Lappeenrannan teknillinen yliopisto
Tietotekniikan osasto

PEAP 802.1x-tunnistuksen toteutus RADIUS-palvelimeen

Diplomityön aihe on hyväksytty Tietotekniikan osaston osastoneuvostossa 22.3.2006

Työn tarkastavat: Professori Heikki Kälviäinen
Diplomi-insinööri Juha Lindström

Esa Nuutinen, 0086450
Hopeahaka 2 F 49
02410 Kirkkonummi
Puh: +358-40-7512611
E-mail: esa.nuutinen@iki.fi

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto

Tietotekniikan osasto

Esa Nuutinen

PEAP 802.1x-tunnistuksen toteutus RADIUS-palvelimeen

Ohjelmistotekniikan diplomityö

2006

83 sivua, 32 kuvaa ja 3 taulukkoa

Tarkastajat: Professori Heikki Kälviäinen

Diplomi-insinööri Juha Lindström

Avainsanat: RADIUS, palvelin, PEAP, 802.1x, WLAN, käyttäjäntunnistus, lähiverkko

Valimo iDServer -tunnistuspalvelin on ohjelmisto, joka tukee eri käyttäjientunnistusmenetelmiä, kuten tekstiviestillä lähetettävää kertakäyttösalasanaa tai normaalia käyttäjätunnusta ja salasanaa. Tässä diplomityössä kuvataan, kuinka palvelimeen on lisätty tuki käyttäjien kirjautumiselle langattoman verkon tukiasemien ja virtuaalilähiverkkoa tukevien kytkinten kautta käyttäen normaaleja Windows-käyttöjärjestelmän mukana tulevia asiakasohjelmistoja.

Työn ensimmäisessä vaiheessa kuvataan lähtökohdat ja vaatimukset tulevalle järjestelmälle. Työn osana käytännössä tehty kokonaisuus muodostuu useista eri määrityksistä koostuvista osista. Työn toisessa vaiheessa käydään läpi korkealla tasolla sovelluksen vaatimat protokollat. Osana näihin protokolliin kuului erilaisten avainten ja tarkisteiden laskenta sekä salausmenetelmien käyttö, jotka myös kuvataan tässä työssä.

Viimeisessä kappaleessa analysoidaan työn tuloksia ja käydään läpi toteutukseen ja itse sovelluksen toimintaan liittyvät ongelmat. Suurin osa havaituista ongelmista liittyi tilanteisiin, joihin itse palvelinsovelluksen toteutuksella ei voitu vaikuttaa. Eniten ongelmia aiheuttivat asiakasohjelmiston sekä verkkokorttien ja niiden ajureiden toiminta ongelmatilanteissa. Asiakasohjelmistoa ei selkeästi ole suunniteltu käytettäväksi kuin muuttumattomien salasanojen kanssa, koska käyttäjän näkökulmasta käyttökokemus ei ollut optimaalinen.

Ongelmista huolimatta työn tuloksena saatiin asiakkaan vaatimukset täyttävä järjestelmä. Myös tuotekehitysnäkökulmasta projektia voitaneen pitää onnistuneena, koska nyt tehty sovellus luo pohjan uusien tunnistustapojen ja menetelmien toteuttamiselle tuotteen jatkokehitystä ajatellen.

ABSTRACT

Lappeenranta University of Technology

Department of Information Technology

Esa Nuutinen

PEAP 802.1x authentication implementation to RADIUS server

2006

83 pages, 32 figures and 3 tables

Supervisors: Professor Heikki Kälviäinen

Master of Science in Engineering Juha Lindström.

Keywords: RADIUS, server, PEAP, 802.1x, WLAN, user authentication, intranet

Valimo iDServer authentication server is a centralized user management system which supports different methods to authenticate users such as one-time-password and normal username and password. This thesis describes the work that was done to add support for the authentication of users from VLAN- and WLAN switches using standard Windows clients.

The first part of the work describes the existing system as well as customer's requirements for the system that will be built. For the server implementation many different specifications had to be followed and combined to form a single working system. The second part of the work describes these specifications in more detail. In addition, the generation of required cryptographic keys, encryption algorithms and message authenticators that were used in different parts of the software are described in this part of the work.

Also, results of the work are analyzed and possible problems and reasons for them are described. Most of the problems were caused by the Windows client software or network interface drivers and could not be solved on the server side of the system. Different network cards, their respective drivers and client software worked differently in different situations. Also user experience was sub-optimal - it is clear that the Windows network login client was not originally designed to be used with one-time-passwords.

Regardless of the problems encountered, the result of this project is that the Valimo iDServer software was successfully modified to include the new features in a manner that satisfied the customer's needs. Also this can be considered successful from a product development perspective, because the created code and authentication methods can be extended to support other advanced authentication methods with significantly less work.

ALKUSANAT

Tämä diplomityö on tehty 1.3.2006-15.11.2006 välisenä aikana työskennellessäni Valimo Wireless Oy:n palveluksessa. Tahdon kiittää kaikkia työtovereitani, yrityksen omistajia ja esimiehiäni saamastani tuesta. Erityisesti tahdon kiittää yrityksen puolesta työni valvovaa Juha Lindströmiä. Projektipäälliköistä haluan erityisesti kiittää Pyry Heikkistä, joka auttoi ja tuki minua projektin eri vaiheissa aina oikeiden määritysdokumenttien löytämisestä itse sovelluksen testaamiseen. Onnistuneen lopputuloksen kannalta Julius Rintasen apu sovelluksen testaamisessa oli korvaamatonta. Lisäksi haluan kiittää esimiestäni Erkki Tapolaa ja projektipäällikkö Ilkka Pietikäistä, jotka mahdollistivat työtehtävien aikataulutuksella sekä motivoimalla tämän työn tekemisen ja loppuun saattamisen.

Luonnollisesti haluan kiittää työni toista tarkastajaa professori Heikki Kälviäistä. Minulle oli erittäin suuri kunnia saada hänet valvojaksi tälle diplomityölle. Hänen apunsa ja tekemänsä työ on ollut onnistuneen lopputuloksen kannalta korvaamatonta.

Suuret kiitokset myös veljelleni Pasille sekä yrityksen omistajista Juha Mitruselle, sillä ilman heidän apuaan tuskin olisin tätä työpaikkaa koskaan löytänyt tai saanut.

Lisäksi haluan kiittää vanhempiani, kaikkia ystäviäni ja opiskelutovereita kaikesta tuesta ja avusta jota heiltä olen kuluneiden vuosien aikana saanut.

I also want to thank Michael Webster of indispensable support and help I have got from him with this as well as with many other projects while I have been working in Valimo.

And of course thank you, to my loved one, Heide of all support, without you my work would have had no purpose.

Kirkkonummi, Marraskuu 2006
Esa Nuutinen

SISÄLLYSLUETTELO

KÄYTETYT MERKINNÄT JA LYHENTEET

||	Merkkijonojen yhdistämisoperaatio.
⊕	XOR-operaatio
CA	Certificate Authority
CGW	Content Gateway
CIMD	Computer Interface for Message Distribution
CHAP	Challenge-Handshake Authentication Protocol
DH_anon	Diffie Hellman anonymous
DHCP	Dynamic Host Configuration Protocol
DHE_DSS	Ephemeral Diffie Hellman with DSS signature
DHE_RSA	Ephemeral Diffie Hellman with RSA signature
DN	Distinguished Name
EAP	Extensible Authentication Protocol
EAPOL	EAP over LAN
EAP-TLS	EAP Transport Layer Security.
HMAC	Hashed Message Authentication Code
IEEE	Institute of Electrical and Electronics Engineers, Inc.
IETF	Internet Engineering Task Force
IP	Internet Protocol
LAN	IEEE 802 Local Area Network, paikallisverkko
LDAP	Lightweight Directory Access Protocol
MAC	Message Authentication Code
MTU	Maximum Transmission Unit
MD5	Message Digest 5
MSChapV2	Microsoft Challenge Authentication Protocol Version 2
MS-MPPE	Microsoft Point-to-Point Encryption Protocol
NAS	Network Access Server
OTP	One Time Password/kertakäyttösalasana
PAE	Port Access Entity
PEAP	Protected Extensible Authentication Protocol
PPP	Point-to-Point Protocol
RADIUS	Remote Authentication Dial In User Service

RC4	Rivest Cipher 4 virtasalain
RFC	Request for Comments
RSA	Ron Rivest:n, Adi Shamir:n ja Len Adleman:n yhdessä kehittämä julkisen avaimen menetelmiin perustuva salausalgoritmi
SDK	Software Development Kit
SHA	Secure Hash Algorithm
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
VLAN	Virtual LAN, virtuaalinen paikallisverkko
VPN	Virtual Private Network
WLAN	Wireless LAN, langaton paikallisverkko

1 JOHDANTO

1.1 Tausta

Diplomityö on tehty Valimo Wireless Oy:lle osana asiakkaalle toteutettavaa keskitettyä käyttövaltuushallinta- ja käyttäjientunnistusprojektia. Valimo Wireless Oy toimittaa asiakkaalle heidän tarvitsemansa RADIUS-protokollaa tukevan tunnistuspalvelimen, jonka on tarkoitus keskitetysti huolehtia käyttäjien tunnistuksesta riippumatta siitä, mihin asiakkaan palveluun käyttäjät ovat kirjautumassa.

1.2 Tavoitteet ja rajaukset

Työn tarkoituksena on kuvata, suunnitella ja toteuttaa useista eri määrityksistä koostuva asiakkaan vaatimusten mukainen uusimpien Windows-järjestelmien mukana tulevia VLAN- (Virtual Local Area Network) ja WLAN-asiakasohjelmia (Wireless Local Area Network) tukeva järjestelmä. Tunnistustapa on tarkoitus toteuttaa Valimo iDServer -tuoteperheeseen kuuluvaan RADIUS-palvelimeen yhtenä tunnistusmenetelmänä. Se tulee toimimaan osana suurempaa kokonaisuutta, jossa se palvelee asiakkaan verkkoon liittyviä omia ja vierailevia VLAN- ja WLAN-käyttäjiä tarjoten heille mahdollisuuden kirjautua palveluun tekstiviestipohjaista kertakäyttösalasanaa tai normaalia käyttäjätunnusta ja salasanaa käyttäen. Käyttäjien ja käyttöoikeuksien hallinta on projektissa toteutettu erillisillä työkaluilla, mutta tunnistuspalvelimen on osattava välittää nämä tiedot asiakkaan verkossa oleville verkkolaitteille, kuten VPN-palvelimelle, VLAN-reitittimille ja WLAN-tukiasemille.

Käytännössä edellä mainitun onnistuminen edellyttää sitä, että Valimo iDServer -tuoteperheen RADIUS-protokollaa tukevaan tunnistuspalvelimeen lisätään tuki 802.1x-standardin mukaiselle protokollakehykselle, jonka päällä käytetään PEAP-protokollaa (Protected Extended Autentication Protocol). Windows-järjestelmät tukevat PEAP:a käytettäessä MSChapV2-standardin (Microsoft Challenge Authentication Password Version 2) mukaista käyttäjätunnukseen ja salasanaan perustuvaa tunnistusta. Kertakäyttösalasanat eivät suoranaisesti ole Microsoftin

asiakasohjelmissa tuettuja, joten tämä puute on yritettävä toteutuksessa kiertää muilla tavoin.

Työn tarkoituksena on kuvata eri määritysten mukaiset palveluun ja protokolliin liittyvät osat sekä toteuttaa ne käytännössä siten, että ne täyttävät asiakkaan vaatimukset.

1.3 Työn rakenne

Kappaleessa 2 käydään läpi alkutilanne, lähtökohdat, asiakkaan vaatimukset sekä kuvataan järjestelmä korkealla tasolla.

Kappale 3 on työn tärkein teoreettinen osuus. Siinä kuvataan kaikki protokollapinon osat ja se, miten niiden avulla muodostuu tarkoituksenmukainen kokonaisuus. Kappaleessa 4 käydään läpi kryptografiaan liittyvät osat, eri avainten ja tarkisteiden laskeminen sekä käsitellään mahdollisesti tunnettuja tietoturvauhkia ja sitä miten niihin on varauduttu.

Kappaleessa 5 kuvataan itse käytännön toteutus, harkitut vaihtoehdot, käytetyt kirjastot sekä esitellään mahdolliset ongelmat ja niiden ratkaisut. Kappaleessa 6 kerrotaan toteutuksen suomista mahdollisuuksista ohjelmiston jatkokehitystä ajatellen ja viimeinen kappale toimii lyhyenä yhteenvetona itse työstä.

2 LÄHTÖKOHTA JA VAATIMUKSET

Valimo iDServer -tuoteperheeseen kuuluva RADIUS-tunnistuspalvelin on rakennettu Valimo Wireless -yrityksen kehittämän oman perustoiminnallisuudet tarjoavan sovelluspalvelimen varaan. Alusta tarjoaa tietyt peruspalvelut kuten tekstiviestien lähettämisen CIMD2-protokollaa (Computer Interface Message Distribution) [1] käyttäen sekä tuen CCC:n kehittämän CGW (Content Gateway):n [2] kautta lähetettäville tekstiviesteille. Alusta tukee suoraan myös Javan avain- ja varmennetiedostojen käyttöä, joihin voidaan tallentaa esimerkiksi palvelimen varmenteet sekä niihin liittyvät salauksessa käytettävät avaimet.

RADIUS-palvelin tukee jo ennestään lukuisia eri käyttäjientunnistustapoja, kuten normaalit tekstiviestillä välitettävät kertakäyttösalasanat, normaalit käyttäjätunnus ja salasana yhdistelmät sekä digitaaliseen allekirjoitukseen perustuvan menetelmän Valimo Validator-MSSP -tuotteen avulla. Näin on mahdollista myös matkapuhelimen sim-kortille tallennetun varmenteen käyttäminen käyttäjän vahvaan tunnistamiseen. Ohjelmisto on rakenteeltaan sellainen, että se on helposti laajennettavissa tukemaan uusia käyttäjientunnistustapoja.

Palvelimen RADIUS-toiminnallisuudet perustuvat AXL RADIUS Server -kirjaston hyödyntämiseen, joka on lisensoitu AXL Softwarelta. Sitä on muokattu vastaamaan tunnistusmenetelmien asettamia erityisvaatimuksia. [3] Tämä kirjasto ei kuitenkaan tarjoa riittävää tukea EAP-pohjaisen (Extensible Authentication Protocol) tunnistuksen toteuttamiseen. Olemassa oleva järjestelmä on kuitenkin suunniteltu siten, että sen laajentaminen tukemaan uusia menetelmiä on mahdollista.

2.1 Alkutilanne

Käytetty alusta tarjoaa valmiiksi tuen kertakäyttösalasanojen välittämiseen tekstiviestillä käyttäjien matkapuhelimeen eri kirjastojen avulla. Mikään asiakkaan vaatimien laitteiden ja ohjelmien käyttämistä tunnistusmenetelmistä ei kuitenkaan ole projektin alussa toteutettuna tai saatavilla valmiina erillisten kirjastojen avulla.

2.2 Vaatimukset

Asiakkaan vaatimuksissa on esitetty käytettäväksi Ciscon VLAN-kytkimiä, WLAN-tukiasemia, HP SelectAccess -verkkotunnistusta sekä Forten VPN-ohjelmistoa. Palvelimen on lisäksi kyettävä välittämään tietoja, jotka HP SelectAccess tallentaa käyttäjistä käyttäjätietokantana käytettävään LDAP-hakemistoon (Lightweight Directory Access Protocol). Näiden perusteella laitteet rajaavat käyttäjän käytettävissä olevat verkkoyhteydet vain sallittuihin palvelimiin ja verkkoihin. Ciscon VLAN-kytkimiin ja WLAN-tukiasemiin on asiakkaan vaatimusten mukaisesti pystyttävä kytkeytymään käyttäen Microsoft Windows -käyttöjärjestelmien uusimpien versioiden mukana tulevaa asiakasohjelmaa ilman mitään normaalista poikkeavia muutoksia käyttöjärjestelmän asetuksiin. Mitään erillisiä ohjelmia tai asetuksia ei käyttäjien koneille voida tehdä, koska monet WLAN-verkon käyttäjistä ovat rakennuksessa vierailevia henkilöitä. Palvelimeen on lisättävä mahdollisuus lähettää käyttäjän tunnistamiseen tarvittavat attribuutit laitteelle RADIUS-attribuutteina. Axl RADIUS -kirjasto, jota Valimo iDServer hyödyntää, tukee attribuuttien lähetystä, mutta niiden lukeminen hakemistosta ja lisääminen oikeassa muodossa RADIUS-viesteihin ei ole tällä hetkellä mahdollista.

Tuki Windows:n asiakasohjelmistoille edellyttää, että RADIUS-palvelimessa on oltava toteutettuna tuki 802.1x-standardille. Se on toteutettava siten että sen avulla on mahdollista ensin tehdä Microsoftin PEAP-toteutuksen mukainen TLS-kättely (Transport Layer Security), jonka toisessa vaiheessa tehdään tunnistus MSChapV2-protokollalla. Siinä käytetään joko normaalia käyttäjätunnukseen ja salasanaan tai käyttäjätunnukseen ja kertakäyttösalasanaan perustuvaa käyttäjän tunnistusta.

Järjestelmän on oltava vikasietoinen. Käytännössä tämä tarkoittaa sitä, että asiakkaan on hankittava RADIUS-protokollaa ja RADIUS-istuntoa tukeva kuormantasaaja tai turvauduttava edullisempaan vaihtoehtoon järjestelmän toteuttamisessa, joka hyödyntää virtuaalista IP-osoitetta. Sen avulla RADIUS-palvelin voidaan korvata ongelmatilanteessa toisella. Valimo iDServer:n sisäisiä tietoja ei kahdenneta, koska järjestelmästä tulisi kahdennetun tietokannan myötä liian kallis asiakkaalle. Valimo iDServer:n sisäisiä tietoja säilytetään kannassa vain muutamia minuutteja, joten sillä

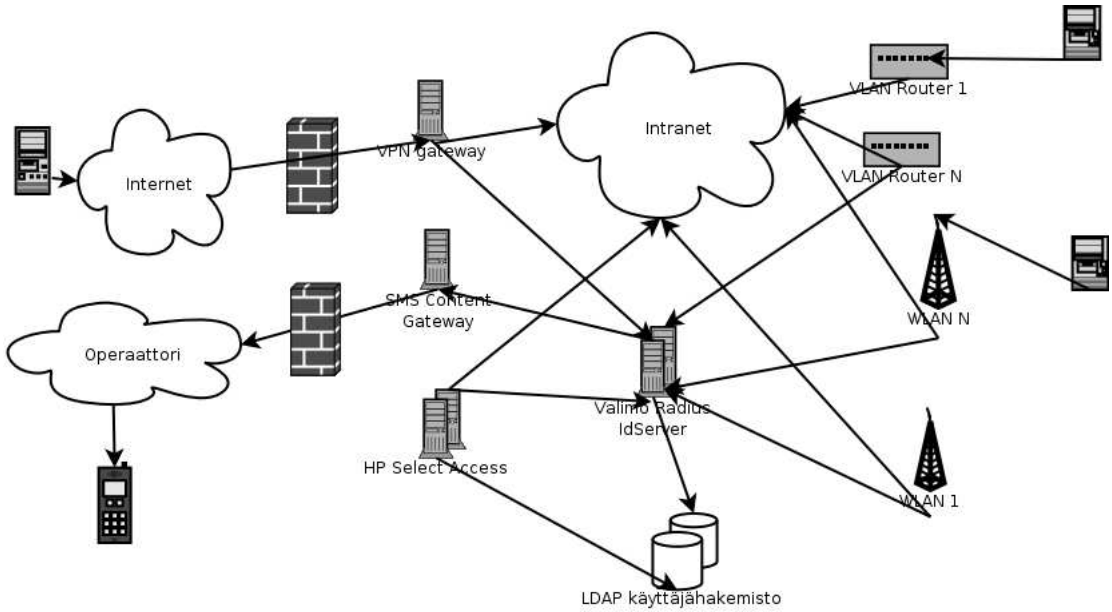
hetkellä käynnissä olevat järjestelmään sisäänkirjautumisyritykset epäonnistuvat, mutta muuta haittaa väliaikaisten tietojen häviämisestä ei ole. Tällaiset tilanteet ovat lisäksi varsin harvinaisia. Näin ollen vikasietoisen tietokannan hankkiminen vain tätä tarkoitusta varten on melko ylimitoitettu vaihtoehto.

Valimo iDServer:n on lisäksi kyettävä tallentamaan tunnistetiedot HP Select Access Audit -palvelimen auditointi tietokantaan, jotta käyttäjien onnistuneista kirjautumisista, epäonnistuneista kirjautumisyrityksistä ja muista tilanteista jää tiedot, joiden avulla esimerkiksi tietomurtojen ja murtoyritysten selvittäminen olisi mahdollista käyttäen HP Select Access Audit -palvelimen omia työkaluja.

2.3 Tavoitejärjestelmä

Asiakkaan toiveiden mukaisessa järjestelmässä HP Select Access huolehtii käyttäjien käyttöoikeuksien hallinnasta. Sen avulla niin vieraileville kuin omille käyttäjillekin voidaan luoda uudet pysyvät tai väliaikaiset käyttäjätunnukset. Näiden avulla käyttäjät pystyvät kirjautumaan yrityksen verkkoon ja saavat pääsyn resursseihin, joihin he ovat oikeutettuja pääsemään käsiksi. Käyttäjien syöttäessä käyttäjätunnuksiaan Valimo iDServer-tuoteperheen RADIUS-palvelin huolehtii käyttäjien tunnistamisesta reitittimien, tukiasemien ja muiden laitteiden kanssa sekä lähettää tarvittaessa kertakäyttösalasanat käyttäjien matkapuhelimiin joko suoralla yhteydellä operaattorin CIMD2-protokollaa tukevaan palveluun tai toisaalla asiakkaan verkossa toimivaan CGW-palveluun.

Kuvassa 1 on esitetty mahdollinen järjestelmä, jossa Valimo iDServer -palvelinta voidaan hyödyntää osana keskitettyä käyttövaltuushallintajärjestelmää. Kuvassa Valimo iDServer huolehtii keskitettyä käyttäjähakemistoa hyödyntäen käyttäjien tunnistamisen tunnistuspyyntöjen tullessa langattoman verkon, VPN:n, HP Select Access ohjelmiston tai VLAN-kytkimien kautta ja sallii näin laitteiden päästävän käyttäjät yrityksen sisäverkkoon. Tekstiviestiyhteydet käyttäjien matkapuhelimiin on toteutettu Content Gateway-ohjelmiston kautta, jonka asiakassovellus on asennettu yrityksen tiloihin.



Kuva 1. Esimerkki mahdollisesta asiakkaan verkosta.

3 PROTOKOLLAPINON KUVAUS

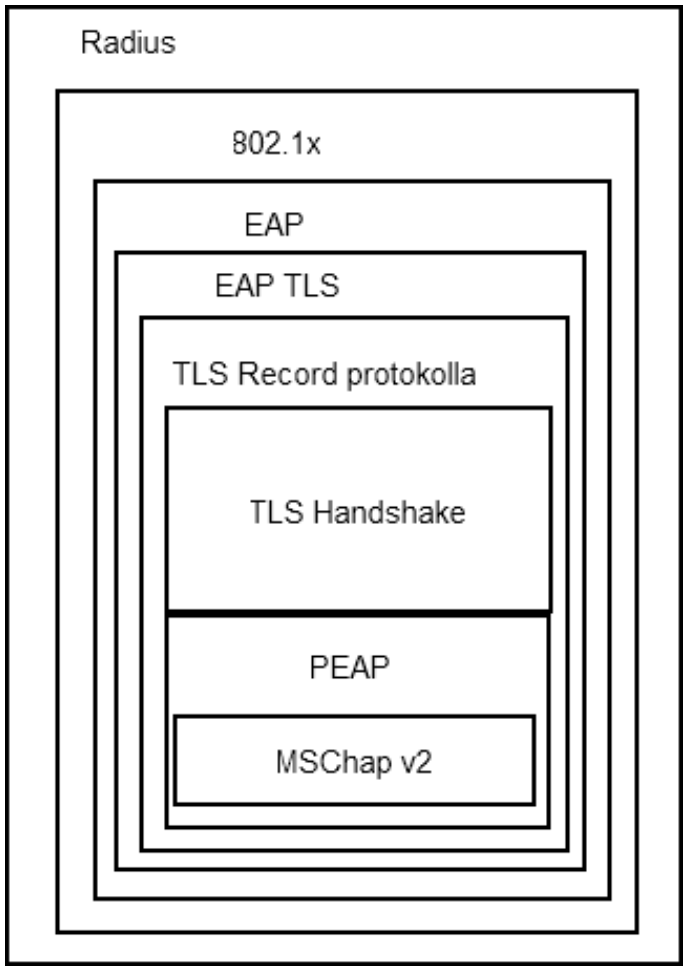
Kehitettävä sovellus vaatii usean eri protokollan toteuttamista, niin että ne yhdessä muodostavat toistensa päälle rakentuneen kokonaisuuden. Lisäksi on toteutettava lukuisia vaadittuja eri protokolliin kuuluvia tarkisteiden ja avaimien laskemiseen tarvittavia menetelmiä. 802.1x-standardi tarjoaa kehyksen, jonka avulla on mahdollista rakentaa erilaisia tunnistusmenetelmiä. Ne toimivat 802.1x-standardia tukevien laitteiden kanssa. Uudet tunnistustavat voidaan toteuttaa muuttamalla palvelinta ja asiakasohjelmaa koskematta itse verkkolaitteeseen.

Protokollapinon rakenne esitellään kuvassa 2. Kuvasta käy ilmi, missä järjestyksessä seuraavissa kappaleissa läpi käytävät eri protokollat rakentuvat toistensa varaan. Pohjalla on siis UDP-pohjainen (User Datagram Protocol) RADIUS-protokolla, jonka varaan muut toiminnot on rakennettu.

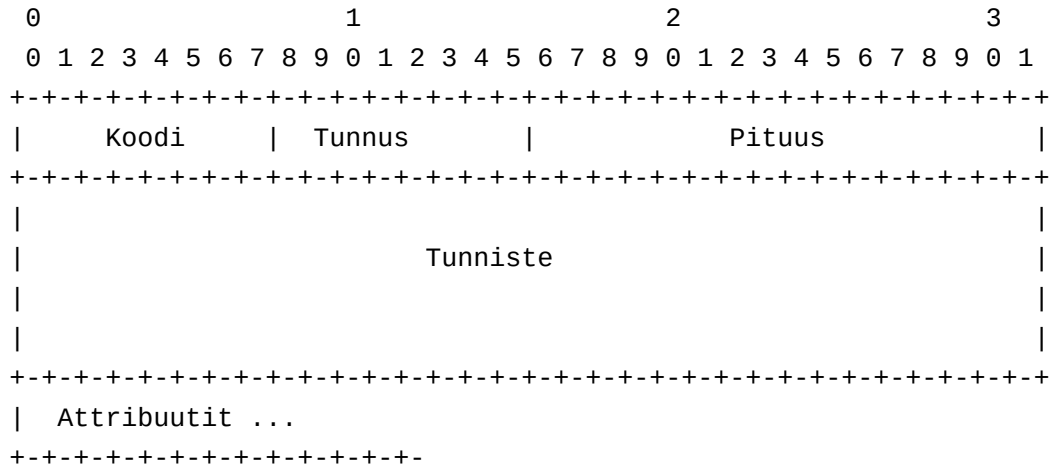
3.1 RADIUS

RADIUS-protokolla on alun perin kehitetty käyttäjien tunnistukseen käyttäjätunnuksen ja salasanan perusteella heidän kirjautuessaan esimerkiksi modeemilla palveluntarjoajan soittosarjaan. Myöhemmin protokolla on otettu käyttöön myös monissa muissa tunnistusta vaativissa verkkopalveluissa. RADIUS-palvelin tunnistaa käyttäjän ja välittää muut tarvittavat tiedot tunnistusta pyytävälle palvelulle. RADIUS-protokolla on rakennettu UDP-protokollan päälle. Protokollassa suojauksesta on huolehdittu käyttäen jaettua salaisuutta. [4]

Protokollassa ovat tuettuna ”Access-request”-, ”Access-challenge”-, ”Access-accept” ja “Access-reject”-viestit. RADIUS-viestin rakenne on esitetty kuvassa 3. Perusrakenteeltaan RADIUS-standardin mukaiset viesti koostuvat koodista, joka kertoo viestin tyypin (request-, challenge-, accept- ja reject-viestit), tunnuksesta jonka avulla voidaan yhdistää pyyntö ja vastaus toisiinsa sekä tiedosta loppuviestin pituudesta. Loppuosa viestistä alkaa RADIUS-viestin oikeellisuuden varmistamiseen käytettävällä arvolla sekä erilaisilla attribuuteilla. [4]



Kuva 2. Protokollapinon rakenne.



Kuva 3. RADIUS-viestin rakenne. [4]

Asiakasohjelmalta tulevissa RADIUS-viesteissä oleva tunnistukseen käytettävä arvo koostuu sattumanvaraisesta 128 bitin mittaisesta bittijonosta. Tätä arvoa hyödynnetään

yhdessä jaetun salaisuuden kanssa käyttäjätunnukseen ja salasanaan perustuvassa tunnistuksessa. 802.1x-standardin mukaisessa tunnistuksessa suojaus on kuitenkin huomattavasti vahvempi ja tätä arvoa käytetään ainoastaan RADIUS-protokollan mukaisen tarkisteen laskemiseen. [4]

Vastausviesteissä tarkisteen arvo koostuu viestin eri tietokentistä lasketusta MD5-tiivisteestä (Message Digest 5). Näin pyritään varmistaa se, ettei kolmas osapuoli pääse muuttamaan RADIUS-viestiä millään tavalla. Kaava muodostuu seuraavasti:

$$\text{tarkiste} = \text{MD5}(\text{Koodi}+\text{Tunnus}+\text{Pituus}+\text{Tunniste}+\text{Attribuutit}+\text{jaettu\_salaisuus}) \quad (1)$$

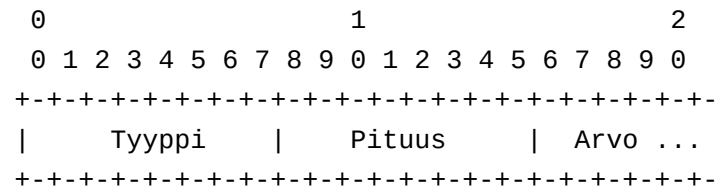
missä eri arvot vastaavat kuvassa 3 esitettyjä RADIUS-viestin kenttiä. Näiden lisäksi kaavassa on mukana jaettu salaisuus, jonka on oltava RADIUS-palvelimen ja tunnistusta pyytävän verkkolaitteen tiedossa. Tämä menetelmä on turvallinen niin pitkään kuin jaettu salaisuus on vain neuvottelevien osapuolien tiedossa. Jos jaettu salaisuus joutuu hyökkääjän tietoon, ei tiivisteen arvoon voida luottaa.

RADIUS-attribuuteissa voidaan välittää kaikkea tarvittavaa tietoa liittyen esimerkiksi käyttäjäntunnistukseen, käyttöoikeuksiin, tiloihin ja salausavaimiin. Näiden samojen attribuuttien avulla välitetään myös RADIUS-protokollassa monimutkaisempien tunnistustapojen vaatimat tiedot. RADIUS-attribuutin rakenne on esitetty kuvassa 4. Jokainen attribuutti muodostuu kolmesta eri kentästä, joista ensimmäinen kertoo attribuutin tyypin, seuraava arvo attribuutin sisällön pituuden ja viimeinen kenttä sisältää itse attribuutin arvon. RADIUS-standardi määrittelee tiettyjen perusattribuuttien sisällön, mutta suurin osa attribuuteista on jätetty avoimeksi myöhemmin kehitettäviä sovelluksia varten. [4]

3.2 802.1x-standardi

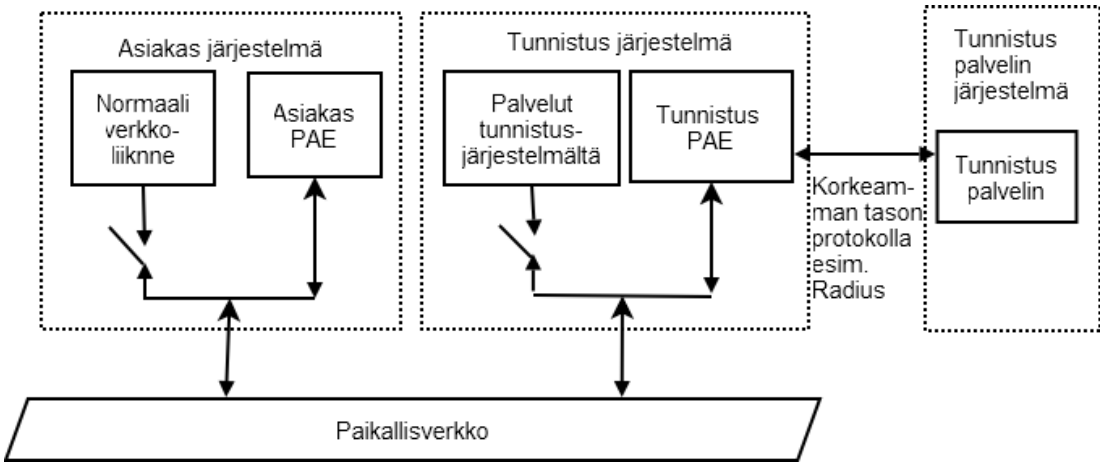
IEEE 802.1x -standardin (Institute of Electrical and Electronics Engineers, Inc.) ajatuksena on tarjota kehys, jonka avulla voidaan toteuttaa käyttäjientunnistus erilaisille verkkolaitteille ja näin estää luvattomien käyttäjien liittyminen esimerkiksi

yrityksen sisäverkkoon. 802.1x-standardi määrittelee lähinnä verkkolaitteen teknisen toiminnan ja sen, että tunnistus toteutetaan käyttäen jotain EAPOL-protokollaa (Extensible Authentication Protocol over LAN). Itse EAP-protokolla on määritelty erillisissä määrityksissä, vaikka 802.1x-standardin mukana on esimerkki EAP-protokollan käytöstä RADIUS-protokollan yli.



Kuva 4. RADIUS-attribuutin rakenne. [4]

802.1x-standardin mukainen verkkolaite toimii kuvassa 5 kuvatun periaatteen mukaan portti- ja protokollatasolla. Laitteiden sisäinen toiminta koostuu kahdesta osasta. Asiakasjärjestelmästä johon käyttäjä kytkeytyy, ja tunnistusjärjestelmästä, joka vastaa järjestelmässä käyttäjäntunnistuksesta. Molemmilla osilla on sekä hallittu että hallitsematon portti. Lyhenne PAE (Port Access Entity) tarkoittaa toiminnallisuutta, joka toteuttaa tunnistusprotokollaan ja portin pääsynhallintaan liittyvät toiminnallisuudet. Käytännössä sillä tarkoitetaan protokollatoteutusta, jonka avulla hoidetaan tunnistus ja siihen liittyvät toiminnallisuudet. Käyttäjä voi asiakasjärjestelmän PAE-portin kautta olla yhteydessä tunnistusjärjestelmän PAE-porttiin ilman rajoituksia. Tunnistusjärjestelmän PAE-portti tarkistaa, onko tunnistukseen liittyviä rajoituksia tai virheitä tapahtunut. Esimerkiksi RADIUS-protokollaa käytettäessä varmistetaan se, että käyttäjän ensimmäinen EAP-viesti on oikean muotoinen. Tämän jälkeen verkkolaite vastaa siihen, mutta seuraavissa viesteissä se toimii ainoastaan tiedon välittäjänä verkkoon kytketyn laitteen sekä käyttäjäntunnistuksesta vastaavan RADIUS-palvelimen välillä, kunnes käyttäjäntunnistusprosessi on suoritettu loppuun asti. Tunnistuspalvelin välittää viimeisessä pyynnössä tiedot tunnistusprosessiin liittyen, jonka perusteella tunnistusjärjestelmä osaa estää tai sallia käyttäjän verkkoliikenteen tai rajata sen vain tiettyihin verkkoresursseihin tai palveluihin vastaanottamiensa attribuuttien mukaisesti. Kuvassa 5 on esitetty standardin määrittelemä malli järjestelmän toiminnasta. [5]



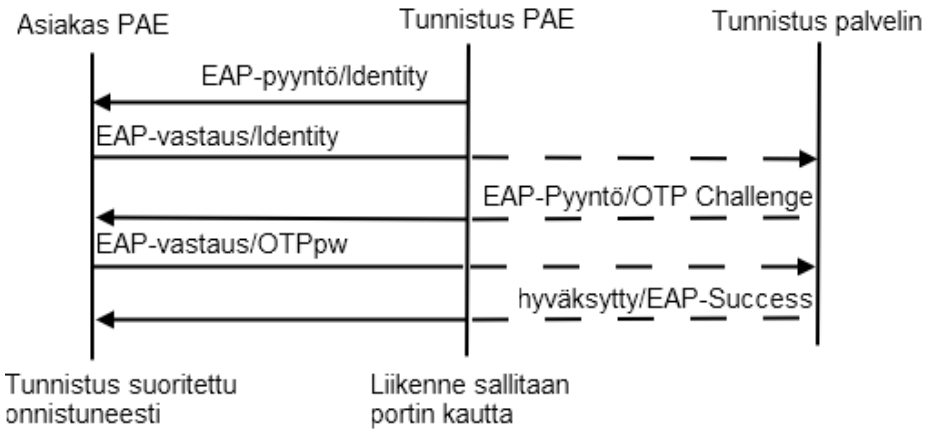
Kuva 5. 802.1x:n toiminta periaate. [5]

802.1x-standardi ei itse ota kantaa minkälaisesta verkkorakenteesta on kyse, mitä protokollaa käytetään käyttäjän tunnistamiseen tai miten itse tunnistus lopulta tapahtuu. Ainut edellytys on, että se tapahtuu käyttäen jotain EAPOL-protokollaa. EAP-standardin mukainen tiedon paketointi on mahdollista toteuttaa lukuisten eri protokollien päälle edellyttäen, että pakettien perusosat ovat ethernet-standardien mukaisia. [5]

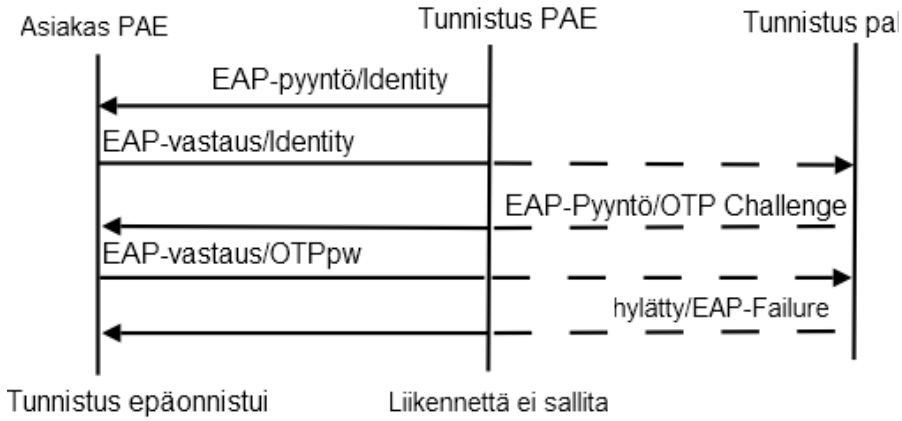
Kuvissa 6 ja 7 esitellään onnistunut ja epäonnistunut 802.1x-standardin mukainen käyttäjäntunnistus, jonka käynnistää verkkolaitteen oman tunnistusjärjestelmän PAE-portti. Tämä vastaa tilannetta, jossa käyttäjä kytkee koneensa liittämällä sen normaalin verkkokaapelin avulla VLAN-kytkimeen. Tällaisessa tapauksessa toki esimerkiksi Windows-käyttöjärjestelmällä varustettu kone pyrkii kytkeytymään verkkoon verkkoasetuksista riippuen käyttämällä DHCP:tä (Dynamic Host Configuration Portocol), mutta se ei osaa automaattisesti käynnistää tunnistusta VLAN-verkkoon. Kun VLAN-kytkin on tunnistanut, että johonkin sen porttiin on kytketty verkkolaite, se käynnistää tunnistusprosessin lähettämällä tälle oikeanlaisen EAP-pyynnön. [5]

Toinen mahdollinen tapaus on esitelty kuvassa 8. Tässä tunnistus on käynnistetty käyttäjän tai käyttäjän koneen käyttöjärjestelmän toimesta. Tämä vastaa tilannetta, jossa käyttäjän kone aloittaa tunnistuksen esimerkiksi langattoman verkon tukiasemaan. Tällöin tunnistustoimenpiteen aloitus ei voi tapahtua tukiaseman

toimesta, koska se ei voi automaattisesti yrittää saada käyttäjien koneita liittymään samaan verkkoon itsensä kanssa. Näin ollen aloituspyynnön on tultava käyttäjältä. VLAN-kytkimessä tilanne on toinen, koska käyttäjän on täytynyt tietoisesti kytkeä verkkokaapeli kytkimeen. [5]



Kuva 6. Onnistunut PAE-portin käynnistämä EAPOL-tunnistus. [5]

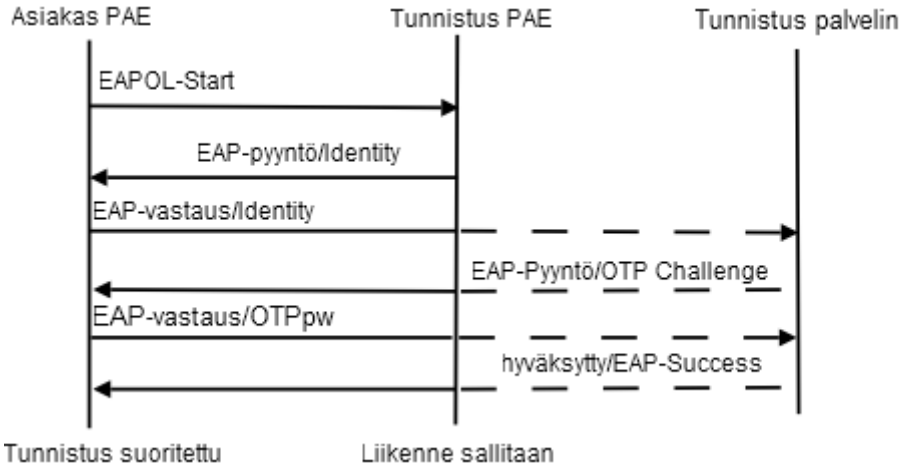


Kuva 7. Epäonnistunut PAE-portin käynnistämä EAPOL-tunnistus. [5]

Kuvista 6-8 käy ilmi, ettei 802.1x-kehys vaikuta millään tavalla itse tunnistuspalvelimen toimintaan, vaan sen näkökulmasta molemmat prosessit vastaavat samaa tilannetta.

Standardi määrittelee myös RC4-EAPOL-avaimen (Rivest Cipher 4) laskemisen. Tunnistuspalvelin laskee avaimet tunnistusprosessin päätteeksi ja välittää ne järjestelmän verkkolaitteelle, johon käyttäjä on liittymässä. Käyttäjän kone laskee

vastaavan avaimen. Näitä käytetään esimerkiksi langattoman verkon liikenteen salaamiseen verkkolaitteen ja käyttäjän koneen välillä. Jotta yhteys voi onnistua, on molempien päiden kyettävä laskemaan oikeat avaimet niiden hallussa olevien tietojen perusteella. Tunnistuspalvelin välittää verkkolaitteelle MS-MPPE-Recv-Key (Microsoft Point-to-Point Encryption Protocol) avaimen. Tämän avulla voidaan avata käyttäjän koneen MS-MPPE-Send-Key avaimella salatut tiedot. Vastaavasti liikenne päinvastaiseen suuntaan salataan tunnistuspalvelimen välittämän MS-MPPE-Send-Key:n avulla, joka käyttäjän koneella voidaan avata sen itse laskeman MS-MPPE-Recv-Key:n avulla. Avaimet muodostetaan EAP-tunnistuksessa syntyneestä avainmateriaalista. [5]



Kuva 8. Käyttäjän käynnistämä onnistunut EAPOL-tunnistusprosessi. [5]

3.2.1 802.1x ja RADIUS

IEEE 802.1x-standardin liitteessä D on kuvattu standardin käyttöä RADIUS-tunnistuspalvelimen kanssa. Tärkeimpiä ovat ohjeistuksessa mainitut valmistajakohtaiset attribuutit sekä IETF (Internet Engineering Task Force) RFC-2866 (Request For Comments) standardin mukaiset tunnistamiseen liittyvät attribuutit, joita VLAN-laitteet yleensä tukevat. Valmistajakohtaisia ja tunnelointiin liittyviä attribuutteja voidaan ja tullaan tässäkin projektissa käyttämään käyttäjien käyttöoikeuksien rajaamiseen ja rajoittamiseen. Tunnistuspalvelin ei ota kantaa näihin attribuutteihin tai niiden sisältöön. Nämä tiedot ovat joko vakioita tai luetaan

käyttäjähakemistosta ja liitetään lähes sellaisinaan mukaan viimeiseen palvelimelta verkkolaitteelle välitettävään viestiin. [5]

3.2.2 Extensible Authentication Protocol (EAP)

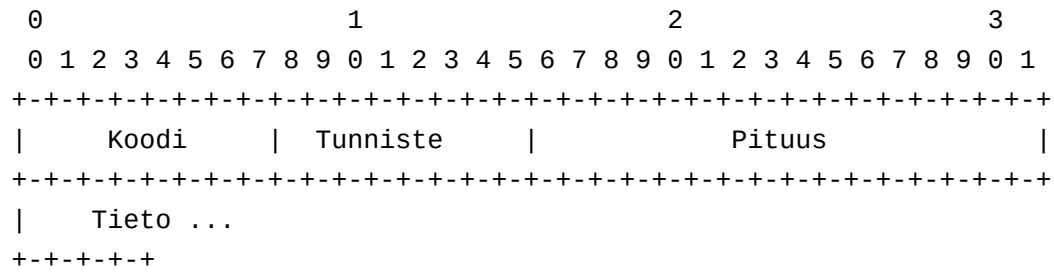
EAP on joustava tunnistusprotokollakehys, joka mahdollistaa vahvemman tunnistamisen toteuttamisen heikomman protokollan sisällä kapseloimalla vaadittu tieto EAP-paketteihin. Ajatuksena on, että edes IP-tason (Internet Protocol) yhteyttä ei vaadita EAP-tunnistukseen. [6]

Normaali EAP-tunnistusprosessi alkaa yleensä siten, että palvelin lähettää käyttäjän järjestelmälle tunnistuspyynnön. Käyttäjä voi aloittaa tunnistuksen, mutta varsinainen EAP-neuvottelun aloituspyyntö tulee tunnistuspalvelimelta. Pyyntö sisältää tiedon siitä, minkä tyyppistä tunnistusta palvelin haluaa. Käyttäjän kone lähettää vastauksen pyyntöön, joka sisältää halutut tiedot. Tämä keskustelu käyttäjän koneen ja tunnistuspalvelun kanssa jatkuu, kunnes käyttäjä on joko tunnistettu onnistuneesti tai käyttäjäntunnistus on todettu epäonnistuneeksi. [6]

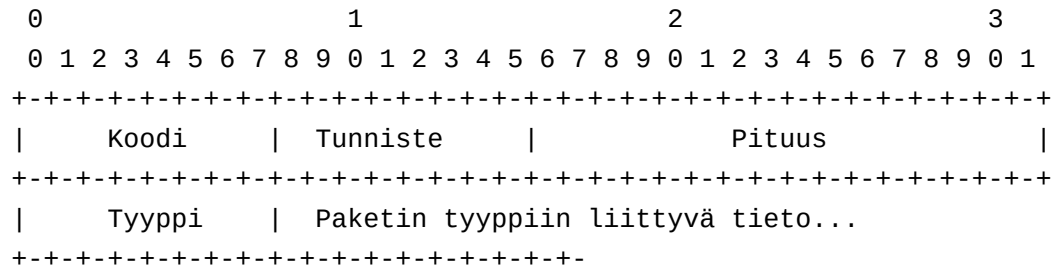
EAP-viestit ovat muodoltaan kuvan 9 mukaisia. Viestejä on 4 eri tyyppiä: pyyntö-, vastaus-, onnistunut- ja epäonnistunut-viestit. Ne ovat hyvin pitkälle identtisiä RADIUS- protokollan viestien kanssa. Viestit sisältävät koodin, joka kertoo viestin tyypin, sekä tunnisteen jonka avulla pyyntö- ja vastausviesti voidaan yhdistää toisiinsa. Pituus-kentässä on määritelty tieto-osan pituus ja itse tieto-kentässä kulkee tunnistustapaan liittyvä varsinainen hyötykuorma. Sen sisältö perustuu muihin standardeihin tai ratkaisuihin, jotka toimivat EAP-tunnistusprotokollan sisällä. [6]

EAP-pyyntö- ja -vastausviestien rakenne on kuvan 10 mukainen. Tieto-osuuden sisältöön on siis lisätty tyyppikenttä. Tämä kuvaa EAP-pyynnön tyyppiä. EAP-standardin mukaisten toteutusten on tuettava Identity-, Notification-, Nak- ja MD5-Challenge-viestejä. Protokolla on laadittu siten, että muut määritykset ja standardit voivat käyttää tätä tyyppi-kenttää omien tunnistenumeroiden avulla. Näin on

mahdollista rakentaa EAP-standardin päälle muitakin tunnistustapoja kuin EAP-standardissa määritelty MD5-haaste. [6]



Kuva 9. EAP-kehyksen rakenne. [6]



Kuva 10. EAP-pyyntö- ja -vastausviestin rakenne. [6]

EAP-standardin mukainen pakollinen Identity-viesti on EAP-protokollassa tunnistusprosessin aloittava viesti. Tämän Identity-pyyntöviestin tunnistuspalvelin lähettää tunnistusta pyytäneelle asiakasohjelmalle prosessin alussa. Tähän vastataan aina Identity-vastausviestillä, sikäli mikäli EAP kuuluu ohjelmiston tukemiin tunnistustapoihin. [6]

Notification-viesti voidaan lähettää käyttäjän ohjelmistolle koska tahansa tunnistus prosessin aikana tunnistuspalvelimelta sillä edellytyksellä, että käyttäjälle ei ole lähetetty mitään muita pyyntöjä, joihin jo odotetaan vastausta. Tähän viestiin vastataan aina Notification-vastausviestillä. Pyyntöön voidaan lisätä ihmisen luettavissa olevaa tekstiä. [6]

Nak-viestiä voidaan käyttää ainoastaan vastausviesteissä. Se lähetetään palvelimelle tunnistusta pyytävän ohjelmiston toimesta, jos palvelimen pyytämä tunnistustapa ei ole tuettu. Käytännössä palvelin siis lähettää EAP-viestejä, kunnes palvelin tarjoaa

asiakasohjelmistolle jotain sen tukemaa tunnistustapaa tai toteaa ettei yhteistä molempien tukemaa tunnistustapaa löydy. [6]

EAP-standardin muut viestit liittyvät itse tunnistusprosessiin. Käytännössä vain muutamia näistä viesteistä on määritelty itse EAP-standardissa ja ne ovat projektin kannalta merkityksettömiä. MD5-challenge-viestiä ei tulla tukemaan, vaikka EAP-standardi niin vaatiikin, koska se olisi huomattavasti toteutettavaa tunnistustapaa heikompi tapa tunnistaa käyttäjiä, eikä toimisi vaaditussa ympäristössä. [6]

3.2.3 EAP RADIUS-protokollan yli

EAP tarjoaa myös RADIUS-palvelimille kehyksen, jonka päälle voidaan rakentaa vaativampia tunnistus tapoja, ilman että jokainen uusi tapa vaatisi muutoksia RADIUS-palvelimeen yhteyttä ottavaan verkkolaitteen (NAS:n). Lisäksi EAP tarjoaa palvelimelle mahdollisuuden tukea useita tunnistustapoja samanaikaisesti, koska EAP:n avulla palvelin ja asiakasohjelma voivat keskenään neuvotella haluamansa tunnistustavan käytöstä ja, jos molempien tukema tapa löytyy, hyödyntää sitä. Tämä tuo paljon joustavuutta kaikkiin tunnistuspalvelimiin. [7]

EAP:n käyttö RADIUS-protokollassa vaatii kahden uuden RADIUS-attribuutin käyttöä: EAP-viesti- ja -viestintarkiste-attribuutteja. RFC 3579 määrittelee, kuinka EAP-viestit voidaan paketoida RADIUS-standardin mukaisten pakettien sisään näiden uusien attribuuttien avulla. [7]

Kommunikointi tapahtuu RADIUS-protokollan sisällä EAP-standardin mukaisesti. RADIUS-protokollan näkökulmasta tunnistusta pyytävälle sovellukselle lähetettävät viestit ovat Access-Challenge-viestejä, joihin se vastaa Access-Request-viesteillä. Nämä viestit sisältävät edellä mainitut EAP-toteutuksen vaatimat RADIUS-attribuutit. Keskustelu jatkuu sovellusten välillä niin pitkään kuin tarvitaan ja päättyy lopulta RADIUS-standardin mukaiseen Access-Accept- tai Access-Reject-viestiin. Asiakasohjelma ei siis saa RADIUS-paketteja koskaan, vaan 802.1x-standardin mukaisesti NAS ja asiakasohjelma kommunikoivat EAP:n avulla ja NAS pakkaa

asiakasohjelman lähettämät EAP-paketit RADIUS-viestien RADIUS-attribuuttien sisään välittääkseen ne tunnistuspalvelimelle. [7]

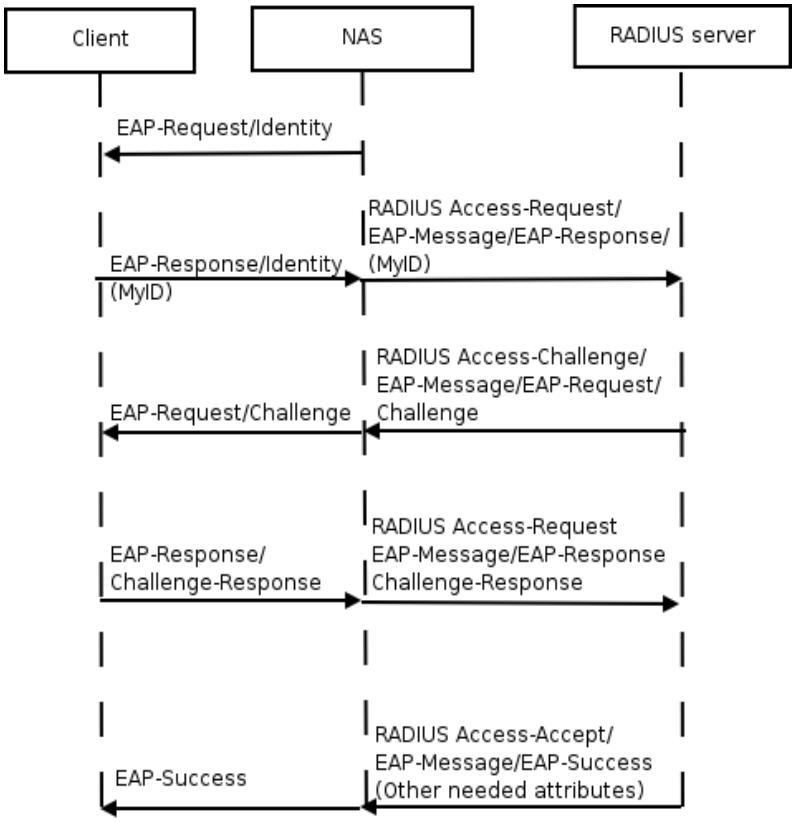
EAP-viestiattribuutti paketoi EAP-viestit siten, että se sallii EAP-tunnistuksen tapahtuvan ulkoisen RADIUS-palvelimen kanssa. NAS toimii vain tiedon välittäjänä, eikä ota kantaa itse EAP-viestien sisältöön. Näin ainut edellytys toiminnalle on, että sekä asiakasohjelmisto että tunnistuspalvelin tukevat samaa tunnistustapaa. Asiakasohjelmisto on osapuoli, joka lähettää ensimmäisen pyynnön tunnistuspalvelimelle EAP:n käyttämisestä. Kommunikaatiossa ensimmäiset EAP-viestit kulkevat ainoastaan verkkolaitteen ja asiakasohjelmiston välillä ensimmäiseen EAP-vastausviestiin asti. Se on ensimmäinen tunnistuspalvelimelle RADIUS-protokollalla välitettävä viesti. [7]

Kuvassa 11 esitellään karkealla tasolla periaate EAP-viestien välityksessä RADIUS-protokollan yli. NAS lähettää ensimmäisenä pyynnön EAP-keskustelun aloittamisesta asiakasohjelmistolle. Kuva esittää vain EAP-osuutta kommunikoinnista yleisellä tasolla, eikä esitä minkään tunnistustavan toimintaa. EAP-viestejä on mahdollista lähettää koneiden välillä niin monta kuin tunnistuksen suorittamiseen niitä on tarvetta lähettää. [7]

NAS välittää saamansa EAP-paketit RADIUS-palvelimelle yhdessä tai useammassa RADIUS-Access-Request-viestin EAP-viesti -attribuutissa. Viesti voi koostua yhdestä tai useammasta EAP-viesti -attribuutista. Jos viesti koostuu useammasta attribuutista, on viestit voitava yhdistää siten, että ne muodostavat yhdistettynä yhden kokonaisen EAP-viestin. [7]

Viestintarkiste-attribuutti on lähes vastaava kuin osana RADIUS-standardia oleva tarkiste. Sen käyttö on pakollista EAP-tunnistuksen yhteydessä, koska on mahdollista, että hyökkääjä voisi päästä käsiksi verkkolaitteen ja tunnistuspalvelimen väliseen viestintään. Käytännössä tämä tarkoittaa sitä, että jos viestiketjun yhtenäisyyttä ei voitaisi varmistaa mitenkään, voisi hyökkääjä yksinkertaisesti vain muokata RADIUS-palvelimen Access-Accept-viestiä sopivalla tavalla voidakseen kirjautua järjestelmään. Asiakasohjelmiston on laskettava Access-Request-viesteihin viestintarkiste, joka on

riippuvainen kaikista aiemmin lähetetyistä viesteistä, sekä tarkistettava tunnistuspalvelimelta tuleva vastaava tarkiste. Näin voidaan varmentua viestiketjun yhtenäisyydestä ja mahdollisuudet ainoastaan viimeisen viestin muokkaamiseen kolmannen osapuolen taholta ovat pienemmät, koska hänen on tunnettava koko viestiketju sekä muuta tietoa pystyäkseen tähän. [7]

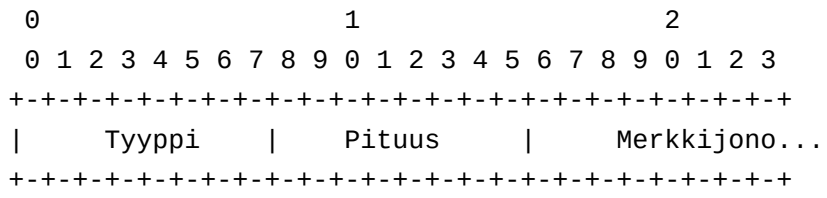


Kuva 11. EAP-keskustelu koneiden välillä. [7]

Kuvassa 12 on esitetty viestintarkiste-attribuutin rakenne, joka vastaa normaalin RADIUS-attribuutin rakennetta. Attribuutin tyyppi kentässä on oltava arvon 80. Attribuutin arvo lasketaan kaavalla:

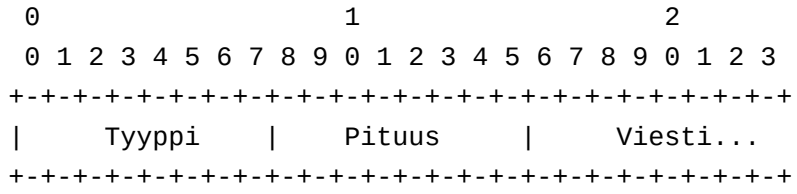
$$MA = \text{HMAC-MD5}(\text{ Tyyppi, tunnistenumero, pituus, RA, attribuutit }) \quad (2)$$

missä, MA on vastausviestin tarkiste ja RA on pyyntöviestin tarkiste. HMAC:n toimintaperiaate on kuvattu tässä työssä kappaleessa 4.4.3. Näin pyritään varmentamaan EAP-viestien muuttumattomuus. [7]



Kuva 12. Viestintarkiste-attribuutin rakenne [7]

EAP-viesti -attribuutin rakenne on varsin yksinkertainen. Se sisältää tyyppi-kentän, pituus-kentän sekä merkkijonon, joka sisältää itse EAP-paketin. Jotta isojen EAP-pakettien lähettäminen olisi mahdollista, voidaan se toteuttaa siten, että viestiin lisätään useampi EAP-viesti -attribuutti, jotka käsiteltäessä yhdistetään yhdeksi kokonaiseksi paketiksi. EAP-viesti -attribuutin rakenne on esitetty kuvassa 13. Se on samanlainen normaalin RADIUS-attribuutin rakennetta. [7]



Kuva 13. EAP-viesti -attribuutin rakenne. [7]

3.2.4 Transport Layer Security (TLS)

TLS-protokollan tarkoituksena on mahdollistaa suojatun yhteyden luonti kahden osapuolen välille, siten että yhteys on salattu eikä viestejä voi muokata kolmannen osapuolen toimesta. TLS-protokolla koostuu kahdesta kerroksesta: TLS Record -protokollasta ja TLS kättely -protokollasta. TLS Record -protokollaa voidaan käyttää tiedonvälitykseen luotettavan protokollan kanssa, jossa TLS Record -protokolla toimii tämän luotettavan protokollan päällä. TLS-protokollassa yhteys salataan symmetrisellä salauksella. Käytetyt salausavaimet ovat uniikkeja ja aina yhteyskohtaisia. Avaimet perustuvat yleensä TLS kättely -protokollan avulla vaihdettuihin ja luotuihin avaimiin. Yhteys on lisäksi aina luotettava, koska viesteistä on laskettu uniikki MAC käyttäen yksisuuntaisia tiivistealgoritmeja. [8]

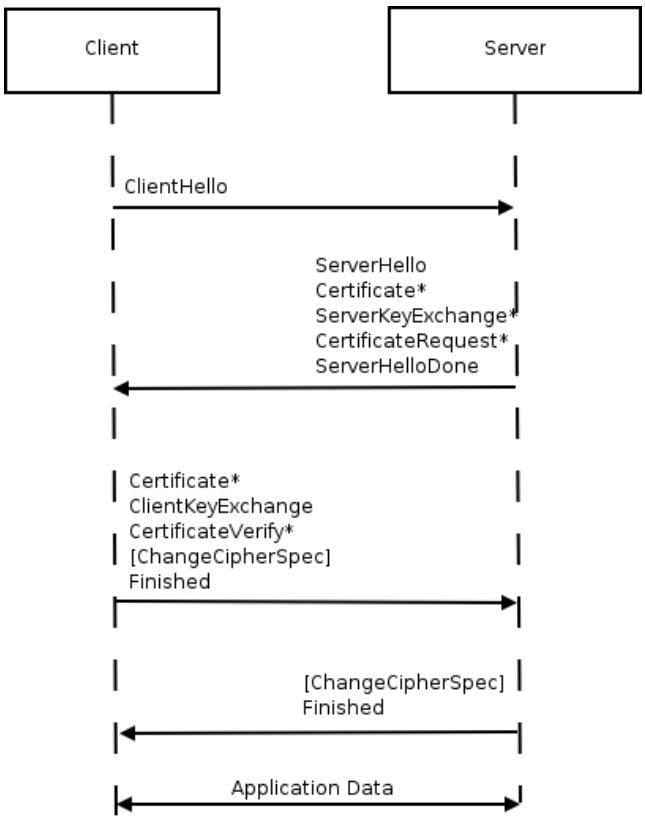
TLS Record -protokollaa käytetään paketoimaan korkeamman tason protokollia. TLS kättely -protokolla on yksi näistä. Tämän protokollan avulla voidaan toimittaa avainten vaihto vahvaa asymmetrisiin salausmenetelmiin perustuvaa salausta käyttäen. Suositeltavaa olisi tunnistaa molemmat osapuolet, mutta RADIUS-palvelimen yhteydessä tullaan käytännön syistä tekemään ainoastaan palvelimen tunnistaminen. Salauksen avulla voidaan neuvotella yhteisesti käytettävä riittävän vahva symmetrinen salausavain muuta tiedonsiirtoa varten. [8]

TLS-protokollan tärkein etu on sen riippumattomuus verkosta ja ohjelmistosta. Tämä tarjoaa mahdollisuuden käyttää sen päälle korkeamman tason protokollia. Näin sen avulla voidaan salata heikommin suojattu tietoliikenne tai tässä tapauksessa suojata heikompi tunnistusmenetelmä vahvalla salauksella. [8]

TLS kättely- protokollaa käytettäessä tunnistus koostuu useasta vaiheesta. Ensimmäisessä vaiheessa sovellukset vaihtavat keskenään Server Hello- ja Client Hello-viestit. Näissä viesteissä siirretään sattumanvarainen tieto, sovitaan käytettävät algoritmit ja tarkistetaan tuki istunnoille. Toisessa vaiheessa vaihdetaan salaukseen tarvittavat tiedot, joiden avulla molemmat voivat laskea tunnetun esisalaisuuden arvon. Tämän jälkeen tapahtuu varmenteiden vaihto, sekä mahdolliset asiakkaan- ja palvelimen tunnistukset. Seuraavaksi lasketaan edellisten arvojen perusteella pääsalaisuus sekä välitetään tarpeelliset tiedot TLS Record -protokollalle. Lisäksi varmistetaan, että molemmat osapuolet ovat saaneet samat tiedot ja laskeneet tarkisteet oikein, niin ettei kolmas osapuoli pääse puuttumaan sovellusten väliseen kommunikointiin. [8]

Kuvassa 14 esitetään normaali TLS-kättely. Neuvottelu alkaa Client Hello -viestin lähettämisellä. Tähän palvelin vastaa Server Hello -viestillä sekä välittää joko palvelimen varmenteen Certificate-viestillä tai vaihtoehtoisesti tarvittavat avaimet Server Key Exchange -viestillä. Jos asiakasohjelmiston tunnistusta vaaditaan, voidaan se tehdä Certificate Request -viestin avulla. Viestiketju päättyy Server Hello Done -viestiin. [8]

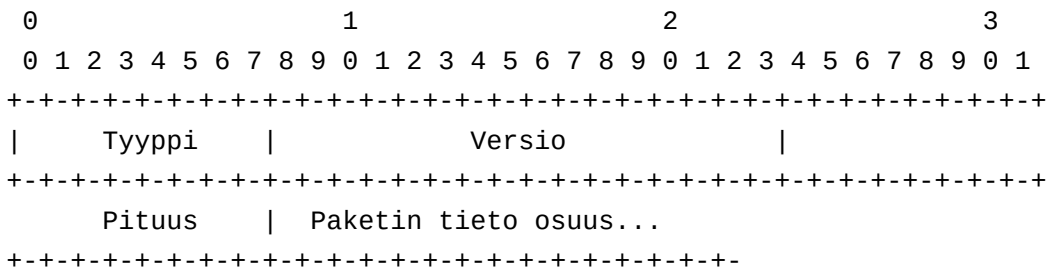
Seuraavassa vaiheessa asiakasohjelma tarkistaa palvelimen varmenteen niin halutessaan sekä vastaa Client Key Exchange -viestillä. Tarvittaessa on mahdollista lähettää myös Certificate- ja Certificate Verify -viestit. Certificate-viesti vaaditaan lähetettäväksi, jos palvelimelta on vastaanotettu Certificate Request -viesti. Edellä mainittujen lisäksi vastaukseen liitetään Change Cipher Spec -viesti, joka toimii merkkinä salatun keskustelun aloittamiselle. Viesti ei liity TLS kättely-protokollaan vaan on osa TLS Record -protokollaa. Finished-viesti on ensimmäinen asiakasohjelman lähettämä symmetrisellä salauksella salattu ja MAC:lla varmennettu viesti. Seuraavassa vaiheessa palvelin avaa asiakasohjelman salaaman viestin ja lähettää oman Change Cipher Spec -viestin sekä salatun Finished-viestin. Jos sekä palvelin että asiakasohjelma saavat viestin salauksen purettua ja tarkistettua MAC:n avulla sen oikeellisuuden, on kättely tapahtunut onnistuneesti. Näin asiakasohjelma ja palvelin voivat jatkaa tietojen vaihtoa salatun yhteyden yli. [8]



Kuva 14. TLS kättely -protokolla. [8]

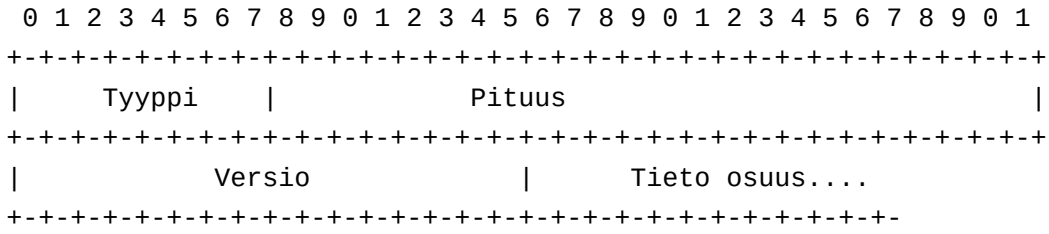
TLS Record -protokollan viesti ovat rakenteeltaan kuvan 15 mukaisia. Paketti koostuu tyyppitiedosta, joka sisältää TLS Record -protokollapaketin sisällä olevan paketin

tyypin, esimerkiksi TLS kättely -protokollaan kuuluvan paketin, versionumeron TLS 1.0 (0x0301) sekä viestin pituus -kentän. Se kuvaa viestin varsinaisen tieto-osuuden pituutta, esim. Client Hello -viestin pituutta. [8]



Kuva 15. TLS Record -protokollan viestin rakenne. [8]

TLS kättely -protokollan tieto siirretään TLS Record -protokollan tieto-osuudessa. Kuvassa 16 esitetään TLS Handshake -protokollan viestin perusrakenne. Se muodostuu TLS kättely -viestin tyypistä, viestin pituudesta, versionumerosta TLS 1.0 (0x0301) sekä tieto-osuudesta, jonka sisältö vaihtelee riippuen mistä viestistä on kyse. [8]



Kuva 16. TLS kättely-viestin rakenne. [8]

Client Hello-viesti

TLS-kättelyn Client Hello -viestin varsinainen tietosisältö koostuu taulukon 1 mukaisista tiedoista. Aikaleimaa ja sattumanvaraista tietoa käytetään avainten laskentaan tarvittavan esisalaisuuden laskemiseen myöhemmissä viesteissä vaihdettavan informaation lisäksi. Istunnon tunnus -kentässä vastaanotetaan asiakasohjelmistolta tunnus, jos se yrittää palata jo olemassa olevaan istuntoon. Kenttä on tyhjä, jos istuntotunnusta ei ole luotu tai asiakasohjelma haluaa luotavaksi uuden istunnon. Salausmenetelmät-kentässä kerrotaan, mitä salaustapoja asiakasohjelma

tukee. Näistä ensimmäisenä listataan salaus, jota halutaan tai suositellaan käytettäväksi. Pakkausmenetelmät-listassa kerrotaan vastaavasti pakkaustavat, jotka ovat asiakasohjelman tukemia. Kaikkien toteutusten on standardin mukaan tuettava vähintään pakkaamatonta kommunikaatiota. [8]

Taulukko 1. TLS kättely -protokollan Client Hello -viestin tietokentät [8]

Nimi	**Tyyppi**	**Pituus (tavua)**	**Pakollisuus**	**Määrä**	**Esimerkki**
Aika	Unix TimeStamp	4	Kyllä	1	Nov 24, 2005 20:00:59.000000000 (0x4385ffdb)
Sattuman-varainen tieto		28	Kyllä	1	
Istunnon tunnus --kentän pituus	Pituus	1	Kyllä, vähintään 0	1	Ei viestiä(0x00)
Istunnon tunnus		N	Ei, jos pituus 0.	1	
Salausmenetel-mät -listan pituus	Pituus	1	Kyllä	1	
Salausmenetel-mät	Salaus-algortimin tunniste	2	Kyllä	1-N	TLS_RSA_WITH_RC4_128_MD5 (0x0004)
Pakkausmentel-mät -listan pituus	Pituus	1	Kyllä, vähintään 0	1	
Pakkausmene-telmät		N	Ei, jos pituus 0.	0-N	

Server Hello-viesti

Kommunikaatiossa seuraavana on Server Hello -viesti, joka on kuvattu taulukossa 2. Se on palvelimen vastaus TLS-kättelyn Client Hello -viestiin. Käytännössä tässä

viestissä kerrotaan mikä asiakasohjelman tukemista salaus- ja pakkaustavoista valitaan käytettäväksi. Jos palvelin ei hyväksy mitään asiakasohjelman tukemista menetelmistä, vastataan TLS-kättely epäonnistui -viestillä. Server Hello -viesti lähetetään kuvan 15 mukaisen TLS Record-protokollan viestin tieto-osan sisällä. Viesti sisältää aikaleimasta ja sattumanvaraisesta tiedoista koostuvan osuuden aivan kuten Client Hello -viestikin. Viestin sattumanvaraisen tiedon on oltava satunnainen, itsenäisesti valittu sekä poikettava asiakasohjelman muodostamasta vastaavasta tiedosta. [8]

Istuntotunnuksen osalta palvelimen vastaus riippuu ratkaisevasti siitä, antoiko asiakasohjelma Client Hello -viestissä jonkin istuntotunnuksen. Jos viestissä tuli tunnus, tarkoittaa se sitä, että asiakasohjelma haluaa palata vanhaan istuntoon. Näin ollen palvelimen on toimittava sen mukaan ja pyrittävä jatkamaan tunnistusprosessia yrittämällä etsiä vanhan istunnon tiedot, käyttää niitä uudestaan ja yrittää niiden avulla lähettää Finished-viesti suoraan vanhojen jo neuvoteltujen salausavainten avulla. Jos palvelin palauttaa istuntotunnuksen tyhjänä tarkoittaa se sitä, ettei istuntoa säilötä, eikä nyt luotavaan istuntoon palaaminen asiakasohjelman toimesta ole mitenkään mahdollista. Salausmenetelmä- ja pakkausmenetelmä-kentissä palvelin palauttaa asiakasohjelmistolle hyväksymänsä tai valitsemansa salaus- ja pakkausmenetelmät, jotka on valittu asiakasohjelman aiemmin tarjoamista vaihtoehdoista. [8]

Server Certificate -viesti

TLS-kättelyssä palvelimen on lähetettävä varmenne käyttäjälle kaikissa tapauksissa joissa käytetty tunnistusmenetelmä ei ole niin sanottu anonyymi menetelmä. Tämä viesti seuraa välittömästi palvelimen lähettämän Server Hello -viestin jälkeen. Varmenteen on oltava asianmukainen ja sovelluttava valitun salausmenetelmän käyttöön. Standardissa varmenteen oletetaan olevan tyypiltään X.509v3-standardin mukainen. [8]

Viesti koostuu varmenneketjusta, jossa ensimmäinen on lähettäjän oma varmenne. Seuraavat varmenteet ovat varmenneketjuun liittyviä varmenteita. Niitä tarvitaan seurattaessa palvelimen varmennetta kohti juurivarmennetta. Itse juurivarmenteen välittäminen ei ole välttämätöntä palvelimen varmenteen ollessa yleisesti tunnetun

tahon allekirjoittama ja jos on oletettavissa että asiakasohjelmistolla on se jo hallussaan. [8]

Taulukko 2. TLS kättely -protokollan Server Hello -viestin tietokentät [8]

Nimi	**Tyyppi**	**Pituus (tavua)**	**Pakollisuus**	**Määrä**	**Esimerkki**
Aika	Unix-TimeStamp	4	Kyllä	1	Nov 24, 2005 20:00:59.000000000 (0x4385ffdb)
Sattumanvarainen tieto		28	Kyllä	1	
Istuntotunnuksen pituus	Pituus	1	Kyllä, vähintään 0	1	Ei viestiä(0x00)
Istuntotunnus		N	Ei, jos pituus 0.	1	
Salausmenetelmä	Salaus-algortimin tunniste	2	Kyllä	1	TLS_RSA_WITH _RC4_128_MD5 (0x0004)
Pakkausmenetel-mä		N	Kyllä.	1	Null (0x00)

Server Key Exchange -viesti

Anonyymissä menetelmässä välitetään Server Key Exchange -viesti välittömästi Server Hello -viestin jälkeen. Muissa tapauksissa viestin lähetys on vaihtoehtoista. Viestiä tarvitaan jos palvelimen Certificate-viesti ei sisällä tarpeeksi tietoa esisalaisuuden välittämistä varten. Esimerkiksi DHE_DSS, DHE_DSS_EXPORT, DHE_RSA, DHE_RSA_EXPORT, DH_anon sekä RSA_Export (jos avaimen pituus

on yli 512 bittiä) ovat tällaisia menetelmiä. Tämä viesti on tarpeeton toteutettavaa sovellusta ajatellen käytettävästä salausalgoritmista johtuen. [8]

Certificate Request -viesti

Muissa kuin anonyymeissä menetelmissä palvelin voi vaatia asiakasohjelmaa välittämään tunnistusta varten oman varmenteensa. Tämä tapahtuu Certificate Request-viestin avulla. Viesti sisältää hyväksyttyjen varmenteiden myöntäjien (CA) yksikäsitteiset nimet (DN), joiden perusteella asiakasohjelma voi etsiä omista tiedoistaan vastaavan varmenteiden myöntäjän allekirjoittaman asiakasvarmenteen. Myös tämä viesti on tarpeeton toteutettavaa menetelmää ajatellen, koska tunnistus tapahtuu anonyymillä menetelmällä jossa asiakasohjelmistoa ei tunnisteta varmenteen, vaan salasanan ja käyttäjätunnuksen avulla. [8]

Server Hello Done -viesti

Server Hello Done -viesti lähetetään asiakasohjelmistolle viimeisenä viestinä Server Hello -viestin ja muiden kättelyprosessin vaatimien viestien jälkeen. Kun tämä viesti on lähetetty, palvelin jää odottamaan asiakasohjelmiston vastausta. Viesti toimii merkkinä siitä, että kaikki viestit mitä palvelin tässä kättelyn vaiheessa tulee lähettämään, on lähetetty. [8]

Client Certificate -viesti

Client Certificate -viesti on ensimmäinen asiakasohjelman palvelimelle lähettämistä vastausviesteistä. Se lähetetään ainoastaan, jos palvelin on lähettänyt Certificate Request -viestin ja vastaava varmenne on löytynyt. Siitä huolimatta, että palvelin on pyytänyt asiakasohjelmalta varmennetta, on edelleen olemassa mahdollisuus anonyymiin tunnistukseen. Asiakasohjelma voi vastata tyhjällä Client Certificate -viestillä, mikä käytännössä tarkoittaa sitä, että vastaavaa varmennetta ei löytynyt. Tunnistus voi edelleen jatkua, jos palvelin hyväksyy sekä anonyymit että varmenteisiin perustuvan tunnistuksen. Jos palvelin hyväksyy ainoastaan varmenteisiin perustuvan tunnistuksen, pitää palvelimen todeta tunnistuksen epäonnistuneen tässä vaiheessa. Projektissa ei tulla toteuttamaan varmenteisiin perustuvaa tunnistusta, vaan toteutettava

ratkaisu tulee käyttämään anonyymiä tunnistusta. Näin tätä viestiä ei tulla tarvitsemaan nyt toteutettavassa ratkaisussa. [8]

Client Key Exchange -viesti

Client Key Exchange -viesti on aina pakollinen viesti asiakasohjelmalta. Jos se ei lähetä Certificate -viestiä, Client Key Exchange -viesti on asiakasohjelman ensimmäinen palvelimelle lähettämä vastausviesti. Itse viestin sisältö on riippuvainen valitusta salausalgoritmista. RSA:ta käytettäessä asiakasohjelma luo 48 tavua pitkän esisalaisuuden, joka koostuu versionumerosta sekä jaetusta salaisuudesta. Viesti salataan joko palvelimen varmenteesta saadulla julkisella avaimella tai väliaikaisella Server Key Exchange -viestistä saadulla avaimella. Asiakasohjelma muodostaa esisalaisuuden, jonka avulla on mahdollista laskea pääsalaisuuden arvo. RSA:ta (Rivest:n, Shamir ja Adleman) tullaan käyttämään viestien salausmenetelmänä nyt tehtävässä toteutuksessa. Diffie Hellman -menetelmää käytettäessä tämän viestin sijasta lähetetään julkisen avaimen arvo tai tyhjä viesti, jos asiakasohjelmiston lähettämä varmenne sisälsi jo tarkoituksenmukaisen arvon. [8]

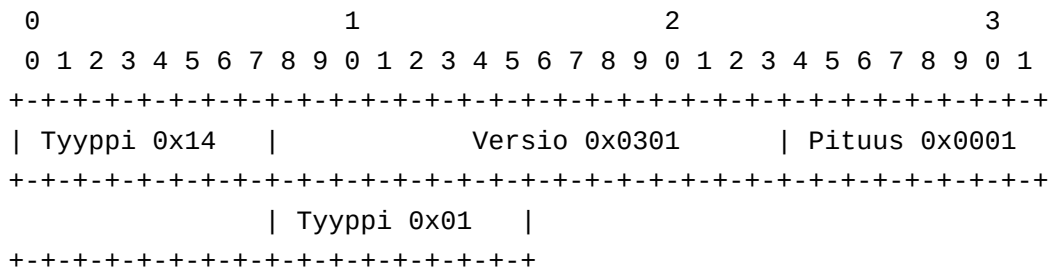
Certificate Verify -viesti

Certificate Verify -viestin tarkoituksena on lähettää käyttäjän varmenteella allekirjoitettu viesti, jos varmennetta voi käyttää allekirjoittamiseen. Allekirjoitettava viesti on tiiviste-arvo, joka on laskettu kaikista edellisistä viesteistä. Anonyymiä tunnistusta käytettäessä viestiä ei tarvita. [8]

Change Cipher Spec -viesti

Change Cipher Spec-viesti ei kuulu TLS kättely -protokollan viesteihin, vaan on TLS Record Layer -protokollan viestejä. Tämä on merkki toiselle osapuolelle siitä, että kaikki tämän jälkeen tulevat viestit ovat salattuja käyttäen valittua symmetristä salausta. Viesti on joka kerta samaa muotoa. Se on TLS Record Layer -viesti oikeilla

tunnisteilla ja dataosassa arvolla 0x01 varustettuna. Viesti kokonaisuudessaan on kuvattu kuvassa 17. Sekä asiakasohjelma että palvelin lähettävät samanlaiset viestit ennen ensimmäisen salatun viestin lähettämistä. [8]



Kuva 17. Change Cipher Spec -viesti.

Finished-viesti

Finished-viesti on sekä asiakasohjelman että palvelimen lähettämä ensimmäinen salattu viesti. Se muodostetaan TLS-standardin mukaan aikaisempien tietojen perusteella laskemalla kaavan 3 mukainen tarkiste luku joka salataan. Jos palvelin laskee saman tarkisteen omien tietojensa perusteella, kuin se on vastaanottanut asiakasohjelmistolta salattuna, on avainten vaihto onnistunut. Palvelimen on laskettava vastaava tarkiste kuin asiakasohjelman, johon mukaan lasketaan myös juuri vastaanotetut viestit ja lähettää sen takasin asiakasohjelmistolle onnistuneen kättelyn merkiksi. Finished-viestin tarkoitus on testata käyttävän salauksen toiminta, sekä varmistaa viestiketjun yhtenäisyys. Kaava on seuraava:

$$VD = PRF(\text{ pääsalaisuus, otsikko, } MD5(HM) + SHA1(HM)) \qquad (3)$$

missä VD tarkoittaa laskettua tarkistetta. HM tarkoittaa kaavassa kaikkia TLS kättely -protokollan viestejä, jotka siihen mennessä on lähetetty. Change Cipher Spec -viesti ei siis kuulu TLS kättely -protokollan viesteihin. Otsikkona kaavassa käytetään ”client finished” -tekstiä, kun tarkiste on asiakasohjelman lähettämä ja ”server finished” -tekstiä kun se on palvelimen lähettämä. TLS-standardin mukaan PRF-funktiolla saadusta arvosta VD otetaan 12 ensimmäistä tavua, jotka välitetään Finished-viestissä

toiselle osapuolelle tarkistettavaksi. PRF-funktion toiminta esitellään tässä työssä kappaleessa 4.3. [8]

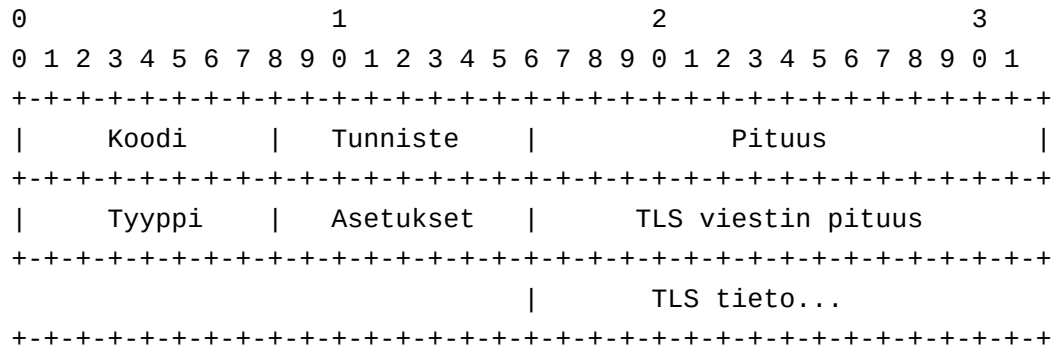
3.2.5 EAP-TLS-standardi

EAP-standardi määrittelee kehyksen sille, miten on mahdollista tehdä uusia tunnistus menetelmiä muuttamatta itse laitteita, jotka vaativat käyttäjän tunnistusta. TLS vastaavasti on tarkoitettu suojatun yhteyden ja kättelyn suorittamiseen yhteyden yli, jossa siirrettävien pakettien ei oleteta hajoavan tai niitä ei ole tarvetta jakaa pienempiin osiin. EAP-TLS-standardi määrittelee sen, kuinka TLS-viestit voidaan tarvittaessa pilkkoa pienempiin osiin ja koota jälleen vastaanottajan päässä yhdeksi kokonaiseksi viestiksi. RADIUS-palvelin voi lähettää EAP-paketin, jonka koko on asiakasohjelman ja NAS:n välillä käytätettävästä siirtotiestä riippuva Framed-MTU (Maximum Transmission Unit) -arvo vähennettynä neljällä tavulla. Neljä tavua on EAP-kehyksen vaatima ylimääräisen tiedon osuus jokaista välitettyä pakettia kohden. RADIUS-protokolla ei lisää tätä kuormaa, koska RADIUS-paketit kulkevat vain NAS:n ja tunnistuspalvelimen välillä. RADIUS-protokollassa pakettien koko on rajoitettu 4096 tavuun. Näin ollen EAP:a käytettäessä attribuutit voidaan joutua jakamaan myös useampaan RADIUS-viestiin, joista yksi RADIUS-viesti voi siis sisältää useita EAP-paketin sisältäviä RADIUS-attribuutteja. Pakettien jakamiseen ja yhdistämiseen tarjoaa ratkaisun EAP-TLS-määrittelyn mukainen toteutus. [9]

Ratkaisu mahdollistaa EAP-viestien lähettämisen useissa paketeissa tai attribuuteissa, jotka kootaan yhdeksi viestiksi. TLS-kättelyn Certificate-viestit voivat teoriassa olla jopa useiden megatavujen kokoisia, joten ne ylittävät helposti RADIUS-protokollan ja 802.1x-standardin pakettikoolle asetetut rajoitukset. Ratkaisuna on lisätä asetus-kenttä, joka kertoo mikä paketti on kyseessä, onko se EAP-TLS-keskustelun ainut paketti vai osa useamman paketin muodostamaa kokonaista EAP-viestiä. [9]

Kuvassa 18 esitetään EAP-TLS-paketin rakenne. Ensimmäisenä kenttänä on koodi, joka kertoo onko kyseessä pyyntö- vai vastaus-viesti. Tunniste-kenttä kertoo paketin yksilöivän tunnisteen, joka siis kasvaa jokaisen pyyntö-viestin myötä. Samaa numeroa

ei saa käyttää kahdessa erillisessä pyynnössä. Kolmanteen kenttään lasketaan viestin pituus. Tähän arvoon lasketaan mukaan EAP-TLS-viesti kokonaisuutenaan. Tyyppi-kenttään tulee viestin tyyppi. Asetukset-kenttään voidaan tallentaa tieto siitä, onko kyseessä koko viesti vai pienempiin osiin pilkottu EAP-TLS-viesti, jotka on myöhemmin koottava yhdeksi kokonaisuudeksi. Viimeisenä syötetään TLS-viestin pituus, jota seuraa varsinainen tietosisältö. [9]



Kuva 18. EAP-TLS-viestin rakenne. [9]

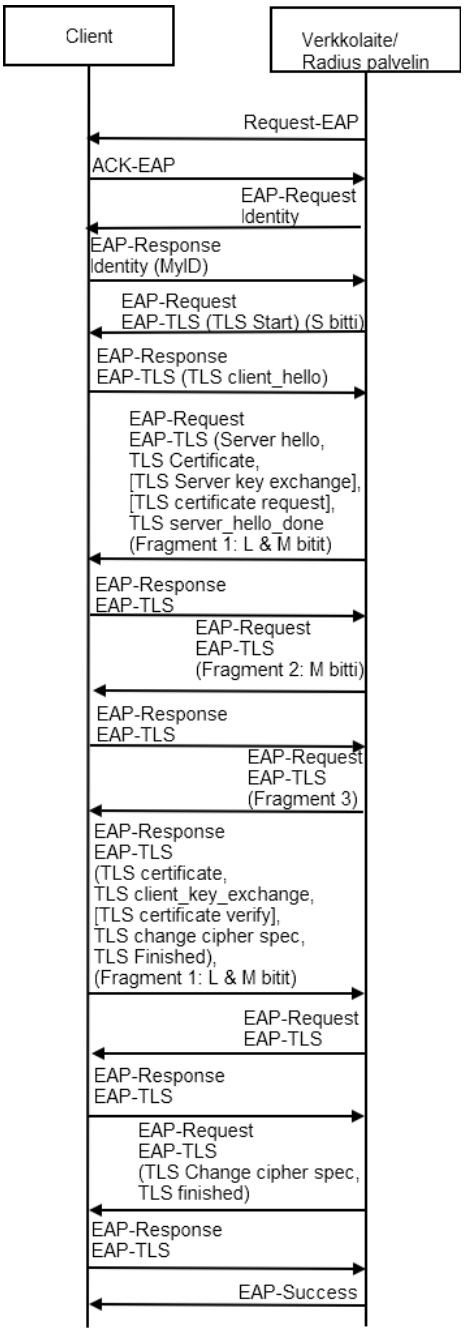
EAP-TLS-viesti sisältää yhden tavun mittaisen kuvassa 18 esiintyvän asetukset-kentän. Kuvassa 19 sen sisältö on esitetty tarkemmin. Kentän sisällöstä tällä hetkellä käytetään ainoastaan 3 ensimmäistä bittiä. Ensimmäinen bitti L kuvaa sitä, onko viestissä mukana 4 tavun mittainen pituus-kenttä. Tämä bitti pitää olla päällä ensimmäisessä viestissä, joka sisältää pituus-kentän. Toisena oleva bitti M on oltava asetettuna päälle kaikissa viestin osissa paitsi viimeisessä. Kolmas bitti S asetetaan päälle ensimmäisessä viestissä, jotta se voidaan helposti tunnistaa ensimmäiseksi viestin osista. [9]

Kuvassa 20 on esitetty EAP-TLS-standardin mukainen kättelyprosessi, jossa itse EAP-viestit on tarvittaessa hajotettu pienempiin viesteihin. EAP-TLS:n avulla tapahtuva kättelyprosessi etenee tässä työssä aiemmin esitellyn TLS-kättelyn mukaisesti sisältäen vastaavat ja vastaavissa tilanteissa lähetetyt TLS-viestit ainoastaan EAP-TLS-pyyntö- ja -vastausviesteihin paketoituna. Osiin jaettu EAP-TLS-viesti kootaan vastaanottavassa päässä ja käsitellään yhtenä viestinä, vaikka alemmalla tasolla se on pienempiin osiin jaettu viesti. Tämä operaatio on täysin näkymätön tunnistusta suorittavalle sovellukselle. Muilta osin malli vastaa jo aiemmin esiteltyjä malleja.

Kuvassa itse tunnistuspalvelimen ja verkkolaitteen toiminta on esitetty yhtenä kokonaisuutena. [9]

```
0 1 2 3 4 5 6 7 8
+-+-+-+-+-+-+-+-+
|L M S R R R R R|
+-+-+-+-+-+-+-+-+
```

Kuva 19. Asetukset-kentän sisältö EAP-TLS-viestissä. [9]



Kuva 20. EAP-TLS-keskustelu, jossa TLS-kättelyn viestit on jaettu useampiin EAP-viesteihin. [9]

3.2.6 Protected EAP (PEAP)

PEAP on kehitetty esimerkiksi langattomia verkkoja varten, missä hyökkääjällä on mahdollisuus päästä käsiksi verkossa liikkuvaan dataan. Se pyrkii ehkäisemään heikkouksia, jotka johtuvat siitä, että normaalissa EAP-käyttelyssä varmenteita ja muuta tunnistamisen kannalta kriittistä tietoa siirretään selväkielisenä. PEAP mahdollistaa istuntojen käytön, avainten vaihdon sekä pakettien hajottamisen ja kokoamisen. [10]

PEAP-keskustelu on kaksiosainen. Ensimmäisessä osassa tapahtuu TLS-istunnon neuvottelu, jossa vähintään palvelin on tunnistettavaa varmenteen avulla. Vaihtoehtoisesti myös asiakasohjelmiston tunnistusta varmenteen avulla voidaan tukea. Neuvottelussa saatua avainta käytetään salaamaan loppuosa keskustelusta. Toisessa osassa käydään täydellinen EAP-keskustelu TLS-istunnon sisällä, sikäli mikäli ensimmäisessä vaiheessa ei asiakasohjelmaa tunnistettu. [10]

PEAP-keskustelu alkaa normaalisti verkkolaitteen ja asiakasohjelman välillä EAP-pyynnöillä. Kun ensimmäiset pyynnöt on välitetty, siirtyy keskustelu tunnistuspalvelimen ja asiakasohjelmiston välille. Verkkolaite toimii vain viestien välittäjänä osapuolien väillä. PEAP-keskustelussa Identity-viestin jälkeen lähetetään PEAP-aloitus -paketti. Jos asiakasohjelma vastaa myöntävästi, voi PEAP-tunnistus osapuolien välillä alkaa. Tämän jälkeen TLS-keskustelu jatkuu normaalisti alkaen Client Hello- ja Server Hello -viesteillä aina Change Cipher Spec- ja Finished-viesteihin asti. PEAP-standardin mukaisten sovellusten ei ole määritelty tai vaadittu tukevan kaikkia samoja salaus- ja tiivistealgoritmeja, kuin TLS-standardissa on määritelty. Ainoastaan TLS-kättelyä käyttäen avainten vaihtoon RSA:ta, salaukseen RC4-algoritmia sekä tiivisteistä tukea MD5- ja SHA-tiivisteille edellytetään. PEAP-keskustelun ensimmäinen vaihe päättyy kun TLS-kättely on suoritettu. [10]

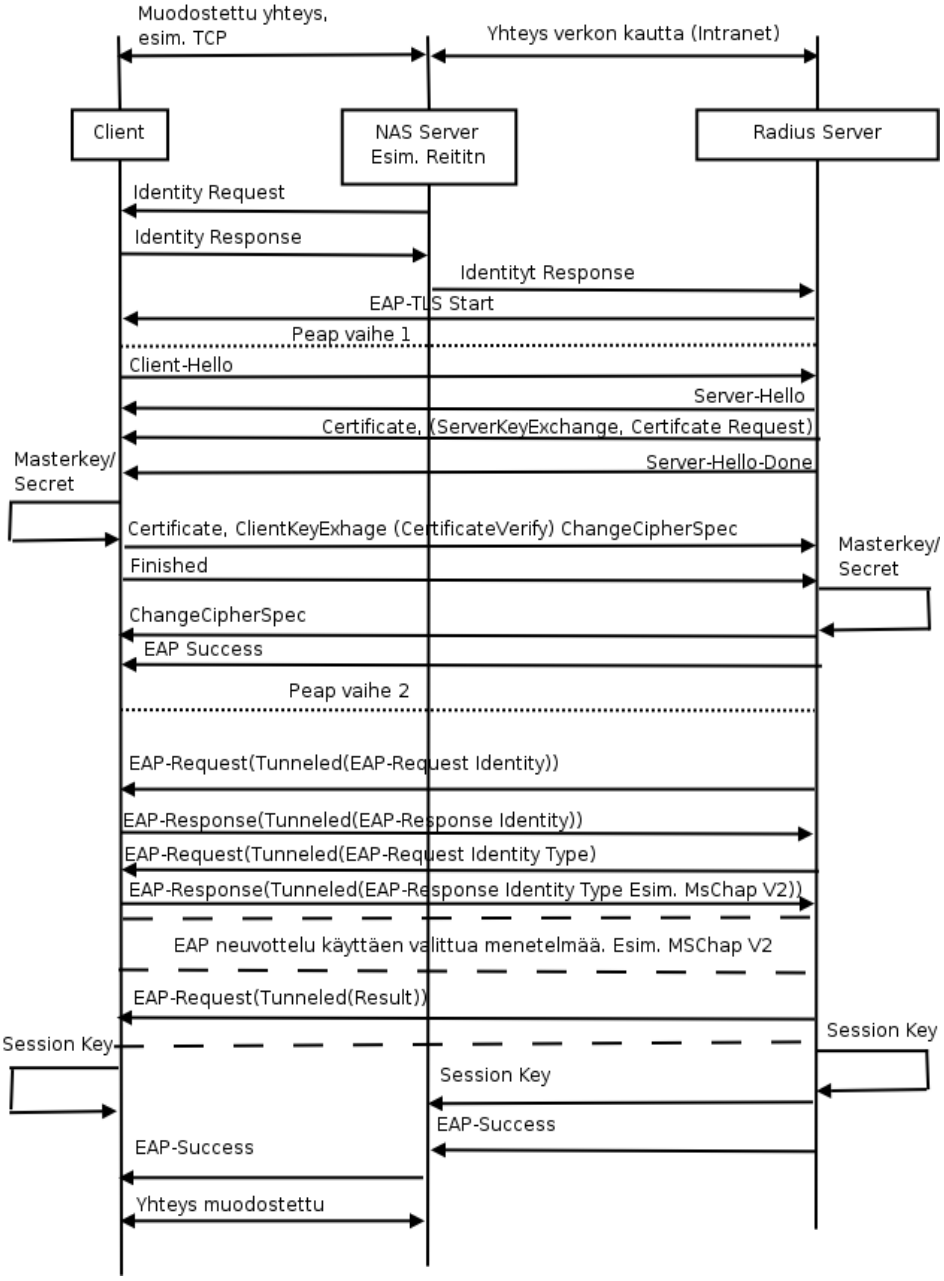
PEAP-keskustelun toisessa vaiheessa alkaa aiemmin neuvotellun TLS-istunnon sisällä salattuna uusi EAP-keskustelu. Näin toisessa vaiheessakin on mahdollista neuvotella käytettäväksi tunnistustapa jota molemmat, sekä palvelin että asiakasohjelma, tukevat ja tehdä tämä kaikki salattuna. Toinen vaihe aloitetaan ainoastaan, jos TLS- ja PEAP-

kättely ensimmäisessä vaiheessa suoritettiin onnistuneesti. Tässä vaiheessa välitetään uudestaan Identity-viestit, sekä neuvotellaan uudestaan palvelimen ja asiakasohjelman käyttämä tunnistusmenetelmä. Keskustelua jatketaan haluttua menetelmää käyttäen kunnes palvelin vastaa joko EAP-Success- tai EAP-Failure-viestillä. Se lähetetään edelleen salattua TLS-istuntoa pitkin. RADIUS Access-Accept- ja Access-Reject-viestien ei tarvitse sisältää enää EAP-attribuutteja, sillä nämä viestit on tarkoitettu ainoastaan itse verkkolaitteelle. Se tekee päätöksen käyttäjän käyttöoikeuksista ja järjestelmään pääsystä näiden RADIUS-viestien avulla, koska salatun EAP-viestin sisällön lukeminen on mahdotonta NAS:lle. EAP-TLS:stä poiketen PEAP-viestit sisältävät myös käytetyn PEAP-standardin versionumeron. Tätä versionumeroa varten on otettu käyttöön kaksi varattua bittiä EAP-TLS-standardin mukaisesta 8 bitin osiosta, jossa oli määritelty viestin jakamiseen osiin ja kokoamiseen liittyvät tiedot. [10]

PEAP:n virheidenhallinta on toteutettu käyttäen EAP:n, TLS:n ja RADIUS-standardin omia virheviestejä. Se ei sisällä omaa erillistä virheidenhallintaa. PEAP:ssa, niin kuin EAP:ssakin, palvelin on vastuussa kaikkiin pyyntöihin liittyvistä uudelleen yrityksistä. Isojen TLS-viestien jakaminen pienempiin osiin on toteutettu käyttäen EAP-TLS-standardin yhteydessä esiteltyä menetelmää. PEAP-standardi sisältää kaikki samat viestin jakamiseen ja kokoamiseen liittyvät bitit vastaavalla tavalla määriteltynä. [10]

Kuvassa 21 on esitetty, normaali PEAP-keskustelu kaikkien osapuolien välillä. Koska keskustelu tapahtuu yhteydettömän UDP-yhteyden yli, on tunnistuspalvelimen pystyttävä pitämään tallessa viestien vaihtoon liittyvä tarpeellinen tilatieto. Kuvassa on esitetty myös istuntoavainten välitys verkkolaitteelle onnistuneen tunnistuksen päätteeksi. Tunnistuspalvelimen ja asiakasohjelmiston on pystyttävä muodostamaan samanlaiset avaimet, joita käytetään salaamaan esimerkiksi verkkokortin ja langattoman tukiaseman välinen liikenne. [10]

Kuvassa 22 on esitetty PEAP-paketin rakenne merkittäviltä osiltaan. Kuvan 18 EAP-TLS-viestiin verrattaessa voidaan havaita PEAP-viestin rakenteen olevan muilta osin samanlainen, mutta sisältävän myös PEAP-protokollan versionumeron, jota varten on otettu käyttöön EAP-TLS-standardissa vapaana olleesta viidestä asetus-bitistä kaksi viimeistä bittiä. [10]

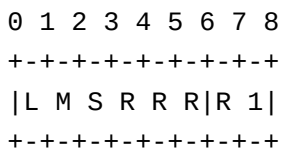


Kuva 21. PEAP-keskustelu asiakasohjelman, verkkolaitteen (NAS) ja tunnistuspalvelimen välillä. [11]

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Koodi     |   Tunniste    |            Pituus            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Tyyppi    |Asetukset  |Ver|  Tieto...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Kuva 22. PEAP-viestin rakenne [10]

Kuva 23 vastaa muilta osiltaan kuvan 19 viestiä, poiketen kuitenkin siten että asetus-kentän kaksi viimeistä bittiä on varattu PEAP-versionumeroa varten. PEAP-standardista on olemassa tällä hetkellä kolme eri versiota. Tämänhetkiset ratkaisut pohjautuvat alkuperäiseen ensimmäiseen Internet Draft:n, jossa käytetään versionumerona arvoa yksi tai jäljempänä esitettyyn Microsoftin versioon. Varatut bitit R ovat viesteissä aina nollia. Joten paketti on käytännössä oikealla versionumerolla varustettuna täsmälleen kuvan 23 esimerkin mukainen. Versionumeron avulla pyritään säilyttämään yhteensopivuus taaksepäin eri PEAP-versioiden välillä. Uusien versioiden myötä on mahdollista tarvittaessa tehdä edellisestä versiosta poikkeavia ratkaisuja. Esimerkiksi avainten ja avainmateriaalin muodostamiseen on uudemmissa versiossa tehty muutoksia. [10]



Kuva 23. Asetukset-bittien rakenne. [10]

3.2.7 Microsoftin PEAP-toteutus

Microsoft on tehnyt oman toteutuksensa PEAP:n ensimmäisestä versiosta, joka on saatavilla Windows 2000 -käyttöjärjestelmälle erillisenä päivityspakettina ja Windows XP-käyttöjärjestelmään Service Pack 1:n mukana. Heidän toteuttamassaan versiossa käytetään kuvassa 23 esitetyn PEAP-paketin versionumero-kentässä arvoa 0. Näin ollen paketista tulee samanlainen kuin EAP-TLS-standardissa, vaikka kyseessä onkin PEAP-standardin mukainen paketti. Syynä tähän on se, että Microsoft on tehnyt oman toteutuksena ennen ensimmäisen virallisen PEAP-version määrittelyn valmistumista. Ensimmäisissä esityksissä oli virheitä, jotka estivät ratkaisua olemasta täysin IEEE 802.1x -standardin mukaisia, esimerkiksi viimeinen viesti alkuperäisissä PEAP-standardissa luonnoksissa oli myös salattu, mikä ei IEEE 802.1x:n mukaisten verkkolaitteiden kanssa toiminut oikein. Näiden ongelmien ratkaisemiseksi yhtiö kehitti hieman varsinaisesta määrittelystä poikkeavan oman version. [12]

Microsoftin toteutuksessa on mahdollista käyttää ainoastaan yhtä tunnistusmenetelmää PEAP-istunnon toisen vaiheen aikana yhtä RADIUS-istuntoa kohden. Tästä ei kuitenkaan nyt tehtävässä toteutuksessa aiheudu mitään ongelmia, koska ainoastaan yhden tunnistustavan toteuttaminen riittää. [12]

3.3 MSCHAPv2-kirjautuminen PEAP-neuvottelun toisessa vaiheessa

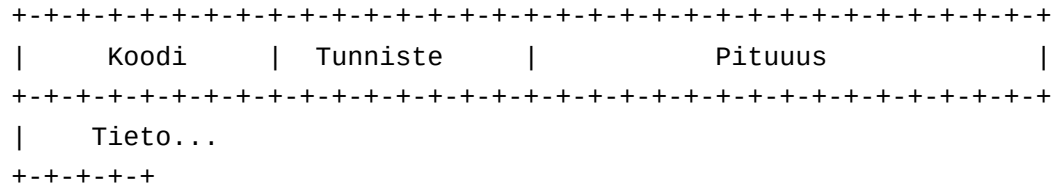
Microsoft on kehittänyt MSCHAPv2-protokollan alun perin käytettäväksi PPP-linkkien yli tapahtuvaan käyttäjätunnukseen ja salasanaan perustuvaan käyttäjän tunnistukseen. Sen kehittäminen on aloitettu tekemällä paranneltu versio CHAP-protokollasta [14], josta on ensin kehitetty MSCHAP [15] ja myöhemmin paranneltu MSCHAPv2. [13]

3.3.1 CHAP

CHAP-protokolla kehitettiin alun perin PPP-linkkien yli tapahtuvaan käyttäjien tunnistukseen. Se oli tarkoitettu esimerkiksi soittosarjoja tai vastaavia laitteita varten, mutta soveltuu yhtälailla muihinkin tarkoituksiin. [14]

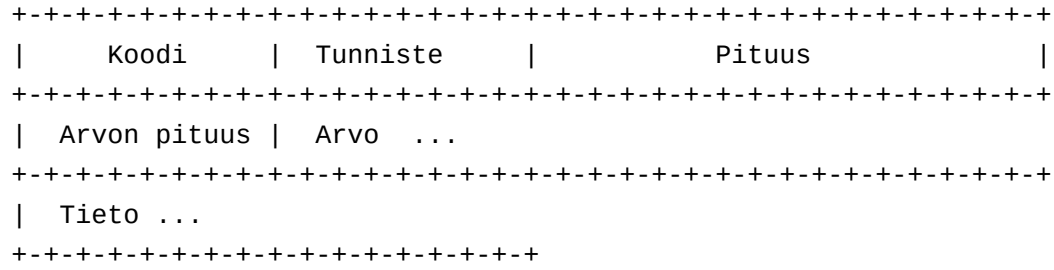
CHAP:n tarkoituksena on mahdollistaa käyttäjän tunnistaminen, joka toistuu aika-ajoin. Käytännössä käyttäjä tunnistetaan aina kun linkki luodaan, mutta tunnistus voidaan toistaa koska tahansa itse operaation jälkeenkin. Kun linkki on ensimmäisen kerran luotu, voidaan lähettää haaste yhteyttä ottavalle taholle. Se vastaa haasteeseen arvolla, joka on laskettu käyttäen tiivistealgoritmia. Palvelin tarkistaa vastauksen verraten sitä itse laskemaansa tiivistearvoa vastaan. Jos arvot täsmäävät, hyväksytään asiakasohjelman vastaus. Muussa tapauksessa yhteys pitäisi katkaista. [14]

CHAP-protokollassa viestit voivat olla neljää eri tyyppiä: challenge-, response-, success- ja failure-viestejä. Ne sisältävät tyyppi-koodin lisäksi paketin yksilöivän tunnisteen, paketin tietokentän pituuden, sekä itse tietosisällön. CHAP-viestin perusrakenne on esitetty kuvassa 24. [14]



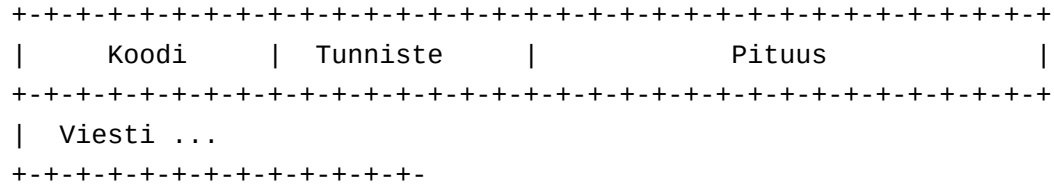
Kuva 24. CHAP-viestin malli. [14]

Kuvassa 25 on esitetty haasteviestin ja haasteen vastausviestin rakenne. Haaste on sattumanvarainen bittijono. Arvo-kentässä välitetään haaste ja haasteen vastaus. Se muodostuu MD5-tiivisteestä, joka lasketaan yhdistämällä viestin tunniste-kentän tieto ennalta tunnettuun jaettuun salaisuuteen ja haastearvoon. Tieto-kentässä voidaan välittää vapaavalintainen merkkijono. Se voi sisältää esimerkiksi viestin käyttäjälle selväkielisessä muodossa. [14]



Kuva 25. CHAP response- ja challenge-viestin muoto. [14]

Kuvassa 26 esitetään viestin rakenne, jota käytetään välittämään tieto kirjautumisen lopputuloksesta. Viestin sisältö vastaa täysin edellisiä, mutta siinä välitetään ainoastaan selväkielinen viesti käyttäjälle, kirjautumisen onnistumisesta tai epäonnistumisen syystä.



Kuva 26. Success- ja Failure-viesti. [14]

3.3.2 MSCHAP

MSCHAP on kaikilta mahdollisilta osiltaan pyritty tekemään yhteensopivaksi CHAP-standardin kanssa. Merkittävin muutos on se, ettei salasanan säilömistä selväkielisenä enää edellytetä, vaan se voidaan tallentaa myös tiivisteenä. Kaikki paketit ovat samanlaisia CHAP-standardin yhteydessä kuvattujen kanssa. Ainoastaan tiettyjen viestikenttien sisältöä on muuttunut siten että ne sisältävät enemmän tietoa. CHAP-vastauksessa ollut arvo-kenttä sisältää Windows NT Challenge Response:n ja LAN Manager -yhteensopivan haasteen. Tämä mahdollistaa salasanan säilömisen NT Password Hash -muodossa käyttäjähakemistoon selväkielisen sijaan. Standardissa mainitaan myös käyttäjätunnukset, jotka ovat muotoa DOMAIN\käyttäjätunnus. [15]

Onnistuneessa kirjautumisessa palvelimen vastaus on identtinen CHAP-standardin määrityksen mukaisen vastauksen kanssa. Epäonnistunut vastaus on rakenteeltaan samanlainen, mutta selväkielistä tekstiä varten varatussa kentässä on MSCHAP-standardin mukaisesti muotoiltu teksti sisältö. Kuvassa 27 on esitetty virheilmoituksen rakenne, joista ensimmäinen kenttä E on virheen numerokoodi ja toisen kentän R avulla voidaan määritellä onko uudelleen yrittäminen mahdollista. Kenttä C kuvaa ascii-muodossa esitettyä uutta haastetta ja kenttä V kuvaa käytettyä versiota. [15]

```
"E=eeeeeeeeee R=r C=cccccccccccccccc V=vvvvvvvvvv"
```

Kuva 27. MSCHAP-virheilmoitus. [15]

MSCHAP-standardin avulla on mahdollista toteuttaa myös käyttäjän salasanan vaihto ja muita käyttäjän tietoihin liittyviä toiminnallisuuksia, mutta ne ovat tämän työn yhteydessä tehtävän toteutuksen kannalta merkityksettömiä.

3.3.3 MSCHAPv2

MSCHAPv2-määritys on kehitetty MSCHAP:n ensimmäisen version pohjalta. Kenttien sisältö ja niiden hyödyntäminen on monilta osiltaan muuttunut, mutta itse viestien rakenne on pääosin sama. Uusi määritys kuvaa ainoastaan muuttuneet osat. Challenge-viesti on täysin identtinen MSCHAP:n ensimmäisen version kanssa. Myös

response-viesti on rakenteeltaan identtinen, mutta siinä olevan haastetta varten varatun kentän rakenne on muuttunut. 16 ensimmäistä tavua muodostavat niin sanotun Peer-Challenge -arvon, seuraavat kahdeksan tavua on varattu myöhempiä tarkoituksia varten ja 24 seuraavaa tavua muodostavat NT-Response -arvon. Lopussa on 8 bittiä asetuksia varten. Peer-Challenge on sattumanvarainen arvo, NT-Response sisältää salasanan, käyttäjätunnuksen, peer-challenge-kentän, sekä palvelimen haasteen tarkoituksen mukaisella tavalla koodattuna. Viestin nimi -kentässä välitetään käyttäjän käyttäjätunnus ja Windows-ryhmä selväkielisenä. [13]

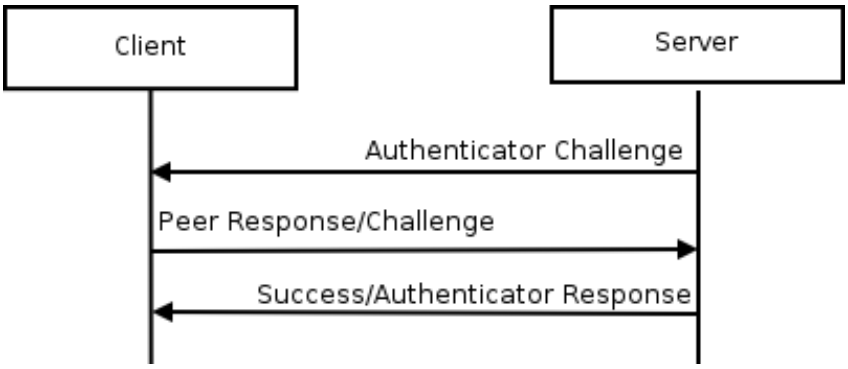
Success-viesti on rakenteeltaan vastaava kuin MSCHAP- ja CHAP-standardeissa. Sen viesti-kenttä on kuitenkin otettu uudestaan käyttöön ja sitä hyödynnetään eri tavalla. Viesti on esitetty kuvassa 28. Tunnistekenttään palvelimen on laskettava aiempien tietojen perusteella 42 bittiä pitkä tarkiste. Järjestelmään kirjautuvan pään on varmistettava tämän tarkisteen oikeellisuus, jotta voidaan olla varmoja neuvottelun jatkumisesta saman osapuolen kanssa, jonka kanssa se on aloitettu. Viesti-kentässä voidaan välittää käyttäjälle selväkielinen viesti. [13]

```
"S=<Tunniste> M=<Viesti>"
```

Kuva 28. Success-viestin viestikentän sisältö. [13]

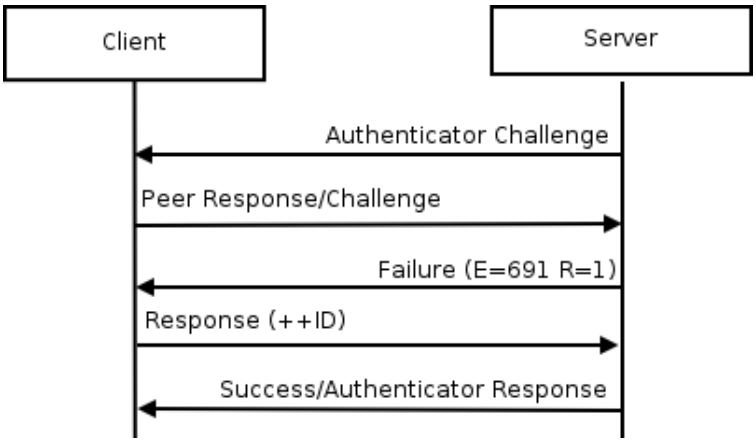
Virheviestin osalta viesti ja sen käyttö ovat identtistä MSCHAP:n aiemman version kanssa. Määrityksessä on kuitenkin tarkennettu tilannetta järjestelmän toiminnan kannalta. Siitä käy yksiselitteisesti ilmi, että jos virheviestissä sallitaan uudelleen yrittäminen, on järjestelmän pyydettävä käyttäjältä uudestaan hänen tunnistuksessa käyttämiä tietojaan, vähintään salasanaa. Lisäksi aina on jatkettava järjestelmässä tunnistukselle varattua aikaa. Tämä mahdollistaa kertakäyttösalasanan kaltaisen toiminnallisuuden toteuttamisen tätä menetelmää käytettäessä. [13]

Kuvassa 28 on esitetty onnistuneen MSCHAPv2-protokollan mukainen kirjautumistapahtuman kulku. Kyseinen neuvottelu tapahtuu toteutettavassa sovelluksessa PEAP-neuvottelun toisessa vaiheessa. [13]



Kuva 29. Onnistunut MSCHAPv2-kirjautuminen. [13]

Kuvassa 30 on esitetty MSCHAPv2-standardin mukainen tunnistus- ja kättelyprosessi, jossa käyttäjä ensimmäisellä kerralla on syöttänyt väärän salasanan tai käyttäjätunnuksen. Viesti jossa muuttujan R arvoksi on annettu 1 tarkoittaa uudelleen yritystä. Tällöin palvelimen puolesta järjestelmän kirjautumisen sallittavaa aikaa kasvatetaan niin, että käyttäjä voi yrittää syöttää käyttäjätunnuksen ja salasanan järjestelmään uudestaan. [13]



Kuva 30. Kirjautuminen, jossa on ensimmäisellä kerralla syötetty väärä salasana. [13]

4 KRYPTOGRAFIA

Valimo iDSever:ssä tullaan pyrkimään toimivaan lopputulokseen, joka toteuttaa tarvittavat menetelmät ja täyttää asiakkaan vaatimukset. Kaikkia standardien vaatimia nykyisen toteutuksen kannalta tarpeettomia toimintoja ei toteuteta ensimmäisessä vaiheessa. Tarjotuista vaihtoehdoista valitaan toteutettavaksi TLS-kättelyssä ja salauksessa algoritmit, joissa kättely tehdään RSA-algortimia käyttäen ja virtasalaimena käytetään 128 bittistä RC4-algoritmilla toimivaa salainta, jossa viestien muuttumattomuus varmennetaan MD5-tiivisteen avulla.

4.1 RSA-algoritmi

RSA on salaukseen ja allekirjoittamiseen tarkoitettu epäsymmetrinen algoritmi. Se on yksi helpoimpia ymmärtää ja toteuttaa. Se perustuu olettamukseen, jonka mukaan erittäin suurien kokonaislukujen tekijöihin jakaminen on hankalaa. [16]

Kaksi avainta muodostetaan valitsemalla kaksi riittävän suurta alkulukua p ja q. Lisäksi on suositeltavaa, että p ja q ovat molemmat yhtä pitkiä. Lasketaan tulo:

$$n=pq \tag{4}$$

Tämän jälkeen valitaan sattumanvarainen salausavain e, siten että e ja (p-1)(q-1) ovat suhteellisia alkulukuja. Tämän jälkeen lasketaan laajennetulla Euklidisella-algoritmilla purkuavain d.

$$ed \equiv 1 \bmod (p-1)(q-1) \tag{5}$$

Toisin sanoen:

$$d = e^{-1} \bmod ((p-1)(q-1)) \tag{6}$$

Myös d ja n ovat suhteellisia alkulukuja. Salausavain e ja n muodostavat julkisen avaimen. Salainen avain on d. Kaksi alkulukua p ja q ovat avainten muodostamisen jälkeen tarpeettomia. Niitä ei koskaan tule paljastaa.

Itse salaus tapahtuu seuraavasti. Viesti m, joka on tarkoitus salata, jaetaan osiin. Jokaisen osan on oltava kooltaan pienempi kuin n. Binäärimuotoisen tiedon ollessa kyseessä suurin mahdollinen tieto on suurin mahdollinen kahden potenssi, joka on pienempi kuin n. Viestistä c otettu osa c_i salataan laskemalla:

$$c_i = m_i^{\,e} \bmod n \tag{7}$$

Kun vastaanottaja tietää salaisenavaimensa d sekä vastaanottaa viestin salaajalta salatun viestin c, voi hän purkaa viestin laskemalla:

$$m_i = c_i^{\,d} \bmod n \tag{8}$$

Tällä hetkellä RSA:ta pidetään turvallisena menetelmänä olettaen, että käytetyt avaimet ovat riittävän suuria. Paras ja tehokkain tunnetuista hyökkäysmenetelmä RSA-algoritmia vastaan on julkisen avaimen osatekijöihin jako. Edellytyksenä on, että hyökkääjällä on käytössään salattu teksti, käytetty eksponentti sekä selväkielinen teksti. On mahdollista, että muitakin tehokkaampia menetelmiä on olemassa, mutta tällä hetkellä muut esitellyt menetelmät ovat hitaampia, kuin mitä osatekijöihin jako olisi. Periaatteessa on osoitettu, että kvanttitietokoneilla tekijöihin jaon voisi suorittaa polynomisessa ajassa. Käytännössä näin ei kuitenkaan tule tapahtumaan lähitulevaisuudessa. [16]

Käytettäväksi valittu RSA-salaus on toistaiseksi turvallinen menetelmä. Avaintenvaihto tapahtuu lähettämällä palvelimen varmenne asiakasohjelmalle. Asiakasohjelma salaa tekstin palvelimen varmenteen julkisella avaimella ja lähettää sen palvelimelle. Näin kaikki elementit, joita salauksen murtamiseen osatekijöihin jakamalla tarvitaan, on saatavilla, jos hyökkääjä tarkkailee kaikkea verkkoliikennettä.

Käytännössä riittävän suuria avaimia käytettäessä menetelmä on kuitenkin turvallinen. [16]

Javan versio 1.4 tukee RSA-avaimia aina 2048-bittiseen avaimeen asti. Javan versiossa 1.5 on tuettuna myös 4096-bittisten RSA-avaimien käyttö. Käytännössä jo 2048-bittisten avaimien käyttäminen tarjoaa riittävän turvan hyökkäyksiä vastaan. Toki suurimman mahdollisen avaimen käyttäminen on suositeltavaa.

4.2 RC4-virtasalain

RC4 on virtasalaus algoritmi. Se on melko yksinkertaista kuvailla ja toimii erittäin nopeasti. RC4:ssä luodaan isoja satunnaislukuja annettujen avainten perusteella. Salattu teksti saadaan yhdistämällä satunnaisluku ja salattava teksti XOR-operaatiolla. Teksti palautetaan selväkieliseksi tekstiksi tekemällä XOR-operaatio samalla satunnaisluvulla samalle salatuntekstin osuudelle uudestaan. Molemmat osapuolet muodostavat samat satunnaisluvut käyttäen samaa molempien tuntemaa avainta, joka näin mahdollistaa tekstin salaamisen ja purkaminen. [16]

4.3 Pseudo Random Function (PRF)

Lukuisat TLS-protokollan osat perustuvat jaetun salaisuuden avulla laskettuun HMAC-tiivisteeseen. HMAC voidaan toteuttaa lukuisten eri tiivistealgoritmien avulla. TLS-standardissa on määritelty käytettäväksi SHA- ja MD5-tiivistealgoritmeja. Lisäksi PRF-algoritmia tarvitaan esimerkiksi satunnaisten salausavainten muodostamiseen. Niitä käytetään esimerkiksi langattomanverkon tukiaseman ja verkkokortin välisen liikenteen salaamiseen. Näiden laskemisessa avainasemassa on PRF-funktion jaettujen salaisuuksien ja molempien osapuolien tunteman otsikon avulla lasketut avaimet. [7]

PRF-funktion turvallisuus TLS-kättelyssä on pyritty varmistamaan sillä, että se perustuu kahteen eri tiivistealgoritmiin. Näin sen pitäisi olla turvallinen ja käyttökelpoinen niin pitkään, kuin yksi käytetyistä tiivistealgoritmeista on turvallinen.

Algoritmi on murrettavissa, vasta kun molemmista käytetyistä tiivistealgoritmeista on löydetty jokin ratkaiseva heikkous. [7]

Ensimmäisessä vaiheessa on määritelty funktio halutun mittaisen tuloksen laskemiseen. HMAC:n toiminta on kuvattu tässä työssä kappaleessa 4.4.3. Funktio on seuraava:

$$\begin{aligned} P\_hash(salaisuus, siemen) = {} & HMAC\_hash(salaisuus, A(1) \,||\, siemen) \,|| \\ & HMAC\_hash(salaisuus, A(2) \,||\, siemen) \,|| \\ & HMAC\_hash(salaisuus, A(3) \,||\, siemen) \,|| \ldots \end{aligned} \tag{9}$$

missä || tarkoitta merkkijonojen liitosoperaatiota ja A(x) määritellään:

(10)

$$A(0) = siemen$$
$$A(i) = HMAC\_hash(salaisuus, A(i-1))$$

missä funktiota kutsutaan iteratiivisesti, kunnes haluttu tietomäärä on saatu muodostettua.

TLS-algoritmissa PRF-funktiota käytetään siten, että sekä SHA- että MD5-tiivistealgoritmeja käytetään samanaikaisesti luomaan haluttu tieto, jonka jälkeen lopputulokset yhdistetään XOR-funktion avulla toisiinsa yhdeksi syötteeksi.

Kuvauksessa S1 ja S2 ovat jaetun salaisuuden eri osista. S1 ensimmäisestä osasta ja S2 toisesta. Tämä tehdään jakamalla salaisuus puoliksi kahteen osaan. Niitä käytetään TLS-protokollan mukaisessa PRF-toteutuksessa, joka perustuu kahteen eri tiivistealgoritmiin. Algoritmissa itse otsikko on ASCII-merkkijono, joka vaihtuu tilanteen mukaan riippuen missä vaiheessa kättelyoperaatiossa ollaan menossa. Esimerkiksi pääsalaisuus muodostetaan esisalaisuudesta PRF-funktion avulla antamalla otsikoksi ”Master secret”.

$$\begin{aligned} PRF(salaisuus, otsikko, siemen) = {} & P\_MD5(S1, otsikko + siemen) \oplus \\ & P\_SHA\text{-}1(S2, otsikko + siemen); \end{aligned} \tag{12}$$

MD5-tiiviste on 16 bitin mittainen ja SHA-1-tiiviste on 20 bitin mittainen. Käytännössä ongelma eripituisten tiivisteiden välillä on ratkaistu siten, että 20 bitin mittaisen syötteen aikaansaamiseksi MD5-algoritmia kutsutaan kaksi kertaa ja SHA-1 vain kerran. Näin saadaan halutun mittainen tulos.

4.4 Tiivistealgoritmit

EAP-protokollan toteutukseen tarvitaan kahta eri tiivistealgoritmia: MD5 ja SHA-1. Niitä käytetään monessa eri kättelyn vaiheessa sekä aina PRF-funktiota käytettäessä.

4.4.1 MD5-tiiviste

MD5-tiivistealgoritmi on yksinkertainen toteuttaa. Se tarjoaa mahdollisuuden tiivisteen laskemiseen viestistä. Tiivisteen pituus on aina 16 tavua. MD5-algoritmin turvallisuutta on selvitetty tarkkaan erilaisissa tutkimuksissa ja algoritmista on löydetty tiettyjä heikkouksia. Sitä vastaan on olemassa hyökkäyksiä. Näiden vakavuus riippuu kuitenkin käyttötarkoituksesta. On esitetty, että vaaditaan noin 2^64 operaatiota, jotta löydettäisiin kaksi viestiä, joilla on sama tiiviste. Lisäksi on esitetty, että vaaditaan noin 2^128 operaatiota, jotta löydettäisiin mikä tahansa tiettyä tiivistettä vastaava viesti. MD5:n merkittävin etu on algoritmin yksinkertaisuus, mikä näkyy siinä, että algoritmi on helppo toteuttaa ja lisäksi erittäin nopea. MD5-algoritmin heikkoudet ovat samat kuin SHA-1-algoritmilla ja muilla iteratiivisilla tiivistealgoritmeilla. Heikkoudet on käyty tarkemmin läpi kappaleessa 4.4.2 SHA-1-algoritmin yhteydessä. [19]

4.4.2 Secure Hash Algorithm (SHA-1)

SHA-1 algoritmia voidaan käyttää MD5:n tavoin tiivisteiden laskemiseen viesteistä tai tiedostoista. SHA-1 on laajasti käytetty esimerkiksi digitaalisten allekirjoitusten tiivisteenä. SHA-1-tiiviste on pituudeltaan MD5-tiivistettä pitempi, 20-tavua. Toteutukseltaan algoritmi on hiukan monimutkaisempi verrattuna MD5-algoritmiin. [18]

Käytännössä on osoitettu, että kaikki iteratiiviset tiivistealgoritmit sisältävät heikkouksia. Heikkoudet liittyvät siihen, että on mahdollista löytää käyttäen nykyisten koneiden laskentatehoja hyväksi kaksi eri tekstiä, joilla on sama tiivisteen arvo. Näin on mahdollista siis löytää toinen teksti, joka johtaa samaan lopputulokseen. SHA-1-algoritmista on vuonna 2005 löydetty vastaavia heikkouksia kuin MD5-algoritmista jo aikaisemmin. Käytännössä nykyisten koneiden laskentateho huomioon ottaen SHA-1-algoritmin käyttöä pidetään turvallisena vielä vähintään vuoteen 2010 asti. Kuten MD5-algoritmissa, haavoittuvuus liittyy siihen, että on mahdollista ajan kuluessa konetehoa käyttäen muodostaa toinen teksti, josta muodostaa sama tiivisteen arvo. Haavoittuvuus vaikuttaa lähinnä digitaalisten allekirjoituksiin. TLS-kättelyssä käytettynä tällä ei ole niin merkittävää vaikutusta, koska sen hyödyntäminen vaatii paljon laskentatehoa verrattuna itse tapahtuman kestoon. Lisäksi vaikka hyökkäyksen avulla löytyisi toinen teksti, jonka avulla sama tiiviste voidaan muodostaa, on kättelyssä oleellisinta tuntea tieto josta tiivisteet on laskettu. Jos hyökkääjä ei saa tiivisteistä laskettua alkuperäistä tekstiä josta ne on laskettu, epäonnistuu hän yrityksessään. Ilman pääsalaisuuden tuntemista kättely epäonnistuu, sillä hyökkääjän ei voi saada selville esimerkiksi virtasalaimessa käytettyä avainta. Näin ollen tällä hetkellä tunnetut tiivistealgoritmien heikkoudet eivät merkittävästi vaikuta TLS-kättelyn tietoturvaan. [21]

4.4.3 HMAC

HMAC on tarkoitettu viestien tunnistamiseen kryptokrafisten funktioiden avulla. HMAC:a voidaan käyttää minkä tahansa iteratiivisen olemassa olevan tiivistealgoritmin kanssa. Tällä hetkellä HMAC lasketaan yleisimmin MD5- tai SHA-1-algoritmeja käyttäen, mitkä siis ovat tämän hetken yleisimmin käytetyt tiivistealgoritmit. Kryptokrafinen luotettavuus HMAC:n tapauksessa riippuu täysin käytetystä tiivistealgoritmista. [17]

HMAC tulee sanoista Hashed Message Authentication Codes. Yleisin käyttötarkoitus HMAC:lle on kahden osapuolen välillä viestien tunnistaminen ja alkuperän varmentaminen. Menetelmä on kehitetty pääasiassa epäluotettavan tai salaamattoman kanavan läpi tapahtuvaan kommunikaatioon viestien alkuperän varmentamiseksi.

MAC-arvoa käytettäessä lähettäjä lähettää vastaanottajalle itse viestin sekä viestistä lasketun HMAC-arvon. HMAC perustuu siihen, että molemmat osapuolet tuntevat salaisen avaimen sekä voivat tarkistaa MAC-arvon laskemalla vastaavan MAC:n samasta viestistä jaetun salaisuuden avulla. Pelkkä tiivistefunktio sellaisenaan ei voi toimia viestin varmenteena, koska salaamattoman tiivisteen muokkaaminen on mahdollista käytettäessä epäluotettavaa viestintä kanavaa. [17]

HMAC:n merkittävimmät edut ovat siinä, että se perustuu jo olemassa oleviin tiiviste-algoritmeihin ja että niitä käytetään ilman muutoksia. HMAC:n laskenta ei merkittävästi heikennä suorituskykyä, vaan suorituskyky riippuu pääasiassa käytetyistä tiivistealgoritmeista. [17]

HMAC lasketaan kaavalla:

$$HMAC(viesti) = h\big((K \oplus opad) \,||\, h((K \oplus ipad \,||\, viesti)\big) \qquad (13)$$

missä viesti tarkoittaa viestiä, josta MAC lasketaan. H() tarkoittaa käytettyä tiivistealgoritmia. K tarkoittaa jaettua salaista avainta. Kaavan arvoista ipad tarkoittaa arvoa 0x36 ja opad 0x5c, joita toistetaan tarvittavan monta kertaa. [17][22]

HMAC:n laskenta etenee periaatteessa kuvan 31 mukaan. Eri vaiheet on selitetty yksityiskohtaisemmin taulukossa 3. [22]

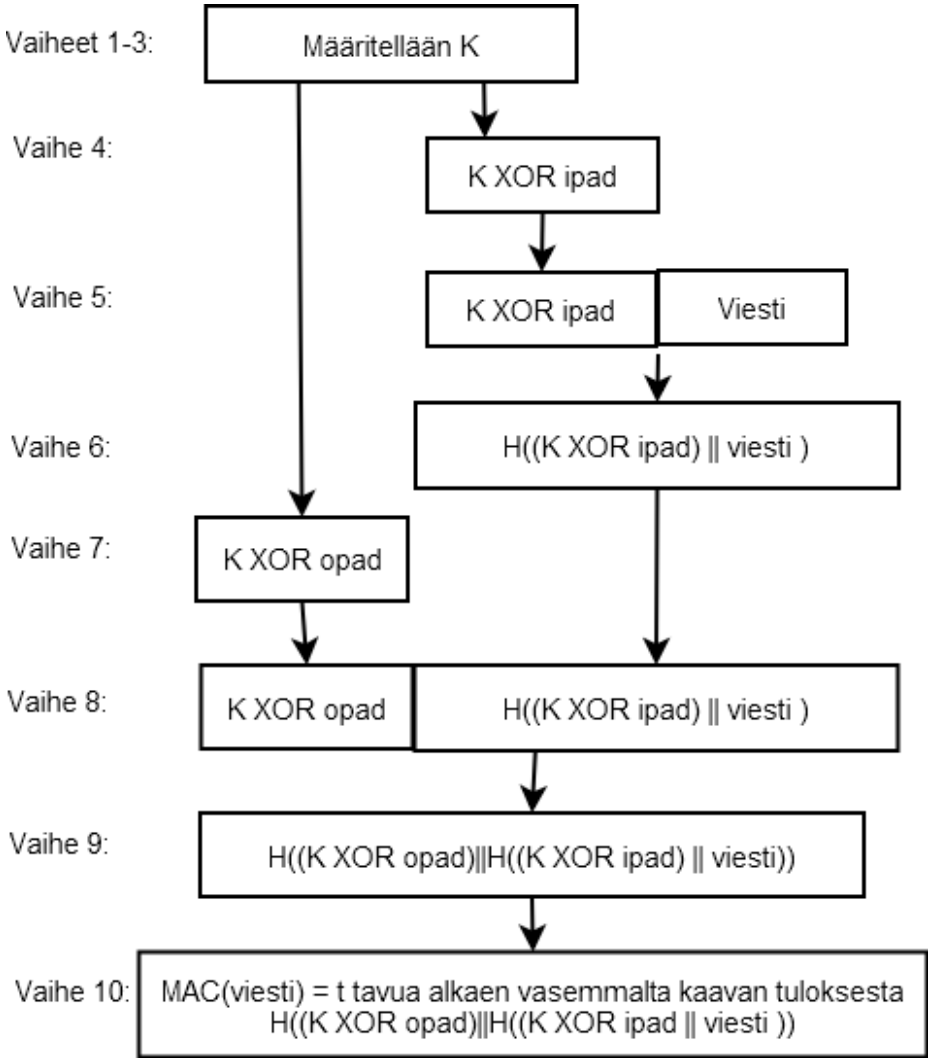
Tiivisteen lyhentäminen alkuperäisestä on yleinen käytäntö. Siinä on omat edut ja haittansa. Hyökkääjä saa vähemmän tietoa viestin tarkisteesta, mutta toisaalta hänellä on vähemmän oikein arvattavaa siitä, jos hän haluaa muodostaa uuden tarkisteen ja väärentää viestin. Suositus on, että tiivisteen pituus tulisi olla vähintään puolet oikeasta viestin tarkisteesta sekä vähintään yli 80 bittiä, jotta pituus olisi riittävä vaikeuttamaan hyökkääjän työskentelyä. TLS-kättelyssä turvallisuutta on lisätty käyttämällä kahden eri tiivistealgoritmin avulla laskettujen HMAC:n yhdistelmää, joka on kuvattu aikaisemmin. [17]

Taulukko 3. HMAC-algoritmin toiminta vaiheittain kuvattuna. [22]

Vaihe	Vaiheen kuvaus
1	Tarkistetaan onko K:n pituus sama kuin tiivistealgoritmin hyväksymän syötteen pituus. Jos on, siirrytään vaiheeseen neljä.
2	Jos avaimen K pituus oli suurempi kuin tiivistealgoritmin hyväksymän syötteen pituus
3	Jos avaimen pituus K oli suurempi kuin B, lisätään K:n loppuun 0x00 bittejä niin paljon kunnes saadaan B:n mittainen bittijono.
4	K XOR ipad tuottaa B:n mittaisen bittijonon: $K \oplus ipad$
5	Yhdistetään vaiheessa neljä saatu bittijono lisäämällä ”viesti” sen jatkoksi. $(K \oplus ipad) \parallel viesti$
6	Lasketaan tiiviste merkkijonosta joka on saatu vaiheessa 5: $H((K \oplus ipad) \parallel viesti)$
7	Lasketaan XOR K:sta ja opad:stä: $K \oplus opad$
8	Yhdistetään tulokset askeleista 6 ja 7: $(K \oplus opad) \parallel H((K \oplus ipad) \parallel viesti)$
9	Lasketaan tiiviste vaiheen 8 tuloksesta: $H(K \oplus opad) \parallel H((K \oplus ipad) \parallel viesti)$
10	Otetaan t bittiä vaiheen 9 tuloksesta MAC arvoksi.

HMAC:n luotettavuus riippuu käytetystä tiivistealgoritmista. Sen luotettavuutta lisää se, että vaikka tiiviste-algoritmeista löytyisi haavoittuvuus ja se saataisiin murrettua, se on kuitenkin ainoastaan varmennettu tiiviste itse tekstistä. Näin ollen, kaikki aiemmin lasketut tiivisteet ovat edelleen voimassa. Salattujen tekstien kohdalla tilanne olisi toinen, koska salauksen murtamisen jälkeen kaikki aiemmin salatut tekstit ovat periaatteessa murrettavissa. Lisäksi TLS-kättelyssä HMAC:n arvoa käytetään ainoastaan väliaikaisesti kättelyssä käytettyjen viestien varmentamiseen ja näissäkin tapauksissa se lasketaan yhdistämällä kahdella eri algoritmilla laskettujen HMAC:n tulokset. Lisäksi HMAC arvo kulkee viesteissä mukana salattuna. Käytännössä vaikka HMAC-algoritmi todettaisiin turvattomaksi, ei itse kättelyprosessi olisi uhattuna, koska lisäksi hyökkääjän olisi murrettava myös salatut viestit ja kaapattava muu osapuolien välinen tietoliikenne. Käytännössä kättelyprosessi itsessään ei ole

tunnistuksessa heikko kohta, vaan järjestelmää vastaan voidaan hyökätä paljon helpommin muilla tavoin.



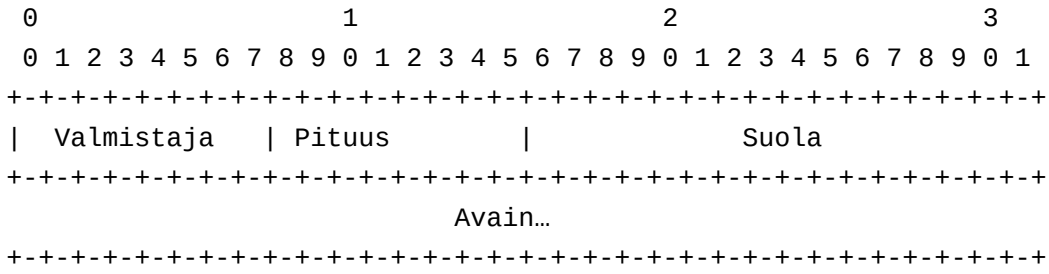
Kuva 31. HMAC:n toiminta esitettynä kaaviona. [22]

4.5 MPPE-salausavainten laskenta

MPPE-salausavaimet (Microsoft Point-to-Point Encryption) eivät sinänsä ole RADIUS-palvelimen itsensä kannalta oleellisia. VLAN-kytkintä käytettäessä niitä ei tarvita ollenkaan. Ne voidaan palauttaa kyllä kytkimelle, mutta liikenne sen ja asiakasohjelmiston välillä ei ole ainakaan oletusarvoisesti salattua. Tietoliikenne kulkee kytkimen ja käyttäjän välillä fyysisesti verkkokaapelissa, joten sitä voidaan pitää salaamattomanakin melko turvallisena siirtotienä. Langattomassa verkossa riskit

ovat kuitenkin toisenlaiset ja liikenne on salattava. Langattoman verkon tukiasema toimii ainoastaan RADIUS-liikenteen välityspalvelimena asiakasohjelmiston ja palvelimen välillä. Sen on kuitenkin pystyttävä salaamaan liikenne kättelyn jälkeen. Koska RADIUS-palvelin on neuvotellut avaimet kättelyn aikana asiakasohjelmiston kanssa salatusti, ei tukiasema voi olla niistä tietoinen. RADIUS-palvelimen on siis välitettävä tieto avaimista tukiasemalle. Tämä tapahtuu RADIUS-protokollassa Access-Accept-viestin yhteydessä välitettävillä MPPE-salausavain -attribuuteilla. Avaimia on kaksi: toinen on palvelimelle viestin salaamista varten ennen sen lähettämistä asiakasohjelmistolle, toinen on asiakasohjelmistolta tulevien viestien salauksen purkamista varten. Asiakasohjelmisto laskee itsellään olevien tietojen perusteella samat avaimet, mutta käyttää niitä päinvastaisessa tilanteessa.

Avainten muoto on kuvattu kuvassa 32. Vastaanotto ja lähetys avaimet ovat muuten samat, mutta valmistaja-koodina käytetään arvoa 16 lähetys avaimelle ja arvoa 17 vastaanotto avaimelle. Viestissä suola on kahden tavun mittainen kenttä, ja sen tarkoituksena on varmistaa, että avaimet että Access-Accept-paketin attribuuttien salaukseen käytetyt avaimet ovat ainutkertaisia. Paketti salataan RADIUS-palvelimella käyttäen jaettua salaisuutta ja RADIUS-viestin muuttumattomuuden varmistamiseen käytettävää viestintarkistearvoa. [20]



Kuva 32. MS-MPPE-avaimen muoto [20]

Itse avaimen laskenta tapahtuu RADIUS-protokollan ollessa kyseessä PPP-EAP-TLS [9] standardin määrityksen mukaisesti. Kaavassa käytetään PRF-funktiota, jolle syötetään aiemmin kättely prosessissa laskettu pääsalaisuus. Otsikkona käytetään merkkijonoa ”client EAP encryption” ja siemenenä käytetään ensimmäisissä TLS-

kättelyn viesteissä vaihdettuja asiakasohjelmiston ja palvelimen muodostamia sattumanvaraisia arvoja yhdistettynä toisiinsa. Kaava on seuraava:

K = PRF(pääsalaisuus, ”client EAP encryption”, ClientRandom || ServerRandom) (14)

mikä sisältää edellä mainitut arvot. Bittijonosta K otetaan 64 tavun mittainen merkkijono. Tukiaseman tiedonvastaanotossa käyttämä salausavain saadaan ottamalla 32 ensimmäistä tavua arvosta K ja palvelimen lähetysavain saadaan ottamalla 32 seuraavaa tavua bittijonosta K. [9]

4.6 Tietoturva

PEAP-tunnistusta ja EAP-TLS-tunnistusta pidetään yleisesti melko turvallisena. Käytetyt algoritmit itsessään ovat turvallisia. Suurimpana uhkana on heikkojen salasanojen käyttö käyttäjien toimesta. Toinen uhka on niin sanottu Man-in-the-middle -hyökkäys, jossa hyökkääjä toimii välittäjänä tukiaseman ja asiakasohjelmiston välillä. Kaikesta huolimatta menetelmä on kuitenkin merkittävästi tuvallisempi kuin esimerkiksi tällä hetkellä monissa paikoissa käytössä oleva pelkkä staattiseen salasanaan perustuva tunnistus ja salaus. Se on melko helppo murtaa tarkkailemalla riittävää määrää verkkoliikennettä. [23][24]

PEAP-tunnistuksen turvallisuutta voitaisiin lisätä käyttämällä nykyisten menetelmien rinnalla sirukortilla tai palvelimella olevan asiakasvarmenteen vaatimista ja tarkistusta. Nykyinen pelkästään palvelimen varmenteeseen perustuva TLS-kättely mahdollistaa sen, että teoriassa tähän väliin on mahdollista päästä, jos asiakasohjelmisto, joka yrittää tunnistusta, saadaan huijattua toimimaan väärin. [25]

4.6.1 Man-in-the-middle -hyökkäys

Man-in-the-middle -hyökkäys on mahdollinen, jos hyökkääjä saa asiakasohjelmiston ottamaan ensin yhteyttä omaan palvelimeensa, joka toimii välittäjäpalvelimena langattoman verkon tukiasemalle. Käytännössä hyökkääjän on saatava asiakasohjelmisto käyttämään heikompaa tunnistusta ja hän tätä kautta saisi selville

käyttäjän salasanan. Koska käyttäjälle ei ole omaa varmennetta, voi kättelyn RADIUS-palvelimen kanssa periaatteessa suorittaa miltä tahansa koneelta. Riittää että tietää käyttäjän salasanan ja käyttäjätunnuksen. [25]

Ongelma voidaan ratkaista tai riskiä vähentää merkittävästi, jos asiakasohjelmisto asetetaan toimimaan oikein. Se voidaan asettaa hyväksymään vain tietyt tunnistusmenetelmät ja tarkistamaan palvelimen varmenteen oikeellisuus. PEAP-tunnistuksesta löydettyjen heikkouksien takia hyökkäys sopivissa olosuhteissa on kuitenkin osoitettu mahdolliseksi. [25]

4.6.2 Heikot salasanat

Suurin riski, kuten kaikissa muissakin salasanaan perustuvissa tunnistusmenetelmissä ovat heikot salasanat. Jos käyttäjä valitsee liian heikon salasanan, joka on helposti arvattavissa, ei mikään suojaus ole riittävä. [25] Käyttäjän käyttäjätunnuksen selvittäminen PEAP-tunnistuksessa on melko helppoa pelkästään verkkoliikennettä seuraamalla, koska ensimmäisissä RADIUS-viesteissä käyttäjätunnus välitetään myös selväkielisenä. Näin hyökkäys yksinkertaisesti kokeilemalla eri salasanoilla olisi mahdollista. Valimo iDServer tarjoaa kuitenkin mahdollisuuden kertakäyttösalasanan käyttöön tai oman salasanan ja kertakäyttösalasanan yhdistelmän käyttöön. Näin ollen vaikka hyökkääjä olisi saanut käyttäjän käyttäjätunnuksen selville, hänen kirjautuessa järjestelmään täytyisi hänen saada haltuunsa saman käyttäjän matkapuhelin, johon salasana tekstiviestillä joka kirjausyrityksen yhteydessä lähetetään. Salasana arvaaminen on vaikeaa, koska nyt toteutetussa 802.1x-ratkaisussa käyttäjä voi yrittää syöttää oikeaa salasanaa vain kerran. Seuraavalla yrityksellä hänelle lähetetään jo uusi salasana. Lisäksi käyttäjä, jonka käyttäjätunnusta murtautumisyrityksessä käytetään, huomaa helposti joutuneensa hyökkäyksen kohteeksi saatuaan tekstiviestejä vaikka ei ole yrittänyt kirjautua järjestelmään.

Lisäturvaa voidaan hakea myös muilla ratkaisuilla, esimerkiksi toteuttamalla järjestelmä siten, että käyttäjätunnuksen käyttö estetään liian monen epäonnistuneen kirjausyrityksen jälkeen. Jos kertakäyttösalasanaa käytetään yhdessä oman salasanan kanssa, täytyy hyökkääjän saada varastettua käyttäjän oman salasanan lisäksi käyttäjän

matkapuhelin, johon kertakäyttösalasana lähetetään. Edellä mainittujen lisäksi on mahdollista toteuttaa myös käyttäjän sirukortilla tai tietokoneelle tallennetun varmenteen tarkistaminen. Järjestelmän turvallisuutta erilaisia hyökkäyksiä vastaan on siis mahdollista kehittää ja nostaa korkeammalle tasolle tarpeen mukaan.

5 KÄYTÄNNÖN TOTEUTUS

Suurin osa työstä alkaen määritysten etsimisestä ja sovelluksen suunnittelusta aina ohjelmointiin saakka on tehty itse tämän diplomityön teon yhteydessä. Ongelmatilanteissa yrityksen muut työntekijät katselmoivat ohjelmakoodia mahdollisten virheiden löytämiseksi ja auttoivat näin ongelmien ratkaisuissa. Valmiita olemassa olevia kirjastoja ja olemassa olevaa ohjelmakoodia hyödynnettiin tai muokattiin siten että se toimi uusien tunnistustapojen kanssa mahdollisuuksien mukaan. Käytännössä koko PEAP-tunnistus mukaan lukien TLS-kättely, avainten sekä varmenteiden laskenta, lukuun ottamatta itse salausalgoritmeja ja tiivisteiden laskentaa, on tehty itse tämän projektin toteutusvaiheessa. Testilaitteiden asennus, testiympäristön pystyttäminen, valmiin sovelluksen systeemitestaus ja asennus tuotantoympäristöön tapahtui pääosin yrityksen muiden työntekijöiden toimesta.

Käytännöntoteutuksen tekeminen aloitettiin siten, että tehtävänmäärittelyn ja määritysten löytymisen jälkeen ensimmäisessä vaiheessa etsittiin ja tutustuttiin mahdollisesti olemassa olevia käyttökelpoisiin kirjastoihin, niiden tarjoamiin toimintoihin ja rajapintoihin.

5.1 Olemassa olevat sovellukset, hyödynnettävät kirjastot ja työkalut

Olemassa olevia kirjastoja tarkasteltaessa pyrittiin ensisijaisesti selvittämään jo tällä hetkellä käytössä olevien kirjastojen hyödyntämistä tai mahdollisesti avoimeen lähdekoodiin perustuvia ilmaiseksi saatavilla olevia kirjastoja. Avoimeen lähdekoodin perustuvat kirjastojen käyttökelpoisuutta tarkasteltaessa on myös varmistettava niiden lisensseistä se, sallitaanko kirjaston hyödyntäminen kaupallisessa sovelluksessa. Toki mahdollisuus myös kaupallisen kirjaston hankkimiselle tarkoitusta varten oli olemassa, jos olisi löytynyt vaihtoehto, joka olisivat merkittävästi vähentäneet työmäärää.

5.1.1 Java SDK

Java SDK (Software Development Kit) tarjoaa työkalut ja mahdollisuudet salattujen yhteyksien avaamiseen käyttäen normaaleja TCP-yhteyksiä (Transmission Control

Protocol). Java SDK ei kuitenkaan tarjoa riittävästi työkaluja yhteyden avaamiseen UDP-yhteyden kautta, eikä rajapintoja joilla SSL(Secure Sockets Layer)/TLS-kättely olisi voitu itse toteuttaa UDP-yhteyden yli. Java SDK:n lähdekoodit ovat myös saatavilla. Mahdollista toteutusta varten käytiin läpi lähdekoodeista saatavilla olevat osat, mutta SSL/TLS-kättelyyn liittyvät mahdollisesti työssä hyödynnettävät osuudet eivät olleet julkisesti saatavilla. [26]

5.1.2 AXL RADIUS -kirjasto

Valimo iDServer:n RADIUS-toiminnot on rakennettu AXL RADIUS -kirjaston päälle. Kirjaston dokumentaatiossa on viitteitä siitä, että kirjasto joiltain osin tukisi EAP-protokollaa. Toteutusta tehdessä kyseisestä kirjastosta oli käytettävissä lähdekoodit. Käytännössä kirjastossa ei ole kuitenkaan käytettävissä mitään TLS-kättelyyn liittyviä osia. Tuki rajoittuu ainoastaan EAP-tyyppisten RADIUS-pakettien lähettämiseen. MSCHAP- ja MSCHAPv2-tunnistukseen kirjastosta löytyy tuki. Lisäksi kirjastosta löytyy tuki avainten laskentaan ja joihinkin muihin tarvittaviin kättelyn osiin. [3]

Käytännössä kyseisen kirjaston EAP-toteutus osoittautui monilta osin varsin puutteelliseksi. Esimerkiksi avainten laskentaan liittyvä koodi oli käyttökelvoton. Eikä MSCHAPv2-tunnistuksen toteutus ollut täysin toimiva. Myös kirjaston käyttäjätunnusten käsittely oli puutteellista. Koska kirjaston lähdekoodit olivat käytettävissä, näiden virheiden ja puutteiden korjaaminen oli kuitenkin mahdollista. [3]

Kirjastoa ei selkeästi ole suunniteltu TLS-kättelyä ajatellen, puhumattakaan, että TLS-kättelyn jälkeen tehtäisiin tunnistus kirjastossa olevaa omaa MSCHAPv2-toteutusta hyödyntäen. Toimintatapa oli niin erilainen, että vain rajallinen osa kirjaston koodista oli hyödynnettävissä. Monien kirjaston toimintojen hyödyntäminen vaati sen, että lähes kaikki tehtiin uudestaan alkaen pakettien purkamisesta bittitasolla. Kun paketit oli saatu purettua, voitiin tiettyjen tarkisteiden ja muiden arvojen laskemiseen käyttää kirjaston valmiita toimintoja. [3]

5.1.3 BouncyCastle-kirjasto

BouncyCastle lienee yksi tunnetuimmista Java-kirjastoista, joka tarjoaa erilaisia toimintoja moniin kryptokrafisiin operaatioihin. Lähdekoodeja ja dokumentaatiot tutkittaessa kävi ilmi, että tämäkään kirjasto ei tarjoa TLS-kättelyyn liittyviä toimintoja valmiina. Ilmeisesti näiden toteuttamista ei ole pidetty tarpeellisena, koska Java tarjoaa toiminnot jo TCP-yhteyksille, mikä täyttää suurimman osan käyttäjien tarpeista. [27]

BouncyCastle-kirjasto tukee monia tiiviste- ja salausalgoritmeja. Nämä toiminnot on toteutettu kirjastossa niin hyvin, että niitä voitiin käyttää sellaisenaan. Suurimmat ongelmat kirjaston käytölle aiheuttikin käytettävän RADIUS-kirjaston ja RADIUS-protokollan toteutus. [27]

Käytännössä RADIUS-protokolla on yhteydetön. Eli jokainen RADIUS-kirjastolle saapuva paketti on riippumaton edellisestä paketista tai kirjasto käsittelee paketteja täysin riippumatta edellisestä paketista. TLS-kättely ja salaukset aiheuttavat riippuvuuden edellisiin paketteihin. Näin ollen oli välttämätöntä tehdä jonkinlainen istuntoihin perustuva toteutus. Käytännössä siis tähän istuntoon tallennetaan tieto kaikista edellisistä paketeista, koska niiden sisällön avulla lasketaan tarkisteet, joiden perusteella voidaan todeta yhteyden avaamisen ja kättelyn onnistuneen. Lisäksi salausavaimet, jaetut salaisuudet sekä salainten tilat on tallennettava tähän istuntoon. Kun salainten tilan tallentaminen oli toteutettu, oli myös BoyncyCastle-kirjaston käyttäminen mahdollista ja voitiin hyödyntää sen tarjoamia valmiita RSA-salaus ja RC4-virtasalain toteutuksia. [27]

Salaimet eivät sinänsä ratkaisevasti helpota työtä, koska ne eivät edelleenkään tarjoa työkaluja siihen, kuinka asiakasohjelmistolta vastaanotettu tieto saadaan avattua ja muutettua salainten tarvitsemaan muotoon. Salaimet voivat avata tiedon, kun se on saatu irrotettua RADIUS-paketeista. Kun salattu tieto on saatu avattua, on tieto jälleen käsiteltävä sopivalla tavalla, varmistettava käyttäjän henkilöllisyys tai vastattava viestiin halutulla tavalla. Tarvittavien toimenpiteiden jälkeen voidaan tieto jälleen salata käyttäen samoja salaimia. Tämän jälkeen salattu tieto on vielä lisättävä RADIUS-protokollan vastausviestin attribuutteihin oikeassa muodossa. BoyncyCastle-

kirjastosta löytyy salausalgoritmit valmiina, mutta sekään ei siis tarjoa valmista ratkaisua kuin pieneen osaan koko ongelmasta. [27]

5.1.4 FreeRADIUS-palvelin

FreeRADIUS on tunnettu ilmainen RADIUS-palvelin toteutus. Se on avoimeen lähdekoodiin perustuva ja saatavilla useiden eri Linux jakeluiden mukana. Toteutus on kuitenkin tehty C/C++:lla, eikä sen lisenssi olisi sopinut sovelluksen tai sen osien hyödyntämistä kaupallisessa tuotteessa. [28]

FreeRADIUS-palvelin tukee kuitenkin 802.1x-protokollakehyksen yli tapahtuvaa PEAP- ja MSCHAPv2-protokollaa käyttävää käyttäjäntunnistusta. Kyseinen toteutus oli siis lähestulkoon vastaava kuin tässä työssä oli tarkoitus tehdä. Tosin se ei tietenkään tukenut kertakäyttösalasanojen, niiden lähetystä tekstiviestillä tai muita vaadittuja ominaisuuksia. FreeRADIUS-palvelin käyttää OpenSSL-kirjastoa TLS-kättelyn toteuttamiseen. Käytännössä lisenssisyistä käyttäjän on käännettävä palvelinsovellus lähdekoodeista ja itse tehtävä käännöstiedostoihin vaadittavat muutokset, jotta palvelin saadaan tukemaan myös TLS-kättelyn vaatimia toimintoja. [28]

FreeRADIUS-palvelinta hyödynnettiin tätä työtä tehdessä siten, että sen avulla voitiin varmistaa käytettyjen kytkimien ja langattoman verkon tukiasemien toiminta. Lisäksi näiden verkkolaitteiden ja FreeRADIUS-palvelimen välinen keskustelu voitiin kaapata, jolloin saatiin käyttöön esimerkinomainen RADIUS-tunnistus. Näin voitiin nähdä, jos itse tehdyssä toteutuksessa oli jotain merkittäviä eroja itse pakettien sisällössä. Käytännössä suurin osa paketeista on kuitenkin salattuja, joten näiden osalta oli mahdollista tarkistaa ainoastaan pakettien pituuteen liittyvää tietoa. [28]

Merkittävimmät hyödyt FreeRADIUS-palvelimella tehdystä esimerkkikättelystä saatiin, kun sen avulla voitiin todeta AXL RADIUS -kirjaston laskevan tukiasemalle, tai kytkimelle lähetettävät salausavaimet väärin. Virhe havaittiin, kun verrattiin toimitettujen avainpakettien pituutta toisiinsa. Käytännössä täysin erilainen toiminta

voitiin havaita myös palvelimen lähdekoodeja tutkimalla. Tämän jälkeen oli mahdollista löytää oikea määritys, jonka mukaan AXL RADIUS -kirjaston avaintenlaskenta oli muokattava toimimaan. Toinen mahdollinen ongelmakohta, jota ei alun perin itse määrityksiä lukemalla havaittu, oli käyttäjätunnusten erilainen käsittely Microsoftin asiakasohjelman ollessa kyseessä. Ongelma oli itse palvelimessa korjattu lisäämällä asetuksiin parametri, joka asetti palvelimen toimimaan Microsoft -yhteensopivassa tilassa. [28]

FreeRADIUS-palvelinta voitiin pitää hyvänä esimerkkinä toimivasta sovelluksesta, vaikka käytännössä mitään sen osia ei voitu hyödyntää, johtuen sen lisenssistä ja käytetystä toteutuskielestä. [28]

5.1.5 JRADIUS

JRADIUS on Javalla toteutettu RADIUS-palvelin, jonka lähdekoodi on avoin. Lähdekoodeja ja dokumentaatioita tutkimalla kävi kuitenkin ilmi, että se on toteutukseltaan paljon puutteellisempi kuin AXL RADIUS -kirjasto on. Se tukee vielä vähemmän nyt tehtävässä sovelluksessa tarvittavia toimintoja kuin AXL RADIUS -kirjasto. Ilmeisesti TLS-kättelyn toteuttamista tässä projektissa on pidetty toistaiseksi liian hankalana. [29]

5.1.6 Wireshark (Ethereal)

Työkaluista hyödyllisimmäksi toteutusta tehtäessä ja virheitä selvittäessä osoittautui Wireshark, jota voi käyttää verkkoliikenteen kaappaamiseen. Wireshark tukee myös RADIUS-protokollaa ja RADIUS-pakettien avaamista. Sen avulla voitiin siis nähdä kaikki attribuuttien salaamaton sisältö liittyen TLS-kättelyyn. Lisäksi oli mahdollista verrata esimerkiksi salattujen pakettien pituuksia oman sovelluksen ja FreeRADIUS-palvelimen välillä. Näin oli mahdollista havaita virheet, vaikka itse virheen syytä ei voitukaan selvittää. Kyseinen sovellus osoittautui erittäin hyödylliseksi tätä työtä tehtäessä. [30]

5.1.7 Muut vaihtoehdot

Etsittäessä kirjastoa, joka olisi tukenut pelkästään TLS-kättelyä, löytyi muutama vaihtoehto. Nämä hylättiin kuitenkin hyvin nopeasti, koska käytännössä niistä saatu hyöty olisi ollut merkityksetön. Ne oli suunniteltu toteuttamaan kättely TCP-yhteyksien yli. Lisäksi ne toteuttivat tuen käyttäen algoritmeja, jotka eivät olisi nyt tehtävässä toteutuksessa tulleet kysymykseen. Kirjastojen muuttaminen toimimaan yhteydettömän UDP-yhteyden yli olisi käytännössä vaatinut vähintään yhtä paljon työtä kuin oman TLS-kättelyn toteuttaminen, joten käytännössä hyvin pian oman toteutuksen tekeminen osoittautui ainoaksi järkeväksi vaihtoehdoksi.

5.2 Valitut menetelmät

Eri standardit antavat paljon liikkumavaraa siihen, mitä menetelmiä RADIUS-palvelimen käytännössä on tuettava. Määritykset asettavat tietyt minimivaatimukset. Käytännössä aikataulun puitteissa ei ollut mahdollista toteuttaa kaikkia määrityksissä vaadittuja kättelymenetelmiä. Ensisijaisesti oli tarkoitus tehdä asiakkaan vaatimukset täyttävä sovellus. Tämä ei siis vaatinut kaikkien eri kättelymenetelmien toteutusta tai erikoistapauksien tukemista. RSA-salausta avaintenvaihdossa ja RC4-virtasalainta itse PEAP-kättelyssä päädyttiin käyttämään, koska nämä menetelmät ovat tuettu Windows-asiakasohjelmassa. Lisäksi nämä kuuluivat määrityksissä toteutettavaksi vaadittuihin menetelmiin. Näin ollen niiden toteuttaminen katsottiin hyödyllisemmäksi itse sovelluksen kannaltakin. Lisäksi näin vältyttiin tiettyjen TLS-kättelyn välivaiheessa välitettävien avaintenvaihtoviestien toteuttamiselta ja itse sovellus oli helpompi tehdä ja nopeampi saada valmiiksi.

RADIUS-palvelimen ei ole vaadittu tukevan istuntoja. Määritysten mukaan niitä on kuitenkin mahdollista tukea. Käytännössä istunnot olisivat osoittautuneet ongelmalliseksi. Jo nyt kättelyn ajaksi jouduttiin muodostamaan istunto, jotta itse sovellus toimisi ongelmitta. Istunnon säilyttäminen pitempiä aikoja olisi periaatteessa ollut mahdollista. Käytännössä näitä tilanteita pidettiin kuitenkin niin harvinaisena,

joten katsottiin riittäväksi, ettei varsinaisia istuntoja ainakaan ensimmäisessä vaiheessa toteuteta. Nämä eivät kuuluneet edes asiakkaan vaatimuksiin. Käytännössä tämä tapahtuisi tilanteessa, jossa henkilö, jolla on yhteys langattomanverkon tukiasemaan, siirtyisi pois ensimmäisen tukiaseman piiristä ja ottaisi yhteyden toiseen tukiasemaan, joka on osa samaa verkkoa. Tässä tilanteessa asiakasohjelmisto ottaisi yhteyttä tukiasemaan ja jos sillä olisi tieto, että RADIUS-palvelin olisi tukenut istuntoa, voisi se yrittää palata vanhaan istuntoon, jos palvelin tämän sallisi. Tätä ei siis tämän projektin puitteissa toteutettu.

Kirjastoista käytännön toteutukseen otettiin uutena käyttöön BoyncyCastle-kirjasto, joka tarjosi käytettäväksi valmiit salausalgoritmien toteutukset. Muilta osin mahdollisuuksien mukaan hyödynnettiin AXL RADIUS -kirjaston toiminnallisuuksia, jonka varaan Valimo iDServer oli rakennettu tai ne toteutettiin itse.

Kahdennus, joka oli yksi asiakkaan vaatimuksista, päädyttiin lopulta toteuttamaan kustannussyistä ilman jaettua tietokantaa. Tämä tarkoittaa käytännössä sitä, että keskeneräiset tunnistusyritykset tulevat epäonnistumaan. Palvelimella on ohjelma, joka tarkkailee palvelimen tilaa. Ongelmatilanteessa, jos palvelin havaitsee ohjelmiston kaatuneen, se käynnistää toisen varmistuspalvelimen ja antaa tälle saman IP-osoitteen. Näin ollen kahdennus on toteutettu itse palvelimessa, eikä verkkolaitteissa. Tietokantaratkaisu olisi ollut laitteiston puolesta liian kallis toteutettavaksi asiakkaan kannalta. Muina vaihtoehtoina esillä olivat mm. RADIUS-istuntoja hallitsevien kytkinten käyttö. Tämäkin kuitenkin hylättiin asiakkaan toimesta liian kalliina.

5.3 Ongelmat

Käytännöntoteutuksessa suurin ongelma oli lukuisat eri olemassa olevat määritykset. Esimerkiksi PEAP:sta ja eri EAP-menetelmistä löytyi monia määrityksiä eri nimillä ja eri versionumeroilla varustettuna. Lisäksi se, että käytännön toteutus koostui monista eri määrityksistä, joista muodostui yksi kokonaisuus, teki työstä monimutkaisen. Tietoa oli saatavilla paljon, osa tiedosta oli päällekkäistä ja epäselvyydet vaadittujen määritysten ja standardien versioista tekivät oikean tiedon löytämisestä hankalaa. Käytännössä oikeat dokumentit ja niiden versiot ongelmallisimmissa tapauksissa

löytyivät vertaamalla FreeRADIUS-palvelimen ja asiakasohjelman välillä kulkevia viestejä kaikkiin löydettyihin määrityksiin tai kokeilemalla.

FreeRADIUS-palvelin auttoi myös selvittämään valmistajakohtaiset poikkeukset. Jälkeenpäin kun ongelman syy paikannettiin, löytyi myös Microsoftin asiakasohjelman tekniseen toteutukseen liittyvä määrittelydokumentti. Käytännössä se oli lähes yhteensopiva myöhemmin julkaistun PEAP-määrityksen kanssa, käyttäjätunnuksen tarkisteen laskentaan liittyvää poikkeusta lukuun ottamatta. Microsoft välittää ryhmänimen käyttäjätunnuksen välityksen yhteydessä, kun muissa toteutuksissa ryhmänimeä ei huomioida kättelyn sisällä tapahtuvan tarkisteen laskemiseen.

Lisää ongelmia tuli esille testausvaiheessa. Ensimmäisessä vaiheessa toteutettu tunnistusmenetelmä tuki ainoastaan tunnistusta PEAP-tunnistusmenetelmällä. Sovellus tarkisti heti alussa, että käyttäjältä tullut paketti sisältää EAP-start paketin, merkkinä EAP-kättelyprosessin aloittamisesta. Normaalia käyttäjätunnukseen ja salasanaan perustuvaa tunnistusta ei tuettu. Laitteiden ylläpitäjät kirjautuvat reitittimiin ja kytkimiin kuitenkin telnet-yhteyden yli. Näin ollen he eivät suorita EAP-kättelyä, vaan tekevät perintisen RADIUS-tunnistuksen. Koska tukiasemat ja reitittimet oli määritelty käyttämään RADIUS-palvelinta käyttäjän tunnistukseen, myös ylläpitäjät tunnistetaan laitteisiin kirjauduttaessa RADIUS-palvelimen avulla. Näin valittujen tunnistusmenetelmien oli siis lisäksi tuettava normaalia perinteistä käyttäjätunnukseen ja salasanaan perustuvaa tunnistusta. Tämä ei sinänsä muodosta tietoturvauhkaa, koska mahdollisuus on vain ylläpitäjillä, joiden pääsy laitteisiin on rajattu. Näin ollen reitittimelle tai langattomaan verkkoon normaalisti kirjautuva käyttäjä ei voi edes yrittää päästä käsiksi laitteeseen telnet-yhteyden kautta. RADIUS-palvelimen oli siis tuettava ylläpitäjiä varten myös normaalia RADIUS-tunnistusta samanaikaisesti.

Toteutusta tehdessä virheiden löytäminen oli ongelmallista. Wireshark auttoi käytännön toteutusta tehdessä huomattavasti niin pitkään kun paketit olivat salaamattomia, sillä se pystyy avaamaan ja näyttämään EAP-pakettien sisällön. Kun kättelyprosessi oli saatu toteutettua niin pitkälle, että yhteys muuttui salatuksi, kävi virheiden löytäminen huomattavasti hankalammaksi. Wireshark auttoi näkemään, että paketit ovat väärän mittaisia, mutta muiden ohjelmakoodissa mahdollisesti olleiden

virheiden löytämiseen niistä ei enää ollut apua. Käytännössä virheiden selvittämisessä suurin osa ajasta kului vertailemalla bittijonoja määrityksiin ja yrittämällä selvittää eroavuudet niiden välillä. Epäselvyydet siitä mitä määritystä, esimerkiksi minkä PEAP:n version mukaisesti asiakasohjelma toimi, teki ohjelmakoodin virheiden selvittämisestä entistä haastavampaa.

Lisäksi itse testilaitteet ja verkkokorttien ajurit aiheuttivat ongelmia. Yhteensopivien laitteiden löytäminen oli ongelmallista. Tunnistuksen olisi pitänyt toimia esimerkiksi Windows 2000:n kanssa, mutta käytännössä testattuihin Windows 2000:lla varustettuihin kannettaviin ei vaadittuja päivityksiä saatu asennettua, koska jokin niihin asennettu päivitys oli liian uusi. Tämä esti vaaditun asiakasohjelmiston asentamisen tällä käyttöjärjestelmällä varustettuihin testikoneisiin.

Käytännönsovellusta tehdessä ongelmana oli lisäksi se, että verkkokortin ajurien vastaanottamat vialliset paketit saattoivat aiheuttaa tilanteita, jotka johtivat verkkokortin ajurien jumittumiseen. Näin ollen vaikka virhe oli jo korjattu, verkkokortti ei siitä huolimatta toiminut yhtään paremmin kuin aikaisemminkaan. Käytännössä monessa tilanteessa ongelma ratkesi vain käynnistämällä testikoneena käytetty tietokone uudestaan. Lisäksi asiakasohjelmasta ei saanut minkäänlaista tietoa siitä, mikä oli kirjautumisessa epäonnistunut tai mikä virhe oli tapahtunut.

5.4 Kertakäyttösalasana

Kertakäyttösalasanan toteuttamisen onnistumisesta ei ollut varmuutta ennen kuin toteutus oli saatu tehtyä niin pitkälle, että voitiin testata miten Windowsin asiakasohjelmisto reagoi vastaanottaessaan MSCHAPv2-protokollan mukaisen ilmoituksen virheellisestä salasanasta. Toteutus toimi tyydyttävällä tavalla. Ongelma ei johtunut RADIUS-palvelimen toteutuksessa, vaan siitä, ettei Windows:n asiakasohjelmistoa ole suunniteltu kertakäyttösalasanoja varten.

Käytännössä tunnistus eteni niin, että käyttäjää pyydettiin syöttämään uusi salasana. Ikkuna, johon salasana voitiin syöttää, oli auki pitempään kuin käyttäjällä todellisuudessa oli mahdollisuutta syöttää uutta salasanaa. Ongelma ei kuitenkaan

näkynyt käyttäjälle mitenkään. Windows-asiakasohjelmisto odottaa tietyn ajan, jonka jälkeen se yrittää jatkaa tunnistusprosessia taustalla ilmoittamatta tästä käyttäjälle millään tavalla. Jos käyttäjä syötti uuden salasanan, ikkuna sulkeutui, mutta tunnistus oli taustalla jo päättynyt epäonnistuneena. Verkkokortin ajureista riippuen tämä saattoi johtaa myös erilaisiin virhetilanteisiin. Vähintään virhetilanteiden toistuessa saatettiin joutua tilanteisiin, jotka eivät lauenneet muuten kuin käynnistämällä käyttöjärjestelmä uudestaan.

5.5 Testaus

Testilaitteena käytettiin Windows XP -käyttöjärjestelmällä varustettua kannettavaa ja Ciscon Catalyst 3500 -reititintä. Langaton verkko testattiin 802.1x-tunnistusta tukevan verkkokortin avulla. Asiakasohjelmistona käytettiin Windows XP:n omaa asiakasohjelmistoa laitteistoajurien mukana toimitetun ohjelman sijaan. Langattoman verkon tukiaseman mallilla ei sinänsä ole väliä. Laitteen ainoana vaatimuksena oli se, että se tukee 802.1x-standardia. Muilla ominaisuuksilla ei ollut merkitystä, koska laite toimii ainoastaan EAP-pakettien välittäjänä RADIUS-palvelimelle.

Reitittimen ja langattoman verkon tukiaseman toiminnan välillä merkittävin ero on siinä, että reitittimelle ei ole väliä, mitkä salausavaimet se vastaanottaa. Langattoman verkon tukiasema sen sijaan käyttää näitä avaimia yhteyden salaukseen sen muodostamisen jälkeen, joiden näiden on oltava oikein. Virhe avaimissa havaittiin vasta kun yhteyttä testattiin ensimmäisiä kertoja langattoman tukiaseman kanssa. Reitittimen kanssa palvelin toimi jo ongelmitta. Lisäksi langattoman verkon tukiasema teki myös RADIUS-accounting pyynnön normaalien RADIUS-pyyntöjen lisäksi.

Testilaitteina kokeiltiin myös muiden valmistajien sovelluksia ja ajureita. Testatessa havaittiin, että sovellukset toimivat hyvin eri tavalla riippuen käytettävistä laitteista. Siitä huolimatta, että laitteet olivat 802.1x-standardin mukaisia, suurin osa niistä vaati vähintään ajurien päivityksen toimiakseen. Aina tämäkään ei auttanut. Virhetilanteissa laitteet saattoivat toimia hyvin eri tavoilla. Jotkut laitteet toipuivat virhetilanteista hyvin, mutta toiset vaativat toipuakseen koko käyttöjärjestelmän uudelleen käynnistyksen. Virhetilanteella tarkoitetaan tässä yhteydessä esimerkiksi tilannetta,

jossa käyttäjä ei riittävän aikaisin syöttänyt puhelimeensa vastaanottamaasa kertakäyttösalasanaa.

5.6 Tulokset

Testauksessa todettiin, että tehty sovellus on toimiva ja täyttää asiakkaan tarpeet ja vaatimukset. Ongelmia voi kuitenkin esiintyä riippuen käytetystä laitteistoista. Nämä ongelmat ovat RADIUS-toteutuksesta riippumattomia ja aiheutuvat eri valmistajien laitteissa ja ajureissa olevista vioista. Ohjelmia ei ole suunniteltu kertakäyttösalasana tyyppistä käyttötarkoitusta varten.

5.6.1 Käyttökokemus

Järjestelmän merkittävin ongelma ei ole itse RADIUS-toteutus, vaan se miten itse Windows-asiakasohjelma on toteutettu. Käyttäjä joutuu muuttamaan verkkokorttinsa asetuksia siten, että se tukee PEAP-tunnistusta käyttäen MSCHAPv2-standardin mukaista käyttäjätunnukseen ja salasanaan perustuvaa menetelmää. Tämä on sinänsä helppo toimenpide, mutta voi vaatia ohjeistusta, jos käyttäjä on kokematon.

Merkittävämpi ongelma liittyen käyttökokemukseen on itse asiakasohjelmiston toimintatapa. Yhteydenotto ja salasanan kysely näkyy käyttäjälle järjestelmän alakulmaan ilmestyvänä Windows:lle tyypillisenä puhekuplana. Käyttäjän on puhekuplan päällä painettava hiiren painiketta kaksi kertaa, jos käyttäjä haluaa yrittää syöttää käyttäjätunnuksensa ja salasanansa. Kertakäyttösalasana ollessa kyseessä, järjestelmä pyytää käyttäjää uuden puhekuplan avulla syöttämään uuden salasanan, koska edellinen oli virheellinen. Tämä tapahtuu jälleen samalla tavalla hiiren painiketta kaksi kertaa puhekuplan päällä painamalla. Tähän ikkunaan järjestelmän asetuksista riippuen käyttäjä joutuu syöttämään käyttäjätunnuksen ja salasanansa uudestaan.

Toisena ongelmana käyttökokemuksessa on, että käyttäjän on toimittava riittävän nopeasti suoriutuakseen kirjautumisoperaatiosta. Suurimpana ongelmana tässä on jo aikaisemmin mainittu tilanne, jossa Windows-asiakasohjelmisto jatkaa toimintaansa taustalla käyttäjän tietämättä. Näin ollen, vaikka käyttäjä syöttää salasanan oikein,

kirjautuminen voi epäonnistua, koska palvelimelle on lähetetty jo uusi väärä vastaus asiakasohjelman toimesta. Tämä ei kuitenkaan näy käyttäjälle mitenkään. Tämä voi johtaa myös virhetilanteeseen, jonka seurauksena käytetystä laitteistosta riippuen käyttäjä voi jopa joutua käynnistämään Windows-järjestelmän uudelleen saadakseen sen toimimaan jälleen normaalisti.

Asiakasohjelmisto ei anna ainakaan oletusasetuksilla käyttäjän yrittää syöttää salasanaa ja käyttäjätunnusta kuin kaksi kertaa, vaikka itse protokollan määritysdokumenttien mukaan kolmea yritystä olisi suositeltavaa tukea. Käytännössä kertakäyttösalasanan ollessa kyseessä tämä tarkoittaa sitä että käyttäjällä ei ole mahdollisuutta yrittää salasanan syöttämistä kuin kerran, koska ensimmäinen yritys menee hukkaan käyttäjä joutuessa ensimmäisellä kertaa syöttämään käyttäjätunnuksensa, jonka perusteella kertakäyttösalasana lähetetään käyttäjän puhelimeen.

RADIUS-protokolla tukee viestin välittämistä käyttäjälle. Valimo iDServer:n ollessa kyseessä käyttäjälle välitetään viesti, jossa häntä pyydetään syöttämään puhelimeen lähetetty kertakäyttösalasana. Windows:n asiakasohjelmisto ei tue tämän viestin näyttämistä käyttäjälle. Näin tilanne näyttää käyttäjälle samalta kuin jos käyttäjä olisi syöttänyt väärän salasanan tai käyttäjätunnuksen. Koska virheilmoituksen viestiä ei näytetä jää, kirjausyrityksen epäonnistumisen syy käyttäjälle tuntemattomaksi.

Käyttäjän kannalta käyttökokemus on kaikin puolin erittäin heikko. Ongelmaa ei kuitenkaan pystytä ratkaisemaan palvelimen päässä, koska RADIUS-palvelin toimii sinänsä oikein. Windows:n asiakasohjelmiston toiminnassa on paljon parannettavaa käyttökokemuksen osalta. Osa ongelmista voidaan toki yrittää välttää ohjeistamalla käyttäjät kunnolla. Kokenutkaan käyttäjä ei välttämättä osaa toimia tilanteessa oikein hänen käyttäessä järjestelmää ensimmäistä kertaa. Ongelmia yritettiin ratkaista tai korjata myös selvittämällä voiko asiakasohjelmiston toimintaan vaikuttaa esimerkiksi käyttöjärjestelmän rekisteriasetuksia muuttamalla. Mitään toimivaa ratkaisua ei kuitenkaan löytynyt.

5.6.2 Yhteensopivuus

Toteutettu järjestelmä on yhteensopiva Windows XP-käyttöjärjestelmään päivityksenä saatavan asiakasohjelmiston kanssa tai Windows 2000-käyttöjärjestelmään erillisenä ladattavan asiakasohjelmiston kanssa. Windows 2000:n erillisenä ladattavan asiakasohjelmiston asennuksessa oli kuitenkin ongelmia. Jotta sen asentaminen olisi onnistunut, ei käyttöjärjestelmän viimeisimpiä päivityksiä saanut olla asennettuna. On kuitenkin mahdollista, että tähän ongelmaan tulee korjaus saataville myöhemmin.

Windows XP:n kanssa yhteensopivuudessa oli myös toivomisen varaa. Käytännössä järjestelmä toimi testeissä eri tavalla riippuen käytetystä verkkokortista ja verkkokortin ajureista. Tiettyjen ajurien kanssa tunnistus ei toiminut ollenkaan. Toisten valmistajien ajurit toimivat paremmin, mutta ongelmia tuli esimerkiksi tilanteissa jossa käyttäjä ei syöttänyt salasanaa ajoissa oikein. Tämän seurauksena verkkokortin ajurit saattoivat joutua tilaan, joka laukesi ainoastaan käynnistämällä käyttöjärjestelmä uudestaan. Ongelmaan ratkaisua etsittäessä tiettyjen verkkokortin ajureiden kanssa ongelma ratkesi, jos verkkokortin asetuksiin liittyvät Windows:n rekisterit palautettiin oletusasetuksille tarkoitusta varten tehdyllä rekisteritiedostolla. Tämä ratkaisu toimi kuitenkin vain muutamien verkkokorttien kanssa.

RADIUS-palvelimen osalta tehty toteutus on tällä hetkellä yhteensopiva vain Windows:n asiakasohjelman kanssa. Yhteensopivuutta muiden valmistajien ohjelmien kanssa ei testattu, koska tätä ei vaadittu. Oletettavaa kuitenkin on, johtuen Windowsin käyttäjätunnuksen käsittelytavasta, että palvelin ei ole yhteensopiva muiden valmistajien ratkaisujen kanssa.

6 JOHTOPÄÄTÖKSET JA SUOSITUKSET

802.1x-standardin toteuttaminen UDP-yhteyden yli vaatii merkittävän määrän oman ohjelmakoodin tuottamista. RADIUS-palvelimen kannalta, kaikki toimii varsin hyvin. Suurin ongelma on itse asiakasohjelmien toiminnassa. Tämän hetkiset asiakasohjelmat on suunniteltu pelkästään käyttäjätunnuksen ja salasanan avulla tapahtuvaa tunnistusta varten. Kertakäyttösalasanan toteuttaminen on mahdollista, mutta ratkaisu ei missään nimessä ole kovin käyttäjäystävällinen.

Itse projekti todettiin huonosta käyttäjäystävällisyydestä ja aikatauluongelmista huolimatta onnistuneeksi. Asiakas on hyväksynyt projektin tuloksena tuotetun sovelluksen ja ottanut sen käyttöön omassa verkossaan.

Yksi menetelmän merkittävimmistä ongelmista on tällä hetkellä verkkokorttien ajurit ja heikosti toimivat asiakasohjelmisto. Oletettavaa on kuitenkin, että verkkokorttien valmistajat ja Microsoft jatkavat asiakasohjelmiston ja ajurien kehitystä. Langattomat verkot ovat yleistyneet huomattavasti. Niitä on tarjolla niin julkisia kuin yksityisiä ja niiden käyttö on lisääntynyt myös yrityksissä. Tulevaisuudessa ne vaativat parempaa suojausta ja entistä vahvempaa käyttäjäntunnistusta. Erityisesti yritykset ovat varmasti kiinnostuneita suojattujen yhteyksien järjestämisestä langattoman verkon kautta omille työntekijöilleen. Näin ne voivat tarjota heille pääsyn yrityksen resursseihin tämän verkon kautta turvallisesti, kun käyttäjät on ensin tunnistettu vahvasti käyttäen esimerkiksi kertakäyttösalasanaa. Normaalissa tapauksessa käyttäjätunnusta ja salasanaa käytettäessä nämä tiedot voidaan periaatteessa varastaa käyttäjän sitä tietämättä. Matkapuhelimeen tekstiviestillä lähetettävän kertakäyttösalasanan ollessa kyseessä voidaan olla kuitenkin varma siitä, että verkkoon kirjautuvalla käyttäjällä on edellisten tietojen lisäksi oltava hallussaan myös käyttäjän matkapuhelin. Uusien käyttäjäntunnistustapojen yleistyminen johtaa varmasti väistämättä myöhemmin myös parempien asiakasohjelmien kehittymiseen.

Nyt toteutettu osuus protokollasta avaa huomattavat mahdollisuudet sovelluksen jatkokehitystä ajatellen. Ensisijaisesti seuraavissa versiossa lienee syytä lisätä palvelimeen tuki istunnoille. Näin mahdollistetaan käyttäjän siirtyminen tukiasemalta

toiselle ja palaaminen siirtymisen jälkeen samaan istuntoon. Tämä vaatii kuitenkin lisäselvityksiä ja mahdollisesti tuen lisäämistä myös muille protokollille. Verkkolaitteista riippuen istunnon siirto voi tapahtua myös muilla protokollilla, jotka on rakennettu RADIUS-protokollan päälle. Käytetty tapa riippuu verkkolaitteista.

RADIUS-palvelimen toiminta on syytä toteuttaa ja testata myös muilla kuin Windows:n asiakasohjelmilla. Ne voivat vaatia pieniä muutoksia nykyiseen toteutukseen. Lisäksi palvelimeen on syytä lisätä määritysten vaatimat muut avaintenvaihto- ja kättelyprotokollat. Nykyinen ratkaisu on suunniteltu ainoastaan täyttämään asiakkaan vaatimukset ja näin ollen se tarjoaa vain yhden ainoan avaintenvaihtotavan.

Tällä hetkellä palvelin käyttää ainoastaan omaa varmennetta salaukseen, eikä asiakkaan varmenteen käyttöä tueta. Itse RADIUS-toteutusta on mahdollista laajentaa melko pienellä vaivalla tukemaan asiakkaan varmenteen vastaanottoa ja tarkistusta. Tulevaisuudessa varmenteen tarkistamiseen olisi mahdollista hyödyntää muita Valimon tuotteita. EAP-toteutus mahdollistaa lisäksi esimerkiksi älykorttien käyttämisen tunnistusvälineenä. Nykyinen toteutus ei ole riittävä tätä tarkoitusta varten, mutta nyt tehdyn työn jälkeen näiden laajennusten tekeminen on mahdollista suhteellisen pienellä vaivalla. Näin Valimo iDServer:n tukemia tunnistustapoja olisi mahdollista laajentaa entisestään.

Yrityksen kannalta yksi kiinnostavimmista tunnistustavoista lienee tunnistus Valimo Validator-MSSP-palvelinta vastaan käyttäen esimerkiksi matkapuhelimen sim-kortille tallennettua mobiilivarmennetta. Windows-asiakasohjelmistoa käytettäessä ongelmana on jo nyt heikko käyttäjäystävällisyys. Mobiilivarmenteen kanssa tunnistus kestää käytännössä kuitenkin vähintään 30 sekuntia, useimmiten minuutteja. Nykyinen Windows-asiakasohjelmisto ei toimi näin pitkää tunnistusta vaativilla menetelmillä. Käytännössä ainoa toteutus tapa on toteuttaa se perustuen useamman kirjautumisyrityksen yhdistelmänä. Tämä johtaa entistä huonompaan käytettävyyteen, koska käyttäjälle ei saada välitettyä minkäänlaisia viestejä siitä mitä on tapahtumassa. Lisäksi eri verkkokorttien kanssa on jo nykyisellä tavalla ongelmia, jos tunnistus kertaalleen epäonnistuu. Tunnistustapa lienee toteutettavissa, mutta se vaatii

huomattavasti lisää suunnittelua. Ratkaisu on myös testattava huolella ennen sen käyttöönottoa.

Mahdollisuudet uusien sovelluksien kehittämiselle nyt rakennetun perustan päälle ovat olemassa. Tulevissa versioissa nykyistä toteutusta tullaan varmasti parantamaan ja uusia toimintoja lisäämään vaatimusten ja asiakkaiden tarpeiden mukaan. Nykyinen EAP-toteutus mahdollistaa kuitenkin kymmenien erilaisten vahvempien tunnistustapojen ja protokollien toteuttamisen ja käyttämisen.

7 YHTEENVETO

Projektin tarkoitus oli kehittää Valimo iDServer RADIUS -palvelimeen uusi langattomia verkkoja ja VLAN-kytkimiä tukeva Windows:n asiakasohjelmalla toimiva tunnistusmenetelmä. Käyttäjien tunnistus tapahtuisi tässä normaalin käyttäjätunnuksen ja salasanan tai tekstiviestillä lähetettävän kertakäyttösalasanan avulla.

Menetelmässä kommunikointi asiakasohjelman ja verkkolaitteen välillä tapahtuu EAP-protokollan mukaisilla paketeilla. Verkkolaite pakkaa EAP-paketit RADIUS-viestin attribuutteihin ja lähettää ne RADIUS-palvelimelle. Verkkolaite toimii ainoastaan EAP-pakettien välityspalvelimena. Toimiakseen Windows:n oman asiakasohjelman kanssa vaati tämä tuen toteuttamista PEAP-protokollalle RADIUS-palvelimeen. PEAP on kehitetty EAP-standardista. Määritysten mukaisesti PEAP-tunnistuksen ensimmäisessä vaiheessa tehdään normaali EAP-TLS-kättely. Toisessa vaiheessa tunnistus tehdään salatun yhteyden yli käyttäen toista mahdollisesti heikommin suojattua tunnistusmenetelmää. Kyseisessä toteutuksessa tämä tehtiin siis käyttäen MSCHAPv2-standardin mukaista käyttäjätunnukseen ja salasanaan perustuvaa menetelmää.

Monista eri määrityksistä huolimatta palvelimen sovelluksesta saatiin lopulta tehtyä toimiva, yhteensopiva ja asiakkaan vaatimusten mukainen. Lukuisten eri standardien ja määritysten sekä niiden eri versioiden takia oikeiden dokumenttien löytäminen oli hankalaa. Lisäksi huonosti toteutetut verkkokortin ajurit ja muut lähinnä asiakasohjelmistoon ja sen toimintaan liittyvät ongelmat toivat omat haasteet virheiden löytämiselle ja korjaamiselle.

Lopulta oikeiden määritysten löydyttyä osittain jopa kokeilemalla, onnistuttiin rakentamaan vaatimukset täyttävä toimiva sovellus. Myös kertakäyttösalasana tunnistuksen toteuttaminen todettiin mahdolliseksi alun perin suunnitellulla tavalla. Vaikka itse palvelin toimii oikein, ei käyttökokemus ole asiakkaan kannalta kovinkaan käyttäjäystävällinen. Suurin osa ongelmista liittyy itse asiakasohjelmiston toteutukseen ja verkkokortin ajureihin. Näihin ei voi vaikuttaa palvelimessa tai sen toiminnalla.

Ehkä tulevaisuudessa vastaavantyyppisten vahvempien tunnistusmenetelmien yleistyessä myös asiakasohjelman toimintaa parannetaan.

Nyt tehty palvelinsovellus on kuitenkin sinänsä toimiva ja se luo pohjan tulevien sovellusten ja tunnistusmenetelmien kehittämiselle. Menetelmä tuo selkeästi lisäturvaa, jos sitä verrataan esimerkiksi tilanteeseen, jossa käytetään heikkoa staattista kaikille yhteistä salasanaa. Käytännössä tällainen ratkaisu on liian turvaton suurimmalle osalle yrityksiä, joten langattomat verkot eivät ole näissä yleistyneet tietoturvasyistäkään. Kertakäyttösalasanan käyttöönotto ei vaadi suuria investointeja yritykseltä verrattuna esimerkiksi varmennekorttien tai muiden vastaavien vahvojen tunnistusmenetelmien käyttöön, koska suurimmalla osalla työntekijöitä on jo matkapuhelin. Tarvittaessa varmenteiden tai varmennekorttien käytön toteuttaminen on nykyisessä ratkaisussa toteutettavissa huomattavasti pienemmällä vaivalla, jos siihen on tarvetta.

Projektia voidaan pitää onnistuneena, koska se täytti asiakkaan vaatimukset, mahdollistaa tuotteen jatkokehityksen sekä tarjoaa uusia entistä monipuolisempia ominaisuuksia palvelimen käyttöä ajatellen. Lisäksi tuki kertakäyttösalasanatunnistukselle esimerkiksi langattomiin verkkoihin tuo merkittävästi lisäturvaa ja mahdollistaa langattomien verkkojen käytön entistä laajemmin myös yrityksissä ja se voidaan yhdistää ongelmitta keskitettyyn pääsynhallintajärjestelmään.

LÄHTEET

[1] Nokia Corporation, Nokia SMS Center 7.0 - CIMD Interface Specification, Document Identifier dn03305755 Issue 2.0 en, [verkkodokumentti], 12.2004, [viitattu 1.4.2006], saatavissa: http://sw.nokia.com/id/a58b0133-4ffa-4e17-8b3b-77877688660f/CIMD_Interface_Specification_SC70.pdf

[2] CCC Group, CCC Content gateway, [verkkodokumentti], 2005, [viitattu 1.4.2006] saatavissa: http://www.cgwbusiness.com/

[3] AXL Software, AXL RADIUS Server & RADIUS Client API, [verkkodokumentti], 2004, [viitattu 1.4.2006], saatavissa: http://www.axlradius.com

[4] C. Rigney, S. Willens, A. Rubens, W.Simpson, RFC 2865 – Remote Authentication Dial In User Service (RADIUS), [verkkodokumentti], 6.2000, [viitattu 14.4.2006], saatavissa: http://tools.ietf.org/html/rfc2865

[5] IEEE Standards 802.1x IEEE standards for Local and metropolitan area networks. Port-Based Network Access Control. 13.12.2004 PDF: SS95298

[6] B. Adoba, L. Blunk, J. Vollbrecht, J. Carlson, H Levkowetz, RFC 3748 – Extensible Authentication Protocol (EAP), [verkkodokumentti], 6.2004, [viitattu 14.4.2006], saatavissa: http://tools.ietf.org/html/rfc3748

[7] B.Adoba, P. Calhoun, RFC 3579 – RADIUS (Remote Autnethication Dial In User Service) Support For Extensible Authentication Protoccol (EAP), [verkkodokumentti], 9.2003, [viitattu 14.4.2006], saatavissa: http://tools.ietf.org/html/rfc3579

[8] T. Dierks, C.Allen, RFC 2246 – The TLS Protocol Version 1.0, [verkkodokumentti], 1.1999, [viitattu 14.4.2006], saatavissa: http://tools.ietf.org/html/rfc2246

[9] B. Adoba, D.Simon, RFC 2716 – PPP EAP TLS Authentication Protocol, [verkkodokumentti], 10.1999, [viitattu 14.4.2006], saatavissa: http://tools.ietf.org/html/rfc2716

[10] H. Andersson, Internet-Draft, Protected EAP Protocol (PEAP), [verkkodokumentti], 22.2.2002, [viitattu 14.4.2006], saatavissa: http://www.drizzle.com/~aboba/IEEE/draft-josefsson-pppext-eap-tls-eap-02.txt

[11] Krishna Sankar, Andrew Balinsky, Darrin Miller, Sri Sundaralingam, EAP Authentication Protocols for WLANs, [verkkodokumentti], 18.2.2005, [viitattu 14.4.2006], saatavissa: http://www.ciscopress.com/articles/article.asp?p=369223

[12] Vivek Kamath, Ashwin Palekar, Mark Wodrich, Microsoft's PEAP version 0 (Implementation in Windows XP SP1), [verkkodokumentti], 25.10.2002, [viitattu: 14.4.2006], saatavissa: http://www.drizzle.com/~aboba/IEEE/mspeap.zip dokumentti: draft-kamath-pppext-peapv0-00.txt

[13] G Zorn, RFC 2759 – Microsoft PPP Chap Extension, Version 2, [verkkodokumentti], 1.2000, [viitattu: 5.5.2006], saatavissa: http://tools.ietf.org/html/rfc2759

[14] W. Simpson, RFC 1994 – PPP Challenge Handshake Authentication Protocol (CHAP) , [verkkodokumentti], 8.1996, [viitattu: 5.5.2006], saatavissa: http://tools.ietf.org/html/rfc1994

[15] G. Zorn, S. Cobb, RFC-2433 - Microsoft PPP CHAP Extensions, [verkkodokumentti], 10.1998, [viitattu: 5.5.2006] saatavissa: http://tools.ietf.org/html/rfc2433

[16] Bruce Schneier, Applied Cryptography, second edition.. John Wiley & Sons, Inc. 1996

[17] H. Krawczyk, M Bellare, R Canetti, RFC-2104 - HMAC: Keyed-Hashing for Message Authentication, [verkkodokumentti], 2.1997, [viitattu: 3.10.2006], saatavissa: http://tools.ietf.org/html/rfc2104

[18] D. Eastlake, P. Jones, RFC-3174 - US Secure Hash Algortihm 1 (SHA1), [verkkodokumentti], 9.2004, [viitattu: 3.10.2006], saatavissa: http://tools.ietf.org/html/rfc3174

[19] R. Rivest, RFC-1321 – The MD5 Message-Digest Algorithm, [verkkodokumentti], 4.1992, [viitattu: 3.10.2006], saatavissa: http://tools.ietf.org/html/rfc1321

[20] G. Zorn, RFC-2548 – Microsoft Vendor-specific RADIUS attributes, [verkkodokumentti], 3.1999, [viitattu: 3.10.2006], saatavissa: http://tools.ietf.org/html/rfc2548

[21] Arjen K. Lenstra, Further progress in hashing cryptoanalysis, [verkkodokumentti], 26.2.2005, [viitattu: 5.10.2006], saatavissa: http://cm.bell-labs.com/who/akl/hash.pdf

[22] Federal Information Processing Standards Publication 198, The Keyed-Hash Message Authentication Code (HMAC), [verkkodokumentti], 6.3.2002, [viitattu: 17.10.2006], saatavissa: http://csrc.nist.gov/publications/fips/fips198/fips-198a.pdf

[23] Marco Carli, Andrea Rossetti, Alessandro Neri, Integrated Security Architecture for WLAN, IEEE Telecommunications, 2003, ICT 2003, vol 2, 23.2-1.3.2003, sivut: 943-947.

[24] Frankie Chan K.L., Ang Hee Hoon, Biju Issac, Analysis of IEEE 802.11b Wireless Security for University Wireless LAN Design, IEEE Networks, 2005, vol 2, 16-18.11.2005, sivut: 1137-1142.

[25] N. Asokan, Valtteri Niemi, Kaisa Nyberg, Man-in-the-Middle in Tunneled Authentication Protocols, [verkkodokumentti], 11.11.2002, [viitattu: 25.10.2006], saatavilla: http://eprint.iacr.org/2002/163.pdf

[26] Sun Microsystems, Sun Developer Network, [verkkodokumentti], 2006, [viitattu: 10.10.2006], saatavilla: http://java.sun.com/

[27] Bouncycastle porject, The Legion of the Boncy Castle, [verkkodokumentti], 2006, [viitattu 10.10.2006], saatavilla: http://www.bouncycastle.org/

[28] The FreeRADIUS Project, FreeRADIUS, [verkkodokumentti], 19.9.2006, [viitattu 10.10.2006], saatavilla: http://www.freeradius.org/

[29] JRADIUS project, JRADIUS, [verkkodokumentti], 10.10.2006, [viitattu 10.10.2006], saatavilla: http://jradius.org/wiki/index.php/Main_Page

[30] Wireshark, Wireshark, [verkkodokumentti], 18.9.2006, [viitattu 10.10.2006], saatavilla: http://www.wireshark.org/