TIETO- JA SÄHKÖTEKNIIKAN TIEDEKUNTA

Aleksi Talman
Jyri Kiljo
Eetu Lassi

KVANTTITIETOKONEET JA KVANTTIALGORITMIT KRYPTOGRAFIASSA

Kandidaatintyö
Tietotekniikan tutkinto-ohjelma
Huhtikuu 2024

Talman A., Kiljo J., Lassi E. (2024) Kvanttitietokoneet ja kvanttialgoritmit kryptografiassa. Oulun yliopisto, Tietotekniikan tutkinto-ohjelma, 41 s.

TIIVISTELMÄ

Tässä kandidaatintyössä perehdymme kvanttialgoritmeihin ja kvanttitietokoneiden ominaisuuksiin ja toiminnallisuuksiin, jotka mahdollistavat kvanttialgoritmien käyttämisen ja luomisen. Kvanttitietokoneiden kehitysten myötä mahdollisten nousevien kryptografisten uhkien ymmärtäminen on ensiarvoisen tärkeää tämänhetkisten kryptosysteemien turvaamiseksi. Kvanttialgoritmeilla on käyttökohteita myös kryptografiassa, minkä vuoksi työssä keskitytään kvanttialgoritmeihin kryptografisesta näkökulmasta, sekä esitellään kvanttiturvallista kryptografiaa.

Tässä työssä tutkitaan myös hakuongelmia nopeuttavaa Groverin algoritmia, jonka käytännön toteutuksen keskiössä on tutkia sen teoreettista nopeutusta hakuongelmiin, sekä väsytyshyökkäyksiin. Käytännön toteutuksella suoritetaan useita etsintöjä kvanttitietokoneella Qiskit-viitekehyksen avulla. Tuloksia pohditaan sekä kryptografisesta että algoritmin tehokkuuden näkökulmista.

Työ tarjoaa tietoa kvanttitietokoneista ja kvanttialgoritmeista kryptografiassa, sekä niiden nykytilasta ja kehitysasteesta. Työssä pohditaan Groverin algoritmin potentiaalia saavutettujen tuloksien pohjalta, sekä annetaan uusia tutkimuskysymyksiä.

Avainsanat: Groverin algoritmi, kvanttiportit, kvanttipiirit, hakuongelmat, hakualgoritmit, kvanttiturvallinen kryptografia

Talman A., Kiljo J., Lassi E. (2024) Kvanttitietokoneet Ja Kvanttialgoritmit Kryptografiassa. University of Oulu, Degree Programme in Computer Science and Engineering, 41 p.

ABSTRACT

In this bachelor's thesis the focus is on exploring quantum algorithms and the features and functionalities of quantum computers that make the creation and utilization of quantum algorithms possible. Understanding potential threats on cryptography that arise when quantum computers keep developing is crucial for modern day cryptosystem security. Quantum algorithms are used widely in cryptography, thus why quantum algorithms in cryptography is also studied in this thesis, with an introduction to post quantum cryptography.

An interesting quantum algorithm for search problems called Grover's algorithm is also examined, from which a practical implementation is done focusing on it's theoretical speed up of search problems and brute force attacks. Using the Qiskit platform, multiple searches with Grover's algorithm are performed on a quantum computer, with the results evaluated from both a cryptographic perspective and a computational efficiency viewpoint.

This work provides information about quantum computers and quantum algorithms in cryptography and the current development state of this field. It also offers reflections on the potential of Grover's algorithm based on found results, and presents new research questions.

Keywords: Grover's algorithm, quantum circuits, quantum gates, search problems, search algorithms, post quantum cryptography

SISÄLLYSLUETTELO

ALKULAUSE

Kvanttilaskenta sekä kvanttietokoneet ovat aihealueina erittäin mielenkiintoisia. Niiden jatkuvan kehittymisen vaikutus kryptografiaan on erittäin ajankohtaista, minkä myötä kiinnostuimme aiheesta heti siitä kuultuamme. Vaikka aiheen valinta oli meille hyppy tuntemattomaan, onnistuimme luomaan mielenkiintoisen kokonaisuuden työstämme.

Isot kiitokset Oulun Yliopiston kyberturvallisuuden professorille Kimmo Haluselle työmme ohjaamisesta, tuesta prosessin aikana sekä erilaisien näkökulmien jakamisesta viikottaisissa ohjaustapaamisissa. Haluamme myös kiittää Michael Nielseniä ja Isaac Chuangia heidän kirjoittamasta teoksesta "Quantum Computation and Quantum Information", joka kertoo laajasti kvanttilaskennasta. Heidän teos toimi pohjana sekä taustatutkimuksessa, että meidän omalle ymmärryksellemme aiheesta.

Oulussa 26. huhtikuuta 2024

Aleksi Talman
Jyri Kiljo
Eetu Lassi

LYHENTEIDEN JA MERKKIEN SELITYKSET

AES	Advanced Encryption Standard
BQP	Bounded-error Quantum Polynomial Time
ECC	Elliptic Curve Cryptography
EPR	Einstein–Podolsky–Rosen paradoksi
LWE	Learning With Errors
NIST	National Institute of Standards and Technology
NP	Nondeterministic Polynomial Time
P	Polynomial Time
QSS	Quantum Secret-Sharing
RSA	Riverst-Shamir-Adleman salausmenetelmä
b	salausavaimen pituus
L	salasanan pituus
N	alkioiden lukumäärä
n	kubittien määrä
S	systeemin tila

1. JOHDANTO

Teoria kvanttitietokoneille syntyi 1980-luvulla [1, 2], mutta ensimmäiset toiminnalliset kvanttitietokoneet saatiin toimintaan vasta 2000-luvulla. Kvanttitietokoneita on siitä lähtien pyritty kehittämään lisäämällä niiden kokoa kasvattamalla kubittien määrää, parantamalla laatua ja vähentämällä laskennassa syntyviä virheitä. Edellä mainituilla tavoilla saadaan kasvatettua kvanttitietokoneiden laskentatehokkuutta. Kvanttitietokoneiden tehokkuus perustuu kubittien, kvanttitietokoneiden kantayksiköiden, hyödyntämiseen kvanttimekaniikan periaatteiden mukaisesti. [3, 4, 5]

Edistyneimmillä kvanttitietokoneilla kapasiteetti on tällä hetkellä noin 1000 kubittia, kun taas julkisesti saatavilla olevat koneet ovat vielä kooltaan 128 kubittia. Kvanttitietokoneet ovat vielä kehitysvaiheessa ja niiden käyttökohteet ovat rajallisia. Koneiden ja algoritmien kehitys jatkuu kuitenkin edelleen, minkä vuoksi niillä tulee olemaan suuria vaikutuksia erityisesti kryptografiassa ja optimoinnissa. [4, 6]

Kvanttitietokoneiden kehitys herättää huolen kryptografian turvallisuudesta. Perinteiset salausmenetelmät perustuvat laskennallisiin ongelmiin, jotka voivat olla haavoittuvaisia kvanttitietokoneiden kehityksen myötä. Tämä huoli juontaa juurensa kvanttitietokoneiden kykyyn ratkaista jotkin laskennalliset ongelmat huomattavasti nopeammin kuin perinteiset tietokoneet[3].

Kryptografiaan suoranaisesti liittyviä kvanttialgoritmeja ovat muun muassa Shorin ja Groverin algoritmit. Shorin algoritmi on suunniteltu tehokkaaseen alkutekijöihin jakamiseen[7], kun taas Groverin algoritmi keskittyy kvanttitehokkaaseen hakuongelmien ratkaisuun [8]. Esimerkiksi Shorin algoritmin on todettu olevan uhka nykyiselle kryptografialle, joka perustuu alkulukujen faktorintiin ja diskreetteihin logaritmeihin[9]. Se saattaa tulevaisuudessa kehityksensä myötä pystyä murtamaan nykyisen pankki- ja liiketoimintamaailman salauksien kulmakivenä toimivan RSA-salausmenetelmän. Kyseisen salausmenetelmän murtuminen aiheuttaisi laajoja ongelmia niin yksityishenkilöille, yrityksille kuin valtioillekin ympäri maailman. Näiden kvanttialgoritmien luoma uhka on kuitenkin vielä kaukana tulevaisuudessa, kunnes molemmista algoritmeista saadaan luotua tarpeeksi tehokas toteutus ajettavaksi kvanttitietokoneilla.[10, 11]

Kvanttitietokoneisiin kohdistuvat hyökkäykset ovat kuitenkin aktiivisesti tutkimuksen kohteena, mikä vaikuttaa uhkien torjuntaan positiivisesti. Niiden mahdollisiin seurauksiin on reagoitu ajoissa ja osa salausmenetelmistä on jo luokiteltu käyttökelvottomiksi kvanttiaikakaudella. Muun muassa NIST tekee jatkuvaa tutkimusta ja kehitystyötä kvanttialgoritmien ja kvanttiturvallisen kryptografian kehittämiseksi [12]. Yksi tuoreimmista tuloksista kryptografiaan liittyen on Yilei Chenin tutkimus, mikä kohdistuu hilapohjaiseen kryptografiaan. Tässä tutkimuksessa Chen on kehittänyt kvanttialgoritmin, jolla saadaan ratkaistua hilapohjaisessa kryptografiassa vaikeana tunnettu matemaattinen *LWE*-ongelma [13], joka toimii pohjana useissa kryptosysteemeissä.

Tässä kandidaatintyössä tutkitaan kvanttialgoritmeja, sekä käymme läpi kvanttitietokoneiden kvanttialgoritmeille tärkeitä ominaisuuksia. Avaamme kvanttialgoritmien nykytilannetta myös kryptografian näkökulmasta ja perehdymme myös hieman kvanttiturvalliseen kryptografiaan. Työn tarkoituksena on erityisesti keskittyä Groverin algoritmiin, josta luomme toteutuksen suoritettavaksi

kvanttitietokoneella. Perehdymme Groverin algoritmin kvanttitietokoneelta vaatimiin ominaisuuksiin, tulemme vertailemaan löydöksiämme niin aikaisempien tuloksien kuin saavuttaviemme tuloksien valossa, sekä keskustelemaan mahdollisista vaikutuksista kryptografiaan.

2. KVANTTITIETOKONEET

Tässä luvussa käydään läpi tärkeitä kvanttitietokoneen ominaisuuksia, jotka mahdollistavat kvanttialgoritmien toiminnan. Nämä ominaisuudet mahdollistavat myös myöhemmin käsiteltävän Groverin algoritmin kvanttipiirin luomisen. Käymme myös läpi kuinka kvanttitietokoneelle algoritmeja kuvaillaan kvanttipiirin ja kvanttiporttien avulla.

2.1. Kvanttitietokoneen ominaisuudet

Klassisissa tietokoneissa käytetään bittejä laskutoimituksissa. Nämä bitit voivat saada joko arvon 0 tai 1. Kvanttitietokoneet sen sijaan hyödyntävät kubitteja, jotka voivat saada myös arvon 0 tai 1, mutta ne voivat myös saada jonkun yhdistelmän 0 ja 1 arvoista. Tätä tilaa kutsutaan nimellä **superpositio**.

Superpositio perustuu kvanttitietokoneissa käyttettyjen partikkeleiden ominaisuuteen. Esimerkiksi elektronin tila voi olla joko 0 tai 1. Jokainen superposition kvanttimekaaninen tila sisältää todennäköisyyden sille, että partikkelin ominaisuus saa arvon 0 tai 1, jota voidaan kuvailla tilalla

$$\alpha|0\rangle + \beta|1\rangle$$

joissa α ja β kertoimet ovat todennäköisyysamplitudit sille, että kubitin arvo on 0 ja 1 vastaavasti. α ja β noudattavat myös sääntöä

$$\left|\alpha^2\right| + \left|\beta^2\right| = 1$$

joka rajaa superposition todennäköisyydet, jolloin kvanttimekaanisella tilalla

$$\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

on yhtä suuri todennäköisyys palauttaa arvo 0 tai 1. Superposition avulla kvanttitietokoneella n-kubitin kvanttimekaaniset tilat ovat siis 2^n ulotteisia, minkä takia niiden esittäminen klassisella tietokoneella vaatisi 2^n eri arvoa. On kuitenkin tärkeä muistaa, kun kubitin arvo lasketaan se voi palauttaa vain arvon 0 tai 1. eli n-kubitin kvanttimekaaniset tilat voi laskemalla palauttaa vain yhden tilan.[5 p.13-14]

Kvanttitietokoneissa on myös toinen ilmiö, jonka voi havaita esimerkiksi 2 kubitin kvanttimekaanisesta tilasta.

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

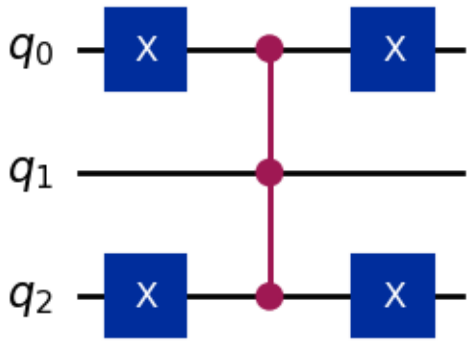
Tästä tilasta käytetään myös nimitystä *Eng. Bell tila* (Bell state) tai **EPR-pari** (Eng. EPR-pair). Kubitit ovat tilassa keskenään lomittuneet. Kun toisen kubitin arvo

lasketaan, tila romahtaa, jolloin seuraavan kubitin arvo määräytyy lasketun kubitin arvon ja valitun Bell tilan mukaan. Tässä tapauksessa bell state on 00 ja 11, kun kumman tahansa kubitin arvo lasketaan on toisen kubitin arvo sama kuin lasketun. Tätä ilmiötä kutsutaan **kvanttilomittumiseksi** (Eng. Quantum entanglement). Bell tilat ja EPR parit ovat myös tärkeäitä osia esimerkiksi kvanttiteleporttauksessa (Eng. Quantum teleportation) ja tiheä koodauksessa (Eng. Super dense coding) [5 p.25-26]

Kolmas ilmiö, joka kvanttitietokoneilla voidaan toteuttaa, on seuraavanlainen: n-kubittia on alustettu superpositioon vastaamaan jokaista 2^n klassissen bitin syötettä. Kun alustettuihin kubitteihin sovelletaan funktiota $f(x)$, jäljellä jäävät kvanttimekaaniset tilat ovat kyseisen funktion tulosteita. Tätä ilmiötä kutsutaan **kvanttirinnasteisuudeksi** (Eng. Quantum parallelism) [5 30-32] [14, 15]

2.2. Kvanttipiirit

Klassisissa tietokoneissa tieto siirtyy ja muuttuu käyttäen hyödyksi sähköpiirejä, joissa on logiikkaportteja. Kvanttitietokoneiden toimintaa kuvataan käyttämällä hyödyksi kvanttipiirejä, jotka koostuvat kvanttiporteista ja niistä muodostuvista oraakkeleista. Kvanttipiirien avulla pyritään kuvailemaan eri algoritmien toimintaa.[5 p.17]



Kuva 1. Kvanttipiiri tilan 010 valitsemiseen

2.3. Oraakkeli

Oraakkeli on keskeinen käsite kvanttialgoritmien toimintaa tarkastellessa. Se toimii eräänlaisena mustana laatikkona, joka ottaa syötteeksi binäärisen syötteen ja palauttaa samankokoisen binäärisen vastauksen. Oraakkeli yleisesti määrittää binäärisen funktion

$$f : \{0,1\}^n \to \{0,1\}^m$$

Kvanttialgoritmit hyödyntävät oraakkelia etsimään ratkaisuja ongelmiin tai suorittamaan jonkin toiminnon. Yllä kuvattua binääristä funktiota kvanttialgoritmit käyttävät kyselynä, jolla on tarkoitus etsiä sellainen tila x, joka antaa jonkin tuloksen binäärisestä funktiosta f.

Groverin algoritmissa oraakkeli O on vaihe oraakkeli, jonka tarkoituksena on muuttaa f funktion toteuttavan tilan vaihe. Tämä oraakkeli voidaan ajatella muodossa

$$O|x\rangle = (-1)^{f(x)}|x\rangle$$

[16, 17].

2.4. Kvanttiportit

Kuten edellä mainittiin, kvanttitietokoneiden informaatio sisältyy koneesta riippuvaan partikkeleiden ominaisuuteen. Tätä ominaisuuden arvoa voidaan muuttaa käyttäen hyödyksi sähkömagneettisen säteilyn pulsseja. Näiden pulssien aikaan saamaa muutosta voidaan kuvailla matriisina, jota kutsutaan **kvanttiportiksi** (Eng. Quantum gates). Kvanttiportit voidaan ajatella olevan kvanttitietokoneen versio klassisten tietokoneiden logiikkaporteista. [5 p.17-19,49]

Esimerkkinä voidaan käytää Groverin algoritmissä usein käytettyä NOT-porttia (Pauli-X gate), joka voidaan ajatella muuttavan kvanttimekaanisen tilan

$$\alpha|0\rangle + \beta|1\rangle$$

tilaksi, jossa 0:n ja 1:n arvot on vaihdettu keskenään.

$$\alpha|1\rangle + \beta|0\rangle$$

matriisina tätä tilan muutosta voidaan esittää matriisi muodossa NOT-portilla.

$$X \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Toinen tärkeä portti groverin algoritmin kannalta on Z-portti (Pauli-Z gate). Portti pitää $|0\rangle$ arvon samana ja kääntää $|1\rangle$ merkin jolloin saadaan $-|1\rangle$. Kvanttimekaaninen tila

$$\alpha|0\rangle + \beta|1\rangle$$

muuttuu tilaksi

$$\alpha|0\rangle - \beta|1\rangle$$

Porttia voidaan kuvailla matriisilla

$$Z \equiv \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Hadamardin portti on yksi tärkeimmistä porteista, jota käytetään paljon kvanttipiireissä, sekä myös Groverin algoritmissa. Hadamardin portin avulla voidaan alustaa kubitit superpositioon. Hadamardin porttia voidaan kuvailla matriisilla

$$H \equiv \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Porttia käytetään alustamaan kubittien todennäköisyys amplitudeja. Esimerkiksi yhden kubitin kvanttimekaaninen tila

$$\alpha|0> + \beta|1>$$

on Hadamardin portin jälkeen

$$\frac{1}{\sqrt{2}}|0> + \frac{1}{\sqrt{2}}|1>$$

riippumatta siitä, mitä arvot α ja β olivat. Tätä käytetään groverin algoritmissä alustamaan n-kubitin tilan, jolloin kubiteilla on yhtä suuri todennäköisyys saada jokaisen 2^n klassisen bitin arvon. [5 p.17-19]

2.5. Ohjausportit

Ohjausportit (Control gates) ovat laajennuksia aikasemmin esitettyihin portteihin. Portit toteuttavat logiikan IF-lauseen. Esimerkkinä voidaan pitää aikaisemman NOT-portin ohjausversiota eli ohjattua-NOT-porttia (Controlled-NOT gate) josta käytetään lyhennystä CNOT-portti. Ohjattuihin portteihin tarvitaan vähintään 2 kubittia joista toinen toimii ohjaus kubittina ja toinen kohde kubittina (Control and target qubit). Ohjausportit toimivat lisäämällä käytetyn portin kohde kubittiin vain, jos ohjaus kubitti saa arvon $|1>$, jos ohjaus kubitti saa arvon 0, kohde kubitin arvoa ei muuteta. CNOT-portti siis kääntää kohde kubitin, jos ohjaus kubittin arvo on 1. CNOT-porttia

voidaan kuvailla matriisina

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Myös muut portit voi laajentaa ohjausporteiksi kuten aikaisemmin mainittu Z-portti on laajennettuna ohjattu Z-portti (Controlled Z-gate) josta käytetään lyhennystä CZ-portti.Tämä portti on matriisina

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

[5 p.20-21]

Ohjausportteja voi myös laajentaa lisäämällä ohjaus tai kohde kubittien määrää esimerkiksi aikaisempi CNOT-portista voidaan tehdä CCNOT-portti, josta käytetään nimitystä Toffoli-portti. Portti suorittaa NOT-portin viimeiselle kubitille jos kaksi ohjauskubittia saavat arvon 1. Tämän portin matriisi muoto on

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Matriisi on kooltaan 8x8, sillä 3 kubittia voi saada 2^3 eri arvoa. [5 p.29-30]

3. KVANTTIALGORITMIT JA KRYPTOGRAFIA

Algoritmit ovat menetelmiä, joilla pyritään ratkaisemaan erinäisiä ongelmia. Yleisesti ottaen algoritmit muodostuvat tietokoneen suorittamista matemaattisista laskuista tai peräkkäin suoritetuista komennoista. Klassisten algoritmien ja kvanttialgoritmien eroavaisuus on se, että ainoastaan kvanttitietokone kykenee suorittamaan kvanttialgoritmeja.[18] Tämä ei kuitenkaan tarkoita, että kaikki ongelmat kuuluvat vain toisen algoritmityypin piiriin. Useissa tapauksissa ongelmat ovat ratkaistavissa klassisilla algoritmeilla kuin kvanttialgoritmeillakin. Esimerkiksi Groverin algoritmi on kehitetty hakuongelmien ratkaisemiseen kvanttitehokkaasti, mutta hakuongelmiin on useita klassisiakin ratkaisuja, kuten lineaariset- ja binääriset hakualgoritmit. [19] Periaatteessa kaikki klassiset algoritmit pystytään ajamaan kvanttitietokoneella, mutta kvanttialgoritmeiksi määritellään vain sellaiset algoritmit, joissa käytetään kvanttitietokoneen ominaisuuksia, esimerkiksi kubitin **superpositiota** tai **kvanttilomittumista**. [18]

3.1. Mihin kvanttialgoritmeja voi käyttää

Tällä hetkellä kaksi merkittävintä kompleksisuusluokkaa ovat *P* ja *NP*. *P* kuvastaa ongelmien luokkaa, jotka voidaan ratkaista polynomisessa ajassa klassisella tietokoneella, ja *NP* kuvastaa ongelmien luokkaa joille ei tunneta tehokkaita ratkaisualgoritmejä klassisella tietokoneilla polynomisessa ajassa. Valmista ratkaisua ehdottamalla *NP* luokan ongelmia voidaan kuitenkin tarkistaa klassisella tietokoneella polynomisessa ajassa. Ongelmaluokkaan *P* voidaan luokitella esimerkiksi kysymys "Onko luku x alkuluku?". Vastaavasti ongelmaluokkaan *NP* voidaan luokitella kysymys "Mitkä ovat alkuluvun x tekijät?". Klassinen tietokone ei siis kykene ratkaisemaan *NP* ongelmia polynomisessa ajassa, mutta kvanttitietokoneen kasvava laskentateho mahdollistaa joidenkin *NP* ongelmien ratkaisun. Myöhemmin hieman tarkemmin käsiteltävällä Shorin kvanttialgoritmilla pystytään ratkaisemaan esimerkiksi edellä mainittu "Mitkä ovat alkuluvun x tekijät?"*NP* ongelma. [20]

Kompleksisuusluokkia on muitakin, sillä vuonna 1993 Ethan Bernstein ja Umesh Vazirani määrittelivät uuden kompleksisuusluokan nimeltä "bounded-error quantum polynomial time"(BQP)[21]. Kyseinen kompleksisuusluokka koostuu ongelmista, joihin voidaan vastata "Kyllä"ja "Ei". Kvanttitietokoneet ovat erittäin tehokkaita ratkaisemaan BQP kompleksisuusluokkaan sisältyviä ongelmia. [20]

Kvanttialgoritmeilla on muitakin käyttökohteita. Niitä käytetään esimerkiksi hakuongelmissa ja optimoinnissa. Hakuongelmissa klassiset algoritmit suorittavat tehtävät yleisesti $O(N)$ ajassa, kun taas Groverin kvanttialgoritmi suorittaa saman tehtävän $O(\sqrt{N})$ ajassa. Groverin algoritmia voidaan myös hyödyntää jokaiseen *NP* kompleksisuusluokan ongelmaan[22]. Groverin algoritmia käsitellään perusteellisesti myöhemmissä kappaleissa.

Monet optimointi ongelmat kuuluvat *NP* kompleksisuusluokkaan, ja kyseisiin ongelmiin haetaan ratkaisuja kvanttilaskennasta, koska niiden ratkaiseminen voi tuottaa suurta arvoa esimerkiksi liiketoiminnallisessa mielessä ympäri maailman. Yksi tunnetuimmista optimoinnin ongelmista on kauppamatkustajan ongelma. Ongelmassa pyritään ratkaisemaan, kuinka kauppias pystyy käymään jokaisessa eri

paikassa mahdollisimman vähäisellä matkustusajalla. Esimerkiksi tämän ongelman ratkaisemalla isojen yritysten toimitusketjujen tehokkuus paranee huomattavasti. Klassisilla menetelmillä kyseinen ongelma on kyetty ratkaisemaan 86 000 kaupungille. Kvanttilaskennassa kauppamatkustajan ongelmaan haetaan ratkaisua tekniikasta nimeltä "vaihe estimointi". [23]

3.2. Kryptografia

Kryptografian käyttäminen on pakollista tietoturvan takaamiseksi. Kautta maailman tietokoneiden kovalevyt sisältävät yksityishenkilöiden, yritysten sekä valtioiden yksityistä dataa jotka voivat aiheuttaa paljon haittaa väärin käytettynä. Kryptografia on tieteen ala joka keskittyy viestien salaamiseen käyttäen monimutkaisia tai muuten vaikeasti laskettavissa olevia matemaattisia ongelmia. Yksinkertaisuudessaan idea on seuraava - lähetetyn viestin sisältö naamioidaan käyttämällä kryptografisia salausmenetelmiä. Tätä naamioitua viestin sisältöä eli dataa kutsutaan yleisesti salakirjoitukseksi (Eng. Ciphertext). Salakirjoitus voidaan purkaa alkuperäiseen muotoonsa käyttämällä salauksen purkamiseen luotua algoritmiä. Kryptografiaan liittyy myös toinen tieteen ala, jota kutsutaan kryptoanalyysiksi. Tämän tieteen alan tarkoituksena on analysoida ja murtaa turvalliseksi luultuja kryptosysteemejä, jotka ovat luotu kryptografian avulla. Erilaisiin kryptosysteemeihin keskitytään tarkemmin myöhemmin tässä kappaleessa. [24]

Kryptografialla on kolme kulmakiveä, ne ovat luottamuksellisuus, eheys ja saatavuus (Eng. Confidentiality, Integrity, Availability). Kryptografian avulla lähetetty data naamioidaan siten, että vain haluttu vastaanottaja pystyy purkamaan salauksen ja lukemaan lähetetyn viestin. Tätä kutsutaan luottamuksellisuudeksi. Kryptografialla pyritään myös takaamaan viestin koskemattomuus. Kryptografiset mekaniikat estävät salatun viestin muokkaamisen vaikka se päätyisikin väärän vastaanottajan käsiin. Kryptografialla pyritään myös takaamaan esteetön saatavuus dataan sellaisille henkilöille, joille se on tarkoitettu. [24]

Kryptosysteemit on jaettu kahteen osaan, symmetrisiin ja asymmetrisiin systeemeihin. Symmetrisessä kryptosysteemissä datan salaus ja salauksen purku suoritetaan samalla avaimella. Symmetrisissä kryptosysteemeissä salaus perustuu siihen, että valittu avainluku on niin suuri ja satunnainen, että nykyaikaisten tietokoneiden laskentateho ei riitä sen purkamiseen väsytyshyökkäystä käyttämällä. [24]

Asymmetrisessä kryptosysteemissä avain jaetaan kahteen osaan, julkinen avain (Eng. Public key) ja yksityinen avain (Eng. Private key). Asymmetrisessä systeemissä viestin saaja antaa lähettäjälle julkisen avaimen, jonka avulla lähettäjä salaa viestin. Tämän jälkeen viestin saaja käyttää yksityistä avainta salauksen purkamiseen ja viesti on luettavissa.[24]

3.3. Kvanttialgoritmit kryptografiassa

Kvanttitietokoneet käyttävät superpositiota sekä kvanttilomittumista. Nämä kaksi kvanttitietokoneen ominaisuutta mahdollistavat suoritettavien operaatioiden

eksponentiaalisesti kasvavan laskentatehon lomittuvien kubittien lisääntyessä. Tähän perustuu myös kvanttialgoritmien muodostama uhka nykyaikaiselle kryptografialle.[25] Tällä hetkellä on olemassa kaksi joukosta erottuvaa kvanttialgoritmia, jotka erityisesti uhkaavat kryptografisia menetelmiä. Peter Shorin vuonna 1994 kehittämä **Shorin algoritmi**[26], ja Lov Groverin kehittämä **Groverin algoritmi** [8], jota käsittelemme tarkemmin seuraavassa kappaleessa. Shorin algoritmin uhka kohdistuu erittäin tunnettuun RSA salausmenetelmään. RSA on asymmetrinen salausmenetelmä, joka perustuu alkulukujen tekijöihin jakoon ja diskreetteihin logaritmeihin. Siinä julkisen avaimen luominen perustuu kahden suuren alkuluvun p ja q tuloon, jossa julkinen avain saadaan kaavasta $N = pq$. RSA:n turvallisuus perustuu näiden kahden alkuluvun p ja q löytämisen vaikeuteen[27]. Shorin algoritmi pystyisi teoriassa ratkaisemaan yhtälöstä tuntemattomat alkuluvut p ja q hyvinkin nopeasti, mikä tekisi RSA:sta epäturvallisen salausmenetelmän.

3.4. Kvanttiturvallinen kryptografia

Tarve kvanttiturvalliselle kryptografialle (eng. post quantum cryptography) on kasvanut kvanttitietokoneiden kehittyessä. Sekä Shorin, että Groverin algoritmien nostamat uhat nykyiselle kryptografialle, ovat kiihdyttäneet kvanttiturvallisen kryptografian tutkimusta[27]. Sen tavoitteena on luoda kvanttikestäviä (eng. quantum resistant) kryptografisia menetelmiä, jotka olisivat turvallisia niin kvantti- kuin klassisilla tietokoneilla, ja ne voisivat olla yhteensopivia nykyisten viestintäprotokollien ja verkkojen kanssa. [28]

Merkittävien askelien ottamista varten kryptografiassa tärkeintä on ymmärtää mitä ei kannata tehdä. Siksi tutkimukset kvanttialgoritmien kryptografisista uhista ovat tärkeitä kvanttiaikakauden turvallisuuden, sekä kvanttiturvallisen kryptografian kehityksen kannalta. Vaikka esimerkiksi RSA salausmenetelmän murtamiseen tarvittaisiin tuhansia kubitteja ja miljardeja operaatioita, toistaiseksi ei ole löydetty esteitä, miksei tuota skaalaa voitaisi tulevaisuudessa saavuttaa[27]. Siksi muun muassa NIST on tehnyt tutkimusta kvanttiturvallisesta kryptografiasta kattavasti torjuakseen kvanttitietokoneiden kehityksen mukana mahdollistuvia uhkia[28].

NIST on linjannut, että suurin uhka kvanttitietokoneiden kehityksen myötä kohdistuu asymmetrisiin kryptosysteemeihin, erityisesti julkisen avaimen menetelmiin, kun taas vaikutukset symmetrisen avaimen kryptosysteemeille eivät ole niin massiivisia. NIST järjestää aika-ajoin julkisia kilpailuhenkisiä prosesseja, joihin kuka tahansa voi lähettää algoritmeja arvioitavaksi. Prosessin tarkoituksena on löytää kvanttiturvallisia kryptografisia algoritmeja, joita voidaan hyödyntää kvanttiturvallisen kryptografian kehittymisessä. NIST kutsuu tätä **NIST Post-Quantum Cryptography Standardization**-prosessiksi. Prosessia hyödyntämällä NIST kehittää kvanttiturvallisen kryptografian standardeja. [29, 12]

Vaikka NIST on valinnut algoritmeja jo useita kymmeniä prosessistaan, ovat he todenneet, että mitä todennäköisimmin voi kulua jopa 15 vuotta tulevien standardien julkaisemisen jälkeen, että niistä saadaan luotua täydellinen toteutus. Erityisesti haasteeksi ilmenee se, että uusien kvanttiturvallisen julkisen avaimen salaukseen perustuvien standardien toteuttaminen voi olla jopa vielä haastavampaa, kuin uusien klassisten kryptografisten algoritmien toteuttaminen. Tämä voi tarkoittaa jopa

vuosikymmenten viivästystä sille, että kvanttiturvalliset kryptografiset menetelmät korvaavat klassiset menetelmät. [30]

4. GROVERIN ALGORITMI

Groverin Algoritmi saa nimensä yhdysvaltalaisesta tieteilijästä Lov Groverista, joka esitteli algoritminsa vuonna 1996 julkaisemassaan artikkelissa "A fast quantum mechanical algorithm for database search"[8]. Groverin algoritmin teoreettinen kehitys perustuu David Deutschin ja Richard Jozsan työhön, jossa he osoittivat, että kvanttitietokone voi tarjota huomattavaa nopeutusta tiettyjen ongelmien ratkaisemisessa [8, 3]. Tässä luvussa käsitellään hakuongelmien historiaa ja niiden ratkaisua klassisilla tietokoneilla, kvanttitietokoneiden mahdollistamaa nopeutusta hakuongelmien ratkaisemiseen Groverin algoritmin avulla, sekä Groverin algoritmin nostamia uhkia kryptografialle.

4.1. Hakuongelmien historiaa

Klassiset hakuongelmat perustuvat tiedonhakuun suurista tietomääristä - halutaan etsiä tietoa joka vastaa haluttuja kriteereitä. Ennen tietokoneita se tarkoitti manuaalisesti tietomäärien läpikäymistä. Ajan kuluessa teknologian kehityksen myötä on kehitetty paljon haku- ja lajittelualgoritmejä, jotka auttavat tiedonhaussa. Algoritmien kehittämisen tarkoituksena on ollut minimoida aika, joka kuluu kriteereitä vastaavan tiedon löytämiseen. Tyypillisimpiä hakualgoritmeja ovat lineaariset- ja binääriset hakualgoritmit [31], joita on käytetty pitkään hakuongelmien ratkaisussa. Myös tietorakenteiden ja tietokantateknologian kehittyminen on parantanut tiedonhakua, muun muassa indeksointien ja erilaisten hakukyselyiden avulla. [32]

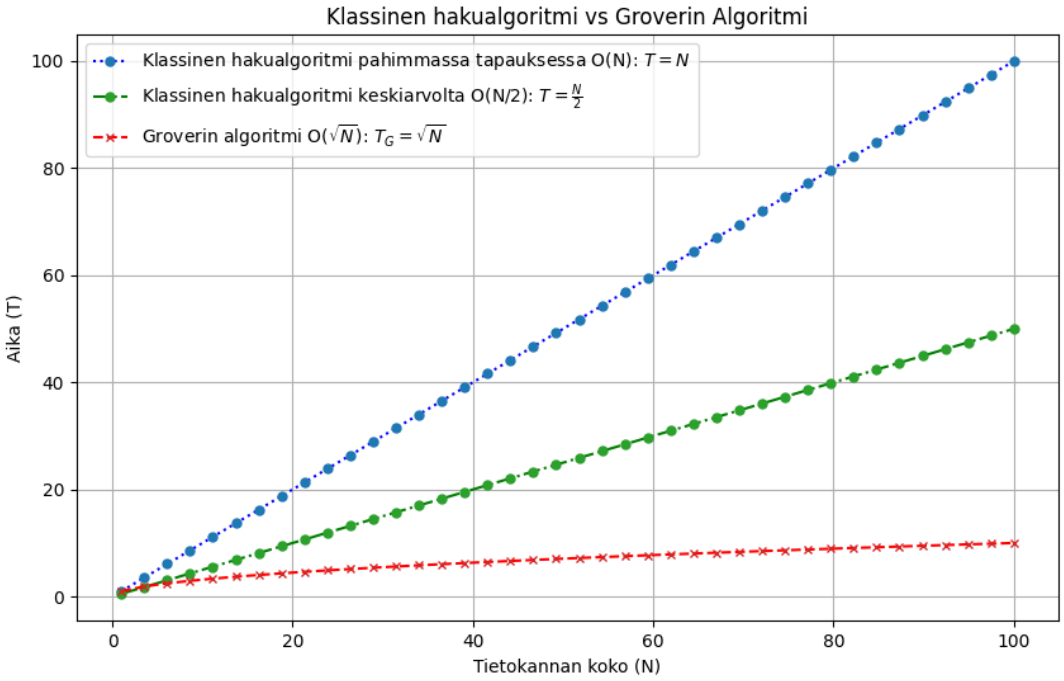
Tyypillisesti tiedonhaku ja hakualgoritmit toimivat klassisilla tietokoneilla seuraavasti: halutaan löytää N verran alkioita sisältävästä tietomäärästä yksi alkio, joka vastaa haluttuja kriteereitä. Käydään läpi jokainen alkio, sekä vertaillaan vastaako alkio haluttuja kriteereitä. Jos alkio vastaa haluttuja kriteereitä, lopetetaan haku. Jos ei, pidetään kirjaa jo läpikäydyistä alkioista ettei sitä tutkita uudestaan. Tätä jatketaan niin kauan kunnes haluttu alkio on löydetty. Tyypillisesti tämä tarkoittaa sitä, että joudutaan vertailemaan keskiarvolta $\frac{N}{2}$ alkiota, jotta oikea alkio löydetään. [8]

4.2. Groverin algoritmi hakuongelmien kvanttitehokkaaseen ratkaisuun

Groverin algoritmi on suunniteltu ratkaisemaan hakuongelmia kvanttitehokkaasti. Sen suunnittelu perustuu kvanttitietokoneiden uniikkien ominaisuuksien, kuten **superposition** ja **interferenssin**, hyödyntämiseen. Kvanttitietokoneilta luvattu laskentatehokkuuden kasvu tekee Groverin algoritmista yhden nopeimmista kvanttialgoritmeista. [3]

Klassiset hakuongelmat, kuten tietokannasta alkion etsiminen ovat aina olleet resursseja ja aikaa vaativia operaatioita, etenkin kun tietomäärät ovat suuria. Kvanttitietokoneiden toimintaperiaatteet mahdollistavat sen, että Groverin algoritmi nopeuttaa hakuongelmia kvadraattisesti. Nykyhetkellä hakuongelmissa tiedon etsimiseen kulunut aika kasvaa lineaarisesti tietokannan koon kasvaessa, kun taas Groverin algoritmilla aika kasvaa neliöjuurtuen. Aikakompleksisuus klassisilla hakualgoritmeilla on yleisimmiten lineaarinen $O(N)$, kun taas Groverin algoritmilla

$O(\sqrt{N})$ [8, 24]. Mitä isommiksi tietomäärät kasvavat, sitä massiivisempi Groverin algoritmin tuoma nopeutus on. Kuvassa 2 havainnollistetaan Groverin Algoritmin tuomaa teoreettista nopeutusta tietokannasta hakemiseen.



Kuva 2. Havannoiva graafi aikakompleksisuuksista: Klassinen hakualgoritmi vs Groverin Algoritmi

Esimerkki Groverin algoritmin nopeudesta hakuongelmissa

Kuvitellaan, että on tietokanta, joka sisältää N kappaletta alkioita täysin satunnaisessa järjestyksessä. Klassisella tietokoneella satunnaisen alkion hakeminen klassisella hakualgoritmilla vaatisi keskiarvolta vähintään $\frac{N}{2}$ vertailua ja pahimmassa tapauksessa N vertailua. Groverin algoritmi kvanttitietokoneella ajettuna tarjoaa kyseiseen hakuongelmaan kvadraattisen (eng. quadratic) nopeutuksen, jolloin se kykenee ratkaisemaan vastaavan hakuongelman tutkimalla vain $\sqrt{N}$ määrällä vertailuja. [8]

4.3. Groverin Algoritmi kryptografiassa

Groverin algoritmin tehokkuus hakuongelmien ratkaisuissa näkyy myös uhkana kryptografialle. Kun Shorin algoritmi uhkaa julkisen avaimen salausmenetelmiä, Groverin algoritmin uhat kohdistuvat lohkosalausmenetelmiin (eng. block cipher algorithms). [33]

Groverin algoritmi nopeuttaa väsytyshyökkäyksiä (eng. brute force attack) symmetrisen avaimen salausmenetelmiä kohtaan. Jotta väsytyshyökkäykset ovat mahdollisia Groverin algoritmia hyödyntäen, tulee ensin toteuttaa kohteen salausmenetelmä kvanttipiirinä, jotta sitä voidaan käyttää kvanttitietokoneessa. [34]

Todennäköinen kohde Groverin algoritmin hyödyntämiseen olisi AES. Vaikka AES-menetelmän 256-bittisen (AES-256) version on ehdotettu olevan kvanttiturvallinen, on mahdollista, että pienempää salaustasoa olevat AES-128 ja AES-192 olisivat murrettavissa väsytyshyökkäyksillä [35, 27]. Jotta näiden haavoittuvuuksia voidaan todeta, on jatkuvasti tehty tutkimusta symmetrisen avaimien salauksien toteuttamisesta kvanttipiirinä [34]. Groverin algoritmia voidaan symmetrisen avaimen menetelmien kohdistuvien väsytyshyökkäyksien lisäksi käyttää etsimään törmäyksiä hajautusfunktioissa (eng. hash function). Tämän myötä on todettu, että moni nykyisistä hajautusfunktioista tulevat olemaan käyttökelvottomia kvanttiaikakaudella [25].

Groverin algoritmin kryptografisiin uhkiin pohjautuvat tutkimukset edesauttavat myös kvanttiturvallisen kryptografian kehittymistä. Uhkien lisäksi, Groverin algoritmin pohjalta on tehty tutkimuksia sen soveltamiseksi kryptografisiin menetelmiin. Esimerkiksi QSS protokollan turvallisuuden vahvistamisesta Groverin algoritmin avulla on tehty muutamia tutkimuksia. [36, 37]

Vaikutus salaustasoon

Salaustaso salausmenetelmissä tarkoittaa salausmenetelmän kykyä vastustaa hyökkäyksiä. Se määritellään yleensä bittimäärällä b, joka vaadittaisiin tehokkaaseen hyökkäykseen salausjärjestelmää vastaan. Mitä suurempi bittimäärä b, sitä tehokkaampi ja vaikeammin murrettavissa salausmenetelmä on. Bittimäärä b viittaa salausmenetelmässä käytettyyn salausavaimen pituuteen. [38]

Väsytyshyökkäyksien uhka perustuu siihen, että kvanttitietokoneella ajettu Groverin algoritmi puolittaa symmetrisen avaimen salausmenetelmien salaustason b-bittisestä $\frac{b}{2}$-bittiseen. [33]. Väsytyshyökkäyksen aikakompleksisuus, joka normaalisti kasvaa eksponentiaalisesti avaimen pituuden kanssa on $O(2^b)$, vähenee huomattavasti Groverin algoritmin avulla $O(2^{\frac{b}{2}})$ [39]. Tämä nopeutus mahdollistaa merkittävän tehon lisäyksen väsytyshyökkäyksille symmetrisen avaimen salausmenetelmiä vastaan ja vähentää tarvittavien vertailujen määrää merkittävästi.

4.4. Algoritmin toimintaperiaate

Groverin algoritmin kvanttipiirin luominen koostuu neljästä vaiheesta:

1. Hadamardin portit

Systeemillä on $S = 2^n$ tilaa, jossa n on systeemin kubittien määrä. Jokainen systeemin tila S alustetaan Hadamardin porteilla samaan amplitudiin, jolloin kaikki kubitit ovat **superpositiossa**. [8] Tiloja voidaan kuvailla seuraavasti:

$$[\frac{1}{\sqrt{S}}, \frac{1}{\sqrt{S}}, \ldots, \frac{1}{\sqrt{S}}]$$

2. Oraakkeli

Groverin algoritmissa oraakkelia käytetään etsimään ensin vaatimuksen $f(x) = 1$ täyttävä tila x, jonka jälkeen vaatimuksen täyttäneen tilan amplitudi käännetään negatiiviseksi kääntämällä sen tilan vaihdetta π verran. Tätä vaihetta varten on määritetty binäärinen funktio C [8]

$$C : \{0, 1\}$$

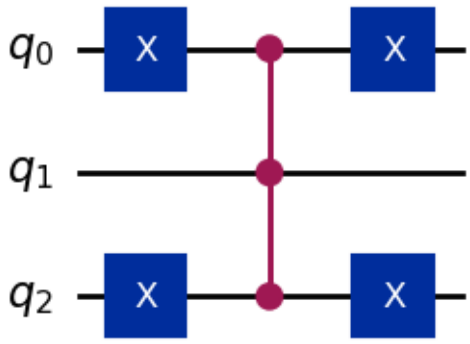
Tila S merkitään ja sen amplitudi käännetään negatiiviseksi, kun se täyttää vaatimuksen

$$C(S) = 1$$

Jos tila S ei täytä vaatimusta, jolloin

$$C(S) = 0$$

jätetään se tila koskemattomaksi.[8]



Kuva 3. Tilan *010* valinta Groverin algoritmin kvanttipiirissä

Esimerkissä (Kuva 3) tilan 010 ensimmäisen ja kolmannen kubitin arvo vaihdetaan käyttäen X-portteja, jolloin saadaan tila 111. X-portit asetetaan niiden tilan 010 kubittien CZ-ohjausportin molemmin puolin joiden arvo on 0. Tila 111 on ainoa, joka toteuttaa ohjausportin ja antaa amplitudille negatiivisen arvon. Muilla tiloilla arvo on eri kuin 111, minkä takia ohjausportti ei toteudu. Tämän jälkeen tilojen arvot palautetaan takaisin alkuperäisiin käyttäen samoja X-portteja.

3. Diffuusio operaatio

Diffuusio operaatio Groverin algoritmissa voidaan määritellä seuraavasti [40]

$$2 \cdot |u\rangle\langle u| - I$$

missä $|u\rangle$ tarkoittaa yhtenäistä kvanttitilaa [40] ja I yksikkömatriisia. Yhtenäinen kvanttitila $|u\rangle$ on yhdistelmä ei-tyhjästä osajoukosta perustiloja, jossa kaikki nollasta poikkeavat amplitudit ovat samoja [41]. I on neliömatriisi, jonka päädiagonaalilla on ykkösiä ja muualla nollia. Tämä operaatio kääntää amplitudeja keskimääräisen amplitudin mukaan, mikä saa Groverin operaation aikana käännetyn merkityn tilan korostumaan samalla kun muiden tilojen amplitudi laskee. Kun merkitty tila oli ennen diffuusio operaatiota keskiarvon alapuolella, on se amplitudien käännön jälkeen yhtä paljon keskiarvon yläpuolella. [8]

4. Mittaus

Mittarit asetetaan kvanttipiirin loppuun jokaiselle kubitille, joiden avulla jokaisen kubitin arvo voidaan lukea yksitellen. Mittaus on myös osa kvanttipiiriä, joten Groverin algoritmin suoritettua etsinnän kubittien arvot luetaan ja mittaustulokset tulkitaan.

5. TOTEUTUS GROVERIN ALGORITMISTA

5.1. Toteutuksen vaatimukset

Toteutuksen luominen vaatii ympäristön, jolla voidaan suunnitella ja toteuttaa Groverin algoritmiin tarvittavat kvanttipiirit, sekä simuloida ja visualisoida sen toimintaa.

Qiskit

Qiskit on avoimen lähdekoodin ohjelmistokehityspaketti, joka mahdollistaa erilaisten kvanttipiirien ja -algoritmien luomisen, sekä niiden ajamisen niin simulaattoreilla kuin oikeilla kvanttitietokoneilla Python-ohjelmointikieltä hyödyntäen. Sen on luonut IBM Research vuonna 2017. [42]

Qiskitin yhteistyö IBM Researchin kanssa mahdollistaa kvanttitietokoneiden etäkäytön hyödyntäen rajapintoja, joiden avulla käyttäjä voi ottaa yhteyttä kvanttitietokoneisiin. Qiskit tarjoaa käyttäjilleen laajasti valmiita ominaisuuksia kvanttipiirien kehittämiseen [42]. Työkalu mahdollistaa kvanttipiirien ja -algoritmien suoraviivaisen kehityksen, tarjoamalla käyttäjilleen valmiita funktioita muun muassa Groverin algoritmiinkin tarvittavien Hadamard-porttien käyttämiseen [43], sekä kvanttipiirien visualisointiin.

Tulemme käyttämään toteutuksessamme Qiskittiä luomaan Groverin Algoritmiin tarvittavan kvanttipiirin, sekä ajamaan toteutuksemme Qiskitin rajapintoja hyödyntäen IBM Quantum Platformin tarjoamilla simulaattoreilla, sekä kvanttitietokoneilla.

Jupyter Notebook

Jupyter on avoimen lähdekoodin työkalu interaktiiviseen dokumentointiin. Jupyterin interaktiivisia dokumentteja voidaan käyttää laajasti kirjoittamaan ja suorittamaan koodia, sekä luomaan visualisointeja dokumentin sisäisesti erilaisten graafien ja mallien avulla. Jupyter tukee useita ohjelmointikieliä. [44]

Jupyter Notebook sopii loistavasti toteutuksemme tarkoitukseen, sillä tulemme tarvitsemaan selkeän dokumentaation tiivistämään ja visualisoimaan toteutustamme. Jupyterin interaktiivisen dokumentoinnin avulla voimme osoittaa selkeästi tuloksiamme, sillä se mahdollistaa Qiskitin funktioita käyttäen muun muassa kvanttipiirien visualisoinnin samaan dokumenttiin.

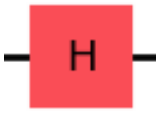
5.2. Toteutuksen havainnollistaminen

Qiskitin ja Jupyter interaktiivisen dokumentoinnin avulla tulemme visualisoimaan vaiheittain toteutustamme. Qiskit mahdollistaa jokaisen kvanttipiirin osan visualisoinnin.

Käytetyt kvanttiportit

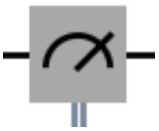
Groverin algoritmi koostuu Hadamardin porteista (Kuva 4), X-porteista (Kuva 5), CNOT-porteista (Kuva 7), sekä kubittien merkkaamiseen käytetyistä CZ-porteista

(Kuva 8). Jotta kubittien lopulliset arvot saadaan luettua, piirin viimeinen vaihe on mittaus (Kuva 6), jossa saadaan arvo 0 tai 1 jokaiselle kubitille.



Kuva 4. Hadamardin portti

Kuva 5. X-portti



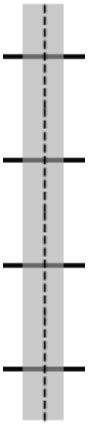
Kuva 6. Mittari

Kuva 7. CNOT-portti

Kuva 8. CZ-portti

Koko Groverin Algoritmin kvanttipiiriä havainnollistaessa Qiskit kuvaa Groverin Operaattoria, joka sisältää Hadamardin portteja, X-portteja sekä CNOT-portteja, helppolukuisempana kokonaisuutena (Kuva 9). Algoritmin osuuden ja mittarit Qiskit erottaa hyödyntämällä katkoviivoin merkittyä barrieria (Kuva 10), mikä helpottaa kvanttipiirin ymmärtämistä.

Kuva 9. Groverin Operaattori



Kuva 10. Barrier

5.3. Toteutuksen alustaminen

Groverin algoritmin luominen aloitetaan mahdollistamalla Qiskitin funktioiden käyttö tuomalla tarvittavat ominaisuudet Qiskitin kirjastosta (Kuva 11).

```python
#tuo tarvittavat moduulit
import math

# Importtaa qiskit ja tarvittavat moduulit
from qiskit import QuantumCircuit
from qiskit.circuit.library import GroverOperator, MCMT, ZGate
from qiskit.visualization import plot_distribution

# Tarvitaan myös moduuli ajoa varten
from qiskit_ibm_runtime import QiskitRuntimeService, Sampler, Batch
```

Kuva 11. Qiskit-kirjaston tuonti

Tuotujen kirjastojen avulla voimme asettaa käyttäjätunnuksen, sekä määritellä kohde laitteen, jolla Groverin algoritmi suoritetaan. Käyttäjä määritellään `token`-parametrilla, joka yhdistää toteutukseemme IBM Quantum-käyttäjätunnuksen, mikä mahdollistaa kvanttipiirien suorittamisen kohde laitteella. `Token`-parametri on käyttäjä kohtainen, jonka takia se on salattu. Tässä esimerkissä asetamme kohde laitteeksi IBM Quantumin tarjoaman simulaattorin. (Kuva 12)

```python
# Tallenna account (api key)
QiskitRuntimeService.save_account(channel="ibm_quantum", token="token", set_as_default=True, overwrite=True)

# lataa credentiaalit
service = QiskitRuntimeService()
backend = service.backend("ibmq_qasm_simulator")
backend.name
```

Kuva 12. Käyttäjän asettaminen ja kohde laitteen valinta

Seuraavaksi määrittelemme Groverin algoritmin koon, mitä tiloja etsimme, sekä kuinka monta kertaa suoritamme etsinnän yhden istunnon aikana (Kuva 13). Etsittävä tila on binääriluku, joka perustuu suoritettavan algoritmin kokoon, eli kubittien määrään.

```python
#kubittien määrä
n = 8

#ajokertojen määrä
ajokerrat = 100

#etsittävä tila(t)
tilat = ["11011100"]
```

Kuva 13. Tarvittavat määrittelyt

Luodaan ensimmäisenä Groverin algoritmin kvanttipiiriä varten Groverin oraakkeli. Oraakkelin luonnissa piiriin merkataan etsittävät tilat CZ-porteilla, sekä asetetaan vaatimusten mukaan X-portteja. Käytimme Groverin oraakkelin luomiseen IBM Quantum Learningin esittelemää funktiota (Kuva 14) [45].

```python
#Groverin oraakkeli funktio: (https://learning.quantum.ibm.com/tutorial/grovers-algorithm)
def grover_oracle(marked_states):
    """Build a Grover oracle for multiple marked states

    Here we assume all input marked states have the same number of bits

    Parameters:
        marked_states (str or list): Marked states of oracle

    Returns:
        QuantumCircuit: Quantum circuit representing Grover oracle
    """
    if not isinstance(marked_states, list):
        marked_states = [marked_states]
    # Compute the number of qubits in circuit
    num_qubits = n

    qc = QuantumCircuit(num_qubits)
    # Mark each target state in the input list
    for target in marked_states:
        # Flip target bit-string to match Qiskit bit-ordering
        rev_target = target[::-1]
        # Find the indices of all the '0' elements in bit-string
        zero_inds = [ind for ind in range(num_qubits) if rev_target.startswith("0", ind)]
        # Add a multi-controlled Z-gate with pre- and post-applied X-gates (open-controls)
        # where the target bit-string has a '0' entry
        qc.x(zero_inds)
        qc.compose(MCMT(ZGate(), num_qubits - 1, 1), inplace=True)
        qc.x(zero_inds)
    return qc
```

Kuva 14. IBM Quantum Learning Groverin oraakkeli-funktio

Oraakkelin valmistelemisen jälkeen luomme kvanttipiirin, joka koostuu Groverin oraakkelista, sekä piiristä, joka korostaa etsittävien tilojen amplitudeja. Tätä kutsutaan Groverin operaattoriksi. Qiskitissä saamme luotua Groverin operaattorin käyttämällä `GroverOperator()` funktiota, joka ottaa parametriksi äskettäin `grover_oracle()` funktiolla luodun oraakkelin.

Jotta Groverin algoritmin kvanttipiiri voidaan kokonaisuudessaan toteuttaa, tulee meidän ensin laskea kuinka monta kertaa Groverin operaattori tulee toistaa Groverin algoritmin kvanttipiirissä. IBM Quantum Learning on esitellyt seuraavan tavan jolla lasketaan tarvittava Groverin operaattorin toistojen määrä (Kuva 15). [45]

```python
iteraatiot = math.floor(
    math.pi/(4 * math.asin(math.sqrt(len(tilat) / 2**grover_operator.num_qubits)))
    )
```

Kuva 15. IBM Quantum Learning Groverin operaattorin toistojen määrä

Groverin algoritmin kvanttipiirin alustamisen ensimmäisenä vaiheena on asettaa jokainen systeemin tila samaan amplitudiin käyttämällä Hadamardin portteja.

Hadamardin porttien jälkeen piiriin tulee sijoittaa Groverin operaattoreita lasketun toistojen määrän verran. Lopuksi jokaiselle tilalle asetetaan mittarit, jotta jokaisen kubitin arvo voidaan lukea. (Kuva 16)

```
[ ]
#kvanttipiiri koko groverin algoritmille
piiri = QuantumCircuit(n)

[ ]
piiri.h(range(n))

[ ]
piiri.compose(grover_operator.power(iteraatiot), inplace=True)

[ ]
piiri.measure_all()
```

Kuva 16. Groverin algoritmin kvanttipiirin luominen

Ennen Groverin algoritmin kvanttipiirin suorittamista asettamallamme kohde laitteella, tulee piiri optimoida suorittamista varten. Asetetaan optimoinnin taso nollaan (`optimization_level = 0`), jolloin käytetty funktio ohittaa Qisktissa oletuksena olevan optimoinnin ja ajaa kvanttipiirin asetetuilla kubiteilla (Kuva 17). Optimointitason pidämme nollassa, sillä Qiskitin eri optimointitasot muuttavat piiriä siten, että se tuottaisi mahdollisimman vähän virhettä. Eri optimointitasoilla kvanttipiirin toiminnallisuus ei muutu, mutta suoritettavat kubitit voivat muuttua.

```
from qiskit.transpiler.preset_passmanagers import generate_preset_pass_manager

target = backend.target
pm = generate_preset_pass_manager(target=target, optimization_level=0)

circuit_ibm = pm.run(piiri)
```

Kuva 17. Groverin algoritmin kvanttipiirin optimointi

Lopuksi Groverin algoritmin kvanttipiiri suoritetaan kohde laitteella (Kuva 18). Yhden istunnon aikana etsintä suoritetaan `shots`-muuttujan asettamien ajokertojen verran. Ajokertojen määrä on määritelty aiemmin.

```
with Batch(backend=backend) as batch:
    sampler = Sampler()
    result = sampler.run(circuit_ibm, skip_transpilation=True, shots=ajokerrat)
```

Kuva 18. Groverin algoritmin suorittaminen simulaattorilla

Kun Groverin algoritmi suoritetaan oikealla kvanttitietokoneella hyödyntäen IBM Quantum Platformin tarjoamia mahdollisuuksia etäyhteyteen, alustetaan kohde ja käyttäjä seuraavasti (Kuva 19).

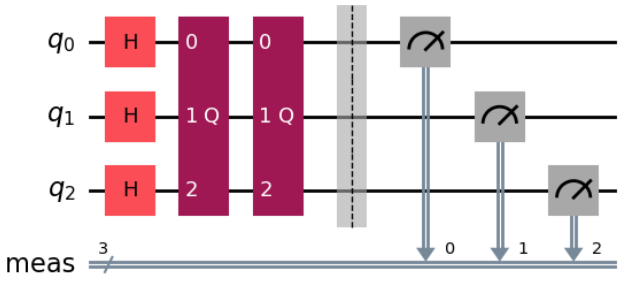
```
# Tallenna account (api key)
QiskitRuntimeService.save_account(channel="ibm_quantum", token="token", set_as_default=True, overwrite=True)

# lataa credentiaalit
provider = IBMProvider()
#print(provider.backends())
backend = provider.get_backend("ibm_kyoto")
#backend.name
```

Kuva 19. Käyttäjän asettaminen ja kohde laitteen määrittely oikealle kvanttitietokoneelle

Muutoin Groverin algoritmin kvanttipiirin luominen suoritetaan samoin, kuin on aiemmin esitelty. Kun Groverin algoritmin kvanttipiiri optimoidaan ajettavaksi oikealla kvanttitietokoneella, suoritetaan se käyttämällä `transpile()` funktiota, joka uudelleenkirjoittaa luomamme kvanttipiirin sopimaan kohde kvanttitietokoneen topologiaan [46]. Pidetään edelleen optimointi nollatasossa (`optimization_level = 0`), jotta voimme ohittaa piirin ylimääräisen optimoinnin ja ajaa kvanttipiirin asettamillamme kubiteilla. Viimeisenä suoritetaan `shots`-muuttujan määrittävät etsinnät hyödyntäen optimoitua Groverin algoritmin kvanttipiiriä kutsumalla `job()` funktiota, jonka parametreiksi annetaan aiemmin `transpile()` funktiolla optimoitu piiri, sekä ajokerrat.

Kuvassa 20 esitetään lopputulos visualisoidusta 3 kubitin Groverin algoritmin kvanttipiiristä.



Kuva 20. 3 kubitin Groverin algoritmin kvanttipiiri

6. TULOKSET

6.1. Hakunopeudet tavallisella tietokoneella

Vertailun vuoksi toteutimme myös hakualgoritmin tavallisella tietokoneella käyttäen Pythonia ja listaa binääriluvuista selvittääkseen, kuinka kauan tavallisella koneella kestää löytää oikea alkio lajittelemattomasta listasta.

Toteutus

Toteutimme hakuaikojen selvittämisen käyttäen hyödyksi Pythonin **time**-kirjaston `perf_counter()`-metodia ja satunnaisen numeron valitsemiseen **random**-kirjaston `randint()`-metodia. Koodi toimii lisäämällä kohtaan `BINARY_NUMBER` halutun binääriluvun maksimiarvoon yhden. Koodi luo luvun pohjalta listan, jossa on kaikki binäärimäärään kuuluvat binääriluvut. Funktio aloittaa ajastimen ja alkaa suorittaa vertailua yksitellen listan jäsenille, kunnes oikea arvo löydetään. Tämän jälkeen ajastin pysäytetään ja suoritusaika kirjataan ylös. Funktio tallentaa valitun määrän suoritusaikoja, minkä jälkeen lasketaan suoritusaikojen keskiarvo.

```python
import time
import random

# Alustetaan tyhjä lista ja halutun (binääri määrän maksimi arvo + 1)
lst = []
BINARY_NUMBER = 16777216

# Luodaan lista välille 0 - (BINARY_NUMBER-1)
for number in range(0,BINARY_NUMBER):
    lst.append(bin(number))

def binary_search(iterations):
    '''
    Valitaan satunnainen binääri luku ja etsitään sille vastaava listasta
    Toteutetaan haku n kertaa ja tulostetaan saatujen suoritusaikojen keskiarvo.
    '''
    runtimes = []
    for _ in range(iterations):
        result = bin(random.randint(0,(BINARY_NUMBER-1)))
        start = time.perf_counter()
        for binary in lst:
            if binary == result:
                stop = time.perf_counter()
                running_time = stop - start
                runtimes.append(running_time)
                break
    print(f"Average of runtimes: {sum(runtimes) / iterations}")

# Funktion suoritus n kertaa.
binary_search(1000)
```

Kuva 21. Koodi toteutukselle

Käytimme kuvassa 21 olevaa koodia etsimään satunnaisia binäärilukuja 1000 kertaa eri binääri määrillä. Otettuamme keskiarvon näistä 1000 suorituskerran

hakuajoista, toteutimme jokaisen testin kolme kertaa jokaiselle binäärimäärälle ja saimme seuraavanlaiset tulokset (Taulukko 1). Tuloksista nähdään, että jopa 24 bitin haku onnistuu tavalliselta tietokoneelta keskimäärin alle sekunnissa.

Taulukko 1. Hakuaikoja eri määrillä bittejä tavallisella tietokoneella

Hakuaikojen keskiarvo 1000 kertaa ajettuna

Toteutus	6 bittiä	8 bittiä	12 bittiä	16 bittiä	20 bittiä	24 bittiä
1. Toteutus	658.09 ns	2.2031 µs	32.85 µs	520.12 µs	9.1425 ms	157.26 ms
2. Toteutus	663.59 ns	2.198 µs	32.447 µs	539.12 µs	8.8488 ms	142.65 ms
3. Toteutus	659.6 ns	2.2106 µs	33.742 µs	532.81 µs	8.900 ms	129.04 ms
Lukuarvo	64	256	4096	65536	1048576	16777216
Nimellisarvo					1 milj.	16 milj.

Kokeilimme myös 28 bitin hakuaikoja satunnaisilla luvuilla, jolloin saimme seuraavat tulokset (Taulukko 2). Tuloksista nähdään, että hakuajat vaihtelevat paljon riippuen siitä, mikä luku valitaan oikeaksi arvoksi. Esimerkiksi luvulla 909243 hakuaika on 0.2711 s, kun taas 266 miljoonan luvulla hakuaika on 64.839 s. Näin ollen hakuaikojen keskiarvo on todennäköisesti 0-65 sekunnin puolivälissä eli 32.5 s

Taulukko 2. Hakuaikoja 28 bitin luvuille

Hakuaikoja 28 bitin (268 milj.) luvuille

Binääriluku	Luku	Nimellisarvo	Hakuaika (sekuntteina)
0b111001001011111000010000100	119926916	119 milj.	25.561
0b110111000001000000000101100	230752300	230 milj.	53.602
0b101101100010000100011000110	190976390	190 milj.	42.73
0b110100010100011111011011011	219446491	219 milj.	52.14
0b111111100101000011011001100	266669772	266 milj.	64.839
0b11011101111110111011	909243	909 tuhatta	0.2711

Näitä tuloksia voidaan pitää suuntaa antavina tavallisten tietokoneiden hakuaikoihin. Hakuaikoja voitaisiin parantaa esimerkiksi vähentämällä koneen taustaprosesseja tai käyttämällä nopeampia prosessoreja. Testin tarkoituksena oli saada suuntaa antavia tuloksia tavallisen tietokoneen hakunopeuksista. Testissä käytetty prosessori oli AMD Ryzen 7 3800X.

6.2. Tuloksia Groverin algoritmin suorittamisesta

Suoritimme vertailua Groverin algoritmin tuloksista erikokoisilla Groverin algoritmin toteutuksilla. Vertailun tavoitteena oli tulkita eroja etsittävän tilan todennäköisyyksissä eri kokoisten algoritmien ja eri suuruisten ajomäärien (`Shots`-muuttuja) välillä, sekä eri suorituskohteilla. Toteutuskokoina toimivat 2, 3 sekä 4 kubitin kokoiset Groverin algoritmit. Ajomäärien suuruuksina vertailussa toimi 1000 ja 10000. Rajoitimme ajomäärien suuruudet näihin lukemiin, sillä yleisen testailun ja toteutuksen luonnin aikana huomasimme isoimpia eroja syntyvän kun ajomäärät olivat näissä suuruuksissa. Myöskin yksi rajoittavista tekijöistä olivat jonotusajat, jotka saattoivat kasvaa hyvinkin suuriksi kun haluttiin ajaa toteutustamme oikealla kvanttitietokoneella. Suorituskohteina toimivat IBM Quantumin tarjoama kvanttitietokone `ibm_kyoto`,

sekä simulaattorina `ibm_qasm_simulator`. Taulukoissa 3 ja 4 on esitetty löytämiämme tuloksia, jossa tulokset ovat kirjattu yhden suorituskerran jälkeen.

Taulukko 3. Tuloksia kvanttitietokoneelta

2 kubittia	Kvanttitietokone (ibm_kyoto, 127 kubittia)			
	Todennäköisyys			
Ajomäärä	Etsitty tila ("10")	Muut tilat yhteensä	Kokonaissuoritusaika (s)	Suoritusaika yhdelle etsinnälle (s)
1000	0,549	0,451	1,46	0,00146
10000	0,559	0,441	3,96	0,000396

3 kubittia	Kvanttitietokone (ibm_kyoto, 127 kubittia)			
	Todennäköisyys			
Ajomäärä	Etsitty tila ("110")	Muut tilat yhteensä	Kokonaissuoritusaika (s)	Suoritusaika yhdelle etsinnälle (s)
1000	0,143	0,857	1,49	0,00149
10000	0,168	0,832	4,37	0,000437

4 kubittia	Kvanttitietokone (ibm_kyoto, 127 kubittia)			
	Todennäköisyys			
Ajomäärä	Etsitty tila ("1101")	Muut tilat yhteensä	Kokonaissuoritusaika (s)	Suoritusaika yhdelle etsinnälle (s)
1000	0,060	0,940	1,77	0,00177
10000	0,064	0,936	5,83	0,000583

Taulukko 4. Tuloksia simulaattorilta

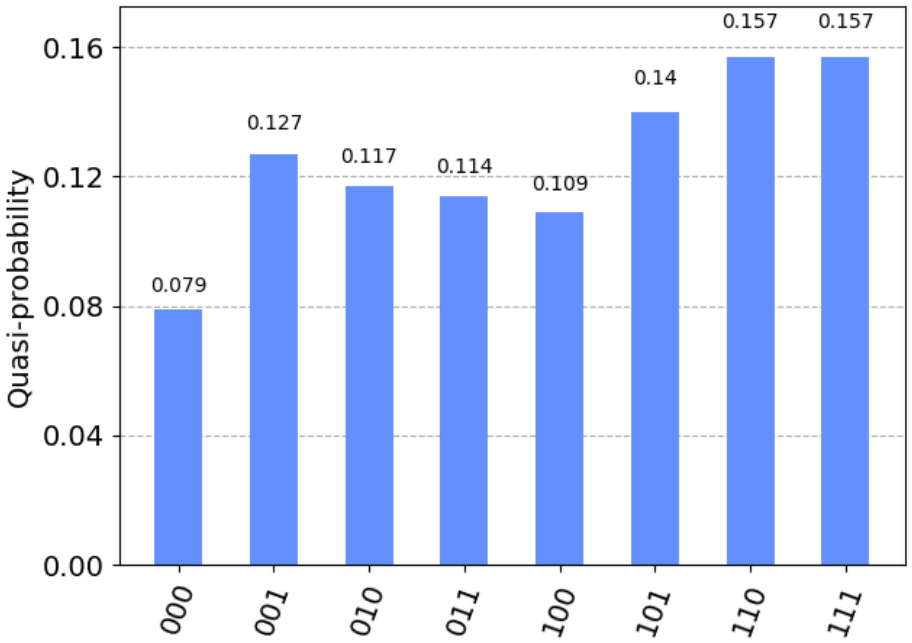
2 kubittia	Simulaattori (ibm_qasm_simulator, 32 kubittia)			
	Todennäköisyys			
Ajomäärä	Etsitty tila ("10")	Muut tilat yhteensä	Kokonaissuoritusaika (s)	Suoritusaika yhdelle etsinnälle (s)
1000	1,000	0,000	0	0
10000	1,000	0,000	0	0

3 kubittia	Simulaattori (ibm_qasm_simulator, 32 kubittia)			
	Todennäköisyys			
Ajomäärä	Etsitty tila ("110")	Muut tilat yhteensä	Kokonaissuoritusaika (s)	Suoritusaika yhdelle etsinnälle (s)
1000	0,950	0,050	0	0
10000	0,946	0,054	0	0

4 kubittia	Simulaattori (ibm_qasm_simulator, 32 kubittia)			
	Todennäköisyys			
Ajomäärä	Etsitty tila ("1101")	Muut tilat yhteensä	Kokonaissuoritusaika (s)	Suoritusaika yhdelle etsinnälle (s)
1000	0,963	0,037	0	0
10000	0,963	0,037	0	0

Yleisimpänä tuloksena olemme huomanneet, niin tätä vertailua tehdessä, kuin yleisen testailun aikana, että kun kasvatetaan ajomäärää, etsityn tilan todennäköisyys kasvaa. Selkeämmin todennäköisyyden kasvun huomaa kvanttitietokoneen tuloksista, kun taas simulaattorilla erot ovat hyvin marginaalisia.

Kun kasvatetaan algoritmin kokoa, kvanttitietokoneella saatavat tulokset alkavat sisältää hyvin paljon virhettä. Kun etsittävän tilan todennäköisyys kahden kubitin Groverin algoritmilla on hieman yli 50 prosenttia, putoaa se 4 kubitin Groverin algoritmilla noin 6 prosenttiin. Ero kubittimäärän kaksinkertaistuessa on miltei kymmenkertainen. Virheen määrä näyttää siis kasvavan isoin harppauksin, mitä suurempiin toteutuskokoihin mennään. Tuloksissa virheen voi tulkita siitä, minkä verran muiden tilojen kuin etsittävän tilan todennäköisyydet ovat. Kuvasta 22 huomataan myös se, että etsittävän tilan todennäköisyys ei myöskään erotu selkeästi muiden tilojen joukosta.



Kuva 22. Todennäköisyysjakauma 3 kubitin Groverin algoritmin tuloksesta, jossa etsitty tila on "001"

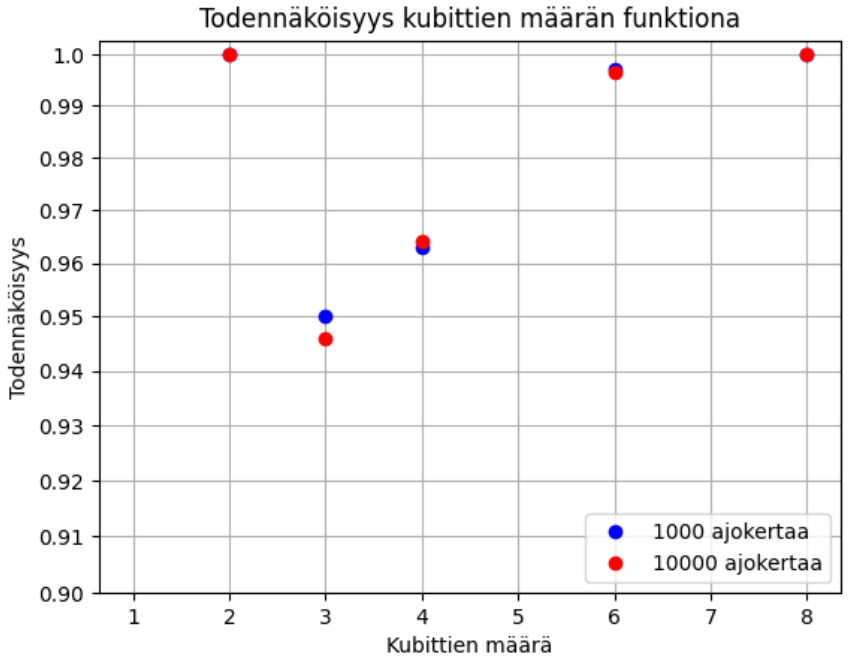
Vastaavasti simulaattorilla ei nähdä näin radikaalia eroa. Vaikka tuloksista voidaan huomata virheen kasvavan kubittimäärän kasvaessa, on etsittävän tilan todennäköisyys silti merkittävän suuri. Simulaattorin on tarkoitus simuloida ideaalia kvanttitietokonetta, joka tuottaisi mahdollisimman vähän virhettä. Myös sen myötä simulaattorin tuloksissa ei nähdä niin merkittäviä muutoksia etsityn tilan todennäköisyyksissä uoritettuna eri ajomäärillä, sekä eri kokoisilla Groverin algoritmeilla.

Toisena testinä (Taulukko 5), suoritimme ajoja `ibm_qasm_simulator` simulaattorilla 4, 6 ja 8 kubitin kokoisilla Groverin algoritmeilla 1000 ja 10000 kokoisilla ajomäärillä.

Taulukko 5. Tuloksia simulaattorilta

	Simulaattori (ibm_qasm_simulator, 32 kubittia)					
	4 kubittia		6 kubittia		8 kubittia	
	Todennäköisyydet		Todennäköisyydet		Todennäköisyydet	
Ajokerrat	Etsitty tila ("1101")	Muut tilat yhteensä	Etsitty tila ("110101")	Muut tilat yhteensä	Etsitty tila ("11010101")	Muut tilat yhteensä
1000	0,963	0,037	0,997	0,003	1,00	0,00
10000	0,964	0,036	0,9964	0,0036	1,00	0,00

Aikaisempaan simulaattorilla suoritettuun 2, 3 ja 4 kubitin (Taulukko 4) vertailun tuloksiin verrattaessa voidaan huomata, että vaihtelevuus etsityn tilan todennäköisyyksissä on kiinnostavaa. Näistä tuloksista erityisen huomattavaa oli se, kuinka etsityn tilan todennäköisyys näyttää putoavan merkittävästi kun siirrytään 2 kubitin toteutuksesta 3 kubitin toteutukseen ja siitä jälleen kasvavan, mitä suurempaan toteutukseen mennään. Odotettua oli, että simulaattorin tulokset seuraisivat samaa kaavaa, kuin kvanttitietokoneen tulokset: etsityn tilan todennäköisyys pienenee mitä suurempaan Groverin algoritmin kokoon siirrytään. Siksi nämä tulokset ovat mielenkiintoisia. Tuloksien vaihtelevuutta on havainnollistettu Kuvassa 23.



Kuva 23. Kuvaaja eroista etsityn tilan todennäköisyyksissä kubittien määrän funktiona simulaattorilla saaduista tuloksista

Muita tuloksia

Groverin algoritmista on tehty tutkimusta 2 kubitin kokoisesta aina 5 ja 6 kubitin kokoisiin toteutuksiin saakka [47]. Siispä kokeilimme testata toteutuksemme kykyä ja rajoja skaalaamalla toteutuksemme 8 kubitin kokoiseksi ja etsimällä kolmea eri tilaa samanaikaisesti ajomäärällä 1000, ja suorittamalla sen `ibm_osaka` kvanttitietokoneella (Taulukko 6). Kvanttitietokone tälle suoritukselle määräytyi sen mukaan, mikä oli vähiten kiireisin suoritusajankohdan aikaan. Vertailukohteena suoritimme myös 8 kubitin kokoisella Groverin algoritmilla kolmen eri tilan etsinnän `ibm_qasm_simulator` simulaattorilla kokeilemalla useita eri ajomääriä (Taulukko 7).

Taulukko 6. Tuloksia kvanttitietokoneelta

	Kvanttitietokone 1 (ibm_brisbane, 127 kubittia)					
	Todennäköisyys					
	Etsittävät tilat			Muut tilat		
Shots	"11010101"	"01000111"	"10010011"	yhteensä	Runtime (all) (s)	Runtime (one shot) (s)
1000	0,0110	0,0130	0,0040	0,9720	20,00	0,020004

`Ibm_osaka` kvanttitietokoneella ajetun 8 kubitin Groverin algoritmin tulokset olivat odotettuja. Skaalatessamme toteutustamme yhä isompiin kokoihin, voidaan huomata, että etsittyä tilaa ei haun aikana löydetä kovinkaan isolla todennäköisyydellä. Virhe alkaa kasvaa huomattavan suureksi. Myös Taulukossa 3 esiteltyjä tuloksia tarkastelemalla huomataan että etsityn tilan todennäköisyys on samassa suuruusluokassa, kuin 4 kubitin Groverin algoritmilla saaduissa tuloksissa.

Vertailtaessa 8 kubitin Groverin algoritmin kvanttitietokoneella saatua tulosta simulaattorilla saatuun tulokseen, voidaan huomata, että simulaattori tuottaa

Taulukko 7. Tuloksia simulaattorilta

	Simulaattori (ibm_qasm_simulator, 32 kubittia)			
	Todennäköisyys			
	Etsittävät tilat			Muut tilat
Shots	"11010101"	"01000111"	"10010011"	yhteensä
1	0,0000	1,0000	0,0000	0,0000
10	0,3000	0,4000	0,3000	0,0000
100	0,3200	0,3800	0,3000	0,0000
1000	0,3170	0,3230	0,3580	0,0020
10000	0,3364	0,3358	0,3247	0,0031
100000	0,3320	0,3340	0,3310	0,0030

todennäköisempiä tuloksia. Simulaattorin tuloksien mukaan etsittyjen tilojen todennäköisyydet jakautuvat lähes tasan, ja virhettä on alle prosentin verran. Tulokset ovat lähes päinvastaisia kvanttitietokoneen tuloksiin nähden, jossa etsittyjen tilojen todennäköisyydet ovat yhteenlaskettuna vain muutaman prosentin luokkaa.

8 kubitin Groverin algoritmia suorittaessa `ibm_qasm_simulator` simulaattorilla testasimme myös eri ajokertojen määrän vaikutusta (Taulukko 5) tuloksiin kun etsitään useaa tilaa. Yhtä tilaa etsiessä ajomäärien lisääminen ei vaikuttanut lopputulokseen juuri ollenkaan, vaan etsitty tila löytyi yli 99 prosentin varmuudella. Useampaa tilaa etsiessä huomasimme virheen määrän kasvavan, kun ajokerrat olivat yhtä suurta tai suurempaa kuin 1000. Tästä voitaisiin tulkita virheen määrän kasvan useampaa kuin yhtä tilaa etsiessä, mitä enemmän etsintojä suoritetaan.

Lineaarinen hakualgoritmi vs Groverin algoritmi

Suoritimme myös erikokoisten alkioiden etsimistä lineaarisella hakualgoritmilla (KPL 6.1), jonka tuloksia verratessamme Groverin algoritmilla saatuihin tuloksiin huomasimme toistaiseksi lineaarisen hakualgoritmin suoriutuvan etsinnästä huomattavasti nopeammin, vaikka sen aikakompleksisuusluokitus on korkeampi kuin Groverin algoritmilla. Yhdestä etsinnästä saatuun suoritusaikaan Groverin algoritmilla tulee kuitenkin kiinnittää huomiota seuraaviin seikkoihin. Esimerkiksi Taulukossa 3 kolumnilla `Kokonaissuoritusaika` on merkitty aika, jonka IBM Quantum platform tarjosi meille kokonaisuudessaan pyydetyn työn suorittamiseen. Viereiselle kolumnilta `Suoritusaika yhdelle etsinnälle` olemme arvioineet kauanko yhden etsinnän suorittamiseen olisi kulunut aikaa kolumnin `shots` määrittävien ajokertojen mukaan. Yhteen etsintään kulunut aika, sekä myös koko työhön kulunut aika ei välttämättä anna meille vertailukelpoista dataa, sillä emme voi tietää sisältyykö työn kokonaisaikaan muutakin, kuin pelkkään `shots`-muuttujan määrittämiin etsintöihin kulunut aika. Tuloksia varovaisesti tarkastellen voimme kuitenkin todeta, että toistaiseksi Groverin algoritmilla ei näillä kubitti määrillä ja käytössämme olevilla kvanttitietokoneilla kyetä suoriutumaan yhden alkion etsinnästä nopeammin, kuin lineaarisella hakualgoritmilla. Kun esimerkiksi 4 kubitin Groverin algoritmi suoriutuu yhden 4-bittisen arvon etsinnästä 2^4 tilan joukosta muutamissa millisekunneissa,

vastaa siihen etsintään arvioimamme aika lineaarisella hakualgoritmilla 20-bittisen arvon etsimistä 2^{20} tilan joukosta. Kvanttitietokoneen tuloksista nähdään myös, että todennäköisyys saada oikea tila on hyvin pieni, kun kubitteja on enemmän kuin kaksi. Tämä kielii siitä, että kvanttitietokoneet vaativat vielä paljon kehittymistä, jotta Groverin algoritmilla voidaan saavuttaa toivottu kvadraattinen nopeutus hakuongelmien ratkaisuun, sekä saada luotettavia hakutuloksia.

7. POHDINTA

Kandidaatintyön prosessin aikana opimme valtavasti kvanttitietokoneista, kvanttialgoritmeista ja niiden kvanttitietokoneilta vaatimista ominaisuuksista, sekä syvensimme osaamistamme kryptografiasta. Erityisesti opimme Groverin algoritmista ja sen ominaisuuksista sekä siihen liittyvistä kryptografisista ongelmista. Prosessin aikana opimme myös tärkeitä taitoja kirjoittamisesta, lähteiden etsimisestä, sekä kokonaisuuden työstämisestä, mitkä tulevat varmasti helpottamaan diplomityön suorittamista. Ajankäyttö työn aikana jakautui tasaisesti jokaisen meidän välillemme, sillä koitimme jakaa niin taustatutkimuksen kirjoittamisen, kuin toteutuksen tekemisen ja testaamisen tasaisesti. Vaikka aihe oli meille entuudestaan liki tuntematon, halusimme haastaa itseämme valitsemalla sen. Olimme kaikki hieman kuulleet kvanttitietokoneista ja niiden potentiaalista, jonka myötä mielenkiintomme tätä aihetta kohtaan heräsi. Myöskään opinnoissamme ei ole käsitelty kvanttitietokoneita tai kvanttialgoritmeja, emmekä ole perehtyneet niiden ohjelmoimiseen. Taustatutkimuksen kirjoittaminen edistyi meillä sujuvasti. Uuden aiheen tuoma motivaatio ja kiinnostus auttoivat perehtymään asioihin syvällisemin ja uskomme että ulosantimme on sen mukaista. Toteutustamme varten käytetty Qiskit oli varsin helppo käyttöönotettava ja sen kanssa toimimista edisti opiskeluidenkin aikana tutuksi tullut Python, johon Qiskit-viitekehys pohjautuu. IBM Quantumin tarjoama loistava dokumentaatio, sekä esimerkit Qiskitin käyttöön auttoivat meitä pitkälle Groverin algoritmin kvanttipiirin luomisessa.

Ongelmia tuotti osittain pitkä jonotusaika toteutustamme ajaessa kvanttitietokoneilla. Välillä jonot venyivät useamman tunnin mittaisiksi, toisinaan ne saattoivat olla muutamia minuutteja. IBM Quantum platformin kautta tarkasteltaessa tuloksia emme saaneet yhtä tarkkoja tuloksia, kuin mitä toteutuksemme olisi meille tulostanut Jupyter Notebook tiedostoomme. Eroavaisuuksia oli myös siinä, minkälaista tietoa IBM Quantum platform tarjosi tarkasteltaessa tuloksia kvanttitietokoneen ja simulaattorilla suoritettujen ajojen välillä. Simulaattorilla suoritetuista ajoista saatiin tarkempaa tietoa tilojen todennäköisyyksistä, kun taas kvanttitietokoneen tuloksilla sitä ei saatu. Tämä yhdessä jonojen pituuden kanssa vaikutti siihen, kuinka laajasti pystyimme vertailemaan tuloksia simulaattorin ja kvanttitietokoneen suoritusten välillä.

Tulokset suorittamistamme ajoista oli osin odotettuja, mutta myös odottamattomia. Simulaattorin on tarkoitus simuloida ideaalia kvanttitietokonetta, siispä varsinkin Groverin algoritmin suorittaminen pienillä kubitti määrillä tuotti odotettuja tuloksia. Vaikka simulaattorilta voitiin odottaa lähes 100 prosentin varmuutta etsittävän tilan löytämiseen, olivat tuloksien vaihtelut osin odottamattomia, kuten taulukossa 7 on havainnollistettu. Olimme asettaneet odotukseksi kvanttitietokoneella suoritettavista ajoista pienemmillä kubitti määrillä lähes samankaltaisia tuloksia, kuin mitä simulaattorilla saadaan. Kuitenkin odotuksemme osoittautuivat vääriksi, sillä virheen määrä jo kahden kubitin kokoisella toteutuksella oli merkittävä. Kun simulaattori tarjosi 100 prosentin todennäköisyyden etsittävän tilan löytymiselle, oli se kvanttitietokoneella suoritettuna vain 50 prosenttia. Kun kubittien määrää kasvatettiin suuremmaksi, tippui etsityn tilan todennäköisyys entisestään. Groverin algoritmin suoritusajat oikealla kvanttikoneella olivat odotettua pienemmät. Kuvassa 3 nähdään, että Groverin yksittäinen suoritus kesti noin 4 ms.

Kryptografian näkökulmasta on oleellista tarkastella Groverin algoritmin mahdollisia vaikutuksia salausmenetelmiin. Kappaleessa 4.3 käsittelimme, kuinka Groverin algoritmi vaikuttaa salaustasoon, sekä kuinka väsytyshyökkäykset nopeutuvat Groverin algoritmilla. Lähin uhka salausmenetelmien murtumiselle kohdistuu vanhentuneisiin salausmenetelmiin, kuten MD5 ja SHA-1 hajautusfunktioihin. Tällä hetkellä näihin hajautusfunktioihin voidaan helposti löytää törmäyksiä käyttäen hyödyksi sanakirjahyökkäyksiä tai sateenkaaritaulukoita. Väsytyshyökkäyksiä hyödyntämällä voidaan löytää lyhyitä salasanoja, edellyttäen kaikkien vaihtoehtojen testaamista. Groverin algoritmin suorittaminen kvanttitietokoneella tarjoaa teoreettisen mahdollisuuden nopeuttaa hajautusfunktioiden murtumista väsytyshyökkäyksillä.

Tällä hetkellä kvanttitietokoneella ajetulla Groverin algoritmilla voidaan löytää yhtä isoja salaisuuksia kuin toteutetun algoritmin kubitti koko n on. Groverin algoritmin kubittimäärilla $n = 2$ ja $n = 3$, jotka vastaavat 2^2 ja 2^3 kokoisia tiloja eivät vielä riitä väsytyshyökkäyksiin esimerkiksi salasanojen selvittämiseen, sillä niissä yksittäisen merkin pituus on 8 bittiä. Tuloksistamme nähdään, että Groverin algoritmi ei usein löydä oikeaa arvoa edes 2 tai 3 kubitin arvoilla. Siispä ei ole vielä järkevää toteuttaa Groverin algoritmilla väsytyshyökkäystä, sillä sille teoreettisesti luvatun nopeutuksen saavuttaminen vaatisi huomattavasti suuremman toteutuksen koon. Nykyhetkellä muun muassa sanakirjahyökkäykset ja sateenkaaritaulukot ovat huomattavasti nopeampia murtamaan hajautusfunktioita kuin väsytyshyökkäykset niin klassisilla tietokoneilla kuin myös Groverin algoritmin avulla toteutettuna. Groverin algoritmilla tällä hetkellä ei kyetä toteuttamaan toimivaa väsytyshyökkäystä edes pienellä kubittimäärällä, sillä oikeaa tulosta ei löydetä isolla todennäköisyydellä. Myöskin suoritusajat jo pienillä kubittimäärillä ovat suuria. Kvanttitietokoneiden kehitystä tulee kuitenkin seurata tarkasti, sillä myös kvanttialgoritmien tehokkuus kehittyy kvanttitietokoneiden laskentatehokkuuden kasvaessa. Vielä teoreettisella pohjalla oleva kvanttivälimuisti (eng. Quantum Random Access Memory) on myös yksi kvanttiteknologian ja kvanttitietokoneiden kehityksen kannalta tärkeimpiä seurattavia kehityskohteita. Kvanttivälimuisti mahdollistaisi 2^n-bittien tallentamisen n-määrällä kubitteja, kun taas klassisten tietokoneiden välimuisti mahdollistaa vain n-bittien tallentamisen n-määrällä bittejä.

Työmme rajatun laajuuden ja aikamääreen myötä pintaan nousi useita tutkimuskysymyksiä, joihin perehtyminen vaatisi lisää aikaa. Esimerkiksi kuinka kauan menee, että Groverin algoritmi on erinomainen vaihtoehto hakuongelmien ratkaisuun? Minkä kokoinen toteutus Groverin algoritmista tulee toteuttaa, että se toteuttaa sille luvatun hakuongelmien kvadraattisen nopeutuksen? Kryptografisesta näkökulmasta tärkeimmiksi kysymyksiksi nousivat esimerkiksi onko Groverin algoritmin käyttämisestä väsytyshyökkäykseen oikeasti hyötyä? Voiko kvanttivälimuistin mahdollinen kehittyminen nopeuttaa Groverin algoritmin hakunopeutta? Näihin kysymyksiin perehtyminen vaatisi lisää aikaa, sekä kehittymistä kvanttiteknologialta.

8. YHTEENVETO

Kvanttitietokoneet jatkavat kehittymistä edelleen ja sen mukana nousevien tietoturvauhkien määrä jatkaa kasvamistaan laskentatehokkuuden kehittyessä. Sitä myötä NIST on tehnyt kvanttiturvalliseen kryptografiaan laajaa tutkimusta torjuakseen näitä uhkia ja kehittänyt kvanttikestäviä salausmenetelmiä. Tuloksiemme valossa kvanttiturvallisen kryptografian kehittämiseen on herätty ajoissa, sillä kvanttitietokoneiden kryptografialle luomien uhkien käytännöllistämistä varjostaa kvanttitietokoneiden tällä hetkellä rajattu skaalautuvuus ja laskentatehokkuus. Täten kvanttiturvallisen kryptografian kehittymisellä on paljon pelivaraa reagoida äkillisiin kasvuihin kvanttitietokoneiden laskentatehokkuudessa, kun uusien kvanttiturvallisten standardien kehittämistä on työstetty jo pitkään.

Kvanttialgoritmien, kuten Groverin algoritmin, teoreettisen tehokkuuden valjastamista varten kvanttitietokoneiden laskentatehokkuuden ja skaalan tulee kasvaa merkittävästi. Myös kvanttialgoritmien kvanttipiirien implementointia suuremmille kubitti määrille tulee tutkia lisää, jotta ne voivat hyödyntää kvanttitietokoneen tarjoamaa laskentatehoa. Vaikka Groverin algoritmi on yksi tunnetuimmista kvanttialgoritmeista, sen tehokas implementointi on vielä varhaisessa vaiheessa. Tuloksiemme pohjalta Groverin algoritmin tehokaaseen käyttöönottoon ja toteutuskoon kasvuun menee vielä useita vuosia, jotta sillä voidaan tehdä tutkimusta sen mahdollisuuksista muun muassa väsytyshyökkäyksissä. Työmme myötä heräsi useita tutkimuskysymyksiä, joihin uskomme löytyvän vastauksia tutkimuksien edetessä maailmalla.

9. VIITTEET

[1] Benioff P. (1980) The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines. Journal of Statistical Physics , ss. 563–591.

[2] Deutsch D. (1985) Quantum theory, the church–turing principle and the universal quantum computer. Proceedings of the Royal Society of London A , ss. 97–117.

[3] Deutsch D. & Jozsa R. (1992) Rapid solution of problems by quantum computation. Proceedings of the Royal Society of London A 439, ss. 553–558.

[4] Isaac L. Chuang Y.Y. (1995) A simple quantum computer. APS journals .

[5] Nielsen M.A. & Chuang I.L. (2010) Quantum Computation and Quantum Information: 10th Anniversary Edition. Cambridge University Press.

[6] Castelvecchi D. (2023) Ibm releases first-ever 1,000-qubit quantum chip. Nature .

[7] Shor P. (1994) Algorithms for quantum computation: discrete logarithms and factoring. Teoksessa: Proceedings 35th Annual Symposium on Foundations of Computer Science, ss. 124–134.

[8] Grover L.K. (1996) A fast quantum mechanical algorithm for database search. Teoksessa: Symposium on the Theory of Computing.

[9] Galbraith S.D. (2012) Mathematics of Public Key Cryptography. Cambridge University Press, 405-416, 485, 504 s.

[10] Albuainain A., Alansari J., Alrashidi S., Alqahtani W., Alshaya J. & Nagy N. (2022) Experimental implementation of shor's quantum algorithm to break rsa. Teoksessa: 2022 14th International Conference on Computational Intelligence and Communication Networks (CICN), ss. 748–752.

[11] Kirsch Z. (2015) Quantum computing: The risk to existing encryption methods.

[12] Chen L., Jordan S., Liu Y.K., Moody D., Peralta R., Perlner R. & Smith-Tone D. (2016), Report on post-quantum cryptography.

[13] Chen Y. (2024) Quantum algorithms for lattice problems .

[14] López-Saldívar J., Castaños O., Nahmad-Achar E., López-Peña R., Man'ko M. & Man'ko V. (2018) Geometry and entanglement of two-qubit states in the quantum probabilistic representation. Entropy 20, s. 630.

[15] National Academies of Sciences E. & Medicine (2019) Quantum Computing: Progress and Prospects. The National Academies Press, Washington, DC.

[16] Qiskit oracles. URL: `https://docs.quantum.ibm.com/api/qiskit/0.19/qiskit.aqua.components.oracles`, accessed 19.04.2024.

[17] Microsoft learn quantum oracles. URL: https://learn.microsoft.com/en-us/azure/quantum/concepts-oracles, accessed 26.04.2024.

[18] Introduction to quantum computing. URL: https://www.quantum-inspire.com/kbase/what-is-a-quantum-algorithm, accessed 2.2.2024.

[19] Roy S.K. (2023) Machine learning showdown: Quantum vs. classical approaches .

[20] Problems that only quantum computers can solve. URL: https://www.qmunity.tech/post/problems-that-only-quantum-computers-can-solve, accessed 10.04.2024.

[21] Bernstein E. & Vazirani U. (1997) Quantum complexity theory. SIAM Journal on Computing 26, ss. 1411–1473.

[22] Montanaro A. (2016) Quantum algorithms: an overview .

[23] Karthik S., Saipriya S., Bikash K.B. & Prasanta K.P. (2018) Efficient quantum algorithm for solving travelling salesman problem: An ibm quantum experience .

[24] Bavdekar R., Chopde E.J., Bhatia A., Tiwari K., Daniel S.J. & Atul (2022), Post quantum cryptography: Techniques, challenges, standardization, and directions for future research.

[25] Mavroeidis V., Vishi K., D. M. & Jøsang A. (2018) The impact of quantum computing on present cryptography. International Journal of Advanced Computer Science and Applications 9.

[26] Shor P.W. (1997) Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Journal on Computing 26, s. 1484–1509.

[27] Bernstein D.J. & Lange T. Post-quantum cryptography. Nature 549, s. 188–194.

[28] Computer Security Division I.T.L. (2017), Post-quantum cryptography: Csrc. URL: https://csrc.nist.gov/projects/post-quantum-cryptography.

[29] Alagic G., Apon D., Cooper D., Dang Q., Dang T., Kelsey J., Lichtinger J., Miller C., Moody D., Peralta R. & et al. (2022), Status report on the third round of the nist post-quantum cryptography standardization process.

[30] Barker W., Polk W. & Souppaya M. (2021), Getting ready for post-quantum cryptography: Exploring challenges associated with adopting and using post-quantum cryptographic algorithms.

[31] Sultana N., Chandra S., Paira S. & Alam S. (2017) A brief study and analysis of different searching algorithms.

[32] Avi S., Michael S. & Ullman J. (1991) Database systems : achievements and opportunities. Communications of the ACM .

[33] Jang K., Song G., Kwon H., Uhm S., Kim H., Lee W.K. & Seo H. (2021) Grover on pipo 10.

[34] Jang K.B., Song G.J., Kim H.J. & Seo H.J. (2021) Grover on simplified aes. Teoksessa: 2021 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), ss. 1–4.

[35] Grassl M., Langenberg B., Roetteler M. & Steinwandt R. (2015), Applying grover's algorithm to aes: quantum resource estimates.

[36] Hsu L.Y. (2003) Quantum secret-sharing protocol based on grover's algorithm. Phys. Rev. A 68, s. 022306.

[37] Yu Z. (2022) The improved quantum secret sharing protocol based on grover algorithm. Journal of Physics: Conference Series 2209, s. 012031.

[38] Lenstra A.K. (2010) Key lengths contribution to the handbook of information security.

[39] Jang K., Kim H., Eum S. & Seo H. (2020), Grover on gift.

[40] Ibm quantum learn grover's algorithm. URL: https://learning.quantum.ibm.com/course/fundamentals-of-quantum-algorithms/grovers-algorithm, accessed 22.02.2024.

[41] Mozafari F., Soeken M. & Di Micheli G. (2020), Automatic uniform quantum state preparation using decision diagrams.

[42] Qiskit documentation. URL: https://docs.quantum.ibm.com/, accessed 16.02.2024.

[43] Qiskit hadamard-gates. URL: https://docs.quantum.ibm.com/api/qiskit/0.37/qiskit.circuit.library.HGate, accessed 16.02.2024.

[44] Jupyter documentation. URL: https://docs.jupyter.org/en/latest/, accessed 16.02.2024.

[45] Ibm quantum learn grover's algorithm tutorial. URL: https://learning.quantum.ibm.com/tutorial/grovers-algorithm, accessed 22.03.2024.

[46] Ibm quantum learn transpiler. URL: https://docs.quantum.ibm.com/api/qiskit/transpiler, accessed 25.03.2024.

[47] Vemula D.R., Konar D., Satheesan S., Kalidasu S.M. & Cangi A. (2022), A scalable 5,6-qubit grover's quantum search algorithm.