QR-koodien tietoturva sovellusten välisessä tiedonsiirrossa

Turun yliopisto
Tietotekniikan laitos
TkK-tutkielma
Tietotekniikka
Joulukuu 2024
Hannonen Onnimanni

TURUN YLIOPISTO
Tietotekniikan laitos

HANNONEN ONNIMANNI: QR-koodien tietoturva sovellusten välisessä tiedonsiirrossa

TkK-tutkielma, 25 s.
Tietotekniikka
Joulukuu 2024

QR-koodien käyttö sovellusten välisessä tiedonsiirrossa on yleistynyt niiden helppokäyttöisyyden ja standardoinnin ansiosta. Erityisesti Covid-19-pandemian aikana QR-koodien käyttö yleistyi, koska niiden avulla voidaan siirtää tietoa täysin kontaktittomasti. QR-koodien yleisimmät käyttökohteet – esimerkiksi sisäänkirjautuminen, vahvistaminen ja maksaminen – ovat tietoturvan näkökulmasta hyvinkin kriittisiä. Standardin mukainen QR-koodi ei kuitenkaan oletusarvoisesti ole tietoturvallinen; on siis sovelluskehittäjän vastuulla huolehtia, ettei QR-koodien käyttö kasvata sovelluksen hyökkäys pinta-alaa.
Tässä tutkielmassa selvitetään systemaattisen kirjallisuuskatsauksen keinoin, minkälaisia uhkaskenaarioita QR-koodeihin kohdistuu ja miten niitä voidaan minimoida. Kirjallisuudessa esitetyt uhkaskenaariot – *vakoilu*, *tietojenkalastelu*, *peukalointi* ja *injektiohyökkäykset* – havainnollistetaan esimerkkien avulla. Uhkaskenaarioiden minimoimiseksi esitetyt ratkaisut – *salaus*, *steganografia*, *piilotus* ja *verifiointi* – kootaan ja analysoidaan niiden rajoitteita. Tunnistettujen rajoitteiden pohjalta tutkielmassa esitetään teoreettista pohjaa uudelle monikerroksiselle QR-koodien tietoturvaa parantavalle menetelmälle.

Asiasanat: QR-koodi, tietoturva, tiedonsiirto

Sisällys

1 Johdanto

QR-koodit ovat jo pitkään vakiinnuttaneet asemaansa helppona tiedonsiirtomenetelmänä kahden sovelluksen välillä, toimien usein linkkinä fyysisen ja digitaalisen maailman välillä [1]. QR-koodien avulla pystytään tekemään monia muuten kosketuskontaktia vaativia asioita täysin kontaktittomasti, minkä seurauksena niiden käyttö yleistyi ennen kaikkea Covid-19-pandemian aikana [2].

Yleisimpiä käyttökohteita ovat käyttäjäprofiilien jakaminen, sisäänkirjautuminen, vahvistaminen, jäsenyyden tason osoittaminen, alennuskoodien esittäminen ja maksaminen. Nämä käyttökohteet ovat tietoturvan näkökulmasta hyvinkin kriittisiä. Siirretty data voi olla henkilötietoja, uniikkeja tunnisteita tai muuta arkaluontoista tietoa, minkä avulla hyökkääjä voi aiheuttaa suurta vahinkoa tai saavuttaa taloudellista etua. [1], [3], [4]

Ihmiselle kryptisen näköinen QR-koodi on standardoinnin ansiosta universaali tietokoneiden ymmärtämä formaatti. Tässä helppokäyttöisyydessä piileekin useita uhkaskenaarioita – standardi ei sisällä tietoturvaa parantavia ominaisuuksia, eli on sovelluskehittäjän vastuulla varmistua, ettei QR-koodien käyttö heikennä sovelluksen tietoturvaa [1]. QR-koodin renderöijien ja lukijoiden käyttö sovelluksessa on helppoa, mutta helppokäyttöisyydellä on hintansa. Sovellus ja sen loppukäyttäjä eivät välttämättä edes tiedä joutuneensa hyökkäyksen kohteeksi. Niillä ei ole keinoja tietää, voiko luettuun standardin mukaiseen QR-koodiin luottaa, eivätkä ne voi olla varmoja, ketkä kaikki saattavat lukea renderöidyn koodin sisällön olan yli tai piilokameran välityksellä [3], [4].

Tässä tutkielmassa selvitetään kirjallisuuskatsauksen keinoin, minkälaisia uhkaske-

naarioita QR-koodeihin liittyy kahden sovelluksen välisessä tiedonsiirrossa. Työssä tarkastellaan myös, mitä keinoja kirjallisuudessa on esitetty sovelluskehittäjille näiden uhkaskenaarioiden minimointiin. Tutkielma pyrkii siis löytämään vastaukset seuraaviin tutkimuskysymyksiin:

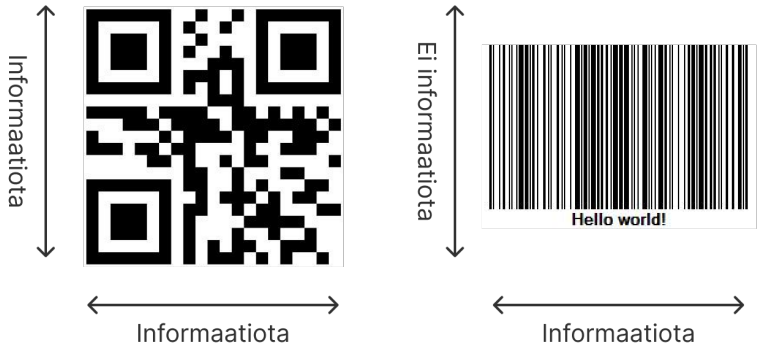
- **TK1:** Minkälaisia uhkaskenaarioita QR-koodeihin liittyy kahden sovelluksen välisessä tiedonsiirrossa?
- **TK2:** Miten näitä uhkaskenaarioita voidaan minimoida?

Luvussa 2 esitetään yleisellä tasolla QR-koodien toimintaperiaate ja tarkastellaan tietoturvan perusperiaatteita. Luvussa tutustutaan myös QR-koodien historiaan ja rakenteeseen, tiedonsiirron virheenkorjaukseen sekä CIA-malliin. Luvussa 3 dokumentoidaan tutkielmassa käytetyt menetelmät ja avataan kirjallisuuskatsauksen vaiheet helposti toistettavaan muotoon. Luvut 4 ja 5 ovat toisiaan täydentäviä. Aluksi luvussa 4 esitetään QR-koodeihin liityviä uhkaskenaarioita, joihin taas etsitään ratkaisuja luvussa 5. Lopuksi luvussa 6 tiivistetään kirjallisuuskatsauksen tulokset ja pohditaan jatkotutkimuksen mahdollisuuksia.

2 QR-koodit ja tietoturva

2.1 QR-koodit

QR-koodi (engl. *Quick Response Code*) on vuonna 1994 kehitetty ja vuonna 2000 standardoitu kaksiulotteinen matriisikoodi. Se kehitettiin alun perin autotehtaille korvaamaan riittämättömäksi todettu yksiulotteinen viivakoodi [5]. QR-koodiin voi tallentaa tietoa vaaka-akselin lisäksi myös pystyakselille, joten saman tiedon tallentaminen QR-koodiin vie noin kymmenen kertaa vähemmän pinta-alaa kuin yksiulotteiseen koodiin. Lisäksi QR-koodeissa on kameran vääristymiä korjaavia elementtejä sekä tiedonsiirron virheitä korjaavia bittejä. Yksiulotteisen ja kaksiulotteisen koodin erot ovat esitelty kuvassa 2.1. [1], [6], [7]



Kuva 2.1: QR-koodi (vasen) ja yksiulotteinen viivakoodi (oikea)

QR-koodin voi renderöidä standardin mukaisella enkooderilla (engl. *encoder*). En-

kooderi ottaa syötteenä QR-koodiin tallennettavan datan, muuntaa sen biteiksi ja renderöi sen pohjalta QR-koodin. Lukeminen taas on käänteinen operaatio renderöintiin nähden. Dekooderi (engl. *decoder*) saa syötteenä kameran kuvan ja muuntaa sen bittien kautta alkuperäiseksi dataksi. [4]–[7]

Mobiililaitteiden yleistyttyä QR-koodeja alettiin käyttämään laajalti tiedon siirtämisessä älylaitteelle, sekä toiselta älylaitteelta että printtimediasta. Yleisimpiä käyttökohteita nykypäivänä ovat linkkien jakaminen, sisäänkirjautuminen, verifiointi sekä maksaminen. [1], [3]–[5]

2.2 QR-koodien toimintaperiaate

Rakenne

QR-koodit koostuvat sekä toiminnallisista että sisällöllisistä osista (kuva 2.2). Toiminnalliset osat auttavat käsittelemään kuvan QR-koodista sellaiseen muotoon, että sen sisällön voi lukea dekooderilla. Ne mahdollistavat siis monet QR-koodeille tyypilliset ominaisuudet, kuten lukemisen mistä tahansa suunnasta ja kulmasta. Sisällölliset osat puolestaan ovat pelkkää dataa. Ne koostuvat lähetettävästä datasta sekä standardin tarvitsemasta metadatasta, kuten versiotiedosta ja virheenkorjauksen vaatimasta datasta. [6]

QR-koodien koko vaihtelee suuresti käytetyn version ja siirrettävän datan määrän mukaan; eri QR-koodin versioita on yhteensä 44 kappaletta. Versiosta riippuen QR-koodi voi sisältää 9–1273 tavua siirrettävää dataa. Mitä suurempi siirrettävän datan määrä on, sitä suurempi QR-koodiversio tarvitaan sen siirtämiseen. Vaikka eri versiot näyttävät eri kokoisilta, on niiden rakenne silti samankaltainen. [6], [7]

Kuva 2.2: QR-koodin rakenne [6]

Kaikki QR-koodiversiot koostuvat standardin mukaisesti seuraavista osista [6]:

- **Suoja-alue** (engl. *Quiet Zone*) on QR-koodia ympäröivä valkoinen reunus. Sen tarkoitus on erottaa QR-koodi mahdollisesta muusta taustasta, jotta itse koodin osat ovat selkeästi luettavissa.

- **Etsintäkuviot** (engl. *Finder Pattern*) sijaitsevat kaikissa QR-koodin kulmissa oikeaa alakulmaa lukuun ottamatta. Niiden avulla QR-koodi voidaan löytää kuvasta ja sen suunta voidaan päätellä. Ne mahdollistavat QR-koodin lukemisen yksiulotteista viivakoodia helpommin. Lukijan ei tarvitse olla ihan kiinni QR-koodissa ja sen voi lukea mistä tahansa kuvakulmasta.

- **Kohdistuskuviot** (engl. *Alignment Pattern*) ovat etsintäkuvion kaltaisia apukuvioita QR-koodin sisällöllisten osien seassa. Mitä suurempi QR-koodi on, sitä enemmän se tarvitsee kohdistuskuvioita. Niiden avulla vääristynyt kuva voidaan suoristaa dekooderille ymmärrettäväksi.

- **Ajoituskuviot** (engl. *Timing Pattern*) ovat peräkkäisiä mustia ja valkoisia neliöitä, joista dekooderi voi päätellä yksittäisen moduulin leveyden. QR-koodeilla on useita eri versioita ja kokoja, ja näistä jokaisella on omat moduulikokonsa.

- **Formaattitieto** (engl. *Format Information*) on metadataa, joka kertoo QR-koodin virheenkorjauksen tason sekä käytetyn peitekuvion (engl. Mask Pattern).

- **Versiotieto** (engl. *Version Information*) kertoo, mitä QR-koodin standardin mukaista versiota enkooderi on käyttänyt QR-koodin luomiseen. Eri QR-koodin versioita on yhteensä 44. Jotta dekooderi voi ymmärtää QR-koodin sisällön, tulee sen käyttää samaa versiota kuin millä kyseinen QR-koodi on luotu.

- **Virheenkorjauksen koodisanat** (engl. *Error Correction Codewords*) ovat joukko kahdeksan bitin mittaisia Reed-Solomon-koodeja, joiden avulla dekooderi voi korjata datasta mahdolliset siirron aikana tapahtuneet virheet.

- **Datan koodisanat** (engl. *Data Codewords*) sisältävät varsinaisen datan, jota QR-koodin välityksellä siirretään. Se koostuu kahdeksan bitin mittaisista osista, joita kutsutaan koodisanoiksi.

Peitekuvio

Jos liian iso osa QR-koodista on pelkästään mustaa tai valkoista pikseliä, on dekooderin vaikea ymmärtää sen sisältöä. Peitekuvio parantaa QR-koodin kontrastia poistamalla liian suuret yksiväriset alueet. Erilaisia peitekuvioita on yhteensä kahdeksan (taulukko 2.1). Niiden ehdot on valittu tuottamaan mahdollisimman erilaisia kuvioita keskenään, jotta vähintään yhdellä niistä saadaan aikaan hyvä kontrasti. Peitekuvio asetetaan aina vain QR-koodin sisällöllisille osille, sillä toiminnallisten osien muokkaus estäisi QR-koodin lukemisen. [1], [6]

Peitekuvio asetetaan QR-koodiin käymällä läpi sen jokainen pikseli. Pikselin koordinaatin vaakakomponentti i ja pystykomponentti j syötetään peitekuvion ehtolausekkee-

seen (taulukko 2.1). Ehtolausekkeen tulos sekä pikselin arvo syötetään XOR-operaattorille (Exclusive OR), jonka tuloksesta tulee pikselin uusi arvo. Käytännössä siis, mikäli pikselin koordinaatit täyttävät ehtolauseen, muutetaan pikselin arvo käänteiseksi (mustasta valkoinen ja valkoisesta musta). Enkooderi asettaa vuorotellen QR-koodille kaikki peitekuviot ja valitsee sen, jolla saavutetaan paras kontrasti. Dekooderi puolestaan lukee käytetyn peitekuvion QR-koodin metadatasta, ja sen perusteella poistaa peitekuvion. [6]

Taulukko 2.1: QR-koodien peitekuviot [6]

Peitekuvio	Ehto
000	(i + j) mod 2 = 0
001	i mod 2 = 0
010	j mod 3 = 0
011	(i + j) mod 3 = 0
100	((i div 2) + (j div 3)) mod 2 = 0
101	(i j) mod 2 + (i j) mod 3 = 0
110	((i j) mod 2 + (i j) mod 3) mod 2 = 0
111	((i+j) mod 2 + (i j) mod 3) mod 2 = 0

Virheenkorjaus

QR-koodien virheenkorjaus perustuu kahdeksan bitin mittaisiin Reed-Solomon-koodeihin. QR-koodeilla on standardin mukaisesti neljä eri virheenkorjauksen tasoa (taulukko 2.2). Mitä suurempi virheenkorjauksen taso on, sitä vähemmän itse dataa QR-koodiin mahtuu. [1], [6]

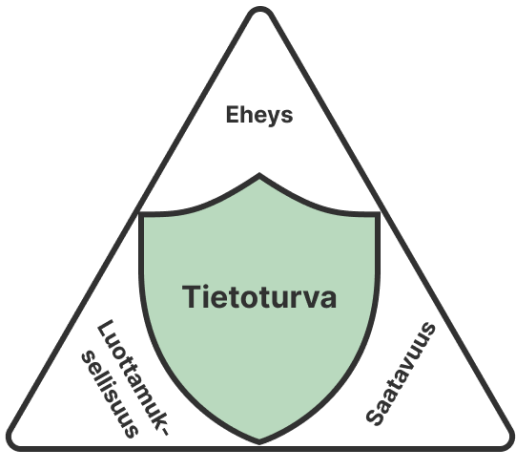
Reed-Solomon-virheenkorjauksen symboleiden määrä **t** on vapaasti valittavissa käyttökohteen mukaan. QR-koodeissa luku määräytyy käytetyn virheenkorjauksen tason sekä QR-koodin version mukaisesti. Mitä suurempi taso tai versio, sitä enemmän QR-koodissa on mukana virheenkorjauksen koodisanoja. Virheenkorjauksen avulla pystytään havaitsemaan **t** virheellistä symbolia sekä korjaamaan **t/2** virheellistä symbolia. [6], [8]

Taulukko 2.2: Reed-Solomon-virheenkorjauksen tasot [6]

Taso	Korjausprosentti	t
L	7%	5 - 750
M	15%	6 - 1372
Q	25%	14 - 2040
H	30%	17 - 2430

2.3 Tietoturva

Kirjallisuudessa tietoturvaa käsitellään usein CIA-mallin avulla (kuva 2.3). Mallin mukaan kokonaisvaltainen tietoturva koostuu kolmesta peruspilarista; Luottamuksellisuus (engl. *confidentiality*), eheys (engl. *integrity*) sekä saatavuus (engl. *availability*). Nämä käsitteet kattavat isossa kuvassa kaikki tietoturvan osa-alueet, joten niitä käytetään usein perustana tietoturva-analyysille yleisellä tasolla. [9]



Kuva 2.3: CIA-malli

Luottamuksellisuus

Tiedon luottamuksellisuudella tarkoitetaan, että yksityinen tieto pysyy yksityisenä. Tietoon on pääsy vain henkilöillä tai tahoilla, joita tieto koskee tai joiden pääsy tietoon on vält-

tämätöntä palvelun toimimisen kannalta. Luottamuksellisuuden vaarantuminen on eräs vaarallisimmista tietoturvaan liittyvistä uhkaskenaarioista, sillä yksityisen tiedon leviäminen usein altistaa myös monille muille henkilöihin, organisaatioihin tai järjestelmiin kohdistuville hyökkäyksille. [9], [10]

Yksityiseen tietoon kohdistuvat hyökkäykset vaihtelevat vaativuudeltaan ja laajuudeltaan suuresti. Yksinkertaisimmillaan hyökkäys voi alkaa bussissa olan yli luetusta salasanasta, kahvilaan unohdetusta tietokoneesta tai väärälle henkilölle lähetetystä sähköpostista. Monimutkaisemmat hyökkäykset taas voivat alkaa tietoturva-aukosta järjestelmässä tai tietokoneelle asennetusta vakoiluohjelmasta. [9], [10]

Eheys

Tiedon eheydellä tarkoitetaan, että tietoa voi muokata ainoastaan siihen luvan saaneet tahot. Luvattoman tiedon muokkaamisen estämisen lisäksi eheyden kannalta on myös olennaista, että mahdolliset muutokset eheyteen kyetään huomaamaan ja perumaan. Kun eheyttä uhkaava muutos huomataan, pitää tieto pystyä palauttamaan viimeisimpään varmasti eheään muotoon. [9]

Eheyden vaarantuminen ei välttämättä tarkoita, että myös luottamuksellisuus olisi vaarantunut. On mahdollista, että hyökkääjä on saanut muokattua tietoa luvatta näkemättä mitään yksityistä tietoa. [10]

Saatavuus

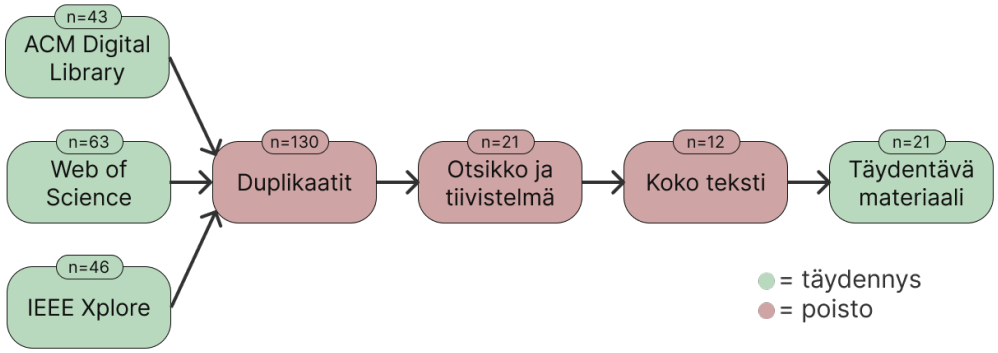
Tiedon saatavuudella tarkoitetaan, että tietoon on pääsy aina kun siihen on tarve. Yleisimpiä uhkaskenaarioita tiedon saatavuudelle ovat luonnonkatastrofit, sähkökatkot, virhetilanteet sovelluksissa ja palvelunestohyökkäykset. Mikäli tiedon saatavuudessa on katkos, on tärkeää, että mitään tietoa ei katoa sen aikana. Tieto tulee pystyä palauttamaan alkuperäiseen tilaan käyttö- tai saatavuuskatkon jälkeen. [9], [10]

3 Metodit

Työ toteutettiin systemaattisena kirjallisuuskatsauksena Kitchenhamin menetelmää [11] noudattaen – sillä erotuksella, että työn luonteen vuoksi lähdeaineiston valitsemisprosessissa oli mukana vain yksi henkilö. Haku suoritettiin 12.10.2024 ja se kohdistettiin ACM Digital Library-, IEEE Xplore- sekä Web of Science -tietokantoihin.

Kirjallisuuskatsauksen vaiheistus on havainnollistettu kuvassa 3.1, ja vaiheet on tarkemmin avattu omissa kappaleissaan. Ennen aineiston karsimista päätettiin sisällyttämiskriteerit. Jokaisessa vaiheessa sisällytettiin vain ne julkaisut, jotka

- ovat kirjoitettu suomeksi tai englanniksi,
- ovat kirjoja, standardeja, tieteellisiä artikkeleita tai konferenssijulkaisuja,
- ovat saatavilla ilmaiseksi verkossa tai Turun Yliopiston kirjaston kautta,
- tuovat esille tai esittävät ratkaisuja QR-koodeihin liittyviin tietoturvaongelmiin,
- ovat sovellettavissa kahden sovelluksen väliseen tiedonsiirtoon.



Kuva 3.1: Tutkimusprosessin vaiheet

Vaihe 1: Hakulausekkeen muodostus ja haku (n=152)

Tutkimuskysymyksistä **TK1** ja **TK2** poimittuja avainsanoja täydennettiin synonyymeillä, tietoturvan peruskäsitteillä sekä muilla aihealueeseen liittyvillä termeillä. Näiden termien pohjalta muodostettiin hakulauseet *(”QR*code” OR ”Quick Response Code”)* sekä *(Confident* OR Integrity OR Availab* OR Secur* OR Threat OR Safe* OR Protect* OR Authentic* OR Valid* OR Encrypt* OR Vulnerabilit* OR Exploit*)*. Nämä hakulauseet yhdistettiin konjunktiolla ja tehtiin tarvittavat rajaukset. Ensimmäistä hakulausetta haettiin vain otsikoista ja toista hakulausetta kaikesta sisällöstä.

Vaihe 2: Duplikaattien poisto (n=130)

Haku suoritettiin kolmeen eri tietokantaan, joista löytyi osittain samoja julkaisuja. Duplikaatit poistettiin, niin että jäljelle jäi vain yksi versio samasta julkaisusta. Mikäli samoilta kirjoittajilta löytyi samasta aiheesta useita erityyppisiä julkaisuja, valittiin aina sisällöltään laajimmat. Valittiin siis kattavat artikkelit tiiviiden konferenssijulkaisujen sijasta.

Vaiheet 3, 4: Poisto otsikon ja tiivistelmän mukaan (n=21), poisto koko tekstin mukaan (n=12)

Poistettiin ne julkaisut, jotka eivät täyttäneet kaikkia sisällyttämiskriteereitä. Tässä vaiheessa korjattiin pois epätäydellisestä hakulausekkeesta johtuvat virheosumat sekä julkaisut, joiden rajaus ei vastaa tutkimuskysymyksiin **TK1** ja **TK2**.

Vaihe 5: Täydennys (n=21)

Lähdeaineistoa täydennettiin yleisellä tasolla tietoturvaa ja virheenkorjausta käsittelevillä teoksilla, QR-koodeihin ja salaukseen liittyvillä standardeilla sekä systemaattisen kirjallisuuskatsauksen kirjallisuudella.

4 Uhkaskenaariot

Tässä luvussa tarkastellaan, minkälaisia QR-koodeihin liittyviä uhkaskenaarioita kirjallisuudessa on esitetty. Uhkaskenaariot on luokiteltu neljään kategoriaan taulukon 4.1 mukaisesti. Tarkastelun tukena käytetään tietoturvan peruspilareita, eli CIA-mallia.

Taulukko 4.1: Lähdeaineiston uhkaskenaariot

Lähde	Uhkaskenaariot			
	Vakoilu	Kalastelu	Peukalointi	Injektiot
Yao ym. 2013 [12]		X		
Xue ym. 2023 [3]	X			
Ismail ym. 2021 [13]		X		X
Kieseberg ym. 2010 [1]	X	X	X	X
Pan ym. 2019 [4]	X			
Zhou ym. 2021 [14]			X	
Sun ym. 2021 [15]			X	
Ahmed ym. 2018 [16]			X	
Loc ym. 2023 [17]			X	
Lei ym. 2021 [18]			X	
Goel ym. 2017 [5]	X		X	
Subairu ym. 2020 [19]		X		

4.1 Vakoilu

QR-koodit ovat kryptisyytensä vuoksi alttiita vakoiluhyökkäyksille (engl. *spying*). Ihminen ei paljaalla silmällä pysty lukemaan QR-koodin sisältöä, joten on helppo langeta ajatteluvirheeseen, että data olisi oletusarvoisesti turvassa [1]. QR-koodit ovat jo niin vakiintuneita, että standardi on sisäänrakennettu useimpien mobiililaitteiden käyttöjärjestelmiin, eli laitteelle sisäänrakennetulla kamerasovelluksella voi lukea QR-koodin sisällön. Mikäli QR-koodeja käytetään arkaluontoisen tai kriittisen datan siirtämiseen, saattaa data päästä helposti vääriin käsiin, ja tiedon luottamuksellisuus vaarantua. Julkisella paikalla olan yli luettu QR-koodi voi vuotaa henkilötietoja kuten nimiä, sähköpostiosoitteita tai puhelinnumeroita [1]. Vakoiluhyökkäys saattaa olla kertaluontoinen ja tarkasti juuri tiettyyn henkilöön kohdennettu tai se voi tapahtua systemaattisesti, vaikkapa kassan läheisyyteen asennetun kameran välityksellä [3], [4].

Etenkin Aasiassa QR-koodeja käytetään paljon maksamiseen tai maksun vahvistamiseen. Ostaja luo taustajärjestelmän kautta maksusitoumuksen, johon linkitetty tunniste renderöidään mobiililaitteella QR-koodiksi. Kassakone lukee tunnisteen QR-koodista, merkkaa maksusitoumuksen taustajärjestelmässä käytetyksi ja suorittaa maksun. Tämän kaltaisissa käyttökohteissa erityisen suurta haittaa aiheuttava vakoiluhyökkäys on niin sanottu STLS-hyökkäys (engl. *Synchronized Token Lifting and Spending*), jossa uhrin kertakäyttöinen koodi varastetaan ja käytetään ennen kuin uhri ehtii itse sitä käyttämään. Maksutilanteissa hyökkääjä voi ottaa kuvan kassajonossa seisovan ostajan QR-koodista, ja käyttää maksusitoumuksen ennen kuin ostaja itse on ehtinyt näin tehdä. [3], [4]

4.2 Tietojenkalastelu

Tiedon luottamuksellisuus voi vaarantua myös tietojenkalastelun (engl. *phishing*) seurauksena. Tällöin yksityinen tieto voi päätyä hyökkääjän käsiin uhrin itse tarjoilemana. Hyökkääjä voi tehdä väärennetyn sovelluksen, jossa on täysin saman näköinen QR-koodin lu-

kija tai renderöijä kuin aidossa sovelluksessa. Luettuaan QR-koodin voi väärennetty sovellus päästä käsiksi yksityiseen dataan, jota QR-koodi sisältää. Vastaavasti väärennetystä sovelluksesta luettu QR-koodi voi ohjata uhrin tekaistulle sivulle, jonka pyrkimyksenä on kalastella henkilö- tai pankkitietoja. Hyökkääjä voi myös ohjata uhrin lataamaan viruksia tai muita haittaohjelmia väärennetyn QR-koodin avulla. [1], [5], [12], [13], [19]

4.3 Peukalointi

QR-koodit ovat yleisyytensä vuoksi erityisen alttiita peukalointihyökkäyksille (engl. *tampering*). QR-koodeja voidaan käyttää esimerkiksi kauppojen alennuskuponkien lukemiseen, kanta-asiakkuuden tason seuraamiseen tai bussilippujen voimassaolon tarkastamiseen. Tämän kaltaiset käyttökohteet ovat hyökkääjille erityisen houkuttelevia paikkoja horjuttaa tiedon eheyttä, sillä suhteellisen helpolla huijauksella voi saavuttaa merkittävää taloudellista etua. QR-koodin datassa voi olla aikaleimoja, numeroita tai uniikkeja tunnisteita, joiden avulla lukija suorittaa varmistuksen. Mikäli hyökkääjä saa muokattua dataa, voi tämä onnistua huijaamaan tarkastajaa antamaan ilmaisia, suurempia tai parempia etuuksia, kuin joihin tämä olisi oikeasti oikeutettu. [1], [14], [17]

Kuten on jo todettu, QR-koodin sisällön saa helposti luettua vaikkapa puhelimen omalla kamerasovelluksella. Enkoodauksen lisäksi standardointi tekee myös dekoodauksesta yhtä helppoa. Internetissä on paljon sovelluksia ja palveluita, joiden avulla kuka tahansa voi luoda uuden QR-koodin mistä tahansa datasta. Hyökkääjän on siis äärimmäisen helppoa lukea QR-koodin sisältö, muokata sitä haluamallaan tavalla ja renderöidä uusi QR-koodi muokatusta datasta. Välttämättä QR-koodia ei tarvitse edes vaihtaa kokonaan, vaan jos muutokset ovat pieniä, voi ne tehdä vanhaan QR-koodiin esimerkiksi valkoisella ja mustalla teipillä. Sovellusten välisessä tiedonsiirrossa koodin renderöijä on näyttö, joten staattisesti näytöllä olevaa QR-koodia voi peukaloida vaikkapa rikkomalla näytön taustavalon ledejä tai yksittäisiä pikseleitä. Peukaloiduista QR-koodeista voi myös luoda uuden huijaussovelluksen, joka näyttää identtiseltä alkuperäisen kanssa. [1], [18]

4.4 Injektiohyökkäykset

Yleinen käyttökohde QR-koodeille on uniikkien tunnisteiden jakaminen. Sosiaalisen median sovelluksissa voi lisätä käyttäjän kaveriksi lukemalla tämän QR-koodin, joka sisältää käyttäjän tunnisteen. Pankkisovelluksissa sisäänkirjautuminen tai maksutapahtuma voidaan hyväksyä lukemalla QR-koodi, joka sisältää session tai maksutapahtuman uniikin tunnisteen. Tämän kaltaiset tunnisteet tallennetaan usein tietokantaan. Jos sovellus luottaa sokeasti QR-koodin sisältöön, saattaa taustajärjestelmä suorittaa tietokantaoperaatioita suoraan koodista poimituilla arvoilla. Tämä avaa mahdollisuuden SQL-injektioille, eli hyökkäyksille, joissa taustajärjestelmälle saadaan salaa syötettyä mielivaltaisia tietokantaoperaatioita. QR-koodin sisältö voidaan peukaloida päättymään komentoon *;drop table <name>*, joka SQL-kyselynä suoritettuna poistaa tietokannasta määrätyn taulun, jolloin tiedon saatavuus vaarantuu. Injektoiduilla SQL-komennoilla voidaan myös muuttaa salasanoja, käyttäjien rooleja, tuotteiden hintoja tai tilien saldoja. [1], [13]

Injektiot voivat myös mahdollistaa komentojen ajamisen palvelimen komentokehotteella. QR-koodeista poimittuja arvoja saatetaan käyttää parametreina komentokehotteella, jolloin QR-koodiin voi suoraan injektoida muita komentoja [13]. Jotkut palvelimet tallentavat ajoitettuja huoltokomentoja SQL-tietokantaan, eli muokkaamalla niihin liittyviä tauluja voidaan palvelimelle ujuttaa omia komentoja [1].

5 Ratkaisut

Tässä luvussa tarkastellaan minkälaisia ratkaisuja kirjallisuudessa on esitetty QR-koodeihin liittyviin uhkaskenaarioihin. Ratkaisut on jaoteltu neljään kategoriaan, kuten taulukossa 5.1 on esitetty. Mielenkiintoista taulukossa on, että kaikki lähdeaineistot esittävät vain yhdenlaisia ratkaisuja, eli useita menetelmiä yhdistävää ratkaisua ei ole tämän kirjallisuuskatsauksen mukaan ole esitetty.

Taulukko 5.1: Lähdeaineiston ratkaisut

Lähde	Ratkaisut			
	Salaus	Steganografia	Piilotus	Verifiointi
Yao ym. 2013 [12]				X
Xue ym. 2023 [3]			X	
Ismail ym. 2021 [13]				X
Kieseberg ym. 2010 [1]				X
Pan ym. 2019 [4]			X	
Zhou ym. 2021 [14]		X		
Sun ym. 2021 [15]		X		
Ahmed ym. 2018 [16]		X		
Loc ym. 2023 [17]		X		
Lei ym. 2021 [18]		X		
Goel ym. 2017 [5]	X			
Subairu ym. 2020 [19]				X

5.1 Salaus

Kirjallisuudessa on esitetty, että tiedon luottamuksellisuuteen liittyviä uhkaskenaarioita voidaan minimoida salaamalla (engl. *encrypt*) lähetettävä data symmetrisellä salausalgoritmilla (AES) [5]. AES on yksi suosituimmista ja eniten käytetyistä salausalgoritmeista, sillä sen matemaattisen pohjan uskotaan olevan murtumaton vielä useiden vuosien ajan [20]. Symmetrisyys tarkoittaa, että data salataan ja salaus puretaan samalla salausavaimella [21]. AES-algoritmissa salausavaimen tulee olla 128, 192 tai 256 bittiä pitkä, mutta oikean mittaisen avaimen voi luoda minkä tahansa salasanan pohjalta KDF-algoritmilla (engl. *Key Derivation Function*) [21]. Kirjallisuudessa avaimen luomiseen on ehdotettu käyttäjän itse luomaa salasanaa. Ennen sekä QR-koodin enkoodausta että dekoodausta käyttäjä syöttää tekstikenttään salasanan, jonka pohjalta luodaan AES-algoritmin käyttämä avain [5]. Näin ollen QR-koodin avulla siirrettyyn dataan ei pääse käsiksi ilman salasanaa.

Monissa käyttötarkoituksissa käyttäjän ei kuitenkaan ole tarkoituksenmukaista syöttää jatkuvasti salasanaansa uudelleen. Datan salausavaimena voidaankin käyttää jotain muuta käyttäjälle uniikkia ja hyökkääjälle tuntematonta tunnistetta. Tämän kaltaisen tunnisteen luomiseen, säilömiseen ja päätelaitteelle lähettämiseen kuitenkin tarvitsee taustajärjestelmän. Vaihtoehtoisesti koko salausoperaatio voidaan tehdä palvelimella ja käsitellä päätelaitteella pelkästään jo valmiiksi salattua dataa. Jos siirrettävä data salataan, voidaan sen joukkoon lisätä verifiointia auttavaa metadataa; Esimerkiksi tietoa ajasta tai paikasta, jossa QR-koodi on enkoodattu. Salauksen ansiosta voidaan varmistua siitä, ettei hyökkääjä pääse peukaloimaan tätä metadataa.

5.2 Steganografia

Tiedon eheyden kannalta on olennaista pystyä havainnoimaan, onko tietoa peukaloitu [9], [10]. Kirjallisuudessa on esitetty steganografisia (engl. *steganography*) menetelmiä, joissa

enkooderi piilottaa QR-koodin dataan tai sen virheenkorjauksen osiin digitaalisia sormenjälkiä tai vesileimoja, joiden pohjalta dekooderi voi varmistua QR-koodin eheydestä [14]–[18]. Tämä turvaominaisuus voidaan siirtää QR-koodin mukana monella eri tavalla. Lähetettävä data voidaan kapseloida, jolloin sormenjälki liitetään itse lähetettävään dataan [16]. Tällöin kuitenkin datan skeema muuttuu, eli dekooderin täytyy osata purkaa kapselointi samalla tavalla. Jos esimerkiksi lähetettävä data on linkki verkkosivulle, ei puhelimen oma kamera osaa avata tätä oikein. Toinen vaihtoehto on piilottaa turvaominaisuus QR-koodiin vasta enkoodauksen jälkeen [17], [18]. Virheenkorjauksen ansiosta QR-koodi pysyy luettavana, vaikka tietyt osat siitä olisivatkin muutettuja. Tällä menetelmällä QR-koodin skeema ei muutu, eli puhelimen omat kamerasovellukset osaavat avata sen sisältämän linkin. Jotta QR-koodin eheys voidaan tarkistaa, täytyy dekooderin kuitenkin osata löytää piilotettu turvaominaisuus QR-koodista. Turvaominaisuudeksi on kirjallisuudessa esitetty:

- DS-algoritmilla (Digital Signature) luotuja sormenjälkiä [16], [18],
- BCH-virheenkorjauskoodeja (engl. *Bose-Chaudhuri-Hocquenghem codes*) [17],
- datan sekoitusta DCT- (engl. *Discrete Cosine Transform*) ja SVD-menetelmiä (engl. *Singular Value Decomposition*) hyödyntäen [15],
- sekä QR-koodia renderöivän näytön virkistystaajuuteen perustuvaa vesileimaa [14].

5.3 Piilotus

Tiedon luottamuksellisuuden parantamiseksi ja STLS-hyökkäysten ehkäisemiseksi kirjallisuudessa on esitetty menetelmiä, joiden tarkoitus on vaikeuttaa QR-koodien lukemista [3], [4]. Menetelmät perustuvat tiedon piilotukseen (engl. *data hiding*), eli niissä QR-koodi on luettavissa vain tietyistä kuvakulmista [4] tai tietyiltä näytöiltä [3]. QR-koodin piilottamiseksi voidaan hyödyntää spatiaalisten taajuuksien epälineaarisuutta [4]. Spatiaalinen

taajuus kuvaa kuinka tiheästi visuaaliset elementit, QR-koodin tapauksessa mustat ja valkoiset alueet, toistuvat kuvassa. Kun kaksi eritaajuista kuviota laitetaan päällekkäin, syntyy uudenlainen niin kutsuttu Moiré-kuvio, jonka ansiosta QR-koodi on näkyvillä vain tietystä kuvakulmasta. QR-koodien piilottamiseksi on myös esitetty näytöille ominaisen PWM-taajuuden (engl. *Pulse Width Modulation*) hyödyntämistä [3]. Tässä menetelmässä QR-koodin lukijalla on keino varmistua sen renderöijän identiteetistä tämän näytön ominaistaajuuden perusteella. Mikäli ominaistaajuus on jotain muuta kuin sen pitäisi olla, ei QR-koodia lueta.

5.4 Verifiointi

Vaikka QR-koodit ovat tietokoneen generoimia, tulee niistä luettua dataa aina käsitellä yhtä epäluotettavana kuin mitä tahansa muuta käyttäjän syöttämää dataa [1]. Kirjallisuudessa on esitetty useita menetelmiä, jotka pyrkivät estämään kalasteluhyökkäyksiä verifioimalla (engl. *verifying*), ettei QR-koodin dataa löydy tunnetuilta mustilta listoilta [12], [13], [19]. Mikäli jokin näistä mustista listoista tunnistaa datan mahdolliseksi huijausyritykseksi, annetaan siitä sovelluksen käyttäjälle varoitus [12], [13], [19]. Datan verifiointi on myös tärkeää SQL- ja komentoinjektioiden välttämiseksi. QR-koodin datalle tulee asettaa tarkka skeema, ja välittömästi dekoodauksen jälkeen tarkistaa, että skeema on oikea [1].

5.5 Tunnettujen ratkaisujen rajoitteet

Kirjallisuudessa on esitetty useita eri menetelmiä, jotka ratkaisevat jonkin tietyn QR-koodeihin liittyvän uhkaskenaarion. Kuten voimme huomata taulukosta 5.1, tämän kirjallisuuskatsauksen perusteella ei ole tehty näitä menetelmiä kokoavaa ja yhdistävää tutkimusta. Yksikään kirjallisuuskatsauksen esittämistä ratkaisuista ei kuitenkaan ole yksinään riittävän vahva tekemään QR-koodeista turvallisia kaikilla tietoturvan osa-alueilla. Edes ratkaisuista vahvin, salaus, ei yksinään turvaa kaikilta hyökkäyksiltä; STLS-

hyökkäyksissä ei tarvitse päästä käsiksi itse salattuun dataan, vaan riittää, että hyökkääjä saa näytettyä saman salatun QR-koodin ennen uhria [3], [4]. Kirjallisuudessa esitetyt ratkaisut STLS-hyökkäysten estämiseksi perustuvat QR-koodien piilotukseen, eli niissä koodi on luettavissa vain tietyistä kuvakulmista [4] tai tietyiltä näytöiltä [3]. Nämä ratkaisut eivät kuitenkaan takaa, ettei datan sisältö pääse vääriin käsiin, sillä itse dataa ei ole salattu. Hyökkääjä voi siis oikeasta kuvakulmasta tai oikealta laitteelta lukea datan sisällön ja käyttää sitä osana hyökkäystä. Kyseiset ratkaisut eivät myöskään esitä datan verifiointia, jolloin järjestelmä saattaa olla altis injektio- tai peukalointihyökkäyksille.

5.6 Teoriaa tietoturvalliselle QR-koodille

Teoriassa QR-koodeista on mahdollista tehdä turvallisia kaikilla tietoturvan osa-alueilla yhdistämällä useita eri tietoturvaa parantavia ratkaisuja, kuten kuvassa 5.1 havainnollistetaan. Tässä tutkielmassa esitetään siis teoriaa uudelle monikerroksiselle menetelmälle QR-koodien tietoturvan parantamiseksi. Menetelmä pohjautuu jo kirjallisuudessa esitettyjen ratkaisuiden yhdistämiseen – sillä lisäyksellä, että QR-koodin piilotukseen käytettäviä vaihtuvia QR-koodeja ei ole kirjallisuudessa tutkittu. Kokoavaan menetelmään on valittu kirjallisuuden ratkaisuista salaus, piilotus sekä verifiointi. Kirjallisuudessa esitetyt steganografiset menetelmät on jätetty huomioimatta, sillä ne ovat salauksen ansiosta merkityksettömiä.

Esitetyssä menetelmässä piilotus perustuu matalaan virheenkorjauksen tasoon sekä datan jakamiseen useaan eri QR-koodiin, joista kukin on vain pienen hetken näkyvillä. Kun virheenkorjauksen taso on matala ja QR-koodi sisältää paljon dataa, pystyy sen lukemaan vain läheltä ja laitteen ollessa paikoillaan. Kun data on pilkottu useampaan QR-koodiin, pitäisi hyökkääjän pystyä lukemaan onnistuneesti useita peräkkäisiä koodeja ilman ainoatakaan virhettä ja samalla reaaliajassa renderöimään koodit omalla hyökkäykseen käytettävällä laitteellaan. Menetelmän esittämää QR-koodien lukumäärä n sekä aikaa T muokataan, kunnes löydetään arvot, joilla saavutetaan QR-koodin piilottava vaikutus.

Kuva 5.1: Tietoturvallinen QR-koodi

QR-koodin luominen

Turvallinen QR-koodi voidaan luoda ja renderöidä päätelaitteella seuraavin askelin:

1. Siirrettävä data kapseloidaan eli paketoidaan yhteen sille olennaisen metadatan, tässä tapauksessa kapseloinnin suoritushetken aikaleiman, kanssa. Päätelaitteen aikaleimojen epäluotettavuuden vuoksi kapselointi tehdään palvelimella, tai aikaleiman synkronointiin käytetään NTP-serveriä (engl. *Network Time Protocol*).

2. Serialisoidaan kapseloitu data ja salataan symmetrisellä salausalgoritmilla (AES).

3. Salattu data jaetaan n yhtä suureen palaseen. Jokaisesta palasesta luodaan QR-koodi matalaa virheenkorjauksen tasoa käyttäen.

4. QR-koodit renderöidään järjestyksessä yksi kerrallaan, siten että jokainen on näkyvillä aina T millisekuntia.

5. Kun kaikki QR-koodit on renderöity, aloitetaan ensimmäinen vaihe uudestaan.

QR-koodin lukeminen

Turvallinen QR-koodi voidaan lukea ja käsitellä päätelaitteella seuraavin askelin:

1. Luetaan vaihtuvia QR-koodeja ja lisätään luettu sisältö pinoon.

2. Aina kun uusi QR-koodi on luettu, yhdistetään pinon n päällimmäisen elementin data, yritetään purkaa symmetrinen salaus ja deserialisoidaan data. Mikäli tämä onnistuu, siirrytään seuraavaan vaiheeseen, muuten jäädään odottamaan seuraavan QR-koodin lukemista.

3. Puretaan kapseloitu data ja varmistetaan aikaleimojen perusteella, että se ei ole yli $n * T$ millisekuntia vanhaa.

Tässä tutkielmassa esitetään vain teoreettista pohjaa jatkotutkimukselle, jonka pohjalta voi mahdollisesti kehittää tietoturvallisen QR-koodin. Salauksen ja verifioinnin osalta teoriaa voi pitää täysin toteuttamiskelpoisena, sillä vastaavia menetelmiä on jo aikaisemmin testattu ja dokumentoitu tieteellisesti. Esitetyn menetelmän vaihtuviin QR-koodeihin perustuva datan piilottaminen taas on huomattavasti kokeellisempi ratkaisu, ja sen soveltuvuus vaatii tarkempia kokeita ja analyysiä. Pitää siis luoda koeskenaarioita mahdollisista hyökkäyksistä, ja kokeilla, pystytäänkö löytämään sellaiset parametrit n ja T, joilla saadaan QR-koodi piilotettua riittävällä tarkkuudella. Haasteena voi olla, että sopivia parametreja ei löydetä, jolloin piilotus on aina joko liian tehokas tai liian epätehokas. Tätä saattaa vaikeuttaa myös se, että laitteiden näytöillä ja kameroilla on eri virkistystaajuuksia, minkä johdosta parametrit voivat toimia yhdellä laitteella ja olla toimimatta toisella.

6 Johtopäätökset

QR-koodit ovat helppokäyttöisyytensä ja kontaktittoman luonteensa vuoksi laajalti käytössä sovellusten välisessä tiedonsiirrossa. Sovelluskehittäjien on helppo ottaa standardin mukainen QR-koodi käyttöön, mutta se ei oletusarvoisesti ole kovin tietoturvallinen. On siis sovelluskehittäjän vastuulla huolehtia, ettei QR-koodien käyttö kasvata järjestelmän hyökkäys pinta-alaa.

Tutkielmassa etsittiin vastauksia tutkimuskysymykseen: ”***TK1**: Minkälaisia uhkaskenaarioita QR-koodeihin liittyy kahden sovelluksen välisessä tiedonsiirrossa?*”. Kirjallisuudessa on esitetty useita erilaisia uhkaskenaarioita, tarkasti kohdennetuista hyökkäyksistä yksittäisiä henkilöitä kohtaan aina laajamittaisiin ja systemaattisiin hyökkäyksiin suuria ihmisryhmiä kohtaan. Suurin osa uhkaskenaarioista liittyy siihen, ettei ihminen pysty itse paljaalla silmällä lukemaan QR-koodin sisältöä. Kun käyttäjä ei tiedä koodin sisältöä, hän ei ymmärrä olla varovainen sen suhteen, eikä hän pysty ilman ulkoisia järjestelmiä tarkistamaan sen eheyttä. Vastauksena tutkimuskysymykseen **TK1** kirjallisuudessa on esitetty seuraavia QR-koodeihin liittyviä uhkaskenaarioita:

- Vakoilu: Hyökkääjä pääsee käsiksi QR-koodin sisältämään arkaluontoiseen dataan.
- Tietojenkalastelu: Hyökkääjä saa väärennetyllä QR-koodilla kalasteltua uhrilta arkaluontoista dataa.
- Peukalointi: Hyökkääjä pääsee muokkaamaan QR-koodin sisältöä aiheuttaen uhrille harmia tai itselleen taloudellista hyötyä.

- Injektiohyökkäykset: Haavoittuvaan järjestelmään voidaan ujuttaa komentokehote- tai SQL-komentoja QR-koodin kautta.

Tutkielmassa etsittiin vastauksia tutkimuskysymykseen: ”***TK2**: Miten näitä uhkaskenaarioita voidaan minimoida?*”. Kirjallisuudessa on esitetty useita eri ratkaisuja korjaamaan yksittäisiä tietoturva-aukkoja QR-koodeihin liittyen. Vastauksena tutkimuskysymykseen **TK2**, kirjallisuudessa esitettyjä ratkaisuja ovat:

- Salaus: QR-koodilla renderöitävä data salataan, jotta yksityinen tieto ei vuoda eikä sitä pääse peukaloimaan.
- Steganografia: QR-koodiin piilotetaan turvaominaisuus, jonka avulla sen eheys voidaan tarkistaa.
- Piilotus: QR-koodin lukemista hankaloitetaan, jotta hyökkääjä ei pysty lukemaan sitä olan yli tai valvontakameran välityksellä.
- Verifiointi: QR-koodin sisältämän datan skeema verifioidaan ja sisältö tarkistetaan mahdollisten huijausten varalta.

Vastaus tutkimuskysymykseen **TK2** löytyi kirjallisuuskatsauksessa kuitenkin vain osittain. Kirjallisuudessa on esitetty monia QR-koodien tietoturvaa parantavaa keinoja, mutta samalla on todettu, ettei yksikään keinoista ole yksinään turvallinen kaikilla tietoturvan osa-alueilla. Tässä tutkielmassa on esitetty teoreettista pohjaa uudelle ratkaisulle, joka on yhdistelmä monista jo kirjallisuudessa esitetyistä ratkaisuista. Tällä monikerroksisella menetelmällä pystyttäisiin teoriassa saavuttamaan turvallinen QR-koodi. Tutkielmassa esitetty monikerroksinen menetelmä on kuitenkin vasta käsitteellinen, joten täydellisen vastauksen esittäminen tutkimuskysymykseen **TK2** vaatii jatkotutkimusta.

Lähdeluettelo

[1] P. Kieseberg, M. Leithner, M. Mulazzani et al., ”QR code security”, teoksessa *Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia*, Association for Computing Machinery, 2010, s. 430–435. DOI: 10.1145/1971519.1971593.

[2] M. Tu, L. Wu, H. Wan, Z. Ding, Z. Guo ja J. Chen, ”The adoption of QR code mobile payment technology during COVID-19: A social learning perspective”, *Frontiers in Psychology*, vol. 12, s. 798 199, 2022.

[3] G. Xue, Y. Li, H. Pan et al., ”ScreenID: Enhancing QRCode Security by Utilizing Screen Dimming Feature”, *IEEE/ACM Transactions on Networking*, vol. 31, nro 2, s. 862–876, 2023. DOI: 10.1109/TNET.2022.3203044.

[4] H. Pan, Y.-C. Chen, L. Yang, G. Xue, C.-W. You ja X. Ji, ”mQRCode: Secure QR Code Using Nonlinearity of Spatial Frequency in Light”, teoksessa *The 25th Annual International Conference on Mobile Computing and Networking*, Association for Computing Machinery, 2019. DOI: 10.1145/3300061.3345428.

[5] N. Goel, A. Sharma ja S. Goswami, ”A way to secure a QR code: SQR”, teoksessa *2017 International Conference on Computing, Communication and Automation (ICCCA)*, Institute of Electrical ja Electronics Engineers, 2017, s. 494–497. DOI: 10.1109/CCAA.2017.8229850.

[6] International Organization for Standardization and the International Electrotechnical Commission, *ISO / IEC18004:2015*, versio 18004:2015, 2015.

[7] S. Tiwari, ”An Introduction to QR Code Technology”, teoksessa *2016 International Conference on Information Technology (ICIT)*, 2016, s. 39–44. DOI: 10.1109/ICIT.2016.021.

[8] S. B. Wicker ja V. K. Bhargava, *Reed-Solomon codes and their applications*. New York: IEEE Press, 1994.

[9] J. Andress, *The basics of information security: understanding the fundamentals of InfoSec in theory and practice*. Syngress, 2014.

[10] M. Stamp, *Information security: principles and practice*. John Wiley & Sons, 2011.

[11] B. Kitchenham, ”Procedures for performing systematic reviews”, *Keele, UK, Keele University*, vol. 33, nro 2004, s. 1–26, 2004.

[12] H. Yao ja D. Shin, ”Towards preventing QR code based attacks on android phone using security warnings”, teoksessa *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, Association for Computing Machinery, 2013, s. 341–346. DOI: 10.1145/2484313.2484357.

[13] S. Ismail, M. H. Alkawaz ja A. E. Kumar, ”Quick Response Code Validation and Phishing Detection Tool”, teoksessa *2021 IEEE 11th IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, 2021, s. 261–266. DOI: 10.1109/ISCAIE51753.2021.9431807.

[14] A. Zhou, G. Su, S. Zhu ja H. Ma, ”Invisible QR Code Hijacking Using Smart LED”, vol. 3, nro 3, 2019. DOI: 10.1145/3351284.

[15] L. Sun, S. Liang, P. Chen ja Y. Chen, ”Encrypted digital watermarking algorithm for quick response code using discrete cosine transform and singular value decomposition”, vol. 80, nro 7, s. 10 285–10 300, 2021. DOI: 10.1007/s11042-020-10075-5.

[16] H. A. Ahmed ja J. W. Jang, ”Document Certificate Authentication System Using Digitally Signed QR Code Tag”, teoksessa *Proceedings of the 12th International Conference on Ubiquitous Information Management and Communication*, Association for Computing Machinery, 2018. DOI: 10.1145/3164541.3164586.

[17] C. V. Loc, T. X. Viet, T. H. Viet, L. H. Thao ja N. H. Viet, ”Deep learning based-approach for quick response code verification”, vol. 53, nro 19, s. 22 700–22 714, 2023. DOI: 10.1007/s10489-023-04712-3.

[18] J. T. Lei Lei, L. S. Chuin ja F. Ernawan, ”An Image Watermarking based on Multi-level Authentication for Quick Response Code”, teoksessa *2021 International Conference on Software Engineering & Computer Systems and 4th International Conference on Computational Science and Information Management (ICSECS-ICOCSIM)*, Institute of Electrical ja Electronics Engineers, 2021, s. 417–422. DOI: 10.1109/ICSECS52883.2021.00082.

[19] S. Subairu, J. Alhassan, S. Abdulhamid ja J. Ojeniyi, ”A Review of Detection Methodologies for Quick Response code Phishing Attacks”, teoksessa *2020 2nd International Conference on Computer and Information Sciences (ICCIS)*, 2020, s. 1–5. DOI: 10.1109/ICCIS49240.2020.9257687.

[20] J. R. Vacca, *Computer and Information Security Handbook*, Second edition. Elsevier Science, 2012.

[21] National Institute of Standards and Technology (US), *Advanced Encryption Standard (AES)*, 2023. DOI: 10.6028/NIST.FIPS.197-upd1.