TIETO- JA SÄHKÖTEKNIIKAN TIEDEKUNTA

Julius Hekkala

JULKISESSA TILASSA OLEVAN EPÄLUOTETUN LAITTEEN KÄYTTÖÄ SUOJAAVA JÄRJESTELMÄ

Kandidaatintyö
Tietotekniikan tutkinto-ohjelma
Syyskuu 2019

Hekkala J. (2019) Julkisessa tilassa olevan epäluotetun laitteen käyttöä suojaava järjestelmä. Oulun yliopisto, Tietotekniikan tutkinto-ohjelma, 29 s.

TIIVISTELMÄ

Tässä työssä esitellään toteutus järjestelmästä, jossa käyttäjä kommunikoi luotetun palvelimen kanssa epäluotetun laitteen välityksellä. Palvelimen lähettämä informaatio näkyy käyttäjälle salattuna visuaalisessa muodossa QR-koodeina epäluotetun laitteen käyttöliittymässä, josta käyttäjä lukee henkilökohtaisella älypuhelimellaan QR-koodeja. Sovellus käyttäjän kännykässä purkaa salauksen ja näyttää käyttäjälle viestien salatun sisällön. Järjestelmää voi käyttää molemminsuuntaiseen kommunikointiin palvelimen kanssa. Se vaikeuttaa julkisissa tiloissa olevien järjestelmien vaarantamista ja käyttäjän selän takaa kriittisen informaation selville saamista. Järjestelmä estää myös siihen kohdistuvia aktiivisia hyökkäyksiä.

Avainsanat: kryptografia, visuaalinen kryptografia, käyttäjän passiivinen tarkkailu, passiivinen hyökkäys, aktiivinen hyökkäys, EyeDecrypt.

Hekkala J. (2019) A system that protects the use of an untrusted device in a public space. University of Oulu, Degree Programme in Computer Science and Engineering, 29 p.

ABSTRACT

In this bachelor's thesis an implementation of a cryptographic system is introduced. In the system the user communicates with a trusted server through an untrusted device. The user is shown the data sent by the server encrypted and visually encoded on the GUI of the untrusted device. The user uses their own smart phone to scan and read QR codes on the screen of the untrusted device. An application on the smart phone decrypts the content and shows the user the plaintext. The implemented system can be used for two-way communication between the user and the server. It prevents shoulder-surfing attacks and improves the safety of using untrusted devices in public spaces. The system also protects against active attacks.

Keywords: cryptography, visual cryptography, shoulder-surfing attacks, passive attacks, active attacks, EyeDecrypt.

SISÄLLYSLUETTELO

ALKULAUSE

Tämä kandidaatintyö tehtiin osana CyberSec4Europe-projektia VTT:llä. Haluaisin kiittää VTT:tä ja Kimmo Halusta mahdollisuudesta tehdä kandidaatintyö näin kiinnostavasta aiheesta. Haluaisin kiittää Teemu Tokolaa ja Kimmoa työn ohjaamisesta ja professori Juha Röningiä työn tarkistamisesta.

Oulu, 11. syyskuuta 2019

Julius Hekkala

LYHENTEIDEN JA MERKKIEN SELITYKSET

HMAC	Hash-based message authentication code, viestin autentikointikoodi
AES	Advanced Encryption Standard, symmetrinen lohkosalausalgoritmi
SHA-256	Secure Hash Algorithm, hash-algoritmi
XOR	exclusive or, poissulkeva tai
QR-koodi	Quick Response code, viivakoodityyppi
PIN	personal identification number, henkilökohtainen tunnusluku
C	yhden QR-koodin sisältö
C_0	salateksti
τ	salatekstin HMAC-autentikointikoodi
i	viestin lohko
f	viestin kehys
IV	initialization vector, salausalgoritmin alkumuuttuja

1. JOHDANTO

Nykyään yhä suurempi osa ihmisten toiminnasta tapahtuu erilaisten tietojärjestelmien kautta. Suuri osa maksutapahtumista tapahtuu internetissä erilaisissa verkkokaupoissa. Ihmiset käyttävät pankki- ja muita palveluja internetin välityksellä ja maksavat korteillaan ostoksista, joista tieto sitten tietojärjestelmien välityksellä kulkee pankille. Tällaisissa tietojärjestelmissä on olennaista, että ne ovat turvallisia, eivätkä pahaa tahtovat tahot pääse käsiksi ihmisten tai yritysten tietoihin tai häiritsemään toimintaa. Mikäli tällaisten järjestelmien turvallisuuteen ei voisi luottaa, ei esimerkiksi maksutapahtumien suorittaminen verkon välityksellä olisi ollenkaan mahdollista. Yksi keino taata turvallisuutta tällaisissa tietojärjestelmissä on salata tietojärjestelmässä kulkeva liikenne kryptografian keinoin.

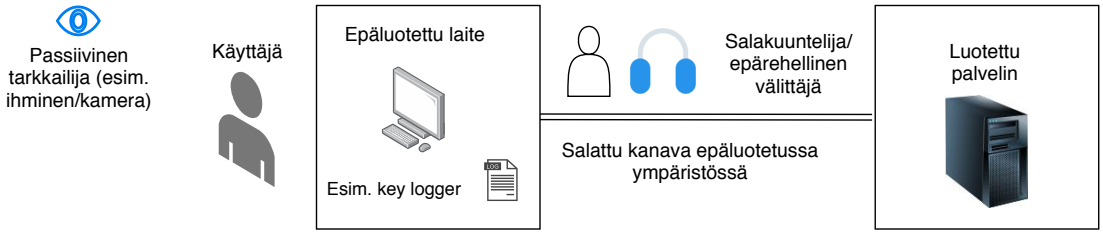
Kryptografiassa keskeinen ajatus on estää muita kuin viestin lähettäjää ja sen haluttua lukijaa selvittämästä, mitä lähetetyssä viestissä piilee. Esimerkiksi verkkokaupankäynnissä on sekä käyttäjän että kaupan etujen mukaista, ettei niiden välistä liikennettä joku pääse kuuntelemaan tai jopa muokkaamaan ja täten aiheuttamaan vahinkoa asiakkaalle tai kauppiaalle. Kryptografiaan liittyy useita erilaisia salauskeinoja ja osa-alueita. Esimerkkejä alueista ovat Diffien ja Hellmanin ensimmäisen kerran vuonna 1976 esittelemä julkisen avaimen salaus [1] ja erilaiset viestien autentikointikoodit, kuten HMAC [2], joita käytetään tarkistamaan, ettei lähetetty viesti ole matkan aikana muuttunut ja että viestin lähettäjä on oikea.

Tällaiset kryptografiset menetelmät suojaavat käyttäjän dataa, mikäli hän voi luottaa käyttämäänsä laitteeseen ja olla varma siitä, ettei hänen toimiaan kukaan tarkkaile esimerkiksi selän takaa tai kameran välityksellä. Jos joku pahaa tarkoittava saa tietoonsa käyttäjän syötteet tai käyttäjälle tarkoitetun informaation tällä tavoin, ei tiedon salaamisesta ole käyttäjälle minkäänlaista hyötyä. Suojatakseen käyttäjän yksityisyyttä esimerkiksi julkisissa tiloissa pankkiautomaattia käytettäessä on otettava huomioon ulkopuoliset tarkkailijat.

Tässä työssä toteutettiin versio Forten ym. esittelemästä toteutuksesta nimeltä *EyeDecrypt* [3], jossa ihmiskäyttäjä ja luotettu palvelin kommunikoivat epäluotetun laitteen välityksellä. Toteutuksessa informaatio liikkuu palvelimelta käyttäjälle salattuna ja visuaalisesti koodattuna. Toteutuksen tarkoituksena on estää sekä käyttäjän toiminnan ulkopuolista tarkkailua (engl. *shoulder surfing*) että tietoliikenteestä informaation urkkimista tai sen manipulointia. Toteutus suojaa sekä käyttäjän syötteitä että käyttäjälle näkyvää informaatiota ulkopuolisten katseilta.

2. TAUSTAA

Nykyään ihmisten täytyy usein käyttää tietoteknisiä laitteita julkisissa tiloissa. Monesti pitää käyttää laitetta, jonka turvallisuuteen ei voi täysin luottaa. Tällainen tilanne on esimerkiksi pankkiautomaattia käytettäessä, jolloin julkinen laite välikätenä on pakollinen käyttäjän kommunikoidessa pankin kanssa. Yksinkertaistettu kuvaus tilanteesta ja siihen liittyviä uhkia on esitetty kuvassa 1.



Kuva 1. Julkisessa tilassa epäluotetun laitteen käyttöön kohdistuvia uhkia.

Jotta tällaisessa tilanteessa käyttäjän ja palvelimen välinen kommunikaatio voidaan turvata, täytyy ratkaista monia eri ongelmia. Esitetyssä skenaariossa kommunikoinnin turvaamista varten on kehitetty *EyeDecrypt* [3]. Skenaarion ja *EyeDecryptin* taustalla on erilaisia konsepteja ja ongelmia, joita esitellään tässä luvussa (esitetty taulukossa 1).

Taulukko 1. Työn taustaan liittyviä ongelmia ja niiden mahdollisia ratkaisuja.

Ongelma	Mahdollisia ratkaisuja
Ihmisen näköaistin hyödyntäminen kryptografisissa järjestelmissä	Visuaalinen kryptografia, salaviestien esittäminen visuaalisessa muodossa
Ulkopuolinen taho voi tarkkailla salasanan syöttämistä	Katseen hyödyntäminen salasanan syöttämisessä, merkkien korvaaminen kuvilla salasanoissa
Viivakoodeilla tapahtuva kommunikaatio on herkkä salakuuntelulle	SBVLC [4]
Kommunikointi palvelimen kanssa tapahtuu epäluotetun laitteen välityksellä	EyeDecrypt [3]
Käyttäjän älypuhelimen näyttöä voi vakoilla	Kännykän sijasta esim. jonkin lisätyn todellisuuden laitteen käyttäminen

Kryptografian avulla voidaan salata erilaisia viestejä laitteiden välisessä kommunikaatiossa. Visuaalinen kryptografia perustuu siihen, että ihmisen

näköaistia käytetään visuaalisessa muodossa olevan salauksen purkamisessa hyödyksi. Menetelmän ensi kertaa esittelivät Naor ja Shamir vuonna 1995 [5]. Heidän työnsä on saanut vuosien varrella paljon jatkoa.

Muiden kryptografisten menetelmien mukaisesti visuaalisen kryptografian tarkoituksena on suojella tärkeää tietoa ulkopuolisilta. Jos joku henkilö pääse tarkkailemaan laitteen käyttöä esimerkiksi julkisessa tilassa käyttäjän sitä huomaamatta, voi kriittistä tietoa salasanoista lähtien joutua vääriin käsiin (engl. *shoulder surfing*). Tätä estämään on kehitetty erilaisia menetelmiä, kuten katseen hyödyntäminen salasanan syöttämisessä [6] tai kuvien käyttäminen merkkeinä salasanoissa kirjaimien ja numeroiden sijaan [7]. Tällaiset menetelmät eivät kuitenkaan estä käyttäjälle julkisessa tilassa laitteella esitettyä informaatiota joutumasta vääriin käsiin, sillä informaatio on kaikkien nähtävillä käytettävän laitteen ruudulla salaamattomana. Mikäli käytettävään laitteeseen ei pysty luottamaan, eivät menetelmät riitä edes pitämään käyttäjän salasanaa hyökkääjän ulottumattomissa. *EyeDecrypt* on kehitetty ratkaisemaan tämä ongelma: kaikki epäluotetulla laitteella oleva informaatio on salattua ja esitetään käyttäjälle visuaalisessa muodossa. Käyttäjällä on oma luotettu laite, jota hän käyttää informaation lukemiseen ja salauksen avaamiseen. Tällöin tietysti oletetaan, että käyttäjän omaa laitetta, esimerkiksi älypuhelinta, ei voisi vakoilla. Käyttäjän laitteen vakoileminen on kuitenkin selkeä tilanteeseen kohdistuva riski. Siksi älypuhelinta parempi laite toteutukseen olisi esimerkiksi Google Glassin kaltainen lisätyn todellisuuden laite [3].

2.1. Visuaalinen kryptografia

Visuaalisten elementtien käyttö kryptografisessa järjestelmässä perustuu vahvasti visuaaliseen kryptografiaan. Visuaalisessa kryptografiassa käyttäjän näköaisti on keskeisessä asemassa. Naor ja Shamir esittelivät paperissaan vuonna 1995 tavan jakaa kuvamuodosssa salaisuus eri osiin useille eri tahoille, ja vain heidän yhdistäessään omat osansa salaisuus paljastuu [5]. Heidän kehittämälleen tavalle jakaa salaisuus on tehty paljon jatkoa, kuten muun muassa parannusta algoritmiin [8]. Esimerkiksi Lin ja Tsai esittelivät tavan soveltaa Naorin ja Shamirin kehittämää skeemaa harmaasävyisille kuville [9], kun aiemmat toteutukset soveltuivat vain binäärikuville. Visuaalisen kryptografian menetelmiä on myös kehitetty värillisille kuville [10].

2.2. Käyttäjän toiminnan passiivinen tarkkailu

Ihmiset joutuvat usein käyttämään erilaisia tietojärjestelmiä paikoissa, joissa on käyttäjän lisäksi muitakin ihmisiä. Missä tahansa julkisessa tilassa vaarana on, että joku saa selville vaikkapa käyttäjän salasanan tai muuta kriittistä informaatiota käyttäjän selän takaa. Tätä käyttäjän toiminnan passiivista tarkkailua voi myös tapahtua kameran tai muun vastaavan laitteen välityksellä. Muun muassa käyttäjän salasana voi joutua hyökkääjän käsiin käyttäjän

kirjautuessa järjestelmään. On kehitetty erilaisia järjestelmiä, jotka tekevät salasanan syöttämisestä julkisisssa tiloissa turvallisempaa.

Kumar ym. [6] hyödynsivät työssään käyttäjän katseen seuraamista salasanan syöttämisessä, minkä seurauksena käyttäjän toiminnan passiivinen seuranta olisi salasanan selvittämisessä hyödytöntä. On kehitetty myös esimerkiksi kirjautumisjärjestelmä, jossa numeroiden ja kirjaimien sijasta salasanassa käytetään kuvia [7]. Lee esittelee artikkelissaan [11] PIN-koodin syöttämiseksi uuden keinon, jonka tarkoitus on ehkäistä PIN-koodin urkkimista. Hänen ratkaisussaan numerot ovat yhdessä rivissä ja niiden alla on rivi merkkejä, joista satunnaisesti valikoituu PIN-koodin ensimmäisen numeron alla oleva merkki session avaimeksi. Käyttäjä sitten hyödyntää merkin siirtämistä alarivillä muiden PIN-koodin numeroiden syöttämisessä. Tällaiset menetelmät suojelevat käyttäjän salasanaa ulkopuolisilta, mutta eivät auta tilanteissa, missä näytöllä näkyy sellaista informaatiota, mitä ei saisi joutua muiden käsiin.

2.3. Turvallinen kommunikointi viivakoodien välityksellä

Zhang ym. [4] kehittivät järjestelmän nimeltä SBVLC, missä kaksi matkapuhelinta pystyvät keskenään kommunikoimaan viivakoodien välityksellä turvallisemmin. Heidän järjestelmässään on kaksisuuntainen kommunikaatio matkapuhelinten välillä, mikä mahdollistaa informaation salaamisen ennen viivakoodimuotoon muuttamista. He esittävät paperissaan erilaisia protokollia datan siirtämiseen viivakoodien avulla, kuten kaksiosaisen viestinlähetyksen ja matkapuhelinten välisen kättelyn. Heidän järjestelmässään kommunikaatio tapahtuu kahden luotetun laitteen välillä, eikä siten auta tilanteessa, jossa epäluotettua laitetta täytyy käyttää kommunikointiin.

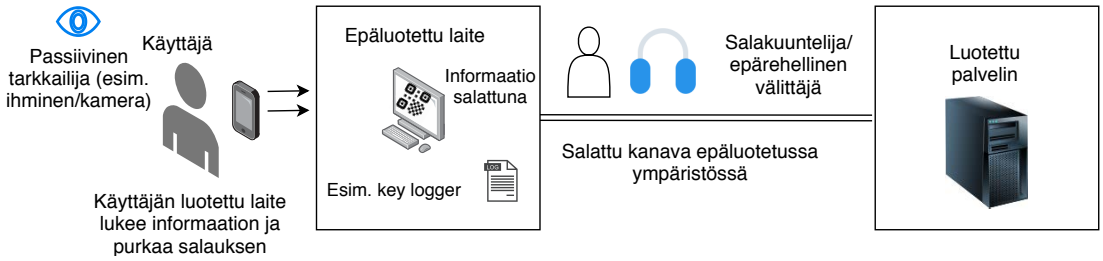
2.4. EyeDecrypt

Mikäli käyttäjän täytyy käyttää julkisessa tilassa olevaa epäluotettua laitetta, ei aiemmin esitellyistä menetelmistä ole hyötyä kommunikoinnin salaamisessa. Forte ym. [3] esittelevät konseptin *EyeDecrypt*, jossa ihmiskäyttäjä on yhteydessä luotettuun palvelimeen, mutta kommunikointi käyttäjän ja palvelimen välillä kulkee epäluotetun laitteen kautta. Forte ym. mainitsevat muutaman esimerkkiskenaarion tällaisesta tilanteesta. Tilanne on mahdollinen esimerkiksi silloin, kun käyttäjä käyttää julkisessa tilassa sijaitsevia kaikkien käytössä olevia tietokoneita tai pankkiautomaattia. Samoin esimerkiksi tilanne voisi olla mahdollinen erittäin korkean turvallisuuden järjestelmissä, jossa ulkopuolisten laitteiden yhteydenottamista palvelimeen ei hyväksyttäisi.

Tällaisissa tilanteissa käyttäjän ja palvelimen väliseen kommunikaatioon kohdistuu erilaisia uhkakuvia. Tietokoneilla voi esimerkiksi olla ohjelma, joka tallentaa käyttäjän syötteitä. Samoin pankkiautomaatin läheisyyteen voi olla asennettu jonkinlainen kamera, joka seuraa käyttäjän toimintaa. On myös mahdollista, että joku käyttäjän takana oleva henkilö tarkkailisi käyttäjän

toimintaa laitteella ja saisi täten haltuunsa esimerkiksi käyttäjän salasanan tai arkoja tietoja.

EyeDecryptin tarkoituksena on estää sekä passiivisia että aktiivisia hyökkäyksiä. Passiivisia hyökkäysskenaarioita ovat esimerkiksi mainitut käyttäjän olkapään yli kurkkiminen ja näppäilytallentimet epäluotetulla laitteella. Aktiivisissa hyökkäyksissä lähetettyä dataa pyritään manipuloimaan. *Epärehellinen välittäjä* -hyökkäys on yksi esimerkki tällaisesta hyökkäyksestä. Vaikka viestien salakuuntelu onnistuttaisiin estämään viestien salauksella ja muokatut viestit onnistuttaisiin tunnistamaan, ei se ole tässä skenaariossa riittävästi. Epäluotetun laitteen tarkkailu tai sillä olevat mahdolliset haittaohjelmat vaarantavat tietojen turvallisuuden, koska viestit ovat salaamattomina epäluotetulla laitteella. Tämän takia tarvitaan jokin luotettu keino kommunikoida palvelimen kanssa. *EyeDecryptissä* ongelma ratkaistaan tuomalla tilanteeseen käyttäjälle luotettu laite, esimerkiksi käyttäjän oma älypuhelin. EyeDecryptin toimintaa yksinkertaistettuna on havainnollistettu tuomalla se kuvassa 1 esitettyyn uhkaskenaarioon kuvassa 2.



Kuva 2. EyeDecryptin toiminta yksinkertaistettuna.

EyeDecryptin [3] idea on seuraavanlainen: ensimmäiseksi palvelin ja käyttäjän luotettu laite suorittavat turvallisen avaintenvaihdon. Sen jälkeen käyttäjä kommunikoi palvelimen kanssa ainoastaan epäluotetun laitteen välityksellä. Palvelin salaa viestinsä ja lähettää ne epäluotetulle laitteelle, joka näyttää viestit käyttäjälle visuaalisessa muodossa käyttöliittymässään. Käyttäjä lukee viestit luotetulla laitteellaan, esimerkiksi älypuhelimen kameran avulla. Luotettu laite avaa salauksen ja näyttää käyttäjälle viestien sisällön. Näin visuaalisessa muodosssa esitetystä datasta ei ulkopuolinen tarkkailija voi päätellä sen sisällöstä mitään. Myöskään epäluotetun laitteen kautta ei tietoa kommunikoinnin sisällöstä pääse vääriin käsiin, sillä epäluotettu laite näkee ainoastaan salatekstin. Paperissa [3] tarkemmin esitellyn salausmallin keskeinen lopputulos on joukko

$$C = (C_0, \tau, i, f), \tag{1}$$

missä C_0 on salateksti, τ viestin todennuskoodi (MAC, engl. *message authentication code*) ja i ja f ovat viestin sijainti epäluotetun laitteen käyttöliittymässä. Käyttäjän on tarkoitus saada salatekstin mukana myös viestin todennuskoodi, jonka avulla käyttäjä voi tarkistaa, ettei hänen vastaanottamaansa salatekstiä kukaan ole päässyt muokkaamaan.

Samoin *EyeDecryptin* avulla käyttäjä voi lähettää viestejä palvelimelle epäluotetun laitteen kautta. Paperissa [3] mainitaan esimerkiksi mahdollisuus

PIN-koodin syöttämiseen siten, että jokaisen syötteen jälkeen näppäinten järjestys epäluotetun laitteen käyttöliittymässä vaihtuu. Näppäinten merkitys selviäisi vain skannaamalla epäluotetun laitteen näyttöä luotetulla laitteella. Näin ulkopuolinen tarkkailija tai epäluotetulla laitteella olevat haittaohjelmat eivät voisi selvittää käyttäjän syötteitä. *EyeDecryptin* tarkat algoritmit ja todistukset toteutuksen oikeanlaisesta toiminnasta on esitelty Forten paperissa [3].

3. TOTEUTUS

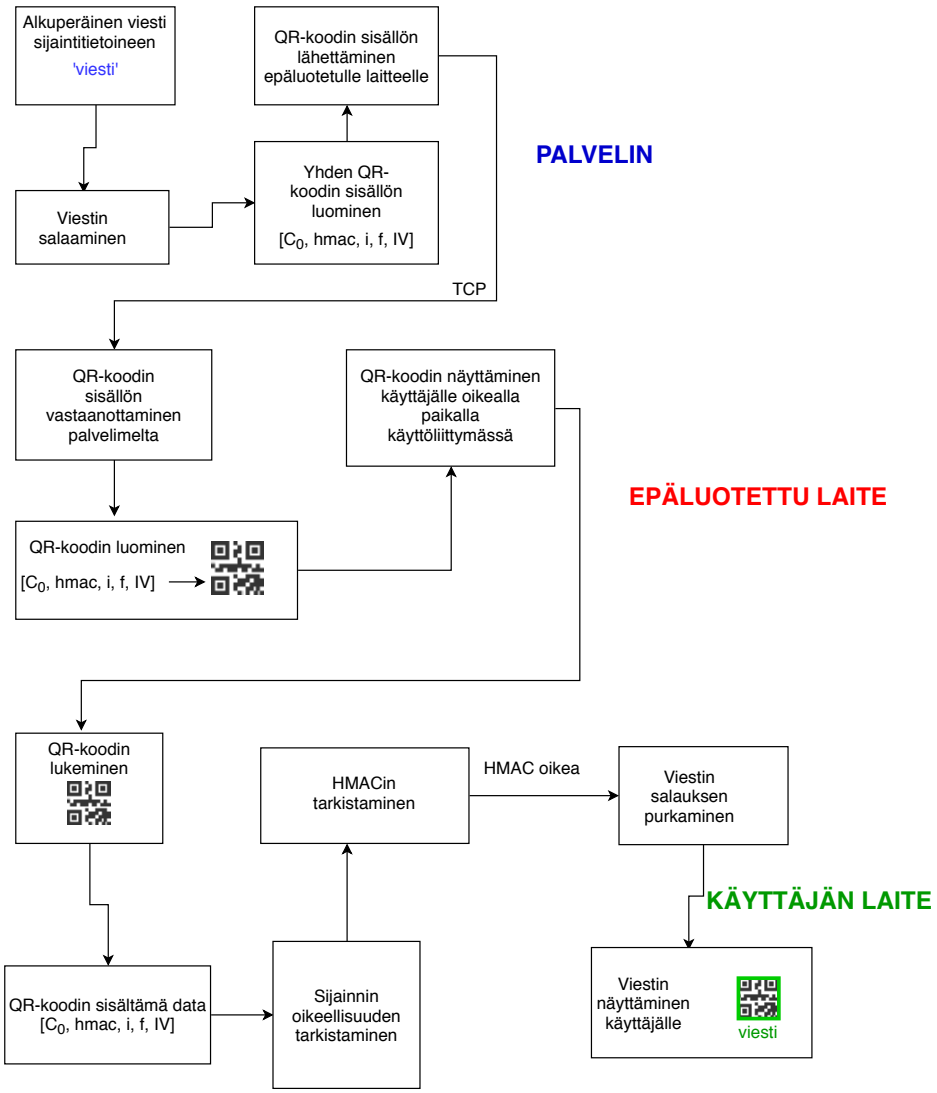
Tässä työssä toteutettiin versio edellisessä luvussa esitellystä Forten ym. [3] *EyeDecryptistä*. Tarkoituksena oli käyttää älypuhelinta salatussa muodossa olevan visuaalisen datan lukemiseen epäluotetun laitteen ruudulta. Samoin *EyeDecryptiä* oli tarkoitus käyttää suojaamaan käyttäjän syötteitä epäluotetun laitteen kautta. Forten paperissa [3] esitetään useampia vaihtoehtoja visualisointiin. Tässä työssä visualisointiin päätettiin käyttää QR-koodeja sen vuoksi, että niitä on helppo luoda ja niitä varten löytyy kirjastoja hyvin eri alustoille.

Tämän työn toteutuksessa salaukseen käytettiin AES-128-algoritmia [12], joka on *EyeDecryptin* vaatimusten mukaan symmetrinen salausmenetelmä ja viestien autentikointiin HMACia [2]. Työssä tehty toteutus koostuu kolmesta komponentista: luotetusta palvelimesta, epäluotetusta laitteesta ja älypuhelinsovelluksesta. Kuvassa 3 on esitetty yksinkertaistettuna toteutuksen toiminta yhden palvelimen lähettämän viestin kannalta. Palvelin luo erilaisia viestejä, jotka on tarkoitus välittää käyttäjälle. Lopputuloksena palvelimella syntyy viidestä komponentista koostuva lista

$$C = (C_0, \tau, i, f, IV), \tag{2}$$

missä C_0 on salateksti, τ HMAC-koodi ja i ja f ovat lohko ja kehys. Lohko ja kehys kuvaavat viestin paikkaa epäluotetun laitteen graafisessa käyttöliittymässä. Kehys f kuvaa sitä, monenteenko samanaikaisesti käyttöliittymässä näkyvään viestikokonaisuuteen viesti kuuluu ja lohko i viestin tarkkaa sijaintia käyttöliittymässä. IV on AES-algoritmiin tarvittava alkumuuttuja (engl. *initialization vector*). Lista on siis luvussa 2 esitelty *EyeDecryptin* salausalgoritmin perusjokko (kaava 1), johon on AES-algoritmia varten lisätty alkumuuttuja IV. Palvelin lähettää listan TCP-yhteyden välityksellä epäluotetutulle laitteelle. Epäluotettu laite muodostaa saamastaan listasta QR-koodin, jonka se näyttää käyttöliittymässään käyttäjälle, joka voi skannata koodin älypuhelimellaan. Käyttöliittymässä on samanaikasesti näkyvillä yhteensä neljä QR-koodia. Käyttäjä skannaa kaikki neljä QR-koodia kännykällään yhtä aikaa. Tällöin voidaan tarkistaa, ovatko QR-koodit niille kuuluvilla paikoilla.

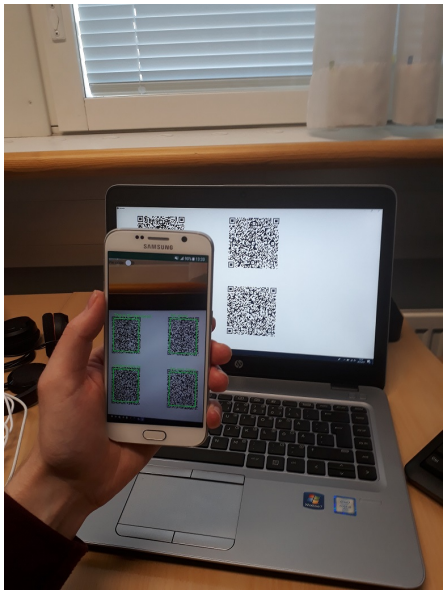
Sovellusta on mahdollista käyttää myös palvelimelle kirjautumisen simulointiin. Tässä työssä toteutettiin soveltuvuusselvitys-tyylinen toteutus tunnistautumisesta palvelimelle epäluotetulla laitteella *EyeDecryptiä* hyödyntäen. Epäluotetun laitteen käyttöliittymässä on asettelu, missä on näppäin jokaiselle mahdolliselle salasanan merkille. Palvelin suorittaa satunnaispermutaation mahdollisille merkeille ja lähettää satunnaisessa järjestyksessä olevat merkit salattuna epäluotetulle laitteelle samalla tavalla kuin salattujen viestienkin tapauksessa. Epäluottu laite näyttää ne palvelimen määrittämässä järjestyksessä QR-koodeina käyttäjälle. Käyttäjä käyttää omaa älypuhelintaan skannatakseen läpi asetelman ja ottaakseen selville, mikä QR-koodi vastaa oikeaa salasanan merkkiä. Käyttäjä sen jälkeen klikkaa kursorilla epäluotetun laitteen käyttöliittymässä QR-koodia, joka vastaa oikeaa salasanan arvoa. Epäluotettu laite lähettää palvelimelle tiedon siitä, mitä



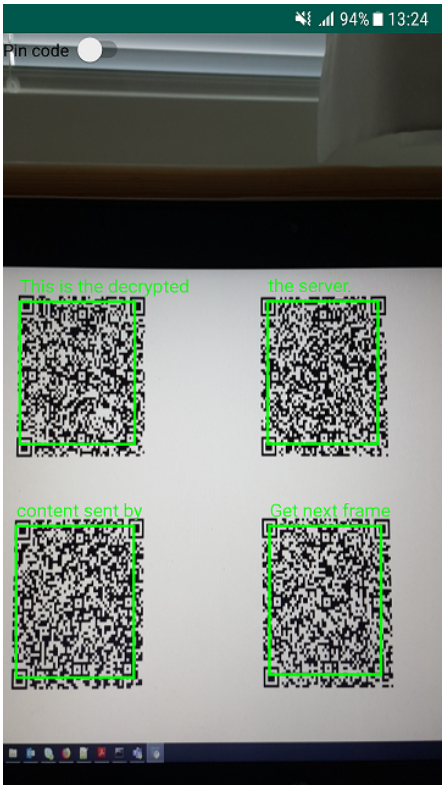
Kuva 3. Yksinkertaistettu esitys yhden viestin matkasta palvelimelta epäluotetun laitteen kautta käyttäjälle. Yhden QR-koodin sisältö on esitetty kaavassa 2.

kohtaa käyttöliittymässä on klikattu. Palvelin tarkistaa, mihin oikeaan arvoon kyseinen kohta viittaa ja tallentaa arvon. Sen jälkeen palvelin suorittaa uudelleen satunnaispermutaation merkkilistalle ja lähettää merkit uudessa järjestyksessä epäluotetulle laitteelle. Tätä toistetaan niin kauan, että käyttäjä on syöttänyt vaaditun määrän merkkejä, minkä jälkeen palvelin tarkistaa syötetyn salasanan oikeellisuuden.

Kuva 4 näyttää toteutetun sovelluksen käyttöä käytännössä. Kuvassa 5 näkyy kännykkäsovelluksen näkymä QR-koodeja skannattaessa, kun sovellus on löytänyt koodit kameran kuvasta ja avannut niiden salatut viestit. Jokaisen QR-koodin päällä kuvassa näkyy sen selkoteksti. Vihreä väri kuvaa sitä, että QR-koodi on oikealla paikalla epäluotetun laitteen käyttöliittymässä.



Kuva 4. Toteutetun sovelluksen käyttötilanne. Käyttäjä skannaa kännykällä epäluotetun laitteen käyttöliittymää. Käyttäjä skannaa neljä käyttöliittymässä näkyvää QR-koodia kerralla.



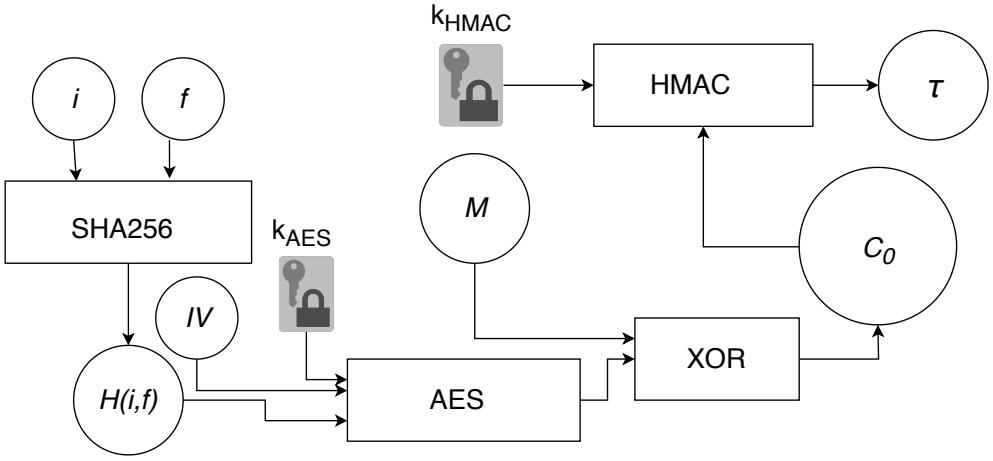
Kuva 5. Käyttäjän kännykän ruudulla näkyy yhden kehyksen QR-koodien sijainti ja niiden avatut selkotekstit. Vihreä väri tarkoittaa sitä, että QR-koodin sijainti on oikea suhteessa muihin kehyksen QR-koodeihin.

3.1. Palvelin

Tässä työssä oletettiin, että palvelin ja käyttäjä ovat suorittaneet jo aiemmin avainten luomis- ja vaihtoprosessin. Palvelimena toteutuksessa toimii Pythonilla kirjoitettu yksinkertainen TCP-palvelin. Palvelin lähettää epäluotetun laitteen yhdistäessä siihen ensimmäisen kehyksen viestejä. Palvelin lähettää pyydettäessä useampia viestejä.

Kuvassa 6 on palvelimella tapahtuva salausprosessi visuaalisessa muodossa. Palvelin ensin muodostaa tiivisteen SHA256-tiivistefunktiolla. Tiivistefunktion argumentteina ovat tiedot viestin kehyksestä i ja lohkosta f, jotka kertovat QR-koodin sijainnista epäluotetun laitteen käyttöliittymässä. Tämä paljastaa täten koodien lukijalle, mikäli ne eivät ole luettaessa enää alkuperäisellä paikallaan. Tiiviste $H(i,f)$ sitten syötetään AES-algoritmiin ja algoritmin tulokselle suoritetaan XOR-operaatio salattavan viestin kanssa. Salateksti C_0 autentikoidaan vielä HMAC-algoritmin avulla. Mikäli joku muokkaisi salatekstiä matkalla palvelimelta käyttäjälle, ei se jäisi käyttäjältä huomaamatta. Jotta salauksen purkaminen Android-puhelimella Java-kielellä onnistuisi

mahdollisimman helposti, muutetaan salateksti C_0 ja HMAC-koodi τ palvelimen puolella ennen eteenpäin lähettämistä Base64-muotoon [13]. Näin salatekstin käsittely Androidin puolella helpottuu. Lopputuloksena palvelin on luonut aiemmin esitellyssä kaavassa 2 olevan listan, jonka se lähettää epäluotetulle laitteelle.



Kuva 6. Salaviestin C_0 ja autentikointikoodin τ luominen palvelimella. Palvelin lähettää epäluotetulle laitteelle listan $C = (C_0, \tau, i, f, IV)$ (kts. kaava 2).

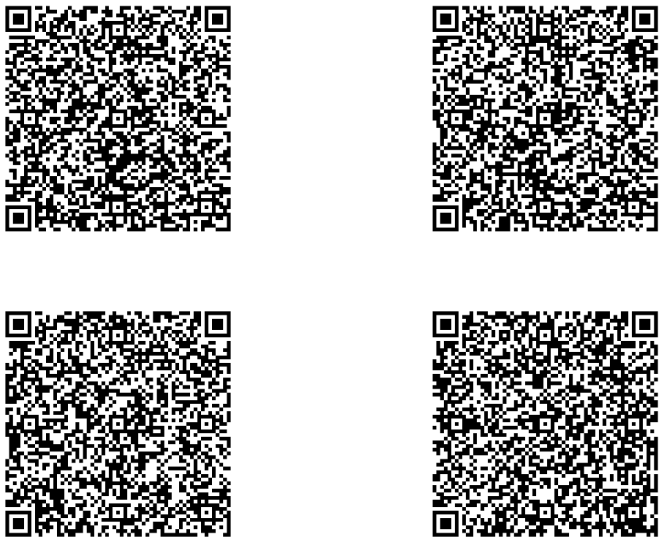
Kun käyttäjä haluaa "kirjautua palvelimelle"ja syöttää kirjautumiskoodin, palvelin arpoo mahdolliset kirjautumismerkit (työssä toteutetussa versiossa numerot 0-5) satunnaiseen järjestykseen ja lähettää ne järjestyksessä epäluotetulle laitteelle. Saatuaan käyttäjän syötteen palvelin tarkistaa, mihin oikeaan arvoon syöte viittaa ja listalle mahdollisista merkeistä suoritetaan uudelleen satunnaispermutaatio. Tämän jälkeen palvelin lähettää listan uudelleen uudessa järjestyksessä epäluotetulle laitteelle. Lähetetyt arvot on salattu samalla tavalla kuin muutkin viestit. Saatuaan oikean määrän syötteitä palvelin tarkistaa, oliko käyttäjän syöttämä salasana oikea.

3.2. Epäluotettu laite

Epäluotettu laite on yhteydessä palvelimeen TCP-yhteyden välityksellä. Tässä työssä keskityttiin *EyeDecrypt*-konseptin totetuttamiseen, joten palvelimen ja epäluotetun laitteen välisen yhteyden turvallisuutta ei tässä paperissa käsitellä.

Epäluotettu laite vastaanottaa palvelimelta luvun alussa kuvailtuja listoja. Listat muunnetaan Python-kirjaston qrcode [14] avulla QR-koodeiksi. Epäluotettu laite näyttää QR-koodit käyttäjälle käyttöliittymässään. Käyttöliittymä on toteutettu Python-pohjaisella Kivy-ohjelmistokehyksellä [15]. Käyttöliittymässä näkyy kerrallaan neljä QR-koodia (yksi kehys), aina kaksi päällekäin (kuva 7). Kun käyttäjä klikkaa tiettyä QR-koodia, asiakasprosessi lähettää palvelimelle pyynnön vastaanottaa seuraava kehys.

Käyttäjän syöttäessä salasanaa epäluotettu laite näyttää käyttäjälle niin monta QR-koodia kuin on mahdollisia salasanan merkkejä. Käyttöliittymä näyttää QR-



Kuva 7. Käyttöliittymän perusnäkymä, jonka myös ulkopuolinen tarkkailija näkee.

koodit aina palvelimen määrittämässä järjestyksessä. Käyttäjän klikatessa jotain QR-koodia epäluotettu laite lähettää palvelimelle tiedon siitä, mitä QR-koodia klikattiin.

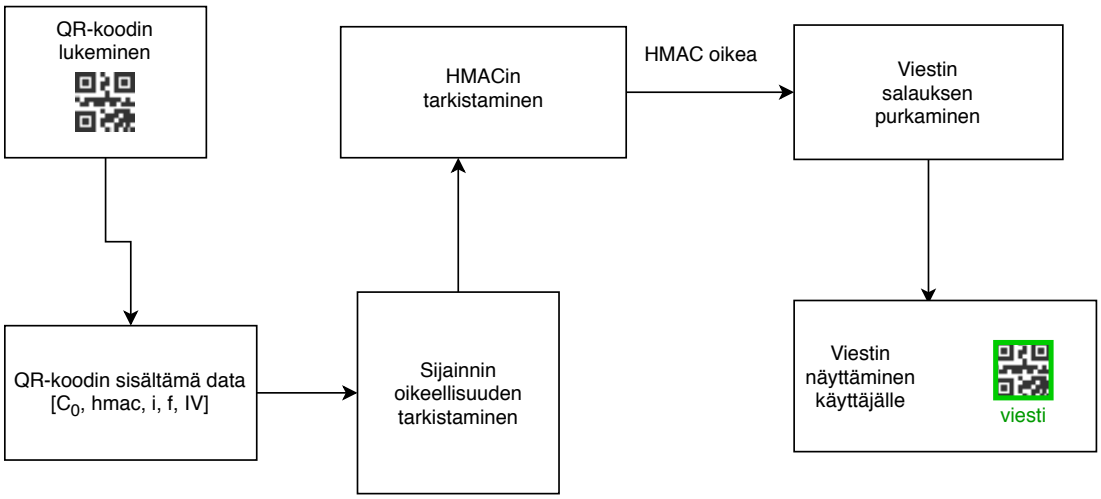
3.3. Käyttäjän laite

Tässä työssä käyttäjän laitteena toimi Samsung Galaxy S6 -älypuhelin. Ideaalitilanteessa käyttäjän laitteena toimisi esimerkiksi Google Glassin kaltainen lisätyn todellisuuden laite [3]. Kännykän näyttö on kuitenkin huomattavasti pienempi kuin esimerkiksi pankkiautomaatin näyttö ja käyttäjä voi lisäksi piilottaa kännykkäänsä uteliailta katseilta tai esiasennetuilta kameroilta helpommin. Tietenkin käynnykän kohdalla edelleen säilyy riski siitä, että sen näyttöä urkitaan ja toteutuksen turvallisuus perustuu siihen, että kännykän turvallisuuteen luotetaan. Kännykkä toimii kuitenkin hyvin esimerkkinä järjestelmän toiminnasta.

3.3.1. Viestien vastaanottaminen älypuhelimella

QR-koodit skannataan älypuhelimen kameralla ja niiden tunnistamisessa käytetään Zxing-viivakoodintunnistuskirjastoa [16]. QR-koodit tunnistetaan kameran esikatselukuvasta. Käyttäjä osoittaa kameran kohti käyttöliittymää siten, että kaikki yksittäisen kehyksen neljä QR-koodia näkyvät näytöllä yhtä aikaa. Kun sovellus tunnistaa kaikki neljä QR-koodia kameran kuvasta yhtä aikaa, otetaan QR-koodit käsittelyyn. Sovellus tarkistaa, että kaikki QR-

koodit kuuluvat samaan kehykseen ja ovat oikeilla paikoillaan käyttöliittymän ruudukossa. Sen jälkeen sovellus tarkistaa, että QR-koodin HMAC-koodi täsmää sen salatekstiin. Mikäli HMAC-koodi täsmää salatekstiin, sovellus avaa salatekstin ja näyttää selkotekstin kännykän kameran esikatselukuvasssa QR-koodin päällä käyttäjälle. Oikeilla paikoillaan olevat QR-koodit ja niiden selkoteksti esitetään käyttäjälle vihreällä värillä ja väärällä paikalla olevat QR-koodit punaisella värillä. Kuvassa 8 on esitetty sovelluksen perustoiminta yksinkertaistettuna.



Kuva 8. Älypuhelinsovelluksen toiminta yksinkertaistettuna.

3.3.2. Palvelimelle tunnistautuminen älypuhelinta hyödyntäen

Sovellusta voi myös käyttää turvalliseen kirjautumiseen palvelimelle. Epäluotetun laitteen käyttöliittymässä näkyy salattuna QR-koodeina mahdolliset syötteet. Käyttäjä skannaa QR-koodit läpi kännykällään ja klikkaa epäluotetulla laitteella sitä QR-koodia, joka vastaa oikeaa merkkiä käyttäjän salasanassa. QR-koodit on salattu samalla tavalla kuin tavalliset viestitkin, ja sovellus hoitaa niiden salauksen avaamisen (kuva 9). Koska merkkien järjestys käyttöliittymässä vaihtuu jokaisen käyttäjän antaman syötteen jälkeen, ei ulkopuolinen tarkkailija voi päätellä mitään muuta salasanasta kuin sen pituuden (tässä työssä salasanan pituus kiinteä). Epäluotetulla laitteella ei myöskään ole tietoa siitä, mitä arvoja käyttäjä syöttää järjestelmään, joten laitteella mahdollisesti olevat tarkkailuohjelmat eivät voi päätellä mitään käyttäjän syötteistä. Tällaista kirjautumistapaa voisi käyttää esimerkiksi järjestelmissä, joissa salasanat ovat kaikki rajatun pituisia.

Kuva 9. Käyttäjä on kirjautumassa palvelimelle ja skannaa sovelluksella epäluotetun laitteen käyttöliittymässä näkyvää näppäinruudukkoa. QR-koodin yläpuolella näkyy, mitä merkkiä kukin QR-koodi vastaa.

4. TULOKSET

Tässä työssä toteutetun järjestelmän tarkoituksena oli suojata käyttäjän kommunikaatiota luotetun palvelimen kanssa tilanteessa, jossa kommunikointi tapahtuu epäluotetun laitteen kautta. Todistukset *EyeDecryptin* oikeanlaisesta toiminnasta löytyvät Forten paperista [3], mutta loppukädessä järjestelmän toteuttajalla on aina vastuu toteutuksen onnistumisesta.

Tässä luvussa arvioidaan työssä toteutetun järjestelmän toimintaa. Tässä luvussa arvioidaan järjestelmän tehokkuutta joitakin uhkaskenaarioita vastaan. Samoin arvioidaan sovelluksen suorituskykyä ja toimintaa käyttötilanteessa.

4.1. Epärehellinen välittäjä -hyökkäys

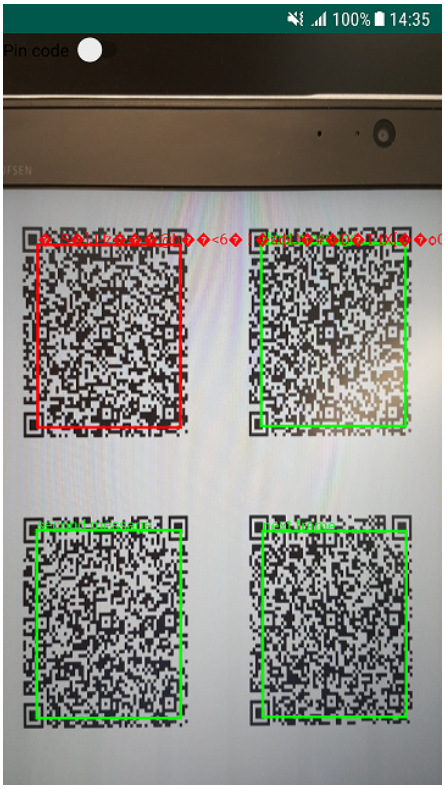
Epärehellinen välittäjä -hyökkäys on yksi skenaario, jota vastaan järjestelmän olisi tarkoitus suojata käyttäjää. *Epärehellinen välittäjä* -hyökkäyksessä (myös esimerkiksi *mies keskellä* -hyökkäys, engl. *man-in-the-middle*) sekä käyttäjä että palvelin luulevat kommunikoivansa keskenään, mutta niiden kommunikaatio kulkeekin kolmannen osapuolen kautta. Hyökkääjä voi tällöin muokata viestejä halutessaan tai jättää joitakin viestejä välittämättä. Koska työssä tehdyssä toteutuksessa on sisällytetty viestin autentikointi, ei hyökkääjä voi saada käyttäjän sovellusta avaamaan omaa viestiään oikean viestin sijasta. Käyttäjän sovellus tarkistaa aina ennen salatekstin purkamista, vastaako käyttäjän vastaanottaman QR-koodin salateksti sen HMAC-koodia. Mikäli hyökkääjällä ei ole hallussaan HMAC- ja salausavaimia, ei se pysty tuottamaan omia salatekstejä, jotka käyttäjä suostuisi avaamaan.

Jos hyökkääjä vaihtaa esimerkiksi salatekstin toiseen, QR-koodin HMAC-koodi ei enää vastaa QR-koodin salatekstiä. Käyttäjän sovellus tunnistaa tämän ja näyttää sen käyttäjälle kameran kuvassa näyttämällä *error*-tekstin punaisella selkotekstin paikalla. Kuva 10 esittää tilannetta, jossa palvelimen ja epäluotetun laitteen välinen liikenne on kulkenut kolmannen osapuolen kautta ja kolmas osapuoli on vaihtanut vasemman yläkulman QR-koodin salatekstin toiseksi. Värikoodatun virheviestin kautta käyttäjä saa tietää, että jokin kyseisen QR-koodin salatekstin avaamisessa ei onnistunut. Mikäli hyökkääjä taas vaihtaa QR-koodin paikkatietoja (lohko tai kehys), ei toteutettu sovellus onnistu enää avaamaan salausta oikein, ja käyttäjän näkymässä QR-koodia ympäröi punainen kehys ja salauksen avaamisyrityksen lopputulos punaisena. Kuvassa 11 kolmas osapuoli on vaihtanut ensimmäisen QR-koodin paikkatietoa viestien kulkiessa palvelimelta käyttäjälle.

4.2. Mitä järjestelmästä jää ulkopuolisen tarkkailijan käteen

Järjestelmän keskeinen ajatus on suojata käyttäjän toimintaa hänen käyttäessään epäluotettua laitetta julkisessa tilassa. Siksi on olennaista tarkastella tarkemmin, mitä kaikkea passiivinen tarkkailija oikein saa selville järjestelmästä käyttäjän selän takaa, joko omin silmin tai sitten esimerkiksi kameran avulla. Epäluotetulla

Kuva 10. Epärehellinen välittäjä on vaihtanut vasemman yläkulman QR-koodin salatekstin. Tällöin QR-koodin HMAC-koodi ei enää vastaa salatekstiä ja sovellus näyttää sen käyttäjälle punaisella *error*-tekstillä.



Kuva 11. Vasemmassa yläkulmassa olevan QR-koodin paikkatietoa on vaihdettu epärehellisen välittäjän toimesta matkalla. Sovellus ei enää pysty avaamaan salausta oikein.

laitteella voi esimerkiksi myös olla käyttäjän toimintaa tai epäluotetun laitteen verkkoliikennettä seuraavia ohjelmia.

4.2.1. Passiivinen tarkkailija käyttäjän takana

Julkisessa tilassa laitetta käytettäessä voi käyttäjän selän takaa joku henkilö tarkkailla hänen syötteitään ja laitteella näkyviä tietoja. Saman informaation voi kolmas osapuoli saada haltuunsa esimerkiksi kiinteän laitteen käyttötilaan asennettujen kameroiden avulla. Kun käyttäjä vastaanottaa viestejä palvelimelta, epäluotetun laitteen käyttöliittymässä näkyy neljä erilaista QR-koodia. Tarkkailija näkee käyttäjän klikkaavan jotain QR-koodeista.

Käyttäjän kirjautuessa teoreettiseen järjestelmään tarkkailija näkee jälleen useaan otteeseen joukon QR-koodeja ja käyttäjän klikkaavan aina jotain niistä. Tarkkailija saa selville käyttäjän tunnistautumiskoodin pituuden, mutta jos kirjautumiseen käytetään esimerkiksi jotain PIN-koodia, on sen pituus

usein vakio ja kaikkien tiedossa muutenkin. Mikäli PIN-koodin pituus ei olisi vakio, voisi ongelman salasanan pituuden selviämisestä ratkaista esimerkiksi lisäämällä merkkeihin jonkinlaisen tyhjän merkin, joita käyttäjä syöttäisi oman PIN-koodinsa syöttämisen jälkeen PIN-koodin maksimipituuteen asti. Toistamalla tismalleen samat askeleet myöhemmin ei tarkkailija onnistuisi tunnistautumaan järjestelmään, sillä kirjautuessa kirjautumismerkkien paikat arvotaan palvelimella uudelleen jokaisen merkin jälkeen. Näin tarkkailijan tunnistautumisen onnistuminen on aivan yhtä epätodennäköistä kuin silloin kun hän ei tietäisi käyttäjän syötteitä, kunhan merkkien järjestyksen arpominen palvelimella on täysin satunnaista.

Jos tarkkailija on tarkan kameran avulla kuvannut epäluotetun laitteen ruutua, saattaa hän onnistua skannaamaan kuvasta yksittäisten QR-koodien sisällön. Tällöin tarkkailija saa haltuunsa QR-koodin sisällön eli kaavassa 2 olevan sisällön. Koska sisältö on salattua, ei tarkkailija saa tietää mitään selkotekstin sisällöstä ilman avaimia.

Oletuksena järjestelmän turvallisuudelle on tietenkin se, että tarkkailija ei onnistu näkemään käyttäjän oman laitteen ruutua. Mikäli esimerkiksi tarkkailija on asentanut tilaan kameran, jolla hän onnistuu näkemään käyttäjän sopivassa paikassa pitämän laitteen ruudun, on kaikki käyttäjän tieto hyökkääjän käsissä. Tämän takia jokin lisätyn todellisuuden laite olisi parempi kuin kännykkä, jonka ruudun vakoilu on vaikeampaa kuin epäluotetun laitteen vakoilu, mutta edelleen tietenkin mahdollista, jos käyttäjä ei pidä varaansa.

4.2.2. Epäluotetulla laitteella oleva tieto

Järjestelmässä voidaan olettaa, että hyökkääjällä on pääsy kaikkeen epäluotetulla laitteella olevaan informaatioon, ja myös epäluotetun laitteen ja palvelimen välisiin viesteihin. Järjestelmän voi helposti rakentaa siten, että käyttäjän klikatessa QR-koodia päästäkseen järjestelmän käytössä eteenpäin, epäluotettu laite lähettää palvelimelle klikatun QR-koodin sijainnin käyttöliittymässä. Tällöin hyökkääjä saa tietoonsa QR-koodien sisällön ja klikattujen QR-koodien sijainnin. Epäluotetulla laitteella ei kuitenkaan järjestelmässä ole tietoa HMAC- tai AES-avaimista, joten hyökkääjä ei voi niitä tätä kautta hankkia. Koska epäluotettu laite ei tiedä, mitä merkkejä käyttäjä syöttää järjestelmään, ei hyökkääjä voi käyttäjän tunnistautumiskoodia saada haltuunsa. Täten hyökkääjän ei pitäisi pystyä päästä käsiksi käyttäjän tietoihin epäluotetun laitteen kautta.

4.3. Suorituskyky, toimivuus ja käytettävyys

Jotta järjestelmä olisi käytännöllinen, on tärkeää, että käyttäjän sovellus toimii luotettavasti ja sitä on helppo sekä miellyttävä käyttää. Käyttäjän sovelluksen keskeinen osa suorituskyvyn ja myös käyttökokemuksen kannalta on QR-koodintunnistusalgoritmi. Tunnistusalgoritmi pyrkii käymään koko ajan kameran esikuvasta kuvakehyksiä läpi ja etsii niistä QR-koodeja.

Sovelluksen QR-koodintunnistusalgoritmin toimintaa testattiin yksinkertaisissa olosuhteissa kolmella eri kameran resoluutiolla: 1920x1080, 1280x720 ja 640x480. Algoritmilla kesti testiolosuhteissa 1280x720-resoluutiolla noin 150 ms käydä yksi kuvakehys läpi. Mikäli QR-koodeja löytyi, niiden käsittely (salauksien avaaminen, käyttäjälle näyttäminen) toi käsittelyaikaan n. 100 ms lisää. Isommalla resoluutiolla suoritusaika piteni luonnollisesti entisestään ja sovellus alkaa olla varsin epäkäytännöllinen. Kuvan tarkkuuden laskeminen testiolosuhteissa 640x480-resoluutioon laski huomattavasti suoritusaikaa. Yhden kuvakehyksen läpikäyminen vei enää 50-100 ms. Samalla sovellus tuntui sulavammalta käyttää, kun algoritmi kävi useampi kehyksiä läpi sekuntia kohden. Resoluution laskeminen kuvassa vaikeuttaa varmasti algoritmin työtä löytää kuvasta QR-koodeja, mutta testiolosuhteissa 640x480-resoluutiolla ei suuria ongelmia sen suhteen esiintynyt. Kun sovellus skannasi neljää 490x490-kokoista QR-koodia yhtä aikaa, löytyivät ne hyvin heikommallakin kuvatarkkuudella, kunhan kameran toi tarpeeksi lähelle QR-koodeja.

Testiolosuhteissa sovellus olikin ehdottomasti sulavamman tuntuinen ja QR-koodit tuntuivat löytyvän helpommin pienemmällä resoluutiolla. Taulukossa 2 näkyy käsitellyt kuvakehykset keskimäärin sekuntia kohden eri resoluutioilla. 640 x 480 -resoluutiolla sovellus käsittelee sekuntia kohden keskimäärin lähes kolme kertaa enemmän kehyksiä kuin 1280 x 720 -resoluutiolla. Vaikka pienemmällä resoluutiolla eivät QR-koodit löytyisi aivan niin helposti, suorituskyvyn parantuminen korvaa heikentynyttä tunnistustarkkuutta. Korkeammalla resoluutiolla tosin kameraa ei tarvitse tuoda niin lähelle QR-koodeja niiden salaisuuksia avatakseen.

Taulukko 2. QR-koodintunnistusalgoritmin käsittelemät kehykset eri kuvaresoluutioilla sekuntia kohden.

Resoluutio	Kehyksiä käsitelty sekunnissa keskimäärin
640 x 480	14.77
1280 x 720	5.45
1920 x 1080	2.49

QR-koodeja tunnistettaessa myös QR-koodin ominaisuudet ovat kriittisiä. QR-koodin koko epäluotetun laitteen käyttöliittymässä vaikuttaa tunnistamistarkkuuteen. QR-koodin on oltava riittävän iso kännykän kuvassa, jotta QR-koodintunnistusalgoritmi löytää sen luotettavasti. Niinpä samaan aikaan käyttäjä pystyy avaamaan kännykällään hyvin rajallisen määrän informaatiota. Toteutetussa työssä 640x480-resoluutiolla neljän QR-koodin yhtäaikainen avaaminen onnistui hyvin, mutta kuusi samaan aikaan kännykän ruudulla oli ehdottomasti maksimi. Tilan tarpeeseen vaikuttaa myös se, että QR-koodi vaatii ympärilleen hiljaisen alueen. Luultavasti QR-koodintunnistustarkkuutta voisi parantaa käsittelemällä kameran kuvaa kuvankäsittelyalgoritmeilla ennen QR-koodien etsimistä, jolloin kuvassa olisi häiriötä vähemmän. Vaikutusta tällä olisi varsinkin siinä tapauksessa, että QR-koodit näkyvät pienenä kameran kuvassa.

QR-koodin tunnistamiseen vaikuttaa myös, minkälaiselta näytöltä käyttäjä QR-koodeja kännykällään lukee. Sovellus löytää QR-koodit paremmin isommalta ja paremmalta näytöltä, ja kameran kuvassa on tällöin vähemmän häiriötä. Näyttöjä skannatessa syntyvä häiriö on hyvin näkyvissä esimerkiksi kuvissa 10 ja 11. Huonommalta näytöltä QR-koodeja luettaessa häiriö vaikeuttaa koodien tunnistamista. Forten paperissa heidän prototyypissään käytettiin kuvanmuokkausalgoritmeja häiriön vaikutuksen estämiseen [3]. Niitä voisi halutessaan lisätä tähänkin sovellukseen, mutta ainakin testiolosuhteissa QR-koodintunnistusalgoritmi toimi varsin hyvin ilman kameran kuvan muokkaamista ennen QR-koodien etsimistä. Todellisissa käyttötilanteissa esimerkiksi auringonvalo voi varmasti vaikeuttaa QR-koodien skannaamista ulkotiloissa olevien laitteiden käyttöliittymistä. Mikäli sovelluksesta toteuttaisi ulkotiloissa käytettävän version, kuvan käsittelystä ennen tunnistamista mahdollisimman selkeäksi olisi todennäköisesti hyötyä.

5. POHDINTAA

Tärkeä kysymys tällaisen järjestelmän kohdalla on tietenkin, onko sille oikeasti käyttöä oikean maailman tietojärjestelmissä. Suuri haittapuoli järjestelmässä on se, että luotettuna laitteena käytetään käyttäjän älypuhelinta. Tällöin kännykän oletetaan olevan turvallinen ja myös käyttäjän oletetaan suojaavan kännykkänsä näyttöä ainakin jotenkin selkänsä takana olevien ihmisten tai kameroiden katseilta. Kännykän kohdalla edelleen on olemassa riski siitä, että näyttökuvaa pääsee näkemään joku muukin kuin oletettu käyttäjä. Siksi esimerkiksi jonkinlainen silmälasien kaltainen lisätyn todellisuuden laite olisi huomattavasti käyttökelpoisempi. Mikäli kännykkä ei ole täysin turvallinen, ei myöskään järjestelmä ole turvallinen. Ehkä parempi olisi tehdä jokin täysin erillinen laite turvallista kommunikointia varten, eikä luottaa siihen, että käyttäjän älypuhelimeen eivät muut tahot pääse käsiksi.

Peruskonsepti kuitenkin on käyttökelpoinen, kunhan käyttäjän laitteen ja palvelimen välinen tunnistautuminen on toteutettu hyvin. Kuten aiemmin mainittua, esimerkiksi pankkiautomaattien kohdalla oikeassa elämässä törmätään tilanteisiin, missä joudutaan käyttämään olosuhteiden pakosta epäluotettua laitetta. Tämä johtuu siitä, että käyttäjän täytyy käyttää korttia tunnistatumiseen laitteelle. Järjestelmä ei mahdollista kovin monimutkaista kommunikointia, mutta esimerkiksi juurikin pankkiautomaatilla on loppujen lopuksi hyvin rajattu määrä toimintoja.

5.1. Työtä tehdessä vastaan tulleita haasteita

Vaikka Forten paperissa [3] mainituista visualisointivaihtoehdoista QR-koodit olivat todennäköisesti helpoiten ainakin tietyllä tasolla toteutettava vaihtoehto, oli toteutuksessa useita erilaisia haasteita. Paperissa painotettiin sitä, että käyttäjän laitteen sovelluksen pitää pystyä lukemaan useita eri QR-koodeja yhtä aikaa, jotta niiden sijaintia voidaan pitää silmällä. Mitään valmista pakettiratkaisua ei kyseiseen ongelmaan löytynyt, ja hyvän ratkaisun toteuttaminen useamman QR-koodin yhtäaikaiseen tunnistamiseen olikin yksi suurimmista haasteista. Vaikka lopulta toteutus onnistui, ei se ole täydellinen. Viivakoodintunnistusalgoritmi tunnistaa tietyissä tilanteissa QR-koodeja kuvista varsin epäluotettavasti. Jotta tämänkaltainen sovellus olisi käytännöllinen, sovelluksen pitäisi tunnistaa QR-koodeja todella luotettavasti.

Järjestelmän voisi toteuttaa myös jollain muulla visualisointikeinolla kuin QR-koodeilla. Eri visualisointivaihtoehdoilla olisi varmasti omat etunsa ja omat haittansa QR-koodeihin nähden. QR-koodien kohdalla suurimpia heikkouksia ovat niiden vaatimat hiljaiset alueet ja vaihteleva koko.

5.2. Jatkoa työlle

Android-sovelluksen toimintaa voisi parantaa monin tavoin. QR-koodien etsimisen optimointi etenkin isommilla resoluutioilla olisi hyödyllistä. Tällä

hetkellä sovellus toimii ainoastaan puhelimen ollessa pystyasennossa, ja sivuasennossa toimimisen implementointi helpottaisi sovelluksen käyttöä useassa tilanteessa. Yksi keino parantaa sovelluksen toimintaa olisi laittaa viivakoodintunnistusalgoritmi erilliseen säikeeseen suorittamaan QR-koodien tunnistamista kameran kuvasta. Hyvällä toteutuksella sovelluksen suorituskyky paranisi varmasti.

Olennainen osa tietenkin on lisätä järjestelmään toteutus Android-sovelluksen ja palvelimen välisestä turvallisesta avaintenvaihdosta, jota ilman koko järjestelmä ei käytännön tilanteessa toimisi. Ideana on lisätä ihmislähtöinen avaintenvaihtoprosessi, jossa ihminen kykenisi seuraamaan prosessia ja vahvistamaan, että avaintenvaihto palvelimen ja käyttäjän laitteen välillä sujui oikein.

6. YHTEENVETO

Tässä työssä toteutettiin versio Forten ym. paperissaan [3] esittelemästä järjestelmästä *EyeDecrypt*. *EyeDecrypt* suojelee käyttäjän tietoja tilanteessa, missä käyttäjä kommunikoi luotetun palvelimen kanssa julkisessa tilassa epäluotetun laitteen välityksellä. *EyeDecryptissä* informaatio kulkee palvelimelta epäluotetulle laitteelle salattuna, ja käyttäjän luotettu laite lukee salatun informaation epäluotetun laitteen graafisesta käyttöliittymästä ja avaa salauksen.

Toteutetussa versiossa palvelin ja epäluotettu laite ovat yhteydessä toisiinsa TCP-yhteydellä. Tässä työssä oletettiin, että käyttäjän älypuhelin ja palvelin olivat suorittaneet luotetun avaintenvaihdon jo aiemmin. Palvelin lähettää informaatiota salattuna epäluotetulle laitteelle ja epäluotettu laite näyttää informaation lohkoina käyttöliittymässään QR-koodeiksi muutettuna. Käyttäjä lukee QR-koodit älypuhelimellaan olevalla sovelluksella. Sovellus tarkistaa, ovatko QR-koodit oikeilla paikoillaan epäluotetun laitteen käyttöliittymässä ja avaa salauksen sekä näyttää käyttäjälle selkotekstin. Tässä työssä toteutettiin myös keino simuloida käyttäjän tunnistautumista palvelimelle PIN-koodin avulla. Palvelin lähettää mahdolliset PIN-koodin merkit epäluotetulle laitteelle salattuna, ja epäluotettu laite näyttää ne käyttäjälle QR-koodeina kuten muutkin viestit. Käyttäjä käy läpi kännykkäsovelluksen avulla mahdolliset merkit ja klikkaa oikeaa merkkiä vastaavaa QR-koodia epäluotetun laitteen käyttöliittymässä. Joka merkinsyötön jälkeen palvelin arpoo uuden järjestyksen merkeille.

Toteutettu järjestelmä vaikeuttaa käyttäjän datan joutumista vääriin käsiin. Epäluotetulla laitteella data on aina salattuna, joten järjestelmän kaltainen konstruktio suojaa käyttäjää esimerkiksi epäluotetulla laitteella olevilta haittaohjelmilta. Samoin toteutus suojaa käyttäjää kameroiden avulla tai silmin tapahtuvalta tarkkailulta ja näytöltä informaation leviämiseltä, koska käyttäjän tiedot näkyvät salaamattomana vain kännykän ruudulla. Forten paperissa [3] tarkemmin esitelty järjestelmä suojaa käyttäjää myös aktiivisilta hyökkäyksiltä, esimerkiksi *epärehellinen välittäjä* -hyökkäykseltä.

7. VIITTEET

[1] Diffie W. & Hellman M.E. (1976) New directions in cryptography. IEEE Transactions on Information Theory 22, pp. 644–654.

[2] (2008) The keyed-hash message authentication code (HMAC) (FIPS PUB 198-1). Standardi, National Institute of Standards and Technology.

[3] Forte A.G., Garay J.A., Jim T. & Vahlis Y. (2014) Eyedecrypt - private interactions in plain sight. In: Security and Cryptography for Networks. SCN 2014. Lecture Notes in Computer Science.

[4] Zhang B., Ren K., Xing G., Fu X. & Wang C. (2016) Sbvlc: Secure barcode-based visible light communication for smartphones. IEEE Transactions on Mobile Computing 15, pp. 432–446.

[5] Naor M. & Shamir A. (1995) Visual cryptography. In: Advances in Cryptology — EUROCRYPT'94. EUROCRYPT 1994. Lecture Notes in Computer Science.

[6] Kumar M., Garfinkel T., Boneh D. & Winograd T. (2007) Reducing shoulder-surfing by using gaze-based password entry. In: Proceedings of the 3rd Symposium on Usable Privacy and Security, SOUPS '07, ACM, New York, NY, USA, pp. 13–19.

[7] Wiedenbeck S., Waters J., Sobrado L. & Birget J.C. (2006) Design and evaluation of a shoulder-surfing resistant graphical password scheme. In: Proceedings of the Working Conference on Advanced Visual Interfaces, AVI '06, ACM, pp. 177–184.

[8] Ateniese G., Blundo C., De Santis A. & Stinson D.R. (1996) Visual cryptography for general access structures. Information and Computation 129, pp. 86–106.

[9] Lin C.C. & Tsai W.H. (2003) Visual cryptography for gray-level images by dithering techniques. Pattern Recognition Letters 24, pp. 349–358.

[10] Hou Y.C. (2003) Visual cryptography for color images. Pattern Recognition 36, pp. 1619–1629.

[11] Lee M.K. (2014) Security notions and advanced method for human shoulder-surfing resistant pin-entry. IEEE Transactions on Information Forensics and Security 9, pp. 695–708.

[12] (2001) Advanced encryption standard (AES) (FIPS PUB 197). Standardi, National Institute of Standards and Technology.

[13] Josefsson S. (2006), The Base16, Base32, and Base64 Data Encodings, IETF RFC 4648. URL: `https://tools.ietf.org/html/rfc4648`.

[14] qrcode 6.1 (vierailtu 22.7.2019). URL: `https://pypi.org/project/qrcode/`.

[15] Kivy: Cross-platform Python Framework for NUI Development (vierailtu 1.8.2019). URL: https://kivy.org.

[16] Zxing ("Zebra Crossing") barcode scanning library for Java, Android (vierailtu 6.8.2019). URL: https://github.com/zxing/zxing.