sekä sovellus- että tietopalvelimen sijainnin tiettyyn palvelinkeskukseen, jos kyseiseen palvelinkeskukseen on luotettavan ulkoisen tarkastajan tarkastamana asennettu pysyvästi laite, joka antaa luotettavaa tietoa palvelinkeskuksen sijainnista.

Vaikka ratkaisu täyttääkin kaikki vaatimukset, sillä on myös omat rajoitteensa. Näitä rajoitteita ja niiden vaikutuksia on arvioitu, ja niiden perusteella voidaan tehdä jatkotutkimusta. Tutkimuksen aiheita voisivat olla muun muassa kierrosajan jakauma, satelliittipaikannustiedon luotettavuus, massamuistin hakuviipeet (jotta tietopalvelin voi vastata haasteisiin), sekä se, olisiko kierrosajan sijasta olemassa muuta menetelmää, joilla läheisyys voitaisiin arvioida. DAGLAP-protokollaston periaatteet voivat myös sopia muihin tarkoituksiin. Aiemmin käsiteltiin mahdollisuutta, että sijaintitietojen sijasta palvelimet voisivat tarkistaa jotain muuta palvelinkeskuskohtaista tietoa. Näin tällä tutkimuksella on mahdollisesti myös arvoa sijainnin todentamisen ulkopuolellakin.

Lähteet

[1] A. Albeshri, C. Boyd ja J. G. Nieto. ”Enhanced geoproof: improved geographic assurance for data in the cloud”. *International Journal of Information Security* 13.2 (2014), s. 191–198.

[2] A. Albeshri, C. Boyd ja J. G. Nieto. ”GeoProof: Proofs of Geographic Location for Cloud Computing Environment”. Teoksessa: *2012 32nd International Conference on Distributed Computing Systems Workshops*. 2012, s. 506–514. DOI: 10.1109/ICDCSW.2012.50.

[3] M. Bartock, M. Souppaya, R. Yeluri, U. Shetty, J. Greene, S. Orrin, H. Prafullchandra, J. McLeese ja K. Scarfone. *Security and Privacy Controls for Federal Information Systems and Organizations*. Tekninen raportti NISTIR 7904. Gaithersburg, MD: National Institute of Standards ja Technology, 2015. DOI: 10.6028/NIST.IR.7904.

[4] D. J. Bernstein. ”The Salsa20 Family of Stream Ciphers”. Teoksessa: *New Stream Cipher Designs: The eSTREAM Finalists*. Toim. M. Robshaw ja O. Billet. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, s. 84–97. ISBN: 978-3-540-68351-3. DOI: 10.1007/978-3-540-68351-3_8. URL: https://doi.org/10.1007/978-3-540-68351-3_8.

[5] J.-C. Bolot. ”End-to-End Packet Delay and Loss Behavior in the Internet”. *SIGCOMM Comput. Commun. Rev.* 23.4 (lokakuu 1993), s. 289–298. ISSN: 0146-4833. DOI: 10.1145/167954.166265. URL: https://doi.org/10.1145/167954.166265.

[6] G. Caronni. ”Walking the web of trust”. Teoksessa: *Proceedings IEEE 9th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE 2000)*. IEEE. 2000, s. 153–158.

[7] R. Chirgwin. ”Sweden leaked every car owners’ details last year, then tried to hush it up”. *The Register* (23. heinäkuuta 2017). URL: https://www.theregister.com/2017/07/23/sweden_leaked_every_car_owners_details_last_year_then_tried_to_hush_it_up/ (haettu 30. 03. 2022).

[8] E. De Win, S. Mister, B. Preneel ja M. Wiener. ”On the performance of signature schemes based on elliptic curves”. Teoksessa: *Algorithmic Number Theory*. Toim. J. P. Buhler. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, s. 252–266. ISBN: 978-3-540-69113-6.

[9] C. C. Erway, A. Küpçü, C. Papamanthou ja R. Tamassia. ”Dynamic Provable Data Possession”. *ACM Trans. Inf. Syst. Secur.* 17.4 (huhtikuu 2015). ISSN: 1094-9224. DOI: 10.1145/2699909. URL: https://doi-org.libproxy.helsinki.fi/10.1145/2699909.

[10] M. Eskandari, A. S. De Oliveira ja B. Crispo. ”VLOC: An Approach to Verify the Physical Location of a Virtual Machine In Cloud”. Teoksessa: *2014 IEEE 6th International Conference on Cloud Computing Technology and Science*. 2014, s. 86–94. DOI: 10.1109/CloudCom.2014.47.

[11] M. Gondree ja Z. N. Peterson. ”Geolocation of Data in the Cloud”. Teoksessa: *Proceedings of the Third ACM Conference on Data and Application Security and Privacy*. CODASPY ’13. San Antonio, Texas, USA: Association for Computing Machinery, 2013, s. 25–36. ISBN: 9781450318907. DOI: 10.1145/2435349.2435353. URL: https://doi.org/10.1145/2435349.2435353.

[12] G. P. Hancke ja M. G. Kuhn. ”An RFID distance bounding protocol”. Teoksessa: *First international conference on security and privacy for emerging areas in communications networks (SECURECOMM’05)*. IEEE. 2005, s. 67–73.

[13] A. R. Hevner, S. T. March, J. Park ja S. Ram. ”Design Science in Information Systems Research”. *MIS Quarterly* 28.1 (2004), s. 75–105. ISSN: 02767783. DOI: 10.2307/25148625. URL: http://www.jstor.org/stable/25148625.

[14] L. Hippelainen, I. Oliver ja S. Lal. ”Towards Dependably Detecting Geolocation of Cloud Servers”. Teoksessa: *Network and System Security*. Toim. Z. Yan, R. Molva, W. Mazurczyk ja R. Kantola. Cham: Springer International Publishing, 2017, s. 643–656. ISBN: 978-3-319-64701-2.

[15] L. Hippeläinen, I. J. Oliver ja S. Lal. *Enhancing integrity of data center specific information*. WIPO-patentti WO2018146180A1. 2019. URL: https://patentscope.wipo.int/search/en/detail.jsf?docId=US224380954&_cid=P20-KYZUEP-05004-1.

[16] C. Jaiswal ja V. Kumar. ”IGOD – Identifying Geolocation of Cloud Datacenter Hosting Mobile User’s Data”. Teoksessa: *2015 16th IEEE International Conference on Mobile Data Management*. Vol. 2. 2015, s. 34–37. DOI: 10.1109/MDM.2015.20.

[17] D. Johnson, A. Menezes ja S. Vanstone. ”The elliptic curve digital signature algorithm (ECDSA)”. *International journal of information security* 1.1 (2001), s. 36–63.

[18] A. Juels ja B. S. Kaliski. ”Pors: Proofs of Retrievability for Large Files”. Teoksessa: *Proceedings of the 14th ACM Conference on Computer and Communications Security*. CCS ’07. Alexandria, Virginia, USA: Association for Computing Machinery, 2007, s. 584–597. ISBN: 9781595937032. DOI: 10.1145/1315245.1315317. URL: https://doi-org.libproxy.helsinki.fi/10.1145/1315245.1315317.

[19] N. Kaaniche, E. E. Moustaine ja M. Laurent. ”A Novel Zero-Knowledge Scheme for Proof of Data Possession in Cloud Storage Applications”. Teoksessa: *2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. 2014, s. 522–531. DOI: 10.1109/CCGrid.2014.81.

[20] A. J. Kerns, D. P. Shepard, J. A. Bhatti ja T. E. Humphreys. ”Unmanned Aircraft Capture and Control Via GPS Spoofing”. *Journal of Field Robotics* 31.4 (2014), s. 617–636. DOI: https://doi.org/10.1002/rob.21513. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21513. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21513.

[21] A. Langley, W. Chang, N. Mavrogiannopoulos, J. Strombergson ja S. Josefsson. ”RFC 7905: ChaCha20-Poly1305 cipher suites for transport layer security (TLS)”. *Internet Engineering Task Force (IETF). Available online: https://www. hjp. at/doc/rfc/rfc7905. html (accessed on 6 October 2021)* (2016).

[22] J. Li, A. Squicciarini, D. Lin, S. Liang ja C. Jia. ”SecLoc: Securing Location-Sensitive Storage in the Cloud”. Teoksessa: *Proceedings of the 20th ACM Symposium on Access Control Models and Technologies*. SACMAT ’15. Vienna, Austria: Association for Computing Machinery, 2015, s. 51–61. ISBN: 9781450335560. DOI: 10.1145/2752952.2752965. URL: https://doi.org/10.1145/2752952.2752965.

[23] D. McGrew ja J. Viega. ”The Galois/counter mode of operation (GCM)”. *submission to NIST Modes of Operation Process* 20 (2004), s. 0278–0070.

[24] National Institute of Standards and Technology. *Announcing the Advanced Encryption Standard (AES)*. Tekninen raportti Federal Information Processing Standards Publications (FIPS PUBS) 197, November 26, 2001. Washington, D.C.: U.S. Department of Commerce, 2001. DOI: 10.6028/nist.fips.197.

[25] A. Noman ja C. Adams. ”DLAS: Data Location Assurance Service for cloud computing environments”. Teoksessa: *2012 Tenth Annual International Conference on Privacy, Security and Trust*. 2012, s. 225–228. DOI: 10.1109/PST.2012.6297945.

[26] A. Noman ja C. Adams. ”Hardware-based DLAS: Achieving geo-location guarantees for cloud data using TPM and Provable Data Possession”. Teoksessa: *2014 17th International Conference on Computer and Information Technology (ICCIT)*. 2014, s. 280–285. DOI: 10.1109/ICCITechn.2014.7073122.

[27] N. Paladi, M. Aslam ja C. Gehrmann. ”Trusted Geolocation-Aware Data Placement in Infrastructure Clouds”. Teoksessa: *2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications*. 2014, s. 352–360. DOI: 10.1109/TrustCom.2014.47.

[28] N. Paladi ja A. Michalas. ”“One of our hosts in another country”: Challenges of data geolocation in cloud storage”. Teoksessa: *2014 4th International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace Electronic Systems (VITAE)*. 2014, s. 1–6. DOI: 10.1109/VITAE.2014.6934507.

[29] K. Peffers, M. Rothenberger, T. Tuunanen ja R. Vaezi. ”Design Science Research Evaluation”. Teoksessa: *Design Science Research in Information Systems. Advances in Theory and Practice*. Toim. K. Peffers, M. Rothenberger ja B. Kuechler. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, s. 398–410. ISBN: 978-3-642-29863-9.

[30] K. Peffers, T. Tuunanen, M. A. Rothenberger ja S. Chatterjee. ”A Design Science Research Methodology for Information Systems Research”. *Journal of Management Information Systems* 24.3 (2007), s. 45–77. DOI: 10.2753/MIS0742-1222240302. eprint: https://doi.org/10.2753/MIS0742-1222240302. URL: https://doi.org/10.2753/MIS0742-1222240302.

[31] I. Poese, S. Uhlig, M. A. Kaafar, B. Donnet ja B. Gueye. ”IP Geolocation Databases: Unreliable?” *SIGCOMM Comput. Commun. Rev.* 41.2 (huhtikuu 2011), s. 53–56. ISSN: 0146-4833. DOI: 10.1145/1971162.1971171. URL: https://doi-org.libproxy.helsinki.fi/10.1145/1971162.1971171.

[32] J. Proos ja C. Zalka. *Shor's discrete logarithm quantum algorithm for elliptic curves*. 2004. arXiv: quant-ph/0301141 [quant-ph].

[33] R. Ramaswamy, N. Weng ja T. Wolf. ”Characterizing network processing delay”. Teoksessa: *IEEE Global Telecommunications Conference, 2004. GLOBECOM '04.* Vol. 3. 2004, 1629–1634 Vol.3. DOI: 10.1109/GLOCOM.2004.1378257.

[34] M.-J. O. Saarinen. ”Cycling attacks on GCM, GHASH and other polynomial MACs and hashes”. Teoksessa: *International Workshop on Fast Software Encryption*. Springer. 2012, s. 216–225.

[35] P. W. Shor. ”Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer”. *SIAM review* 41.2 (1999), s. 303–332.

[36] M. Suárez-Albela, T. M. Fernández-Caramés, P. Fraga-Lamas ja L. Castedo. "A Practical Performance Comparison of ECC and RSA for Resource-Constrained IoT Devices". Teoksessa: *2018 Global Internet of Things Summit (GIoTS)*. 2018, s. 1–6. DOI: 10.1109/GIOTS.2018.8534575.

[37] *TCG Specification Architecture Overview, Revision 1.4*. Trusted Computing Group. 2007.