Soluautomaattien käyttö kryptografiassa: uusi Langtonin muurahaiseen perustuva salausalgoritmi

Pro gradu
Turun yliopisto
Tulevaisuuden teknologioiden laitos
Tietojenkäsittelytiede
2020
Vilho Kivihalme

Tarkastajat:
Antti Hakkala
Jarkko Kari

Turun yliopiston laatujärjestelmän mukaisesti tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck-järjestelmällä.

TURUN YLIOPISTO
Tulevaisuuden teknologioiden laitos

VILHO KIVIHALME: Soluautomaattien käyttö kryptografiassa: uusi Langtonin muurahaiseen perustuva salausalgoritmi

Pro gradu, 93 s., 1 liites.
Tietojenkäsittelytiede
Huhtikuu 2020

Salausta on käytetty tuhansia vuosia piilottamaan arkaluontoisia yksityisiä viestejä vihollisilta, tai miltä tahansa muilta tahoilta, joille viestin viestien sisältöä ei haluta paljastaa. Salauksen käyttötarkoitus on edelleen sama, mutta käyttökohteet ovat yleistyneet arkipäiväisemmiksi. Koko internetin olemassaolon ja toimivuuden ehtona on, että salausmenetelmät ovat tehokkaita ja turvallisia. Internetin välityksellä tehdään paljon yksityisiä asioita, kuten maksetaan laskuja ja kirjaudutaan erilaisiin palveluihin. Internetissä kulkevaa liikennettä voidaan kuitenkin seurata melko helposti, joten on ensiarvoisen tärkeää, että kaikki viestit lähetetään salattuna. Näin esimerkiksi pankkitunnukset ja yksityisviestit kahden henkilön välillä pysyvät ainoastaan kommunikoivien osapuolten tiedossa.

Soluautomaatti on tietojenkäsittelytieteessä tutkittu dynaaminen malli, jossa eräänlaisella ruudukolla simuloidut solut ovat vuorovaikutuksessa keskenään. Vähimmillään soluautomaattina voi ajatella ruutupaperia, jonka solut väritetään mustaksi tai valkoiseksi riippuen siitä, ovatko ne eläviä vai kuolleita. Simulaation aikana solut syntyvät ja kuolevat riippuen niiden lähellä olevien solujen tilasta. Yksinkertaisillakin säännöillä solut muodostavat monimutkaisia rakenteita, kuten toistuvia syklejä, rajatonta ja kiihtyvää laajenemista, sekä itsensä kopioimista.

Langtonin muurahainen on eräänlainen soluautomaatti ja Turingin kone. Muurahainen on ruudukolla liikkuva osoitin, joka liikkuu ruudusta toiseen niiden värien perusteella, vaihtaen ruutujen värejä tarkkaan määriteltyjen sääntöjen perusteella. Tarkoista säännöistä huolimatta muurahaisen liikkeitä ei voi ennustaa etukäteen. Liikkuminen näyttää usein satunnaiselta ja epäsäännölliseltä, mutta tietyillä säännöillä syntyy myös säännöllisiä rakenteita ja usein myös epäsäännöllisesti liikkuvat muurahaiset päätyvät lopulta päättymättömään säännölliseen jaksoon.

Tutkielmassa perehdytään soluautomaattien ja salausalgoritmien perusteisiin esimerkkien avulla, jonka jälkeen tehdään kirjallisuuskatsaus soluautomaattien avulla toteutettuihin salausjärjestelmiin. Tämän jälkeen esitellään uudenlainen Langtonin muurahaisen avulla toteutettu salausjärjestelmä ja pohditaan sen käyttökelpoisuutta ja turvallisuutta.

Asiasanat: kryptografia, salaus, soluautomaatti, Turingin kone, Langtonin muurahainen

UNIVERSITY OF TURKU
Department of Future Technologies

VILHO KIVIHALME: Cryptography with cellular automata: A novel encryption algorithm based on Langton's ant

Master's Thesis, 93 p., 1 app. p.
Computer science
April 2020

Encryption has been used for thousands of years in order to hide private messages from enemies and any other entities that should not be able to read them. The usage of encryption has remained the same, but the applications have become more mundane. Powerful and secure encryption methods are vital for transferring messages on modern day internet. Money transfer and accessing data in various services are two very common online activities that should remain private. Because internet traffic can be monitored easily, encryption of all communication should be considered almost mandatory.

A cellular automaton is a discrete dynamical system where artificial cells arranged into a regular grid interact with each other. Simplest example would be a grid paper where black and white squares represent dead and alive cells. During a simulation certain cells die, and new cells are born depending on the state of adjacent cells. Even the simplest rules can cause complex behavior like repeating cycles, infinite and accelerating growth and self-replication.

Langton's ant is a cellular automaton and a Turing machine. The ant is a pointer that moves on a regular grid depending on the color of the cell it occupies, changing the colors as it moves. Even though the color changing rules of are known, it is impossible to predict how the ant moves. While the moving is usually irregular and almost random-like, with certain rules the ant can also produce regular patterns. Often the irregularly moving ants also transition to regular infinite period of moves.

This thesis covers the basics of cryptography and cellular automata using various examples. This is followed by a survey of existing encryption algorithms that are based on cellular automata. Finally, a novel encryption algorithm that is based on Langton's ant is presented and analyzed in terms of usability and security.

Keywords: cryptography, encryption, cellular automata, Turing machine, Langton's ant

Sisältö

1 Johdanto **1**

2 Kryptografia **4**

2.1 Salausjärjestelmien vaatimukset 5

2.2 Salausalgoritmien ominaisuuksia 7

2.2.1 Avaimeen perustuva salaus 7

2.2.2 Muihin salaisuuksiin perustuva salaus 9

2.2.3 Merkkisalakirjoitus ja lohkosalakirjoitus 10

2.3 Kryptoanalyysi 11

2.3.1 Menetelmien luokittelu 11

2.3.2 Katsaus menetelmiin 13

2.4 Salausjärjestelmiä 17

2.4.1 Klassisia salausjärjestelmiä 17

2.4.2 Enigma 20

2.4.3 RSA 21

2.4.4 AES 22

2.5 Täydellinen salaus 23

3 Soluautomaatti **24**

3.1 Määritelmä 25

3.2 Kääntyvyys 26

3.3 Turing-täydellisyys 27
3.4 Ratkeamattomuus 28
3.5 Erilaisia soluautomaatteja 28
3.5.1 Alkeissoluautomaatti 28
3.5.2 Game of Life 29

4 Langtonin muurahainen 34
4.1 Merkinnät 37
4.2 N-tilainen muunnelma 38
4.3 N-ulotteinen muunnelma 38
4.4 Yleinen muoto 39
4.5 Käyttäytyminen äärellisellä ruudukolla 41
4.5.1 Tilojen määrä äärellisellä ruudukolla 44
4.5.2 Tilojen saavutettavuus 44
4.5.3 Ruutujen tilojen riippuvuus toisistaan 45
4.5.4 Pariteetti 45
4.5.5 Alkutilaan palaaminen 46
4.5.6 Liikehdinnän satunnaisuus 48
4.6 Muurahaisen esitys soluautomaattina 50

5 Soluautomaatit salausalgoritmina 52
5.1 Soluautomaatilta vaaditut ominaisuudet 52
5.2 Aiempia toteutuksia 53
5.2.1 Yksiulotteinen soluautomaatti satunnaislukugeneraattorina 53
5.2.2 Kuvansalaus Solautomaatin avulla 54
5.2.3 Kuvansalaus Langtonin muurahaisen avulla 56

6 Langtonin muurahainen salausalgoritmina 57
6.1 Algoritmin toimintaperiaate 58

6.1.1 Muurahaisen loppusijainnin ongelma 61
6.1.2 Muurahaisen alkusijainnin ongelma 62
6.1.3 Sähköisen koodikirjan ongelma 63
6.2 Paranneltu algoritmi 65
6.2.1 Siirtomäärän arviointi 68
6.2.2 Tilamuutoskaavion laajennus 76
6.3 Kryptoanalyysi 78
6.3.1 Purkaminen väärällä avaimella 78
6.3.2 Purkaminen väkisin 80
6.3.3 Avaimen päättely salatekstistä 81
6.3.4 Herkkyys selkotekstin muutokselle 81
6.4 Ongelmat kuvansalauksessa 82
6.5 Salausavaimen generointi 82

7 Johtopäätökset **84**
7.1 Avoimet kysymykset 86
7.1.1 Systeemin tilojen saavutettavuus 86
7.1.2 Satunnaiskävelyn saavuttamat tilat 86

Lähdeluettelo **88**

Liitteet

A Sääntö 1 **A-1**

Luku 1

Johdanto

Salakirjoitus on tiedon muuttamista sellaiseen muotoon, että sen pystyy lukemaan ainoastaan, jos tuntee käytetyn salakirjoitusmenetelmän. Yksinkertaisia salakirjoitusmenetelmiä on ollut olemassa jo Julius Caesarin aikaan.[1] Ajan mittaan salakirjoituksen ympärille on kehittynyt matematiikan osa-alue, salakirjoitustiede eli *kryptografia*. Kryptografia tutkii salausalgoritmien tehokkuutta ja turvallisuutta pyrkien kehittämään uusia ja parempia salausmenetelmiä. Eräs kryptografiaan läheisesti liittyvistä tieteenaloista on kryptoanalyysi, joka pyrkii löytämään salausalgoritmien ongelmia tai heikkoja kohtia, joita voisi mahdollisesti käyttää salattujen viestien purkamiseen. Vaikka kryptoanalyysiä käytetään salaisuuksien selvittämiseen ja haitan aiheuttamiseen, kryptoanalyysi on tärkeää myös salauksen kehittäjille. Yleisessä käytössä olevien salausmenetelmien on oltava ehdottoman turvallisia, joten on erittäin tärkeää pyrkiä murtamaan tunnettuja salausmenetelmiä ja havaita niiden mahdolliset puutteet ennen kuin niistä aiheutuu haittaa.

Modernissa tietoyhteiskunnassa tiedon salaamista käytetään päivittäin, eivätkä monet arkipäiväiset asiat olisi mahdollisia ilman kehittynyttä salausteknologiaa. Esimerkiksi verkkopankkiin tai muuhun verkkopalveluun kirjautuessa tietoliikenne salataan, jotta kirjautumistiedot ja muu viestinvaihto asiakkaan ja palvelun välillä ei paljastuisi ulkopuolisille. Toinen salausalgoritmien käyttökohde on digitaalisen kommunikaation osapuolten todentaminen. Ilman todentamista tietoverkkoon kytketty laite voi käytännössä esiintyä

kenenä tahansa. Tämä ongelma on ratkaistu digitaalisella allekirjoituksella, jolla viestin vastaanottoja voi varmistua, että viestin lähettäjä on se, joka väittääkin olevansa.[2] Digitaaliset allekirjoitukset perustuvat kryptografisiin menetelmiin, erityisesti tiivistefunktioihin. Tiivistefunktion avulla viesti voidaan tiivistää vakiomittaiseksi merkkijonoksi, eräänlaiseksi sormenjäljeksi.

Salakirjoitusta voidaan käyttää myös haitan aiheuttamiseen. Eräs viime vuosien merkittävimmistä tapauksista on vuonna 2017 levinnyt Wannacry -kiristysohjelma.[3] Kuten muutkin samankaltaiset ohjelmat, Wannacry perustuu tehokkaaseen salausalgoritmiin. Kun ohjelma suoritetaan tietokoneella, se käy läpi mahdollisimman paljon käyttäjän tiedostoja, salaten ne erittäin tehokkaalla salausalgoritmilla. Tämän jälkeen ohjelma pyytää käyttäjää lähettämään maksun esimerkiksi tiettyyn BitCoin -lompakon osoitteeseen. Kiristyksessä esitetään, että maksun jälkeen käyttäjälle luovutetaan salauksen purkamiseen tarvittavat tiedot. Koska salauksen purkaminen on käytännössä mahdotonta ilman avainta, käyttäjän täytyy maksaa summa ja toivoa parasta.

Soluautomaateista

Soluautomaatti on tietojenkäsittelytieteessä ja matematiikassa tutkittu diskreetti dynaaminen järjestelmä, jossa keinotekoista elämää jäljittelevät solut ovat vuorovaikutuksessa toistensa kanssa. Soluautomaateista kenties tunnetuin esimerkki lienee Life -peli (engl. *Game of Life*) jossa solut "syntyvät" ja "kuolevat", kun niillä on oikea määrä muita soluja ympärillään.

Monet soluautomaatit perustuvat yksinkertaisiin sääntöihin. Sääntöjä noudattamalla soluautomaatit tuottavat monimutkaisia malleja, jotka ovat usein näennäisesti satunnaisia tai kaaottisia, mutta joskus solut muodostavat ajan mittaan myös hyvin järjestelmällisiä ja toistuvia kuvioita. Soluautomaateilla on useita paljon tutkittuja ominaisuuksia, joista esimerkiksi kaaottisuus, deterministisyys ja tiettyjen ominaisuuksien matemaattinen ratkeamattomuus antavat aihetta pohtia, voisiko soluautomaatteja käyttää tiedon salaamiseen.

Tässä tutkielmassa esitellään mielikuvituksellinen soluautomaatteihin perustuva salausjärjestelmä. Toisessa luvussa tutustutaan kryptografian perusteisiin ja hyvän salausjärjestelmän vaatimuksiin. Näiden avulla analysoidaan ja vertaillaan historiallisia salausjärjestelmiä ja modernin tietoyhteiskunnan vaatimukset täyttäviä salausjärjestelmiä. Kolmannessa luvussa perehdytään soluautomaattien peruskäsitteistöön ja niiden erilaisiin ominaisuuksiin esimerkkien avulla. Neljännessä luvussa tutustutaan Langtonin muurahaiseen ja tutkitaan sen ominaisuuksia ja käyttäytymistä erilaisissa tilanteissa. Viidennessä luvussa selvitetään, minkälaisia ominaisuuksia salaukseen käytettävällä soluautomaatilla tulee olla ja listataan aiempia toteutuksia alan kirjallisuuden perusteella. Kuudennessa luvussa esitetään uusi salausjärjestelmä, jossa salaukseen käytetään Langtonin muurahaista ja arvioidaan sen suorituskykyä ja turvallisuutta.

Luku 2

Kryptografia

Kryptografia tai kryptologia, eli salakirjoitustiede tutkii ja kehittää salakirjoitusta. Salauksen tarkoituksena on muuntaa alkuperäinen tieto, eli *selkoteksti* (engl. *plaintext*) sellaiseen muotoon, ettei sitä pysty lukemaan tietämättä käytettyä salausmenetelmää. Ideaalitilanteessa salausmenetelmästä voidaan sopia jopa kolmannen osapuolen läsnä ollessa. Käytännössä kaksi samassa huoneessa olevaa henkilöä voivat sopia käytettävästä menetelmästä ja käydä salattua keskustelua ilman, että muut huoneessa olijat pystyvät selvittämään keskustelun sisältöä, vaikka he kuulevatkin sekä salausjärjestelmästä sopimisen, että kaiken salatun keskustelun. [4] Tällaisessa tilanteessa muut huoneessa henkilöt voivat yrittää selvittää viestinvaihdon sisältöä. Tällöin heitä kutsutaan *hyökkääjiksi*.

Nimestään huolimatta selkoteksti ei ole välttämättä tekstiä. Salausmenetelmät perustuvat matematiikkaan ja näin ollen tekstiäkin käsitellään yleensä numeerisina arvoina. Taulukossa 2.1 on esitetty eräs tapa esittää kirjaimet numeroina. Matematiikkaan perustuvat menetelmät mahdollistavat sen, että algoritmien syötteenä voidaan käyttää mitä tahansa numeroina esitettävissä olevaa dataa. Salausalgoritmilla salattua dataa kutsutaan salatekstiksi (engl. *ciphertext*) ja tietyn salauksen tekemiseen ja purkamiseen tarvittavat algoritmit muodostavat *salausjärjestelmän* (engl. *cryptosystem*).

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Taulukko 2.1: Englanninkielisen aakkoston kirjaimia vastaavat numeeriset arvot.

2.1 Salausjärjestelmien vaatimukset

Auguste Kerckhoffs määritteli hyvän salausjärjestelmän ominaisuudet vuonna 1883. Kaikkia hänen esittämiään ominaisuuksia ei voida enää pitää ehdottoman tärkeinä, sillä ne perustuivat 1800-luvun tietämykseen ja laitteistoihin. Monet Kerckhoffsin listaamat vaatimukset perustuvat ihmisen ominaisuuksiin, kuten rajalliseen muistiin. Nykyisin käytössä olevat pienet ja kannettavat tietokoneet ovat mahdollistaneet ihmisen kykyihin verrattuna huomattavasti suuremman laskenta- ja muistikapasiteetin, jolloin osa Kerckhoffsin esittämistä ominaisuuksista voidaan jättää huomioimatta, tai ainakin tulkita eri tavalla. Auguste Kerckhoffsin esittämät vaatimukset ovat seuraavat [5]:

Matemaattinen murtumattomuus. Salausjärjestelmän on oltava matemaattisesti murtumaton. Matemaattisesti murtamattomalla salauksella tarkoitetaan, että salatekstistä ei saa olla mahdollista laskea tai päätellä alkuperäistä selkotekstiä. Salakirjoituksen analysointia ja matemaattisia ominaisuuksia käsitellään aliluvussa 2.3.

Salausjärjestelmä itsessään ei saa vaatia salassapitoa. Jos salaus perustuu ainoastaan salaiseen menetelmään, sen paljastuessa kaikki aiemminkin salatut viestit ovat kaikkien purettavissa. Menetelmän tulee siis olla yleisesti tunnettu, mutta salauksen apuna käytetään *salausavainta*, joka toimii eräänlaisena salasanana. Salausavain säätää tai muokkaa salauksen toimintaa niin, että eri salausavainta käyttämällä samakin viesti muuttuu täysin erilaiseksi salatekstiksi. Tämä vaatimus tunnetaan nimellä Kerckhoffsin periaate. Salausavaimen käsitteeseen tutustutaan tarkemmin aliluvussa 2.2.1. Salausavaimen vaatimuksen muotoili uudelleen Claude Shannon vuonna 1949 seuraavalla tavalla[6]: "Järjestelmiä tulisi suunnitella sillä oletuksella, että

viholliset saavat niiden täsmällisen toimintaperiaatteen tietoonsa välittömästi.[1]

Salattujen viestin siirto tulee olla mahdollista lennätinjärjestelmän avulla. Käytännössä siis normaaleista aakkosista koostuva teksti pitäisi salauksen jälkeenkin olla sellainen, että se on mahdollista siirtää lennättimen avulla samalla tavalla kuin salaamattomat viestit. Lennätinjärjestelmiä ei juurikaan ole enää käytössä, mutta ehtoa voidaan soveltaa myös internetiin. Salattujen viestien tulee siis olla sellaisia, että ne voidaan siirtää modernin tietoverkon välityksellä.

Salausavaimen tulee olla muistettavissa ilman sen kirjoittamista muistiin. Avaimen paljastumisriski pienenee sen ollessa ainoastaan sitä käyttävien henkilöiden tiedossa, eikä kirjoitettuna paperilla. Modernien salausjärjestelmien avaimet ovat melko pitkiä, eikä ihminen voi muistaa niitä kovinkaan helposti. Nykyisten salausjärjestelmien käyttäjät ovat kuitenkin pääasiassa keskenään kommunikoivia tietokoneita ja pitkäkin avain voidaan säilöä tietokoneen muistiin erittäin helposti.

Salausjärjestelmää tulee pystyä käyttämään ja siirtämään yksin. Jos järjestelmä perustuu usean henkilön yhteistyöhön, tai se vaatii valtavia laitteita, se ei ole käyttökelpoinen hankalissa olosuhteissa, kuten sodan keskellä. Varsinainen salaus tehdään nykyään useimmiten tietokoneen avulla ja tietokone kykenee käyttämään sellaistakin salausta, joka vaatisi useampien henkilöiden yhteistyötä tai laskentakapasiteettia. Lisäksi nykyiset tietokoneet mahtuvat kämmenelle, joten sellaisen kuljettaminen mukanaan on varsin helppoa.

Ohjeiden ja avaimien pitää olla yksinkertaisia, jotta käyttäjän on helppo muistaa kaikki ulkoa. Tämäkin ehto on vanhentunut, sillä tietokone selviää monimutkaisista ja pitkistä operaatiosta todella nopeasti ja pystyy muistamaan todella pitkiä algoritmeja.

[1] alkup. "one ought to design systems under the assumption that the enemy will immediately gain full familiarity with them"

2.2 Salausalgoritmien ominaisuuksia

Salausalgoritmeilla on erilaisia piirteitä, joiden perusteella niitä voidaan luokitella. Kaksi merkittävintä luokittelukriteeriä ovat algoritmin tai sen osien salassapito ja se, kuinka suurissa osissa algoritmi salaa dataa. Tässä aliluvussa käsitellään edellä mainittuja piirteitä ja niihin liittyviä seikkoja, joista merkittävimpänä käsitellään salausavaimen perusperiaatetta.

2.2.1 Avaimeen perustuva salaus

Salakirjoituksessa on oleellista, että salausalgoritmin toiminta on julkisesti kaikkien tiedossa. Tällöin salausalgoritmin on jo lähtökohtaisesti oltava turvallinen, koska muuten kuka tahansa voisi tutkia algoritmia ja murtaa salatut viestit. Julkinen algoritmi vaatii toimiakseen eräänlaisen salasanan, eli *avaimen*. Avain säätelee salausalgoritmin toimintaa ja mahdollistaa sen, että samakin selkoteksti muuttuu täysin erilaiseksi, jos avainta muutetaan. Jos salausavain päätyy taholle, jolle se ei kuulu, salauksen käyttäjät voivat sopia uuden avaimen. Kun avainta muutetaan, aikaisemmasta avaimesta tai puretuista viesteistä ei ole hyökkääjälle kovinkaan suurta hyötyä, jos algoritmi on suunniteltu hyvin. Salausalgoritmi säilyy siis edelleen käyttökelpoisena, vaikka yksittäinen viesti tai avain paljastuikin.

Avaimia on kahdenlaisia. *Symmetrisen avaimen salausalgoritmi* (engl. *Symmetric-key algorithm*) on sellainen, jossa samaa ennalta valittua avainta käytetään sekä salaukseen, että salauksen purkamiseen. Tämän voi mieltää eräänlaiseksi salasanaksi. Symmetrinen avain ei kuitenkaan sovi tilanteeseen, jossa salattu viesti on tarkoitus siirtää muualla olevalle vastaanottajalle. Ongelmana on, että viestin lisäksi pitäisi siirtää myös salausavain, jotta vastaanottaja saa salatun viestin purettua. Internetin tai puhelinverkon viestejä voidaan salakuunnella ja postin kautta lähetetty kirje saatetaan kaapata ja lukea. Jos viesti ja avain lähetetään saman väylän kautta, hyökkääjä voi kaapata ne molemmat ja selvittää viestin sisällön helposti. Symmetrinen avain soveltuu siis paremmin tilanteeseen, jos-

sa informaatio pyritään ainoastaan suojaamaan, eikä sitä välttämättä ole tarkoitus siirtää. Symmetristä salausta voidaan toki käyttää myös viestien vaihtoon, jos avaimesta sovitaan etukäteen, tai jos avain siirretään jollakin turvallisella tavalla. Avaimien siirtäminen tapahtuu erityisillä avaimenvaihtomenetelmillä (engl. *key exchange*), kuten Diffie-Hellman-avaimennvaihtoprotokollalla[7].

Toinen joukko avaimiin perustuvia salausmenetelmiä on epäsymmetrisen avaimen salausmenetelmät (engl. *asymmetric key cryptography*), joihin edellä mainittu avaimenvaihtoprotokollakin perustuu. Epäsymmetrisessä avaimessa on kaksi osaa: julkinen osa ja salainen osa. Epäsymmetrisen avaimen menetelmistä käytetään yleisesti myös nimeä *julkisen avaimen salausalgoritmi* (engl. *public key cryptography*). Epäsymmetrinen salaus perustuu siihen, että salaukseen käytettävä avain on julkista tietoa. Henkilö, joka haluaa vastaanottaa salattuja viestejä, ilmaisee kaikille julkisen avaimensa. Tästä avaimesta ei voi päätellä avaimen salaista osaa. Viesti salataan julkisella avaimella, mutta sitä ei voi enää purkaa samalla avaimella. Salauksen purkamiseen tarvitaan avaimen salainen osa, joka on ainoastaan viestejä vastaanottavan tahon tiedossa. Esimerkkialgoritmi ja sen matemaattinen perusta käsitellään tarkemmin aliluvussa 2.4.3.

Sekaannus ja diffuusio

Claude Shannon [8] määritteli salaisessa raportissaan vuonna 1945 salauksessa tarvittaviksi ominaisuuksiksi sekaannuksen (engl. *confusion*) ja diffuusion (engl. *diffusion*). Erityisesti avainpohjaisessa salauksessa, jossa varsinainen algoritmi on kaikkien tiedossa, avaimen on tärkeä tuottaa molemmat ominaisuudet. Sekaannus pyrkii siihen, että salatekstin kukin bitti riippuu useasta avaimen osasta. Näin salatekstin ja avaimen välille ei muodostu yhteyttä, joka mahdollistaisi avaimen päättelyn salatekstin perusteella. Diffuusio puolestaan pyrkii siihen, että pienikin muutos selkotekstissä johtaa mahdollisimman suureen muutokseen salatekstissä. Tällöin puolestaan selkotekstin ja salatekstin välinen yhteys hälvenee. Molemmat ominaisuudet yhdessä johtavat siihen, että selkotekstin,

salatekstin ja avaimen väliset yhteydet eivät ole havaittavissa, eikä esimerkiksi pienillä avaimen muutoksilla saada purettua osaa salatekstistä.

Digitaalinen allekirjoitus

Julkinen avain ei ole vielä vastaus kaikkiin ongelmiin. Jos salattavan viestin lähettäjä noutaa julkisen avaimen esimerkiksi vastaanottajan verkkosivustolta, hän ei voi olla varma, onko joku mahdollisesti hakkeroinut verkkosivuston ja syöttänyt sinne oman julkisen avaimensa. Tätä varten on kehitetty menetelmiä, joilla data voidaan allekirjoittaa omistajan varmentamiseksi. Eräs digitaalisen allekirjoituksen muoto on viestin salaaminen *yksityisen avaimen* avulla. Tämän salatekstin saa purettua ainoastaan avainta vastaavan julkisen avaimen avulla. Jos viestin purkaminen ei onnistu, viestin vastaanottajalle selviää, ettei hänellä oleva julkinen avain ole oikea. [2, 9] Toki tässä on edelleen se ongelma, ettei vastaanottaja voi tietää, onko huijari onnistunut ujuttumaan hänelle väärän julkisen avaimen, sekä väärällä yksityisellä avaimella allekirjoitetun viestin. Tätä varten on sertifikaattiauktoriteetiksi kutsuttuja tahoja, jotka myöntävät digitaalisia sertifikaatteja, eli käytännössä allekirjoittavat julkisia avaimia. Näin julkisen avaimen vastaanottaja voi olla varma, että hänellä on oikea avain, jonka avulla hän voi varmistua avaimen yksityisen osan haltijan identiteetistä. [10] Toki edelleen on mahdollista, että joku saa käsiinsä sertifikaattien allekirjoitukseen käytettävän avaimen, jolloin käytännössä koko internetin tietoturva vaarantuu. Täysin pomminvarmaa ratkaisua koko ongelmaan ei siis ole.

2.2.2 Muihin salaisuuksiin perustuva salaus

Jos salaus perustuu vain algoritmin salassapitoon, on lähtökohtaisesti erittäin turvaton. Vaikka salaus olisi matemaattisesti murtumaton, algoritmin paljastuttua kaikki kyseisellä algoritmilla salattu informaatio voidaan purkaa ja algoritmi on lopullisesti käyttökelvoton. Esimerkki tällaisen algoritmin käytöstä on DVD-levyissä sovellettu CSS-suojaus. CSS perustui vain algoritmin salassapitoon ja algoritmin algoritmi murrettiin vuotaneen

lähdekoodin avulla vuonna 1999. Vaikka salauksessa käytettiin myös eräänlaista avainta, niitä oli hyvin rajallinen määrä ja ne olivat helposti laskettavissa. Vuoden 2019 teknologialla CSS-algoritmin avulla salattu informaatio on mahdollista murtaa sekunneissa tavallista tietokonetta käyttäen.[11]

Kun kyseessä on salausjärjestelmä, joka perustuu ainoastaan algoritmin monimutkaisuuteen ja salassapitoon, puhutaan *vaikeaselkoisuudella saavutetusta turvallisuudesta* (engl. *Security by obscurity*). Monimutkaisinkin salausalgoritmi saattaa vahingossa vuotaa julkisuuteen, jolloin kaikki salatut viestit paljastuvat. Esimerkiksi National Institute of Standards and Technology suosittelee olemaan käyttämättä tämänkaltaisia salausmenetelmiä.[12]

2.2.3 Merkkisalakirjoitus ja lohkosalakirjoitus

Merkkisalakirjoitus (engl. *Stream cipher*) on salakirjoitusalgoritmi, joka salaa viestin yksi merkki tai bitti kerrallaan. Lähtökohtaisesti yhden merkin salaamiseen käytetään yhtä avaimen osaa. Luvussa 2.4 esiteltävä toisen maailmansodan aikainen Enigma on myös eräänlainen merkkisalakirjoituskone. Modernimmat versiot ovat käytännössä matemaattisia funktiota, jotka tuottavat pseudosatunnaisia lukuja tiettyjen sääntöjen mukaan. Luvut ovat yleensä yksittäisiä bittejä, joita käytetään datan salaamiseen bitti kerrallaan. Merkkisalakirjoitusten ongelmaksi voi muodostua niiden tuottaman satunnaislukujakson pituus. Ideaalitilanteessa jakso olisi ääretön, jolloin voitaisiin puhua kerta-avaimesta ja päästäisiin lähelle aliluvussa 2.5 käsiteltävää täydellistä salausta. Käytännössä lineaarisella avaingeneraattorilla jokin tietyn mittainen toistuva jakso, vaikka se voidaankin suunnitella riittävän pitkäksi useisiin käyttötarkoituksiin. Kuitenkin tällaisen avaingeneraattorin tuottaman avaimen pystyy päättelemään myös jakson pituutta lyhyemmästä bittimäärästä[13].

Lohkosalakirjoitus (engl. *Block cipher*) puolestaan järjestää datan tasamittaisiin lohkoihin ja salaa lohkon sisältämän datan keskenään. Lohkosalausten eräs potentiaalinen ongelma on tietyn datan salautuminen aina samalla tavalla. Tällöin huonon lohkosalausalgoritmin datasta saattaa vuotaa tilastollista informaatiota. Tämän ongelman ominaisuuk-

sia ja sen ratkaisua käsitellään tarkemmin luvussa 6 esiteltävän uuden algoritmin yhteydessä aliluvussa 6.1.3.

2.3 Kryptoanalyysi

Kryptoanalyysi on salakirjoituksen murtamista. Tarkoituksena ei välttämättä ole haitan aiheuttaminen, vaan murtamisella pyritään varmistamaan jatkuvasti, että käytössä olevat algoritmit ovat edelleen turvallisia. Lisäksi uusia menetelmiä voidaan pyrkiä kehittämään niin, että tunnetut murtomenetelmät eivät ole käyttökelpoisia niitä vastaan. Jos salausjärjestelmästä löytyy jokin kriittinen haavoittuvuus, kyseinen järjestelmä on tärkeää poistaa yleisestä käytöstä, ennen kuin jokin muu taho löytää saman heikkouden ja alkaa purkaa salattuja viestejä haitallisiin tarkoituksiin.

Salakirjoitustiede on myös kilpajuoksua matematiikan ja laskentatehon välillä. Tehokkaammiksi kehittyvät tietokoneet tuovat jatkuvasti lisämahdollisuuksia raa'an voiman käyttöön salakirjoituksen murtamisessa. Yksinkertaisimmat salakirjoitukset pystytään murtamaan tavallisella tietokoneella sekunnin murto-osissa. Myös verrattain moderneja salausjärjestelmiä on murrettu alati kasvavan tietokonetehon ansiosta. Periaatteessa äärimmäisellä tietokoneteholla voitaisiin kokeilla kaikki mahdollisia avaimia ja näin purkaa mikä vain tunnetulla algoritmilla salattu teksti. Monet nykyisinkin käytössä olevat algoritmit perustuvat yksinkertaisesti siihen, että kokeiltavia avaimia olisi enemmän kuin universumissa on alkeishiukkasia, jolloin salaus on käytännössä murtumaton.

2.3.1 Menetelmien luokittelu

Salakirjoituksen murtamiseen käytettävät menetelmät erottuvat toisistaan sen perusteella, millaisia tietoja hyökkääjällä on käytettävissään. Vaihtoehtoja ovat seuraavat:

Pelkkä salatekstipohjainen hyökkäys on sellainen, jossa hyökkääjällä on ainoastaan kokoelma salattuja viestejä ja hän yrittää näiden avulla selvittää tekstien sisältöä.

Tunnettu selkotekstihyökkäys on sellainen, jossa hyökkääjällä on käytössään esimerkki salatusta tekstistä ja sitä vastaava alkuperäinen selkoteksti.

Valittu selkotekstihyökkäys on selkotekstihyökkäyksen erikoistapaus, jossa hyökkääjän on mahdollista saada valitsemiaan selkoteksti-salatekstipareja. Tämä ei välttämättä vielä tarkoita, että hyökkääjä saisi purettua minkä tahansa salatekstin tai että hän tuntisi käytetyn algoritmin toiminnan.

Mukautuva selkotekstihyökkäys on hyökkäys, jossa on mahdollista saada uusia selkoteksti-salatekstipareja edellisten parien perusteella.

Toisiinsa liittyvät avaimet on tilanne, jossa hyökkääjällä on sama selkoteksti salattuna kahdella eri avaimella.

Luokittelu murtumisen laajuuden perusteella

Erilaisia salakirjoituksen murtumisia voidaan luokitella myös niiden vaikutuksen perusteella. Kaikki menetelmät eivät välttämättä aiheuta täsmälleen yhtä suurta vahinkoa. L. Knudsen esittää esimerkiksi seuraavaa luokittelua [14]:

Täydellinen murtuminen jolloin hyökkääjä pystyy päättelemään salausavaimen. Kaikki lohkopohjaiset salaukset murtuvat, jos kokeillaan kaikkia mahdollisia avaimia yksi kerrallaan.

Globaali päättely (engl. *global deduction*) jolloin hyökkääjä onnistuu löytämään algoritmin salauksen tekemiseen tai purkamiseen, tuntematta salausavainta.

Paikallinen päättely (engl. *local deduction*) Hyökkääjä onnistuu löytämään selkoteksti-salateksti parin.

Informaatiovuoto (engl. *information deduction*) jolloin hyökkääjälle selviää jotakin informaatiota joko avaimesta, selkotekstistä tai salatekstistä.

Menetelmät ovat hierarkkisessa järjestyksessä. Ylempänä oleva johtaa aina alempana olevaan.

Lohkosalakirjoitukset hajoavat varmasti, jos koitetaan jokaista mahdollista avainta. Esimerkiksi jos kaikki avaimet on mahdollista kokeilla lyhyessä ajassa, voidaan puhua täydellisestä murtumisesta. Tieteellisesti tarkasteltuna tietty salausmenetelmä voidaan katsoa murtuneeksi, jos löytyy pienikin algoritmin ilmoitettua turvallisuutta heikentävä hyökkäys. Esimerkiksi SHA-1 -tiivistefunktion suunniteltu turvallisuus oli 2^{80} bittiä, mutta myöhemmin on osoitettu sen murtuvan jo noin $2^{60.3}$ kokeilulla.[15] Määrä on edelleen todella suuri, mutta joka tapauksessa algoritmi on huonompi, kuin sen on suunniteltu olevan.

Luokittelu vaadittujen resurssien suhteen

Salauksen murtamiseen vaaditaan kolmea eri resurssia; aikaa, muistia ja dataa. Muistilla tarkoitetaan murtoalgoritmin suorittamiseksi tarvittavaa työmuistia ja data voi olla esimerkiksi salatekstiä, tai salateksti-selkotekstipareja. Esimerkiksi jokaisen salausavaimen kokeileminen ei vaadi juurikaan muistia, eikä siihen tarvita dataa, mutta se kuluttaisi todella paljon aikaa. Lopputuloksena salateksti saadaan purettua, mutta tämä ei kuitenkaan johtaisi muiden viestien purkamiseen. On myös mahdollista, että salausalgoritmista löytyy jonkin heikko kohta, jos varastoidaan tietty määrä samankaltaisia selkoteksti-salateksti pareja. Vaikka dataa olisi mahdollista hankkia, fyysinen kapasiteetti ei välttämättä riitä datan tallentamiseen tai sen käsittelemiseen, mikä tekee kyseisestä murtomenetelmästä epäkäytännöllisen.

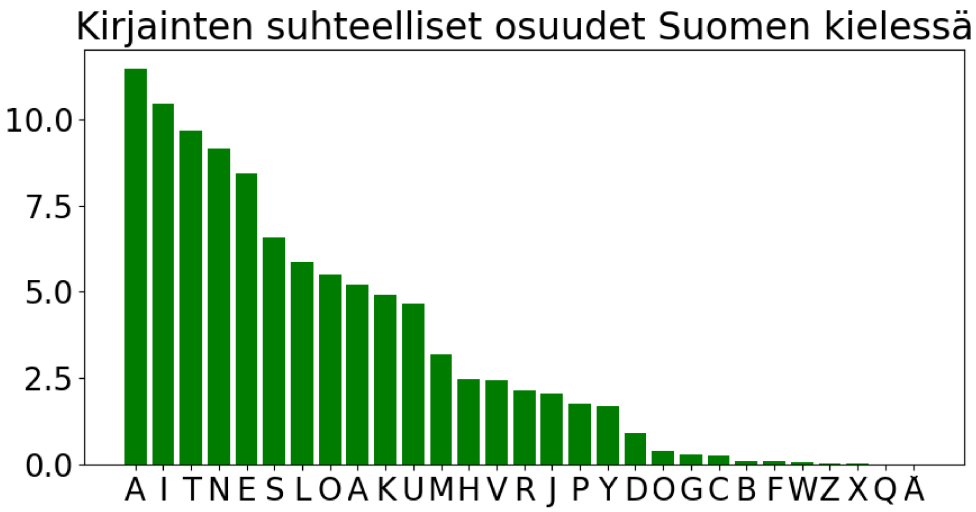
2.3.2 Katsaus menetelmiin

Tässä aliluvussa esitellään muutamia menetelmiä, joilla salakirjoituksen voi yrittää murtaa. Modernien salakirjoitusmenetelmien vastaan tehtävät hyökkäykset perustuvat yleensä kyseisen salausalgoritmin matemaattisiin puutteisiin ja murtomenetelmä kehitetään täs-

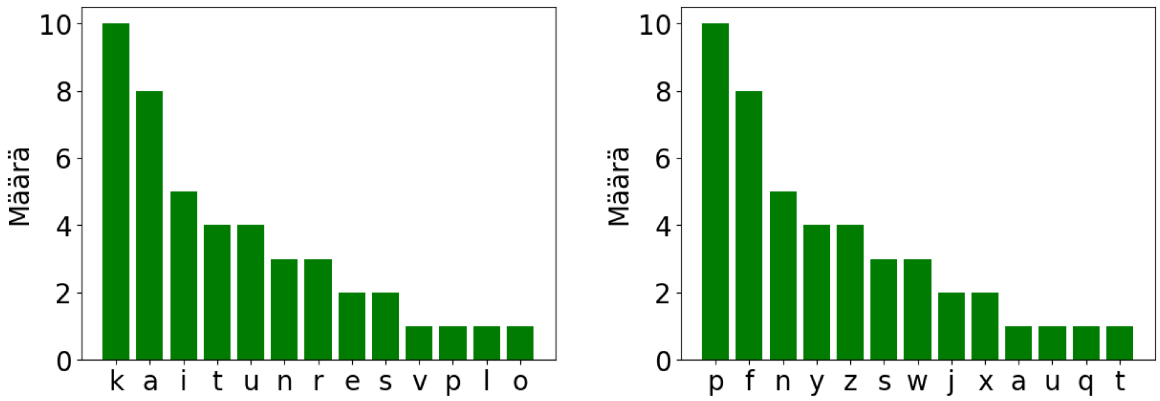
mälleen tiettyä algoritmia varten. Tässä esiteltävät menetelmät ovat puolestaan hiukan yleiskäyttöisempiä.

Frekvenssianalyysi

Erityisesti korvaussalakirjoituset (engl. *substitution cipher*), joissa tietty merkki korvataan aina samalla merkillä, on helppo murtaa viestin ollessa tarpeeksi pitkä. Tämä johtuu siitä, että kielessä käytettyjen kirjainten suhteelliset osuudet ovat vakiintuneita. Kuvassa 2.1 on esitetty aakkosten suhteelliset osuudet suomenkielisessä tekstissä. Kun valitaan jokin riittävän pitkä teksti, a-kirjain on tilastollisesti yleisin. Kaaviossa 2.2(a) on lauseen "viekas kettu punaturkki laiskan koiran takaa kurkki" kirjainten frekvenssit. Koska viesti on melko lyhyt, yleisin kirjain ei ole sama kuin tilastollisesti pitäisi olla, vaan tässä tapauksessa kirjain sattuu olemaan "k". Kuitenkin voidaan huomata, että suomen kielen kolme yleisintä kirjainta löytyy esimerkkitekstin neljän yleisimmän kirjaimen joukosta ja vielä samassa järjestyksessä. Kuvassa 2.2(b) nähdään frekvenssit, kun salaus on tehty siirtämällä jokaista kirjainta eteenpäin viidellä. Näin ollen esimerkiksi A-kirjaimesta tulee F, B-kirjaimesta tulee G ja niin edelleen. Kuvia 2.2(a) ja 2.2(b) vertaillessa käy ilmi, että kirjainten suhteelliset osuudet ovat samat. Salakirjoitusta voisi lähteä murtamaan kokeilemalla selkotekstin oletetun kirjoituskielen yleisintä kirjainta salatekstin yleisimmän kirjaimen kohdalle. Näin kokeilemalla salateksti ratkeaa yleensä melko nopeasti. Tietokoneella tämänkaltainen salaus on mahdollista murtaa sekunnin murto-osissa kokeilemalla läpi kaikki eri vaihtoehdot.



Kuva 2.1: Kirjainten suhteelliset osuudet suomenkielisessä tekstissä.



(a) Kirjainten osuus selkotekstissä

(b) Kirjainten osuus salatekstissä

Kuva 2.2: Kirjainten suhteelliset osuudet säilyvät, vaikka niitä vaihdettaisiin toisiksi.

Differentiaalinen kryptoanalyysi

Differentiaalinen kryptoanalyysi on pääasiassa lohkosalausalgoritmeja vastaan käytetty hyökkäystekniikka. Kyseessä on yleensä valittu selkotekstihyökkäys, mutta variaatiota löytyy jopa pelkkään salatekstipohjaiseen hyökkäykseen. Tyypillisesti hyökkäyksessä otetaan tietyn selkotekstin useampia salatekstejä ja lasketaan niiden keskinäinen eriävyys, tavoitteena löytää jotakin tilastollista informaatiota, jolla paljastetaan avaimen vai-

kutus salatekstiin. Jos avaimen vaikutus syötteeseen ei ole tarpeeksi suuri, algoritmi ratkeaa luultavasti nopeammin, kuin kaikkia avaimia kokeilemalla. [16] Esimerkiksi moderni AES -algoritmi on suunniteltu niin, että differentiaalinen kryptoanalyysi olisi mahdollisimman tehoton sitä vastaan [17].

Epäsuorat hyökkäykset

Salattujen viestien sisällön selvittämiseen ei välttämättä tarvita varsinaisen algoritmin murtamista. Joskus informaatiota voidaan hankkia epäsuorasti huonosti toteutetun tai kaapatun järjestelmän kautta. Tähän käytettäviä keinoja ovat esimerkiksi virrankäytönseurantahyökkäys (engl. *power-monitoring attack*) ja sähkömagneettinen hyökkäys (engl. *electromagnetic attack*), jolloin mitataan täysin ulkopuolisia tekijöitä, kuten tietokoneen tuottamaa sähkömagneettista säteilyä tai virrankulutusta ja pyritään näiden avulla analysoimaan salauksessa käytettyjä laskutoimituksia. Jos taas hyökkääjällä on osittainen pääsy järjestelmien sisään, hän voi pyrkiä tutkimaan järjestelmän muistia ja pyrkiä löytämään jälkiä käytetyistä avaimista.[18] Avaimia ei välttämättä ole poistettu oikealla tavalla, esimerkiksi ylikirjoittamalla käytetyt muistipaikat tai tallennustila. Muita keinoja laitteeseen tunkeutumiseen on esimerkiksi peukaloidut laiteohjelmistot, jotka tarjoavat takaportin järjestelmään ilman kirjautumista. Murrettuun järjestelmään voidaan asentaa ohjelmia jotka lähettävät jatkuvasti käyttötietoja, kuten näppäinpainalluksia hyökkääjän palvelimelle. [19]

Väsytyshyökkäys

Väsytyshyökkäys (engl. *brute-force -attack*) on eräs yksinkertaisimmista, mutta samalla tehottomimmista menetelmistä. Väsytyshyökkäys tarkoittaa, että salaus murretaan kokeilemalla jokaista mahdollista avainta, kunnes oikea avain löytyy. Tätä varten on kehitetty jopa laitteistotason ratkaisuja. Vuonna 1998 kehitettiin verrattain edullinen, noin 200 000 dollarin hintainen laite, joka kykeni selvittämään yleisesti käytössä olleen DES -

salausjärjestelmän salausavaimen muutamassa päivässä. [20] Modernimpien salausjärjestelmien, kuten 256 bittisen AES salauksen murtaminen kestäisi huomattavasti kauemmin. Maailman nopein supertietokone Summit (2019, [21]) kykenee $2 \cdot 10^{14}$ liukulukuoperaatioon sekunnissa. Jos oletettaisiin reippaasti yliarvioiden, että yksi operaatio vastaisi yhtä avainkokeilua, supertietokone saisi kokeiltua jokaisen avaimen noin $1.8 \cdot 10^{55}$ vuodessa, mikä on moninkertaisesti enemmän kuin universumin nykyinen ikä. Hyvin toteutetun algoritmin eräs ominaisuus on siis myös se, että algoritmi on immuuni kulloinkin käytössä olevan teknologian avulla toteutetulle väsytyshyökkäykselle.

2.4 Salausjärjestelmiä

Salausjärjestelmiä on käytetty historiallisista ajoista lähtien ja erilaisia salausjärjestelmiä on niin paljon, ettei olisi mielekästä listata jokaista. Tässä aliluvussa luodaan pintapuolinen katsaus muutamaan tunnettuun salausjärjestelmään. Esitellyt salausjärjestelmät muodostavat riittävän poikkileikkauksen salakirjoituksen historiasta ja antavat esimerkkejä eri periaatteilla toimivista salausjärjestelmistä.

2.4.1 Klassisia salausjärjestelmiä

Ennen tietokoneiden olemassaoloa käytössä oli alkeellisia salausjärjestelmiä. Salattava data oli tyypillisesti selkokielistä tekstiä tavallisella aakkostolla kirjoitettuna. Salauksen tarkoitus oli muuttaa teksti satunnaisen näköiseksi kirjainten sekamelskaksi. Nämä algoritmit perustuvat siihen, että jokaiselle kirjaimelle annetaan numeerinen arvo, esimerkiksi

$$A = 0,\ B = 1,\ \ldots\ , Z = 25$$

jolloin niitä voidaan käsitellä matemaattisesti. Täysi englanninkielinen aakkosto esiteltiin aiemmin taulukossa 2.1. Matemaattisena pohjana näille salausjärjestelmille on modulaarinen aritmetiikka. Aakkosto voi myös koostua vain osasta aakkosia, tai siihen voidaan

lisätä mikä tahansa äärellinen määrä muita merkkejä, joille kaikille annetaan yksikäsitteinen numeerinen arvo.

Caesar-salakirjoitus

Caesar-koodi [1] on jo Julius Caesarin aikaan käytössä ollut historiallinen salausjärjestelmä. Siinä jokaista aakkosta siirretään tietty määrä eteenpäin. Salataan esimerkkinä sana **`gradu`**.

1. Valitaan salausavaimeksi luku $x = 3$, eli jokaista kirjainta siirretään eteenpäin 3 askelta.

2. Muutetaan selkoteksti numeroiksi edellä mainitulla järjestelmällä:

$$g = 6$$

$$r = 17$$

$$a = 0$$

$$d = 3$$

$$u = 20$$

3. Lisätään kaikkiin lukuihin x:

$$6 + 3 = 9$$

$$17 + 3 = 20$$

$$0 + 3 = 3$$

$$3 + 3 = 6$$

$$20 + 3 = 23$$

4. Muutetaan luvut takaisin kirjaimiksi:

$$9 = j$$

$$20 = u$$

$$3 = d$$

$$6 = g$$

$$23 = x$$

5. Selkoteksti on nyt muutettu salatekstiksi: **judgx**

Tämä salaus on ollut murrettavissa helposti jo ennen tietokoneita. Viestiä purkava henkilö voi kokeilla purkaa salatekstin kaikilla x:n arvoilla verrattain lyhyessä ajassa sillä vaihtoehtoja x:lle on vain aakkoston kirjaimia vastaava määrä.

Affiinisalaus

Affiinisalaus (engl. *affine cipher*) voidaan katsoa parannelluksi versioksi caesar-salauksesta. Salausfunktio $f(x)$ on affiinikuvaus

$$f(x) = (ax + b)\ mod\ m$$

jossa x on salattava numero, m on aakkoston pituus ja a ja b ovat salauksen avain. Avain a on valittava siten, että a ja m ovat keskenään jaottomia, eli niiden suurin yhteinen tekijä $syt(a, m) = 1$. Affiniisalauksen etu caesar-salaukseen on, että purkaminen ei onnistu ainoastaan kokeilemalla siirtää jokaista kirjainta yhtä monta kertaa. Muuttujien a ja b mahdolliset arvot ovat kuitenkin verrattain rajallisia. Koska a täytyy valita siten, että $syt(a, m) = 1$ ja englanninkielisen aakkoston koko on 26, sellaisia lukuja a, joille $syt(a, 26) = 1$, on yhteensä 12.[2] Muuttujalle b on puolestaan 26 eri arvoa. Tämä johtuu siitä, että kun lausekkeesta $(ax + b)$ otetaan jakojäännös luvulla m, luvuilla $b > m$ pätee

[2] $syt(a, 26) = 1$, kun $a = 1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25$

$b \equiv b + km \ (mod\ m)$. Esimerkiksi siis b:n arvolla 1 ja 27 salauksen lopputulos on identtinen. Ottaen huomioon a:n ja b:n mahdolliset arvot, saadaan erilaisten avainten määräksi $12 \cdot 26 = 312$. Jos salauksen purkaminen on tärkeää, päättäväinen laskija voi murtaa tämänkin salauksen väkisin kokeilemalla kaikki 312 eri avainta verrattain lyhyessä ajassa kynällä ja paperilla, puhumattakaan tietokoneen käyttämisestä. Lisäksi tämänkin salauksen heikkoutena on se, että samasta merkistä tulee aina sama merkki. Tällöin voidaan soveltaa frekvenssianalyysiä, kuten esitettiin aliluvussa 2.3.

2.4.2 Enigma

Toisen maailmansodan aikana saksalaisten käyttämä Enigma on eräs tunnetuimmista historiallisista salausjärjestelmistä.[22] Enigma on fyysinen laite, jossa on kirjoituskoneen näppäimistö. Selkoteksti salataan kirjoittamalla se Enigman näppäimistöllä. Näppäiltäessä tietty merkki, koneessa syttyy lamppu, joka ilmaisee mihin kirjaimeen syötetty kirjain muuttuu. Enigman avaimena käytettiin roottoreita, joiden sisällä oli eri tavoin järjestettyjä johtoja, joiden kautta lamppuihin johdettiin sähköä. Kirjoittaessa roottorit naksahtelivat eteenpäin, muuttaen laitteen asetuksia jatkuvasti. Tämä mahdollisti sen, että kaksi peräkkäin syötettyä samaa kirjainta muuttuivat eri kirjaimiksi ja vastaavasti saattoi olla mahdollista, että kaksi eri kirjainta muuttui samaksi kirjaimeksi. Tästä on suuresti hyötyä, sillä vaikka salatekstin tietty sana saataisiin selville, tietoa ei voida käyttää koko viestin ratkaisemiseen. Myöskään frekvenssianalyysiä ei voida käyttää samaan tapaan kuin yksinkertaisissa korvaussalakirjoituksissa.

Enigmassa oli lukuisia heikkouksia. Ensinnäkään sama kirjain ei voinut koskaan kuvautua itseensä. Tällöin viestiä purkava henkilö tietää, että salatekstin kirjain k ei koskaan voi olla selkotekstissä kirjain k. Toinen ongelma oli avainten hallinta. Jotta teksti voidaan purkaa, pitää tietää salauksen tekijän käyttämä roottorien alkuasento. Jotta kaikkea salattua tekstiä ei voisi purkaa samalla avaimella, roottorien asento piti vaihtaa mahdollisimman usein. Tämän takia päivittäin vaihtuvat roottorien alkuasennot oli kirjattu

paperille. Tämä on aiemmin esitettyjen Kerckhoffsin periaatteiden vastaista, sillä fyysinen kokoelma salauksessa tarvittavia tietoja aiheuttaisi ongelmia päätyessään vääriin käsiin. Avainlistan paljastuessa kaikki tulevien päivien avaimet paljastuvat, eikä salausta voi enää käyttää ennen kuin kaikkien osapuolten omistamat tulevien päivien koodit on saatu vaihdettua. Enigmaa käytettiin sodan aikana ja silloin uusien fyysisten avainlistojen kuljettaminen kaikille osapuolille olisi ollut erityisen hankalaa.

Eräs Engiman merkittävä heikkous oli saksalaisten viestien ennalta-arvattavat rakenteet, kuten otsikot ja allekirjoitukset. [23] Tällöin salatekstin ja selkotekstin välillä on tunnettu yhteys ja avain on helpompi selvittää, kuten mainittiin luvussa 2.3.

2.4.3 RSA

RSA (Rivest–Shamir–Adleman) on laajassa käytössä oleva julkisen avaimen salausjärjestelmä. RSA:n toiminta perustuu myös modulaariseen aritmetiikkaan, mutta apuna käytetään eksponenttifunktioita.

Kokonaislukujen alkutekijöihinjaon ongelma

RSA:n turvallisuus perustuu kokonaislukujen tekijöihinjaon ongelmaan. Jos valitaan kaksi alkulukua, kuten 11 ja 17, voidaan helposti laskea niiden tulo $11 \cdot 17 = 187$. Jos taas tiedetään ainoastaan kahden alkuluvun tulo, esimerkiksi $x \cdot y = 391$, mitkä ovat x ja y? Nopeimmatkin tunnetut algoritmit tämän ratkaisemiseen ovat varsin hitaita.[24] Kun käytetään riittävän suuria lukuja x ja y, vaikka niiden selvittäminen on teknisesti mahdollista, kokeiltavia vaihtoehtoja on liikaa.

RSA:n toimintaperiaate

Avainten generointi tapahtuu seuraavalla tavalla[2]:

1. Valitaan kaksi suurta alkulukua p ja q, $p \neq q$ ja lasketaan $N = pq$

2. Valitaan kokonaisluku e jolle pätee $1 < e < N$ ja $syt(e, (p-1)(q-1)) = 1$

3. Valitaan d niin, että $ed \equiv 1\ (mod (p-1)(q-1))$

Nyt N ja e muodostavat julkisen avaimen ja N ja d yksityisen avaimen. Jotta selkotekstin merkistä m saadaan salatekstin merkki c, lasketaan:

$$c \equiv m^e\ mod\ N$$

Purkaminen tapahtuu laskemalla:

$$m \equiv c^d\ mod\ N$$

2.4.4 AES

AES (engl. *Advanced Encryption Standard*) on standardoitu lohkosalausmenetelmä. Algoritmi tunnetaan myös alkuperäiseltä nimeltään *Rijndael*, kehittäjiensä J. Daemenin ja V. Rijmenin mukaan. Se on niin laajassa käytössä, että eräiden prosessorien käskykannat tukevat salauksen tekoa laitteistotasolla. Tämä tekee salauksesta erittäin nopeaa. Salauksessa data jaetaan 128 bitin eli 16 tavun pituisiin, $4 \cdot 4$ kokoisiin lohkoihin. Käytettävät avaimet ovat 128, 196 tai 256 bittisiä. Salauksessa jokaiselle 128 bitin lohkolle tehdään seuraavat operaatiot[25]:

Kierrosavainten johtaminen salausavaimesta. Jokaiselle kierrokselle luodaan oma avain, joka johdetaan alkuperäisestä salausavaimesta.

Kierrosavaimen lisääminen dataan bittitason XOR-operaatiolla.

Avaimesta riippuva määrä kierroksia, joilla jokaisella tehdään neljä eri operaatiota. Ensin dataa muutetaan ennalta määritetyn hakutaulun mukaan eri arvoiksi. Sen jälkeen jokaista riviä siirretään rivin numeron verran vasemmalle, palauttaen lohkon yli päätyvät arvot takaisin ruudukon oikealle puolelle. Tämän jälkeen kukin sarake kerrotaan ennalta määritellyllä matriisilla ja vielä lopuksi lohkoon lisätään uusi

kierrosavain. Avaimen pituudesta riippuen tämä kaikki tehdään 9, 11 tai 13 kertaa kullekin lohkolle.

Viimeinen kierros on erilainen. Se on muuten sama kuin edellinen, mutta sarakkeiden matriisikertolasku jää pois.

AES on saatu osittain murtumaan, jos kierroksia on pienempi määrä. Tosin silloinkin laskennalliset vaatimukset ovat epäkäytännöllisiä, jotta murrosta olisi hyötyä. [26] Vuonna 2020 AES-salausta vastaan ei ole tiedossa käytännöllistä hyökkäystä, jolla salattu data saataisiin luettua ilman avainta.

2.5 Täydellinen salaus

Murtamaton salaus on keksitty, mutta se ei ole käyttökelpoinen. Kyseessä on kerta-avaimeen (engl. *one-time-pad, OTP*) perustuva menetelmä, jossa tiettyä salausavainta käytetään ainoastaan kerran. Lisäksi avain tulee valita niin, että se on salattavan selkotekstin mittainen. Jatkuva avainten vaihto olisi haastavaa ja erityisesti pitkien viestien lähettäminen olisi epäkäytännöllistä. Kerta-avain toimii siten, että selkotekstin jokainen bitti salataan käyttämällä täsmälleen yhtä bittiä avaimesta. Jos kerta-avain on muodostettu oikein, eli valitsemalla mahdollisimman satunnaisia lukuja, salatekstiksi muodostuu vain jono toisistaan riippumattomia bittejä. Tilastollisista menetelmistä ei ole apua, sillä salausalgoritmista tai -avaimesta ei vuoda tilastollista informaatiota salatekstiin. Salattu teksti ei eroa täysin sattumanvaraisesti valituista merkeistä, eikä kahden peräkkäisen merkin välillä ole minkäänlaista yhteyttä.

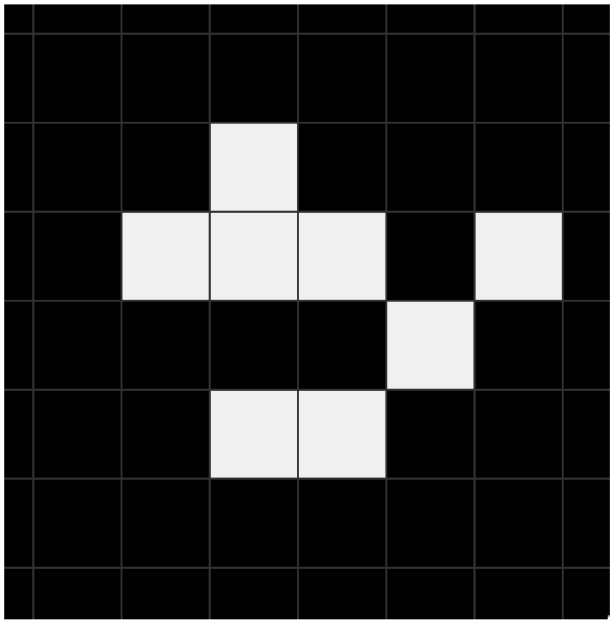
Luku 3

Soluautomaatti

Soluautomaatti on diskreetti dynaaminen malli, joka kuvaa säännöllisellä verkolla olevien solujen tilaa. Tässä tutkielmassa käsitellyt soluautomaatit perustuvat äärelliseen tai ääret-tömään neliömäiseen verkkoon, jota kuvaamaan käytetään termiä *ruudukko*. Ruudukko koostuu soluista, joista jokaisella on jokin tarkasti määritelty tila. Tilat voidaan esittää esimerkiksi väreinä tai numerona. Yksinkertaisen soluautomaatin soluilla voisi olla kaksi tilaa: elävä ja kuollut. Nämä tilat voidaan esittää numeroina, kuten 1 ja 0. Visualisoinnin helpottamiseksi solut voidaan esittää väreillä esimerkiksi niin, että elävä solu on valkoi-nen ja kuollut solu musta. Esimerkki soluautomaatin visualisoinnista on kuvassa 3.1.

Solujen tilaa tietyllä ajanhetkellä kutsutaan sukupolveksi. Sukupolvi vaihtuu määrittä-mällä kaikille soluille uusi tila kunkin solun vieressä olevien solujen, eli naapurien perus-teella. Uuden tilan selvittämisen jälkeen kaikki solut siirtyvät uuteen tilaan yhtäaikaisesti.

Tämän luvun tarkoituksena on käsitellä soluautomaattien perusteet ja keskeiset omi-naisuudet. Luvussa 6 salauksen kannalta tärkein soluautomaatti käsitellään laajuutensa vuoksi erikseen luvussa 4.



Kuva 3.1: Eläviä ja kuolleita soluja ruudukolla.

3.1 Määritelmä

Matemaattisesti soluautomaatti A on monikko $A = (d, S, N, f)$, jossa [27]

- $d \in \mathbb{Z}_+$ on soluautomaatin avaruudellinen ulottuvuus
- S on äärellinen joukko tiloja. Jokaisella solulla on aina jokin näistä tiloista.
- $N = (\vec{n_1}, \vec{n_2}, \ldots, \vec{n_m})$ on naapurustovektori, joka määrittää, mitkä solut ovat vierekkäin, eli toistensa naapureita.
- $f : S^m \to S$ on tilanmuutosfunktio, joka kuvaa solujen tilan muuttumisen, eli sen millä ehdoilla solut vaihtavat tilasta toiseen.

Avaruudellinen ulottuvuus d voi olla mikä tahansa kokonaisluku. Tässä tutkielmassa jokaisella soluautomaatilla on kaksi ulottuvuutta tai vähemmän. Esimerkiksi *ruudukko* viittaa aina kaksiulotteisella tasolla esitettyyn soluautomaattiin, mutta samat säännöt pätevät myös n-ulotteisille soluautomaateille, ellei toisin mainita.

Solun tilojen ei tarvitse rajoittua kahteen. Kolmantena tilana elävän ja kuolleen lisäksi voisi olla vaikka kuoleva solu, joka muuttuu kuolleeksi vasta seuraavan sukupolven aikana. Tyypillisesti tilat kuvataan abstraktimmin numeroina ja niitä voi olla mikä tahansa äärellinen määrä.

Soluautomaattia voidaan simuloida äärellisellä tai äärettömällä ruudukolla. Lisäksi ruudukko voi olla rajattu, kuten munkkirinkilän pinta, jolloin reunoja ei ole. Tällöin siis tiettyyn suuntaan liikkuessa päädyttäisiin lopulta takaisin lähtöpisteeseen.

Naapurustot

Naapurusto määrittää, mitkä solut ovat toistensa vieressä. Naapurusto voi olla millainen tahansa, mutta yleensä naapurustot ovat säännöllisiä ja jokaisella solulla on sama määrä naapureita ja ne ovat suhteellisesti samoissa paikoissa kustakin solusta katsottuna. Kaksi yleisesti käytettyä naapurustoa ovat Von Neumannin naapurusto ja Mooren naapurusto[28].

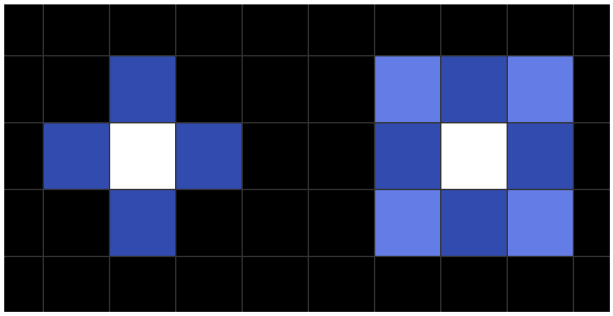
Von Neumannin naapurusto on sellainen joukko ruutuja, joihin pääsee siirtymällä lähtöruudusta yhden reunan yli viereiseen ruutuun.

Mooren naapurusto käsittää Von Neumannin naapuruston lisäksi ruudut, joihin pääsee siirtymään lähtöruudun kulmasta viistoon. Täten yhdellä solulla on kahdeksan naapuria.

Molemmat naapurustot ovat esillä kuvassa 3.2. Naapurustoja voidaan myös laajentaa pidemmiksi, esimerkiksi koskemaan kaikki ruutuja, jotka ovat kahden siirtymisen etäisyydellä.

3.2 Kääntyvyys

Kääntyvän soluautomaatin jokaisella sukupolvella on yksikäsitteisesti määritelty edeltäjä. Tällöin soluille voidaan määrittää seuraavan sukupolven lisäksi yksikäsitteinen edellinen



Kuva 3.2: Von Neumannin (vas.) ja Mooren naapurustot. Valkoisen ruudun naapurit on esitetty sinisen eri sävyillä.

sukupolvi. Näin ollen soluautomaattia voidaan simuloida sekä eteen, että taaksepäin. [29]

Kääntymätön soluautomaatti on puolestaan sellainen, jolle on määritelty yksikäsitteisesti ainoastaan seuraava sukupolvi. Kääntymättömän soluautomaatin tunnusmerkki on, että kahdella tai useammalla erilaisella sukupolvella on yhteinen seuraava sukupolvi. Tällöin tietystä sukupolvesta ei voi määrittää edellistä sukupolvea. Soluautomaattia voisi tietenkin *tarkastella* myös takaperin, jos sitä simuloidaan ensin eteenpäin ja myöhemmin toistetaan nämä tilat takaperin, mutta tällöin kyse on teknisesti vain kuvien katselusta. Edellisen tilan on oltava laskettavissa matemaattisesti, jotta kääntyvyyden määritelmä täyttyy.

3.3 Turing-täydellisyys

Turing-täydellisyydellä tarkoitetaan järjestelmää, joka voi simuloida universaalia Turingin konetta. Turingin kone on abstraktin laskennan malli, joka määrittelee eräänlaisen abstraktin tietokoneen. Universaali Turingin kone on ohjelmoitava järjestelmä, joka kykenee simuloimaan mitä tahansa Turingin konetta.[30] Churchin–Turingin teesin mukaan kaikki laskettavissa oleva on laskettavissa Turingin koneen avulla. Toisin sanoen, jos on mahdollista määritellä funktio tai algoritmi jonkin ongelman ratkaisuun, kyseinen ongelma on myös ratkaistavissa Turingin koneen avulla.[31]

Vaikka soluautomaatit ovat varsin yksinkertaisia ja luonteeltaan usein kaaottisia, asettamalla solut tiettyyn alkutilaan on mahdollista luoda sellaisia järjestelmiä, jotka ovat Turing-täydellisiä. [32, 33, 34]

3.4 Ratkeamattomuus

Ratkeamattomuus (engl. *Undecidability*) on laskentateorian määritelmä ongelmalle, jolle ei ole todistettavasti mahdollista muodostaa yksikäsitteisen vastauksen antavaa algoritmia. Tiettyyn soluautomaattiin liittyvä ratkeamaton ongelma voisi olla esimerkiksi se, johtaako jokin alkutila tiettyyn lopputilaan. Jos kyseinen ongelma on ratkeamaton, ainoa keino vastata kysymykseen on simuloida soluautomaattia tarpeeksi pitkään ja kokeilla, saavuttaako se kyseisen tilan.

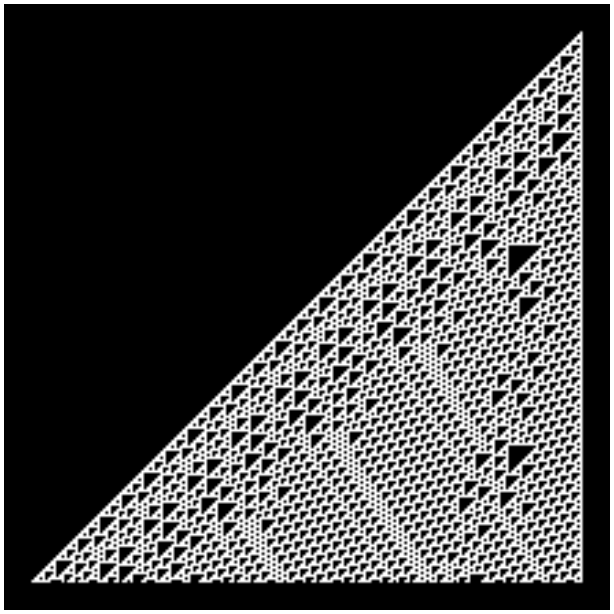
3.5 Erilaisia soluautomaatteja

Tässä aliluvussa käsitellään lyhyesti kaksi tunnettua soluautomaattia. Ensimmäisenä esitellään alkeissoluautomaatti, jonka voidaan katsoa olevan yksi yksinkertaisimmista soluautomaateista. Alkeissoluautomaattia on käytetty myös salaukseen. Soluautomaattien salauskäyttöön tutustutaan luvussa 5. Toisena esitellään Life-peli (engl. *Game of Life*), joka toi soluautomaatit suuremman yleisön tietoisuuteen ja kiihdytti soluautomaattien tutkimusta. [35]

3.5.1 Alkeissoluautomaatti

Alkeissoluautomaatti (engl. *Elementary cellular automaton*) on yksiulotteinen soluautomaatti. Tässä soluautomaatissa on yksi äärettömän pitkä rivi soluja, joiden tila on joko 1 tai 0. Seuraava sukupolvi lasketaan tarkastelemalla solun omaa tilaa ja solun kahden viereisen naapurin tilaa ja laskemalla näiden perusteella uusi tila. Solu ja sen naapurit ovat

voivat olla ykkösiä tai nollia 2^3 eri tavalla ja kullekin yhdistelmälle voidaan määrittää seuraavan sukupolven olevan joko 1 tai 0, joten mahdollisia sääntöjä alkeissoluautomaatille on $2^{2^3} = 256$. Alkeissoluautomaatti esitetään usein kahdessa ulottuvuudessa, jolloin seuraavat tilat piirretään edellisten tilojen alle. Esimerkki alkeissoluautomaatista on kuvassa 3.3. Alkeissoluautomaatti on tietyillä säännöillä Turing-täydellinen ja kykenee universaaliin laskentaan[36]. Halutessaan alkeissoluautomaattia voi laajentaa ottamalla huomioon useampia naapureita, esimerkiksi kaksi molemmilta puolilta.



Kuva 3.3: Alkeissoluautomaatti 110. Alkutilana ylimmällä rivillä on yksi valkoinen piste ja seuraavat tilat ovat sen alapuolella, kukin omalla rivillään.

3.5.2 Game of Life

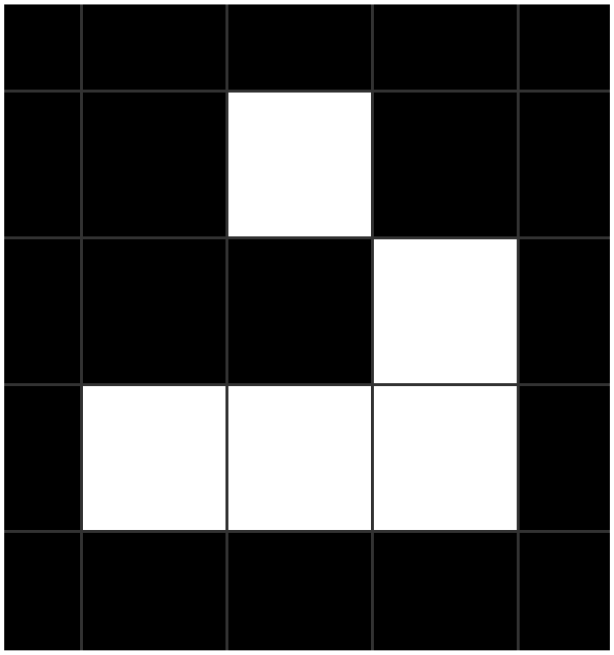
Kenties tunnetuin soluautomaatti on John Conwayn Life -peli. Life -pelissä soluilla on kaksi eri tilaa: elävä ja kuollut. Tilanmuutosfunktio on seuraavanlainen:

- Sellainen kuollut solu, jolla on täsmälleen kolme elossa olevaa naapuria (Mooren naapurusto), muuttuu eläväksi.
- Sellainen elävä solu, jolla on täsmälleen 2 tai 3 elossa olevaa naapuria, säilyy elossa.

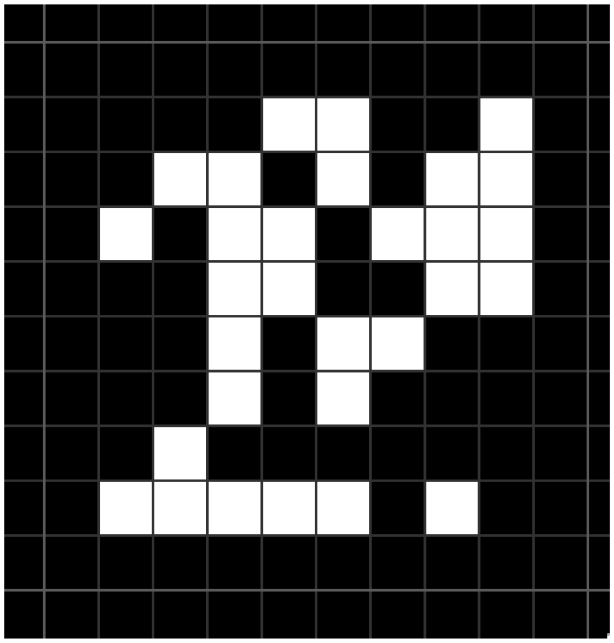
- Kaikki muut solut kuolevat tai pysyvät kuolleina.

Näillä yksinkertaisilla säännöillä syntyy varsin monimutkaisia rakenteita. Eräs merkittävistä on liukuri, eli glider, joka muuttuu ja liikkuu ennalta-arvattavasti. Neljän sukupolven välein liukuri liikkuu yhden askeleen eteenpäin sekä x- että y-akselin suhteen. Suunta määräytyy liukurin asennon perusteella. Kuvan 3.4 liukuri liikkuu oikealle ja alas. Liukuri on siitä merkittävä, että niitä syntyy helposti muiden solujen vaikutuksesta ja liukurien keskinäisistä törmäyksessä voi syntyä toisenlaisia rakenteita. Tämä prosessi tunnetaan nimellä liukurisynteesi [37]. Koska liukuri liikkuu ennalta-arvattavalla tavalla, sen avulla voidaan siirtää informaatiota paikasta toiseen. Life-pelissä voidaan rakentaa Turing-täydellisiä rakenteita [33].

Esimerkki Life-pelin simulaatiosta on annettu kuvissa 3.5 ja 3.6. Ensimmäisenä mainitussa on satunnaisesti valittu alkutilanne ja jälkimmäisessä sama tilanne 101 sukupolven kuluttua. Jälkimmäiseen kuvaan on merkattu eri väreillä tiettyjä tunnettuja muotoja, mutta varsinaisessa simulaatiossa kaikki elävät solut ovat samassa tilassa, eikä niillä ole erityisiä värejä.

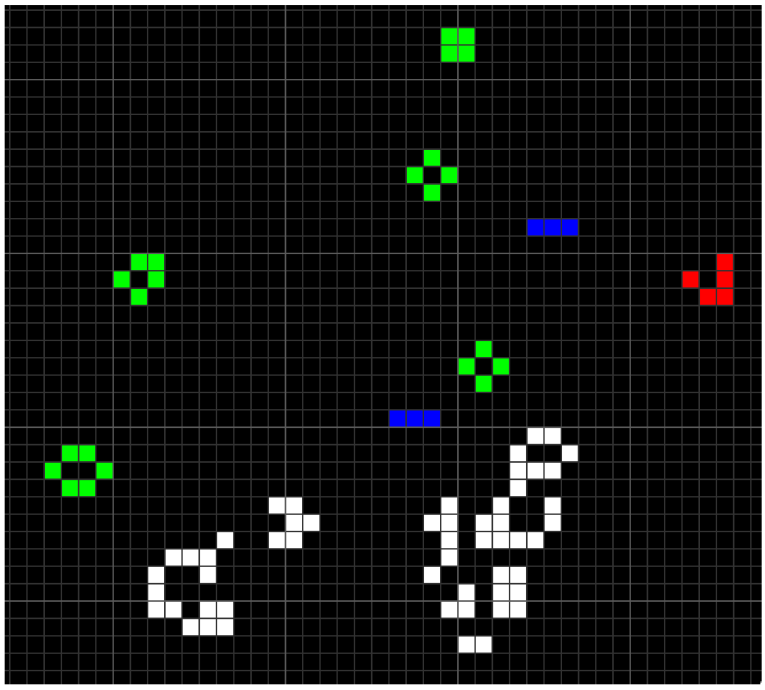


Kuva 3.4: Liukuri eli glider

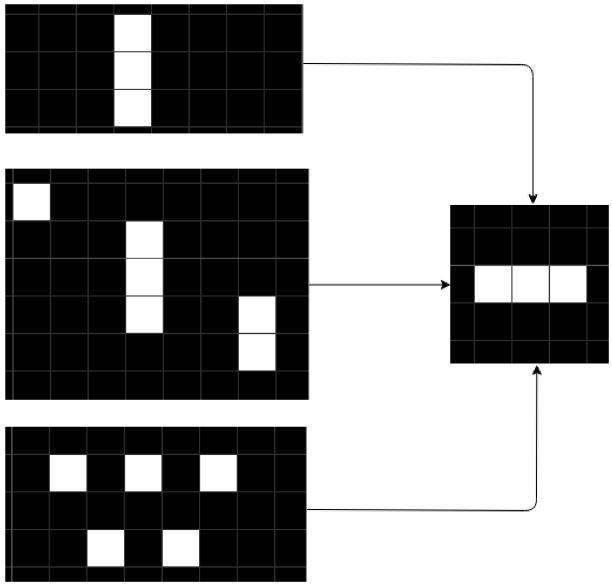


Kuva 3.5: Sattumanvaraisesti valittu alkutilanne

Life-peli on kääntymätön. Tarkastellaan esimerkiksi sellaista Life -soluautomaattia, jossa kaikki solut ovat kuolleita. Tällä tilalla on äärettömän monta edellistä tilaa: mikä tahansa yksittäinen solu on saattanut olla elävä edellisen sukupolven aikana. Kuvassa 3.7 on toinen esimerkki useista eri tiloista, jotka johtavat samaan tilaan.



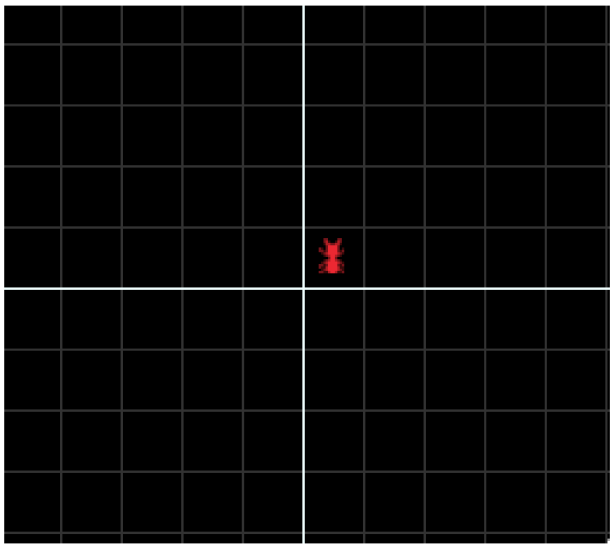
Kuva 3.6: Kuvan 3.5 alkutilanteen perusteella tehty simulaatio 101 sukupolven kuluttua. Prosessissa on syntynyt staattisia rakenteita (vihreät), sykkiviä rakenteita (siniset) ja yksi liukuri (punainen). Lisäksi simulaatio etenee vielä kaaottisesti (valkoiset solut).



Kuva 3.7: Life -peliä simuloidessa jokainen vasemmalla oleva tila on yhden sukupolven päässä oikealla olevasta tilasta. Pelkästään oikealla olevan tilan perusteella ei voida sanoa, mistä tilasta siihen on päästy.

Luku 4

Langtonin muurahainen



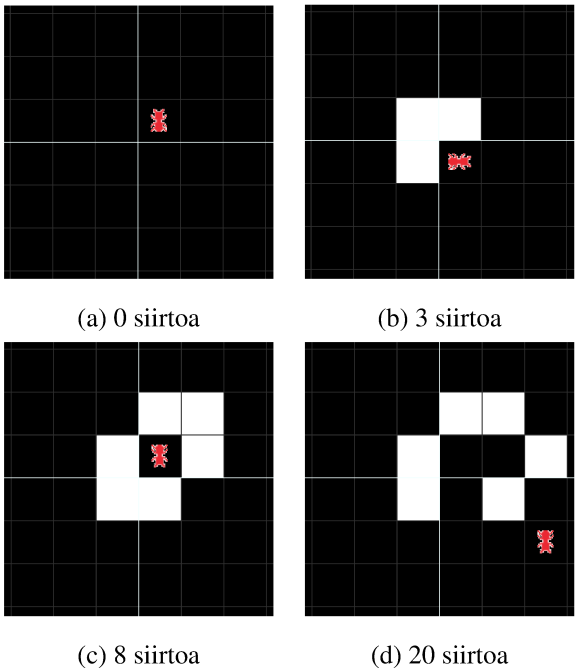
Kuva 4.1: Muurahainen alkutilassaan tyhjällä ruudukolla

Langtonin muurahainen on kaksiulotteinen soluautomaatti tai Turingin kone, jonka kuvasi ensimmäisen kerran Chris Langton vuonna 1986.[38] Langtonin muurahainen liikkuu kaksiulotteisella ruudukolla kahdella eri säännöllä:

Saapuessaan mustaan ruutuun, muurahainen kääntyy 90 astetta **vasemmalle**, vaihtaa ruudun värin valkoiseksi ja liikkuu yhden ruudun eteenpäin.

Saapuessaan valkoiseen ruutuun, muurahainen kääntyy 90 astetta **oikealle**, vaihtaa ruu-

dun värin mustaksi ja liikkuu yhden ruudun eteenpäin. Yhtä kääntymisen ja liikkumisen yhdistelmää kutsutaan *siirroksi*. Muurahaisen ensimmäisiä siirtoja on kuvassa 4.2

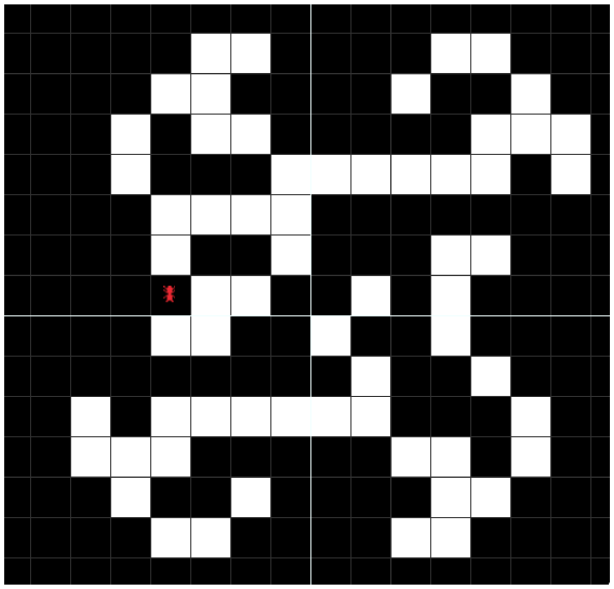


(a) 0 siirtoa (b) 3 siirtoa

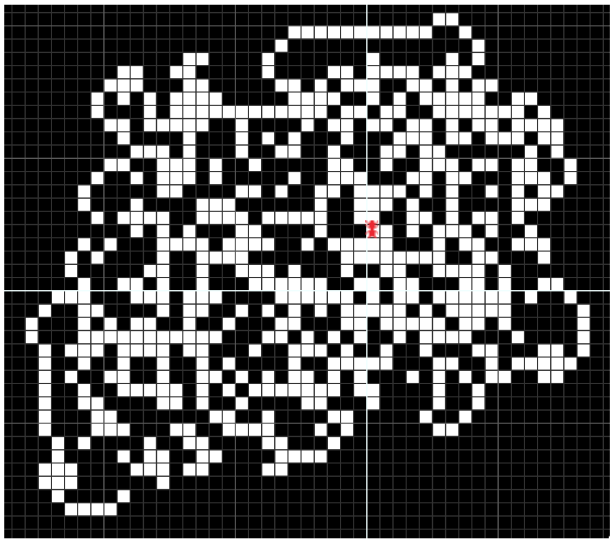
(c) 8 siirtoa (d) 20 siirtoa

Kuva 4.2: Muurahaisen ensimmäisiä siirtoja.

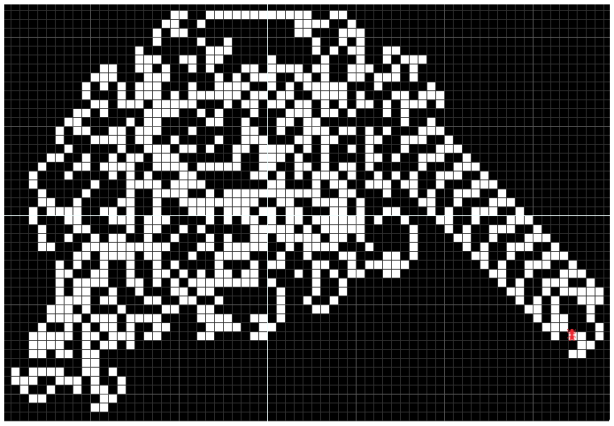
Edellä mainittuja sääntöjä seuraamalla tyhjältä ruudukolta aloittavan muurahaisen käyttäytymisessä on havaittavissa kolme eri vaihetta. Aluksi se tekee yksinkertaista symmetristä kuviota muutaman sadan siirron ajan (kuva 4.3). Tämän jälkeen symmetrisyys häviää ja liikkuminen muuttuu kaaottisemmaksi (kuva 4.4). Lopulta muurahainen ajautuu liikkumaan 104 siirron mittaista jaksoa, *valtatietä* (engl. *highway*) (kuva 4.5). Valtatietä tehdessä muurahainen toistaa samoja siirtoja ja ajautuu jatkuvasti eteenpäin. Äärettömällä ruudukolla muurahainen käy jatkuvasti uusissa ruuduissa [39, 40], mutta valtatien alkamista kaikista mahdollisista alkutiloista ei ole pystytty todistamaan. Kuitenkin kaikista kokeilluista alkutiloista muurahainen ajautui aina tekemään valtatietä [41].



Kuva 4.3: 400 siirron jälkeen muurahainen on tehnyt yksinkertaisia symmetrisiä kuvioita.



Kuva 4.4: 8000 siirron jälkeen muurahainen on muuttunut selvästi kaaottisemmaksi.



Kuva 4.5: 11000 siirron jälkeen muurahaisen tekemä toistuva kuvio, eli valtatie erottuu selkeästi oikealla alhaalla.

4.1 Merkinnät

Sijainti ja asento

Määritellään, että ruudukkoa käsitellään xy-koordinaatistona. Näin tietty ruutu voidaan esittää koordinaatteina `(x,y)`. Muurahainen on aina täsmälleen yhdessä ruudussa, mutta sijainnin lisäksi se on tietyssä asennossa. Asento voi olla ylös (u)[1], alas (d), vasemmalle (l) tai oikealle (r). Jos muurahainen osoittaa ylöspäin ja se sijaitsee ruudussa `(0,0)`, merkitään muurahainen notaatiolla `(0,0,u)`. Kun puhutaan samaan aikaan sekä asennosta että suunnasta, voidaan käyttää termiä *paikka*.

Säännöt

Jokaisella ruudulla on aina jokin tila. Määritellään, että ruutujen tilat esitetään numeroina. Aiempien esimerkkien musta ruutu on tilassa `0` ja valkoinen ruutu tilassa `1`. Muurahaisen kääntymistä vasemmalle merkataan kirjaimella **L** ja kääntymistä oikealle merkataan kirjaimella **R**. Alkuperäinen Langtonin muurahainen kääntyy tilassa `0` vasemmalle ja tilassa `1` oikealle. Tämän muurahaisen säännöt ovat siis **LR**. Vaihtoehtoisesti tilan voi esittää

[1]Lyhenteet ovat peräisin sanojen englanninkielisistä vastineista.

yksikäsitteisesti binääri- tai desimaalilukuna. L -kirjain merkkaa ykköstä ja R merkkaa nollaa, jolloin **LR** = 10_b tai desimaalilukuna 2.

Merkkijonoina esitetyille säännöille on määritelty toisto operaatio (*) ja katenaatio (+). Esimerkiksi sääntö **R + LLR** on sama kuin **RLLR** ja sääntö **RL*5** on **RLRLRLRLRL**.

4.2 N-tilainen muunnelma

Ruudukon tilojen määrän ei tarvitse rajoittua kahteen. Ruuduilla voi olla n tilaa ja jokaiselle ruudun tilalle määritellään yksi kääntymissääntö muurahaista varten. Liikkuessaan muurahainen muuntaa ruudun tilan aina seuraavaan tilaan, kasvattaen tilan numeerista arvoa yhdellä. Yleisessä muodossa n-tilaisen ruudun tilat ovat siis $0, 1, 2, ..., n-2, n-1$ ja muurahainen vaihtaa niitä syklisesti niin, että tilasta 0 siirrytään tilaan 1 ja lopulta tilasta $n-1$ siirrytään takaisin tilaan 0. Näitä n-tilaisia muurahaisia ovat määritelleet mm. D. Gale *et al.* [42]. Esimerkki 5-tilaisen muurahaisen säännöistä on taulukossa 4.1. Kun tiloja on käytössä enemmän, muurahainen piirtää selkeästi erilaisia kuvioita verrattuna tavalliseen **LR** -muurahaiseen. Muutama esimerkki on esitetty kuvissa 4.6, 4.7 ja 4.8.

Ruudun tila	0	1	2	3	4
Kääntymissuunta	L	R	L	R	R

Taulukko 4.1: Esimerkkikaavio 5-tilaiselle muurahaiselle, säännöillä **LRLRR**

4.3 N-ulotteinen muunnelma

Muurahaista on mahdollista simuloida myös N-ulotteisessa avaruudessa, jota ovat tutkineet mm. Dorbec *et al.* [43]. Eräs havainto oli valtatien kaltaisten kuvioiden syntyminen myös N-ulotteisessa avaruudessa. Dorbec *et al.* kutsuvat kolmiulotteisia muurahaisia Langtonin kärpäsiksi (engl. *Langton's fly*).

4.4 Yleinen muoto

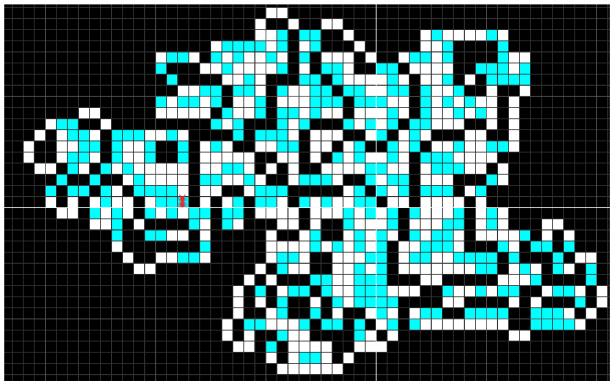
Turmite[44] on Langtonin muurahaisen kaltainen soluautomaatti, mutta muurahaisesta poiketen turmitella on myös oma sisäinen tilansa. Sisäinen tila voi muuttua ja tämä muuttaa turmiten ominaisuuksia. Turmite saattaa kääntyä mustalla ruudulla vasemmalle, mutta sen sisäinen tila voi muuttua ja uuden tilansa johdosta turmite kääntyykin seuraavaksi mustassa ruudussa oikealle. Turmitella on myös mahdollisuus olla kääntymättä ollenkaan, tai kääntyä 180 astetta kerralla. Lisäksi turmite saattaa olla vaihtamatta ruutujen värejä. Turmiten toiminnot voidaan esittää tilanmuutoskaaviolla (engl. *state transition table*), joka on esillä taulukossa 4.2. Turmiten kääntyminen määritellään samoilla kirjaimilla kuin muurahainen, mutta lisäksi määritellään kaksi muuta kääntymistapaa: N (ei käännöstä, (engl. *no turn*)) ja U, eli 180 asteen "U-käännös". Lisäksi tilat eivät etene syklissä, vaan turmite kirjoittaa ruutuun sen tilan, joka tilanmuutoskaaviossa määritellään.

		Ruudun tila					
		0			**1**		
		Ruudun uusi tila	**Käännös**	**Oma uusi tila**	**Ruudun uusi tila**	**Käännös**	**Oma uusi tila**
Oma tila	**0**	1	R	0	1	R	1
	1	0	N	0	0	N	1

Taulukko 4.2: Esimerkki turmiten tilanmuutoskaaviosta. Tällä turmitella on kaksi tilaa ja se liikkuu ruudukolla, jolla on myös kaksi tilaa.

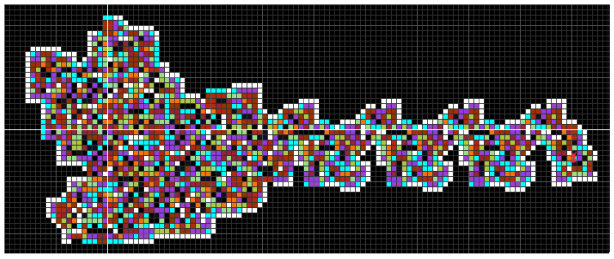
Myös Langtonin muurahainen on esitettävissä samanlaisella tilanmuutoskaaviolla. Muurahainen on käytännössä turmiten erikoistapaus, jossa sisäinen tila ei muutu ollenkaan tai sitä ei varsinaisesti ole, käännös tehdään aina vasemmalle tai oikealle ja ruutujen väriä muutetaan aina samalla tavalla.

Turmiten on mahdollista ajautua sykliseen tilaan, josta se ei pääse pois. Triviaalitapauksessa turmite päätyy kahteen vierekkäiseen ruutuun, joissa käännytään 180 astetta, eikä ruutujen väriä muuteta. Näin Turmite jää liikkumaan näiden kahden ruudun välillä loputtomasti. Turmite ei myöskään ole kääntyvä, paitsi tietyillä säännöillä. [44]



Kuva 4.6: Kolmetilainen muurahainen **LRL** 10000 siirron jälkeen.

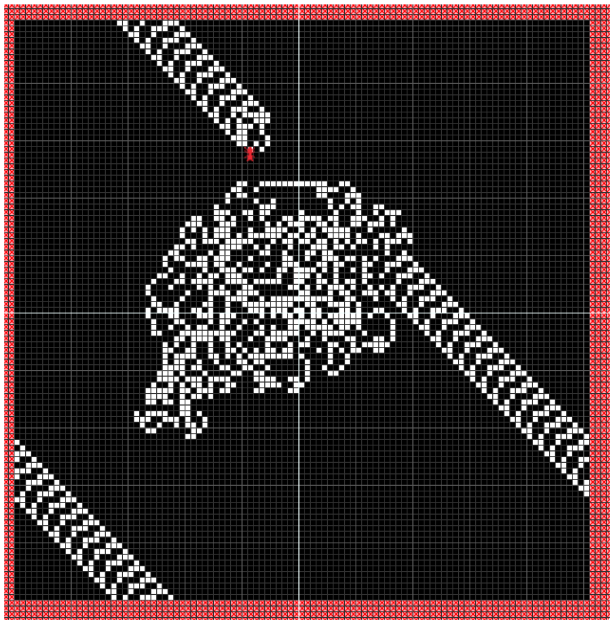
Kuva 4.7: **RRLLLRLLLRRR**(3143) siirron 50000 jälkeen. Tämä muurahainen tekee alaspäin liikkuvaa ja kasvavaa kolmiota.



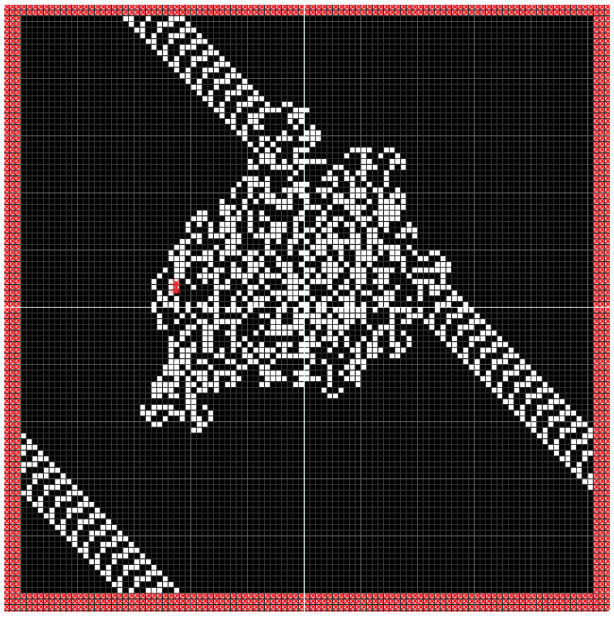
Kuva 4.8: **RRLLLRLRLRRL**(3158) 36000 siirron jälkeen, valtatien yhden jakson pituus on 1968 siirtoa.

4.5 Käyttäytyminen äärellisellä ruudukolla

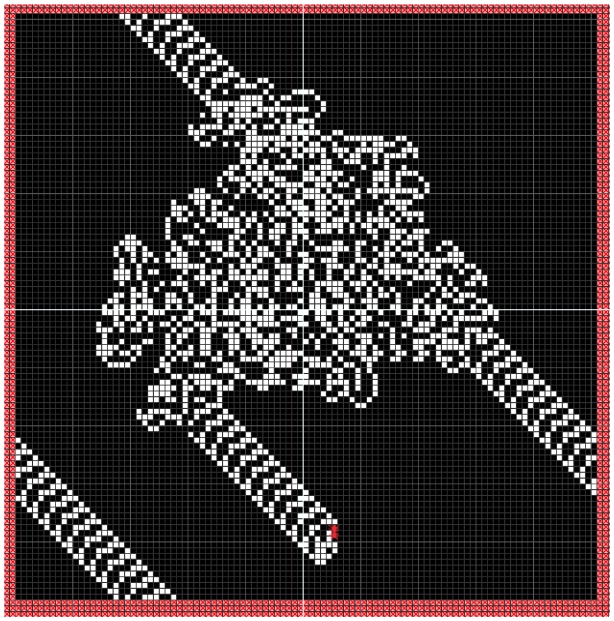
Aiemmissa esimerkeissä oletettiin, että muurahainen kykenee liikkumaan rajattomasti mihin suuntaan tahansa. Muurahaista on mahdollista simuloida myös äärellisellä alueella, mutta tällöin tulee määritellä, mitä tapahtuu muurahaisen mennessä reunan yli. Eräs mahdollisuus on, että rajan yli mennessään muurahainen siirtyy alueen vastakkaiselle puolelle. Koska muurahaisen tutkima alue kasvaa jatkuvasti, rajatulla pinnalla mahdollinen valtatie päättyy lopulta siihen, että muurahainen liikkuu takaisin ruutuihin, joissa se on jo käynyt. Tarkastellaan esimerkkinä muurahaista **LR** rajatulla $100 \cdot 100$ kokoisella alueella. Muurahainen aloittaa täysin tyhjältä ruudukolta paikasta `(0,0,u)`. Kuvassa 4.9 muurahainen on liikkunut 14000 siirtoa, ajautunut tekemään valtatietä ja päätynyt monta kertaa reunan yli, lähestyen aiemmin jälleen aiemmin tutkittuja ruutuja. Kuvassa 4.10 muurahainen on liikkunut 18000 siirtoa ja se on päätynyt takaisin aiemmin tutkittuihin ruutuihin. Muurahaisen liikehdintä on jälleen kaaottista. Kuvassa 4.11 muurahainen on aloittanut jälleen uuden valtatien. Jos tätä jatketaan tarpeeksi pitkään, koko alue täyttyy mielivaltaisilla tiloilla, eikä aiemmin havaittua toistuvaa kuviota enää synny.



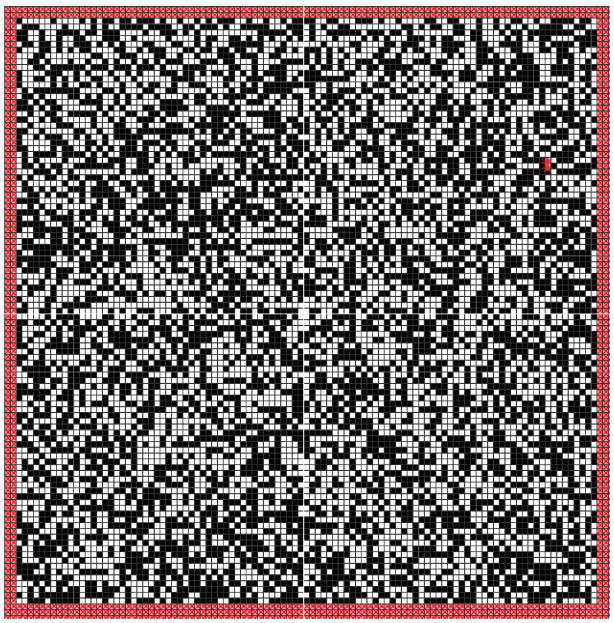
Kuva 4.9: Muurahainen on liikkunut 14000 siirtoa rajatulla alueella. Punaiset ruudut merkkaavat alueen rajoja. Muurahainen on liikkunut rajojen yli ja siirtynyt vastakkaiselle puolelle.



Kuva 4.10: 18000 siirron jälkeen muurahainen on palannut kaaottiseen liikehdintään törmättyään aiemmin tutkittuihin ruutuihin.



Kuva 4.11: 26000 siirron jälkeen muurahainen on päätynyt uudelleen tekemään valtatietä.



Kuva 4.12: Tilanne 400000 siirron jälkeen. Koko alue on täyttynyt, eikä erityisiä järjestelmällisiä rakenteita ole havaittavissa.

4.5.1 Tilojen määrä äärellisellä ruudukolla

Äärellisellä ruudukolla koko ruudukon tiloja on rajallinen määrä. Ruudukon tilalla tarkoitetaan kaikkien sen ruutujen yhteistä tilaa. Tämä määrä on enintään yksittäisen ruudun tilojen määrä potenssiin ruutujen määrä. Valitaan esimerkiksi ruudukon kooksi $3 \cdot 3$ ja muurahaisen säännöiksi **LRL**. Ruutuja on siis yhteensä $3 \cdot 3 = 9$ ja kukin voi olla kolmessa eri tilassa. Tämä tarkoittaa sitä, että ruudukolla on yhteensä $3^9 = 19683$ eri tilaa.

Määritelmä 4.5.1. Äärellisellä ruudukolla mahdollisten tilojen määrä X on enintään $X^{n \cdot m}$, jossa n ja m ovat ruudukon sivujen pituudet ja X on yksittäisen ruudun tilojen määrä.

Koko systeemin tilalla tarkoitetaan ruudukon ja muurahaisen yhteistä tilaa. Tällöin tilaan lasketaan ruudukon tilojen lisäksi muurahaisen sijainti ja suunta. Ruudukolla, jonka koko on $3 \cdot 3$ muurahainen voi olla $3 \cdot 3$ eri sijainnissa, neljässä eri asennossa. Täten systeemin tilojen yhteenlaskettu määrä on $3^9 \cdot 3 \cdot 3 \cdot 4 = 708588$.

Määritelmä 4.5.2. Systeemin tilojen määrä on ruudukon kaikkien tilojen määrä $X^{n \cdot m}$ kerrottuna muurahaisen kaikilla mahdollisilla sijainneilla ja asennoilla. Merkataan systeemin tilaa kirjaimella Z. Muurahainen voi olla $n \cdot m$ eri paikassa ja neljässä eri asennossa. Systeemin tilojen määrä $Z = X^{nm} \cdot 4nm$

4.5.2 Tilojen saavutettavuus

Kaikki systeemin tai ruudukon tilat eivät välttämättä ole saavutettavissa. Muurahainen liikkuu vain viereisiin ruutuihin ja muuttaa aina yhden ruudun tilaa liikkuessaan. Kokeiltaessa selvisi, että aloittaessa tyhjältä $3 \cdot 3$ ruudukolta käyttäen sääntöä **LRL**, ruudukko palaa tyhjään alkutilaansa 93 siirron jälkeen. Muurahaisen paikka on nyt `(1,0,r)`, joten systeemi ei ole vielä alkutilassaan. Kuitenkin yhteensä 372 siirron jälkeen muurahainen palaa alkupaikkaan `(0,0,u)` ja kaikki ruudut ovat jälleen tyhjiä, joten systeemi on palannut alkutilaansa. Näin ollen kaikki seuraavat tilat on nähty jo aikaisemmin. Tämä jää

huomattavan paljon määritelmien 4.5.1 ja 4.5.2 tilojen kokonaismääristä X ja Z. Lisäksi tarkastellessa kaikkia muurahaisen saavuttamia ruutuja ja asentoja näissä ruuduissa huomataan, ettei muurahainen saavu koskaan ruutuun `(0,0)` muussa asennossa kuin d tai u. Kokeilussa selvisi, että tilojen määrää tai ruudukon kokoa kasvattaessa ongelma käy harvinaisemmaksi tai katoaa kokonaan.

Havainto 4.5.1. Äärellisellä ruudukolla systeemi ei aina saavuta kaikkia eri tiloja.

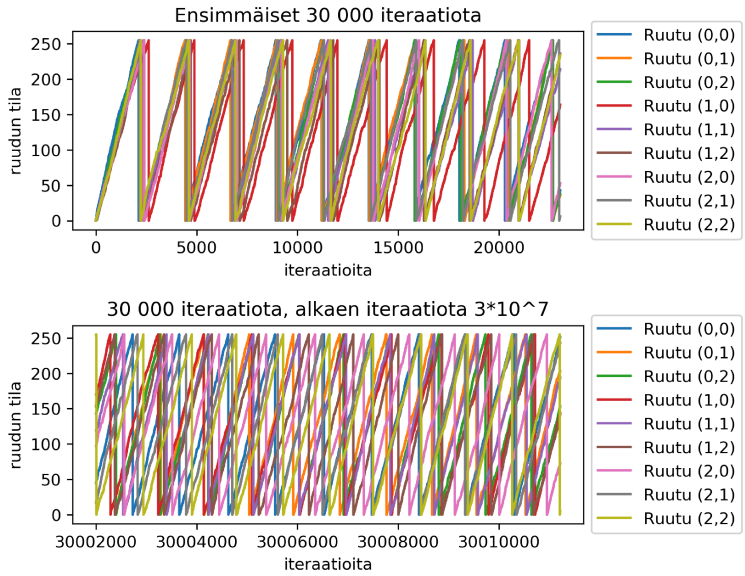
Havainto 4.5.2. Äärellisellä ruudukolla simuloituna muurahainen ei aina saavuta jokaista mahdollista sijaintia kaikissa mahdollista asennoissa.

4.5.3 Ruutujen tilojen riippuvuus toisistaan

Valitaan satunnaiset säännöt 256-tilaiselle muurahaiselle ja ruudukon kooksi 3^2. Kun muurahainen on liikkunut 3^2k kertaa, huomataan, että riittävän pienillä k arvoilla jokaisen ruudun tila on melko lähellä arvoa k,. Tämä johtuu siitä, että pienellä alueella ei ole kovinkaan montaa ruutua, joihin muurahainen voi mennä. Koska säännöt ovat pitkät ja satunnaisesti valitut, muurahaisen liikehdintä on melko satunnaista ja näin ollen se käy kaikissa ruuduissa lähes yhtä usein. Lopulta ruudut saavuttavat arvon 255 ja palaavat takaisin arvoon 0. Tämä tapahtuu kaikille ruuduille hyvin pienellä aikavälillä ja ruutujen arvot pysyvät melko kauan lähellä toisiaan. Kun muurahainen on liikkunut paljon enemmän, arvot hajautuvat kauemmas toisistaan. Kuvassa 4.13 on esimerkkisimulaation jokaisen ruudun numeerinen tila muurahaisen siirtomäärän funktiona. Viimeistään noin 30 miljoonan siirron jälkeen ruutujen tilat ovat erkaantuneet toisistaan.

4.5.4 Pariteetti

Parillisen leveyden ruudukolla muurahainen ei pääse kaikkiin sijainteihin kaikissa asennoissa, mikä johtuu muurahaisen liikkumistavasta. Muurahaisen eräs ominaisuus on suunnan muuttuminen aina liikkuessa. Muurahaisen ei ole mahdollista kääntyä liikkumatta,



Kuva 4.13: Yhdeksän solun tilat ajan funktiona.

tai liikkua kääntymättä. Paikasta `(0,0,u)` muurahainen ei voi mitenkään liikkua suoraa eteenpäin paikkaan `(0,1,u)` koska sen tulee ensin kääntyä. Muurahainen voi siis liikkua esimerkiksi paikkaan `(1,0,r)`. Kääntymisestä ja samaan aikaan siirtymisestä seuraa, että parillisella ruudukolla paikasta `(0,0,u)` aloittava muurahainen ei voi olla milloinkaan saavuttaa paikkaa `(0,0,r|l)`. Siirtyessään x-akselin suuntaan jokaista siirtoa varten tarvitaan kaksi liikkumista: jos muurahainen osoittaa x akselin suuntaan, seuraavaksi muurahainen liikkuu y-akselin suuntaisesti, koska sen täytyy kääntyä vasemmalle tai oikealle. Tämän jälkeen y-akselin suuntaan osoittava muurahainen voi siirtyä x-akselin suuntaisesti, jonka jälkeen sen asento on edelleen sama kuin se oli alun perin. Tällöin muurahainen on joka toisessa ruudussa aina ylös tai alaspäin ja joka toisessa vasemmalle tai oikealle.

4.5.5 Alkutilaan palaaminen

Alkutilaan palaamisella tarkoitetaan sitä, että tietyn siirtomäärän jälkeen systeemi on palannut alkutilaansa niin, että muurahainen ja kaikki ruudut ovat samassa tilassa kuin aloittaessa. Esimerkiksi aliluvussa 4.5.2 mainittu **LRL** muurahainen palasi alkutilaan jo 372

siirron jälkeen.

Lemma 4.5.1. *Muurahainen ei voi ajautua rajatulla alueella sellaiseen päättymättömään sykliin, jonka pituus on pienempi kuin koko systeemin tilojen määrä.*

Todistus. Muurahaisella on seuraavat ominaisuudet:

- Systeemi on deterministinen, eli jokaisella tilalla on tarkasti määritelty yksikäsitteinen seuraava tila.
- Systeemi on kääntyvä, eli jokaisella tilalla on täsmälleen yksi mahdollinen edellinen tila.

Valtatien tekeminen vaatii, että muurahainen toistaa samoja siirtoja jatkuvasti. Rajallisella alueella muurahainen palaa lopulta takaisin ruutuihin, joissa se on jo käynyt ja valtatie pysähtyy. Toisenlainen päättymätön sykli olisi sellainen, että muurahainen tekee tiettyjä siirtoja, pysyen jatkuvasti tiettyjen ruutujen sisällä, kulkien esimerkiksi ympyrää rajatun alueen sisällä. Kuvataan Z-tilaisen systeemin peräkkäisiä tiloja $s_0, s_1, s_2, \ldots, s_{Z-1}$. Jotta muurahainen tekisi päättymätöntä sykliä, jonka pituus on pienempi kuin systeemin tilojen määrä, täytyy olla olemassa tila s_y josta systeemi siirtyy aiemmin nähtyyn tilaan s_x, eli $x < y$. Systeemin tilat ovat siis järjestyksessä

$$s_0, s_1, \ldots, s_{x-1}, s_x, \ldots, s_y, s_x \ldots, s_{Z-1}$$

Jos tämä on mahdollista, tilalla s_x on kaksi erilaista edellistä tilaa, s_y ja s_{x-1}. Tällöin esitetty systeemi ei ole Langtonin muurahainen, sillä muurahainen ehtona on yksikäsitteinen edeltäjä. Tämä tarkoittaa, että s_y jälkeen tuleva tila voi olla s_x ainoastaan, jos $x = 0$ ja $y = Z - 1$.

□

Lause 4.5.2. *Rajallisella ruudukolla systeemi palaa väistämättä alkutilaansa.*

Todistus. Koska muurahainen ei voi ajautua valtatiehen tai muuhun päättymättömään silmukkaan rajallisella alueella (4.5.1), jokainen systeemin saavuttama tila on aina erilainen kuin aiemmat tilat. Koska systeemillä tilojen määrä Z on äärellinen, systeemi palaa väistämättä alkutilaansa enintään Z siirron jälkeen. □

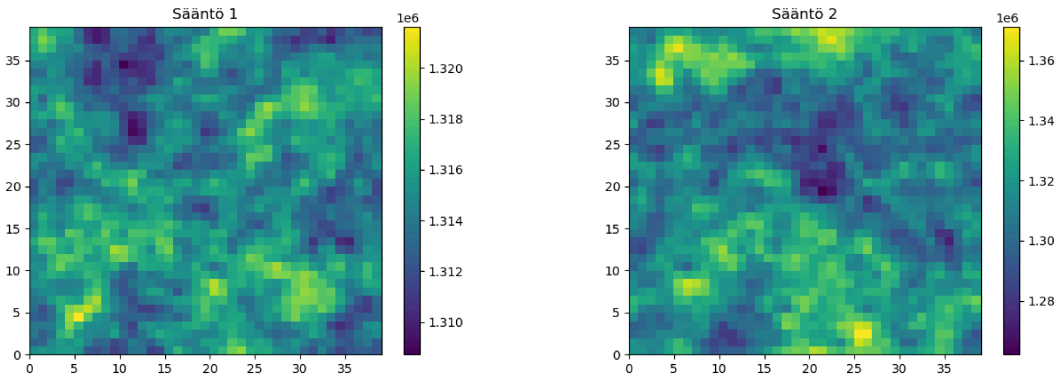
4.5.6 Liikehdinnän satunnaisuus

Vaikka muurahainen liikkuu melko arvaamattomasti, kaikki säännöt eivät ole yhdenvertaisia. Tarkastellaan kolmea erilaista sääntöä:

1. Sääntö-1 (Liite A)
2. **L*256 + R**
3. **RRRRRRLL*32**

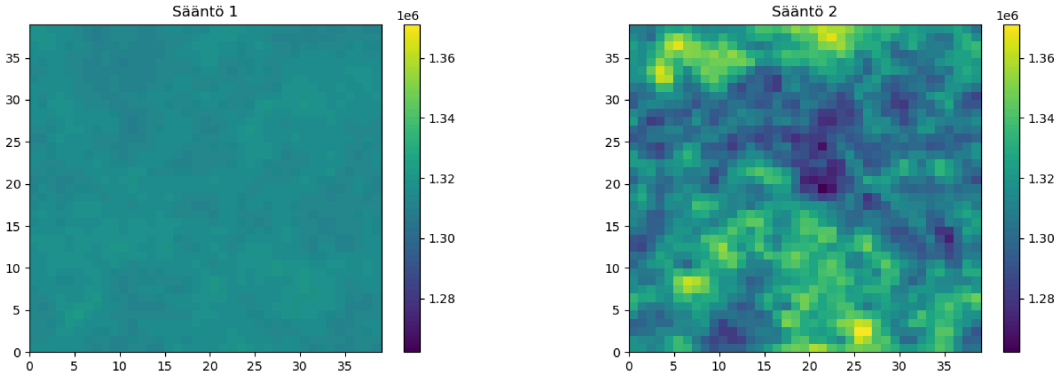
Simuloidaan kutakin sääntöä tyhjällä 39^2 kokoisella ruudukolla kaksi miljardia siirtoa ja tarkastellaan muurahaisen liikkeistä muodostettua lämpökarttaa. Kuva 4.14 esittää sääntöjen 1 ja 2 lämpökarttoja. Lämpökartat ovat melko satunnaisen näköisiä, mutta merkittävin havainto on, että säännöllä 1 arvoalue on paljon pienempi. Kuvassa 4.15 on samat kuvat, mutta nyt myös lämpökarttojen arvoalueet ovat samat. Säännöillä 1 muurahainen käy siis tasaisemmin alueen kaikissa ruuduissa.

Sääntö 3 puolestaan käyttäytyy poikkeavalla tavalla. Tyhjältä ruudukolta aloittaessa muurahainen käy paljon useammin keskimmäisessä ruudussa ja se lähistöllä, eikä se ehdi saavuttaa alueen reunoja annetulla aikavälillä. Sääntö 3 on kuvassa 4.16. Kuvan väriasteikko on logaritminen. Jos ruudukon alkuarvot satunnaistetaan, myös sääntö 3 liikkuu satunnaisesti, kuten nähdään kuvasta 4.17. Arvoalue on tällöin hyvin lähellä sääntöä 2.



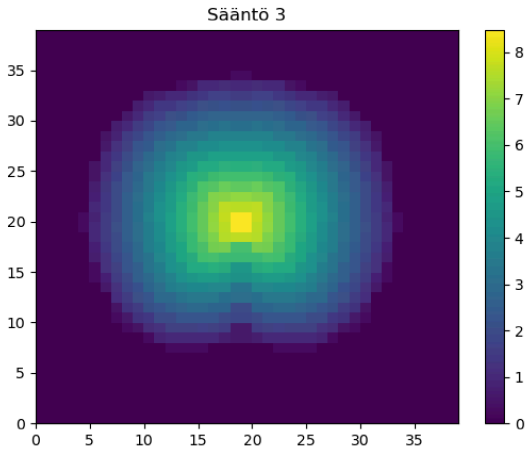
(a) Sääntö 1 (b) Sääntö 2

Kuva 4.14: Lämpökartat säännölle 1 ja 2

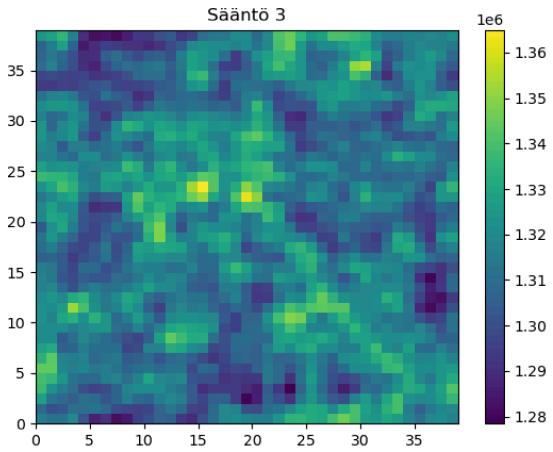


(a) Sääntö 1 (b) Sääntö 2

Kuva 4.15: Lämpökartat säännölle 1 ja 2, väriasteikot suhteutettuna toisiinsa.



Kuva 4.16: Lämpökartta säännölle 3. Huomaa, että asteikko on logaritminen.



Kuva 4.17: Lämpökartta säännölle 3, kun alkutila koostuu satunnaisista ruuduista.

4.6 Muurahaisen esitys soluautomaattina

Muurahaisen voidaan katsoa olevan Turingin kone, sillä se on käytännössä osoitin, joka liikkuu nauhalla. Muurahainen lukee nauhalta tietoja, kirjoittaa tietoja ja liikkuu niiden perusteella. Muurahainen voidaan esittää myös soluautomaattina. Tällöin ruuduille tarvitaan useampia tiloja niin, että jokaista ruudun väriä kohden on olemassa 4 muuta väriä,

jotka kuvaavat muurahaisen kutakin sijaintia ja asentoa, kun alla oleva ruutu on tietyn värinen.[39] Kun pohditaan käytännön toteutusta, Langtonin muurahaista kannattaa ehdottomasti simuloida Turingin koneena. Jokaisella siirrolla lukupää eli muurahainen lukee allaan olevan tilan ja liikkuu sen mukaisesti. Yksi tällainen operaatio tapahtuu vakioajassa $O(1)$. Jos Langtonin muurahaista simuloitaisiin soluautomaattina, tulisi soluautomaatille ominaiseen tapaan jokaisen solun tila päivittää uutta sukupolvea laskettaessa. Tällöin jokaisen sukupolven laskeminen olisi aikavaativuusluokkaa $O(n)$ jossa n on solujen määrä. Kirjallisuudessa Langtonin muurahaiseen viitataan usein soluautomaattina, vaikka se onkin helpompi kuvata Turingin koneena toimintaperiaatteensa perusteella.

Luku 5

Soluautomaatit salausalgoritmina

Soluautomaattien avulla on tehty useita onnistuneita salausalgoritmeja. Tässä luvussa pohditaan ensin soluautomaatilta vaadittuja ominaisuuksia, minkä jälkeen tutustutaan alan kirjallisuuteen.

5.1 Soluautomaatilta vaaditut ominaisuudet

Jotta soluautomaattia voidaan käyttää salauksessa, soluautomaatin tulee olla kääntyvä, jotta salaus voidaan purkaa. Esimerkiksi Life -peli ei ole kääntyvä, joten on epätodennäköistä, että se toimisi sellaisenaan salausalgoritmin osana. Kuitenkin esimerkiksi M.V. Jahan *et al.* kertovat tutkimuksessaan Life -pelin avulla toteutetusta salauksesta. Kyseessä on kuitenkin enemmänkin loogisten porttien rakentamista liukureista, eli Life -peliä käytetään abstraktina laskukoneena, kuten mitä tahansa muutakin Turing-täydellistä soluautomaattia voisi käyttää.[45]

Soluautomaatin tulisi myös olla helposti muunneltavissa, jotta siitä voidaan johtaa avaimia. Esimerkiksi alkeissoluautomaatilla on 256 erilaista sääntöä, jotka edustavat samaa soluautomaattia, mutta käyttäytyminen vaihtelee. Myös Langtonin muurahaisella on tämä ominaisuus. Muurahaisella voi olla n-pituiset säännöt ja tällöin erilaisia sääntöjä on 2^n. Vaihtoehtoisia sääntöjä voidaan käyttää mahdollisen salausavaimen perustana.

Lisäsi voidaan katsoa, että tietystä ratkeamattomasta piirteestä on hyötyä. Jos salausalgoritmi perustuu useiden sukupolvien peräkkäiseen simulaatioon, mutta kyseisen soluautomaatin tila k sukupolven kuluttua saadaan laskettua alkutilanteen perusteella vakioajassa, tai alkutila saadaan laskettua lopputilasta, salauksen purkaminen saattaa osoittautua liiankin helpoksi. Toisaalta siirtomäärän ollessa tuntematon, tästäkään ei välttämättä ole haittaa.

5.2 Aiempia toteutuksia

5.2.1 Yksiulotteinen soluautomaatti satunnaislukugeneraattorina

M. Tomassini *et al.* [46] käyttivät yksiulotteista soluautomaattia pseudosatunnaislukugeneraattorina, jota voi edelleen käyttää merkkisalakirjoituksen tekemiseen. Tutkimuksen mukaan alkeissoluautomaatin säännöt 30, 90,105,150 ja 165 ja näiden mikä tahansa yhdistelmä toimii parhaiten satunnaislukugeneraattorina. Yhdistelmässä eri ruuduilla on eri säännöt. Soluautomaattia varten alustetaan äärellisen pituinen rivi soluja, joka kytketään molemmista päistä yhteen. Tämä täytetään satunnaisilla tiloilla, jotka muodostavat avaimen. Soluautomaattia simuloidaan laskemalla alkutilan perusteella uusia tiloja ja tämän soluautomaatin tuottamia bittejä käytetään salauksen tekoon. Bittejä luetaan ennalta valitusta sarakkeesta ja bitti kerrallaan suoritetaan XOR -operaatio selkotekstin biteille. Samankaltaista satunnaislukugeneraattoria ovat tutkineet myös F. Seredynski *et al.* [47]. Heidän mukaansa myös automaatit 86, 101 ja 153 tuottavat riittävän satunnaisia numeroita. Satunnaislukujen laadun arviointiin on käytetty niin kutsuttua Diehard-testiä (G. Marsaglia)[48, 49], joka mittaa satunnaislukujen entropiaa ja jakaumaa useilla eri tilastollisilla ja matemaattisilla menetelmillä. Molemmissa tutkimuksissa satunnaislukujen laatu on osoitettu riittäväksi salauksen tekemiseen. Kun valitaan solurivin pituudeksi p, erilaisia avaimia on käytettävissä 2^p.

Koska rajallisella solurivillä on äärellinen määrä tiloja, tiettyjen solujen tuottama sykli

palaa lopulta alkutilaansa ja alkaa toistaa samoja numeroita. Jos tämä sykli on liian lyhyt, salatekstiin vuotaa tilastollista informaatiota, mikä helpottaa salauksen murtamista. Aikaisemmassa tutkimuksessa S. Wolfram [50] tutki syklien pituuksia samaisilla soluriveillä. Suurin osa mahdollisista tiloista on samalla syklillä, mutta myös lyhyempiä syklejä löytyy tietyistä symmetrisistä alkutiloista. Täten avaimen valintaan tulee kiinnittää huomiota, että voidaan välttää sellaisia avaimia, jotka tuottavat liian pieniä syklejä. Lisäksi tutkimus osoittaa, että mitä suurempi solurivin pituus on, sitä pidempi pisimmästä syklistä tulee. Liian lyhyttä soluriviä ei siis kannata käyttää.

W. Meier *et al.* mukaan [51] Wolframin esittämä algoritmi on altis tunnetulle selkotekstihyökkäykselle. Jos tunnetaan selkoteksti ja sitä vastaava salateksti, voidaan yksikäsitteisesti laskea salaukseen käytetty bittijono. Koska salaus tehdään XOR-operaatiolla niin, että $selkoteksti \oplus bittijono = salateksti$, voidaan laskea myös $selkoteksti \oplus salateksti = bittijono$. Näin avaingeneraattorin tuottama bittijono selviää ja tällöin saadaan myös purettua samalla bittijonolla salatut viestit, jotka ovat yhtä pitkiä tai lyhyempiä kuin kyseinen bittijono. Jos viestit ovat pidempiä, täytyisi tietää varsinaisen avaingeneraattorin alkutila. Tätä voidaan koittaa selvittää avaimen avulla. Jos avaimen ensimmäinen bitti on esimerkiksi 1, valitaan edelliseksi sukupolveksi sellaiset bitit, joiden seuraava tila on 1 ja kokeillaan simuloida tätä eteenpäin ja katsoa, tuottaako soluautomaatti samat bitit kuin tunnetut avaingeneraattorin tuottamat bitit. Jos ei, vaihdetaan bittejä ja kokeillaan uudelleen. Aiemmin esitetty toteutus, jossa yksittäisen ruudun säännöt eivät ole ennalta tiedossa tekee avaimen laskemisesta haastavampaa.

5.2.2 Kuvansalaus Solautomaatin avulla

Kuvansalausta samankaltaisin menetelmin ovat tutkineet mm. X. Wang *et al.* [52] sekä M. Kumar *et al.* [53]. Molemmissa on käytetty differenssiyhtälöitä sekaannuksen luomiseksi ja Kumar *et al.* viittaavatkin Wang *et al.* käyttämiin nivottuihin logistisiin kuvauksiin (engl. *intertwining logistic map*). Wang *et al.* mukaan useamman toisistaan riippu-

van kuvauksen käyttäminen rajoittaa kuvausten huonojen ominaisuuksien ilmaantumista. Huonoja ominaisuuksia ovat mm. epätasaiset jakaumat ja stabiilit syklit.

Differenssiyhtälöiden ohella molemmissa tutkimuksissa käytetään soluautomaatteja erilaisilla tavoilla. Wang *et al.* käsittelevät pikselien yksittäisiä bittejä oman nimeämättömän soluautomaatin avulla niin, että bitti ja sen neljän naapuria von Neumannin naapurustossa vaikuttavat bitin uuteen arvoon. Jokaiselle naapurin ja solun oman tilan yhdistelmälle on määritelty tietty tilanmuutos. Algoritmissa tätä logistisen kuvauksen ja soluautomaattisukupolven yhdistelmää suoritetaan useita kertoja, kunnes salaus on valmis.

Kumar *et al.* puolestaan asettavat datan yhteen pitkään vektoriin ja suorittavat ensin sekoitteluoperaatioita kahdella eri tavalla käyttäen Wang *et al.* esittelemää logistista kuvausta, jonka avulla luodaan alkuperäisen kuvan perusteella bittijono, joka myöhemmin yhdistetään XOR-operaatiolla alkuperäiseen kuvaan. Vasta algoritmin lopuksi käytetään alkeissoluautomaattia sekoittamaan dataa kuvan tavumäärää vastaavalla iteraatiomäärällä.

Mahdollinen ongelma

Vaikka molemmat julkaisut toteavat algoritmin turvalliseksi, Kumar *et al.* antaa esimerkin, jossa salatekstistä, eli salatun kuvan keskeltä poistetaan pala ennen salauksen purkamista. Alkuperäinen kuva tulee lähes kokonaan näkyviin pieniä muutoksia lukuun ottamatta. Tämän perusteella vaikuttaisi, että algoritmi ei ole erityisen herkkä salatekstin muutokselle. Koska Wang *et al.* käyttää samaa logistista kuvausta, on mahdollista, että sama ongelma esiintyy myös heidän algoritmissaan. Toisaalta siinä käytetty soluautomaatti toimii eri tavalla ja salauksen aikana tehdään myös useampia kierroksia, joten on mahdollista, että tältä ongelmalta vältytään.

5.2.3 Kuvansalaus Langtonin muurahaisen avulla

Myös Langtonin muurahaista on käytetty salauksessa aikaisemmin. X. Wang *et al.* [52] ovat tehneet tutkimustyötä myös tämän parissa. Tässä algoritmissa logistisen kuvauksen tilalla on paloittainen lineaarikuvaus (engl. *PWLCM, Piece-wise chaotic linear map*), jonka avulla generoidaan pseudosatunnaisia arvoja. Salauksessa luodaan salattavan kuvan kokoinen ruudukko, joka täytetään edellisessä vaiheessa tuotetuilla satunnaisluvuilla. Lisäksi luodaan uusi tyhjä ruudukko salatulle kuvalle. Kun muurahainen liikkuu mustavalkoisella ruudukolla ja saapuu vakoiseen ruutuun, muurahaisen koordinaattia vastaavaan ruutuun asetetaan alkuperäisen kuvan ensimmäinen pikseli. Jos ruudussa on jo pikseli, ei tehdä mitään. Tätä jatketaan yhtä monta kertaa kun kuvassa on pikseleitä. Koska muurahainen käy lähes varmasti samoissa ruuduissa useita kertoja, kaikkia arvoja ei ehditä latoa uuteen ruudukkoon. Näiden siirtojen jälkeen käyttämättömät arvot ladotaan salattuun kuvaan. Tässä kohtaa julkaisu ei kuvaile tätä kovinkaan tarkasti, mutta voidaan olettaa, että täytetään tyhjät kohdat järjestyksessä. Lopulta käytetään vielä paloittaista lineaarikuvausta tuomaan diffuusiota.

Algoritmi esitetään toimivaksi erilaisin tilastollisin menetelmin, mutta tässäkin voisi miettiä, tuleeko lineaarikuvauksen mukana edellisen menetelmän ongelma, jossa salatekstin tietyn osan poistaminen ei vaikuttanut salauksen purkuun merkittävästi.

Luku 6

Langtonin muurahainen salausalgoritmina

Tässä luvussa esitetään tutkielmaa varten kehitetty toteutus soluautomaattipohjaisesta salauksesta. Aluksi esitetään yksinkertainen versio, minkä jälkeen pohditaan sen ongelmia ja mahdollisia ratkaisuja. Ratkaisujen pohjalta esitetään paranneltu versio, jota analysoidaan suorituskyvyn ja turvallisuuden näkökulmista.

Algoritmi poikkeaa edellisen luvun menetelmistä siten, että se käyttää ainoastaan Langtonin muurahaista sekä sekaannuksen, että diffuusion aiheuttamiseksi. Lisäksi luvussa 5 esitetty Langtonin muurahaista käyttävä menetelmä käsittely yksinomaan kuvien salaamista. Tässäkin luvussa suurin osa visualisoinneista tehdään kuvatiedostojen avulla, mutta algoritmi on laadittu yleiskäyttöiseksi, eikä se ota kantaa syötteeseen. Esiteltävä algoritmi on suunniteltu salaamaan mitä tahansa 8-bittisistä arvoista, eli tyypillisistä tavuista koostuvaa dataa. Datan pituus ei vaikuta salauksen toimintaan, sillä kyseessä on yleiskäyttöinen lohkosalausalgoritmi, jossa data jaetaan tietyn mittaisiin lohkoihin salauksen aikana.

Esimerkkejä varten on kolme kuvaa: tiikeri.png, laskuvarjo.png ja ruoka.png. Salausta ei tehdä koko kuvatiedostolle, vaan ainoastaan sen väridatalle, jolloin kuvatiedoston

rakenne säilyy ja salattua dataa voidaan edelleen tarkastella kuvana. Kuvissa näkyvään dataan ei ole lisätty mukaan viimeisen lohkon tasausta, tai muita myöhemmin selostettavia arvoja. Näin kuvatiedoston pituus ei muutu. Kuvat ovat $523 \cdot 523$ pikselin kokoisia ja jokainen pikseli ilmaistaan kolmella tavulla, jolloin salattavan datan pituus on $3 \cdot 523^2$. Alkuperäiset kuvat ja niiden esimerkkisalaukset esitetään myöhemmin kuvassa 6.4.

6.1 Algoritmin toimintaperiaate

Salausalgoritmi perustuu luvussa 4 esiteltyihin muurahaisen ominaisuuksiin. Algoritmi simuloi Langtonin muurahaista rajatulla ruudukolla ja ruudukon alkutilaksi asetetaan merkkejä selkotekstistä, yksi jokaiseen ruutuun. Tämä jälkeen muurahainen liikkuu ruudukolla, muuttaen ruudukossa olevien numeroiden tilaa. Valitaan salausta varten ensin seuraavat arvot:

Lohkon koko L**,** jonka tulee olla pariton positiivinen kokonaisluku $L = 2x + 1, x > 0, x \in \mathbb{N}$, jolloin muurahainen pystyy liikkumaan kaikkiin ruutuihin kaikissa asennoissa.

Siirtomäärä K**,** mahdollisimman suuri kokonaisluku K.

Siirtosääntö S_n**,** joka määrittelee n-tilaisen Langtonin muurahaisen säännöt. Muurahaisen sääntöjen pituus n tulee valita siten, että siinä on vähintään yhtä monta eri tilaa, kuin selkotekstin aakkostossa. Järkevä valinta on $n = 256$, jolloin tietokoneella toteutetussa algoritmissa voidaan salata mikä tahansa 8-bittisen tavun arvo ilman aakkoston uudelleenkoodausta. Tällöin myös erilaisia siirtosääntöjä on käytettävissä $2^{256} - 2$. Kokonaismäärästä vähennetään 2, sillä 256 peräkkäistä samaa kirjainta, eli $L \ldots L$ ja $R \ldots R$ ovat kelvottomia sääntöjä. Näillä säännöillä muurahainen vain pyörisi paikallaan neljässä ruudussa.

Nämä määrittävät Langtonin muurahaisen simulaatiossa käytettävät parametrit. Salauk-

sessa data jaetaan lohkoiksi ja muurahaista simuloidaan erikseen jokaisella lohkolla, käyttäen lohkon sisältämää dataa ruutujen tiloina. Algoritmi 1 kuvaa salausalgoritmin kokonaisuudessaan ja Algoritmi 2 kuvaa salauksen purkamisen.

Algoritmi 1 Datan salaaminen

1: Luodaan L^2 kokoinen ruudukko, jolla muurahainen liikkuu.
2: Asetetaan muurahainen paikkaan `(0,0,u)`
3: Ladotaan L^2 merkkiä salattavaa dataa ruudukkoon
4: Jos data loppui, lisätään mielivaltaisia täytearvoja lohkon loppuun ja kirjataan lohkon loppuun käytetty täytearvojen määrä
5: Simuloidaan muurahaista K siirtoa.
6: Luetaan muuttunut data takaisin ruudukosta ja lisätään se salatekstiin.
7: Jos salattavaa dataa on vielä jäljellä, palataan kohtaan 3. Muurahaisen nykyinen paikka on muurahaisen uusi paikka seuraavalla lohkolla.
8: Jos data loppui, algoritmi on päässyt loppuun ja data on salattu.

Esimerkkitekstin salaus on esitetty kuvassa 6.1. Selkeyden vuoksi esimerkissä käytettiin aakkostona englanninkielisiä kirjaimia, jolloin arvoja on käytettävissä 26. Kuvia vertaillessa salaus vaikuttaa onnistuneelta. Esimerkiksi kirjain K on muuttunut useaksi erilaiseksi merkeiksi eri kohdissa ja vastaavasti useasta eri kirjaimesta on tullut A. Tämä tarkoittaa, että kyseessä ei ole yksinkertainen korvaussalakirjoitus, eikä salatekstin tietystä merkistä ole välttämättä mahdollista päätellä selkotekstin merkkiä.

Algoritmi 2 Salauksen purkaminen

1: Luodaan L^2 kokoinen ruudukko, jolla muurahainen liikkuu.
2: Asetetaan muurahainen paikkaan, johon se jäi viimeisen lohkon salauksen jälkeen.
3: Ladotaan L^2 merkkiä purettavaa dataa ruudukkoon, aloittaen purettavan datan lopusta
4: Simuloidaan muurahaista K -siirtoa, tällä kertaa taaksepäin.
5: Luetaan muuttunut data takaisin ruudukosta ja lisätään se selkotekstiin.
6: Jos purettavaa dataa on vielä jäljellä, palataan kohtaan 3.
7: Jos data loppui, luetaan viimeisen lohkon lopusta ylimääräisten täytearvojen määrä ja poistetaan puretusta datasta vastaava määrä merkkejä, jotta jäljelle jää ainoastaan alkuperäinen määrä merkkejä.
8: Algoritmi päättyy ja data on purettu.

Kuva 6.1: Vasemmassa kuvassa on selkoteksti "esimerkkitekstin salauskoe" ladottuna ruudukolle vasemmalta oikealle ja ylhäältä alas ilman välilyöntiä. Oikealla vastaava teksti salattuna.

6.1.1 Muurahaisen loppusijainnin ongelma

Algoritmin eräs ongelma on muurahaisen kaaottinen liikehdintä. Salauksen pääteeksi muurahainen jää ennalta määräämättömään ruutuun. Koska ruutujen alkutila vaikuttaa muurahaisen liikeisiin, lopullinen paikka ei ole etukäteen tiedossa. Vaikka käytettäisiin samaa siirtomäärää ja sääntöjä eri datalle. Ilman tietoa tästä ruudusta salausta ei voi purkaa. Tämän ratkaisemiseen on kolme erilaista tapaa, joista jokainen heikentää salausta omalla tavallaan.

Lopullisen paikan tallennus avaimeen johtaa siihen, että avain muuttuu salauksen jälkeen, jolloin purkamiseen ja salaamiseen tarvitaan eri avain. Tällöin salauksen purkajan on saatava käsiinsä myös purkuavain, jolloin salaus ei ole erityisen käyttökelpoinen julkisen tiedonsiirtokanavan kautta käytettynä. Salatun datan - joka saa olla luettavissa - lisäksi vastaanottajalle pitäisi siirtää myös purkuavain, joka ei saa päätyä vääriin käsiin.

Lopullisen paikan tallennus datan perään heikentää turvallisuutta. Kolmas osapuoli voi lukea muurahaisen paikan suoraan datasta ja yrittää purkamista sijoittamalla muurahaisen kyseiseen sijaintiin ja asentoon. Purkajan täytyy tietenkin arvata käytetyt siirtosäännöt, joita on $2^{256} - 2$, joten kokeiltavaa riittää edelleen. Lisäksi tämä menetelmä saattaa paljastaa käytetyn lohkokoon. Jos datan pituudesta voidaan päätellä lohkokooksi joko 13 tai 39, mutta datan mukana tuleva paikka on esimerkiksi `(34,2,d)`, lohkon kooksi voidaan tämän perusteella todeta 39, koska lohkokoolla 13 muurahainen ei voisi olla ruudussa `(34,2)`.

Ylimääräisen siirtomäärän tallennus dataan ei paljastaisi muurahaisen paikkaa. Tässä menetelmässä alkuperäiseen avaimeen valitaan ennen salausta satunnainen paikka, johon muurahaisen tulee päästä viimeisen lohkon jälkeen. Viimeisen lohkon salauksen jälkeen simuloidaan vielä tarpeeksi monta ylimääräistä siirtoa viimeisellä lohkolla, kunnes muurahainen on saavuttanut vaaditun sijainnin ja asennon. Tämä

ylimääräinen siirtomäärä liitetään salatun datan perään. Tällöin salateksti paljastaa salauksesta sen, että viimeisellä lohkolla on siirrelty hiukan enemmän siirtoja, mutta ei muuta. Salauksen purkamisessa muurahainen asetetaan avaimessa määriteltyyn paikkaan, minkä jälkeen muurahaista siirretään takaperin salatekstissä ilmaistu siirtomäärä. Näin muurahainen saapuu siihen paikkaan, jossa se oli viimeisen lohkon salaamisen jälkeen. Tämän jälkeen aloitetaan varsinainen purkaminen, siirtäen muurahaista K siirtoa taaksepäin.

Tämän menetelmän ongelmana on muurahaisen siirtoihin liittyvä epävarmuus. Määritelmän 4.5.2 mukaan muurahainen ei välttämättä saavuta kaikkia mahdollisia asentoja kaikissa sijainneissa. Jos tietylle lohkokoolle löydetään tällainen avaimen ja datan yhdistelmä, kyseinen lohkokoko ja samalla koko algoritmi osoittautuu melko käyttökelvottomaksi. Vaikka tämä tapahtui lohkokoolla 3 ja säännöillä **LRL**, isommalla lohkokoolla ja pidemmillä säännöillä esimerkkiä ei ole löytynyt. On täysin mahdollista, ettei sellaista tapahdu ollenkaan pidemmillä säännöillä. Eräs mahdollisuus olisi käyttää jonkinlaista hybridiä. Jos muurahainen ei tietyn siirtomäärän jälkeen saavuta haluttua paikkaa, perutaan ylimääräiset siirrot ja koodataankin muurahaisen paikka datan perään, kuten ensimmäisessä ehdotuksessa. Vaihtoehtoisesti koodataan pelkkä sijainti ja koodataan muurahaisen asento datan perään. Lisäksi tarvittaisiin vielä pari bittiä kertomaan, mitä näistä vaihtoehdoista on käytetty.

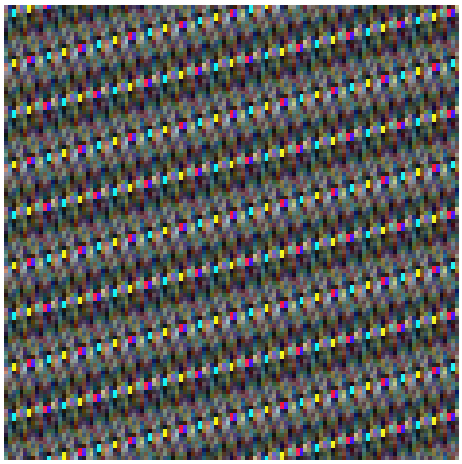
6.1.2 Muurahaisen alkusijainnin ongelma

Jos muurahainen aloittaa salauksen aina paikasta `(0,0,u)`, tämä antaa hyökkääjälle vihjeen salauksen purkamisesta. Murtaessa salausta kokeilemalla riittää, jos tutkitaan purettavan sisältöä ainoastaan silloin, kun muurahainen saapuu paikkaan `(0,0,u)`. Tämän johdosta myös alkupaikka pitää valita satunnaisesti ennen salauksen tekoa. Koska joko avain tai data sisältää tiedon muurahaisen lopullisesta paikasta, eikä lopullinen paikka riipu salauksen aikana tapahtuvista asioista, alkupaikan voi huoletta valita satunnaises-

ti. Tästä saadaan myös se etu, että sama data salautuu eri tavalla kahdella peräkkäisellä salauskerralla.

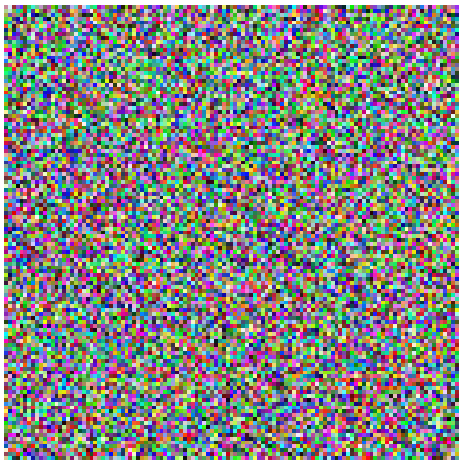
6.1.3 Sähköisen koodikirjan ongelma

Tilastollisesti tarkasteltuna pelkästään hyvä datan sekoitus ei riitä vahvan salauksen tekemiseen, vaikka visuaalisesti tarkasteltuna tulos näyttäisikin lähes satunnaiselta kohinalta. Jos kahden lohkon arvot ovat keskenään samat ja muurahainen aloittaa lohkon samasta asennosta, lohkojen lopputilat ovat myös keskenään samat salauksen päätyttyä. Lisäksi tilanteessa, jossa lohkon jokaisella ruudulla on sama arvo, salattu lohko sisältää aina tietyt arvot. Muurahaisen asennon ja sijainnin muuttuminen lohkojen välissä saa lohkojen arvot vaihtamaan suhteellisesti paikkoja, mutta ne ovat kuitenkin aina samat. Tällöin hyökkääjä voi jakaa salatekstin lohkoiksi ja verrata niitä keskenään, paljastaen mahdolliset toistuvat rakenteet ja näin tehdä mahdollisia päätelmiä selkotekstin sisällöstä. Triviaali esimerkki on salata tiedosto, joka sisältää pelkästään yhtä arvoa, kuten nollaa. Valitaan lohkon kooksi 13 ja salataan $117 \cdot 117$ kokoinen kuvatiedosto. Salatusta kuvasta 6.2 huomataan, että data ei salautunut juuri ollenkaan. Tämä tunnetaan sähköisen koodikirjan ongelmana (engl. *ECB, Electronic Code Book*), joka on tyypillinen lohkosalausmenetelmille. Koska salausprosessi on deterministinen, se ei sisällä satunnaistekijöitä. Näin siis tietty data muuttuu aina samalla tavalla tietyllä avaimella salattuna, samaan tapaan kuin caesar-salauksessa käy yksittäiselle kirjaimelle. Lohkot ovat keskenään identtiset, koska alkutilanteessa jokainen salattava arvo oli täysin sama.



Kuva 6.2: Yksivärinen kuva salattuna. Koska kukin lohko salautuu samalla tavalla, sama rakenne toistuu salatussa kuvassa.

Koodikirjaongelma voidaan poistaa ottamalla mukaan satunnaisuutta. Eräs tekninen toteutus on lisätä selkotekstin alkuun yhden lohkon verran satunnaista dataa, eli niin kutsuttu alkuvektori (engl. *Initialization vector*), joka salataan muun datan mukana. Salauksen aikana jokainen lohko salataan, minkä jälkeen otetaan seuraavan lohkon arvot ja tehdään niille binäärinen XOR-operaatio edellisen lohkon salattujen arvojen kanssa. Koska ensimmäinen lohko oli satunnaista dataa ja jos salausalgoritmi ei poista satunnaisuutta, XOR -operaation jälkeen myös seuraavan lohkon alkuarvot ovat satunnaisia, eikä niillä ole yhteyttä alkuperäisiin arvoihin. Vasta tämän jälkeen seuraava lohko salataan. Näin selkoteksti, joka on kokonaan samaa arvoa, saadaan muutettua täysin satunnaisen näköiseksi kohinaksi. Alkuvektorin lisääminen salattavaan dataan näkyy kuvassa 6.3. Data on sekoittunut selkeästi paremmin.



Kuva 6.3: Sama yksivärisen kuvan salaus täysin samoilla parametreilla kuin kuvassa 6.2, mutta tällä kertaa käyttämällä satunnaista alkuvektoria.

6.2 Paranneltu algoritmi

Valitaan aiemmin esitettyjen arvojen L,K ja S_n lisäksi:

Koordinaatit x ja y, $x < L$, $y < L$ jotka edustavat sellaista ruutua, johon muurahainen siirretään salauksen jälkeen.

Muurahaisen suunta D, joko ylös, alas, vasemmalle tai oikealle, joka määrittää suunnan, johon muurahainen osoittaa salauksen jälkeen.

Paranneltu algoritmi toimii niin, että ensin salataan mielivaltaista dataa, joka liitetään salattavan datan alkuun. Tätä käytetään ketjuttamaan kaikki lohkot toisiinsa, jolloin vältytään sähköisen koodikirjan ongelmalta. Koska kaikki lohkot ketjutetaan toisiinsa, alkuvektorin lisääminen vaikuttaa jokaisen lohkon sisältöön. Algoritmi 3 kuvailee parannellun salausalgoritmin ja Algoritmi 4 salauksen purkamisen.

Esimerkkisalaus

Parannellun algoritmin esimerkkisalaus on kuvassa 6.4. Kolmeen eri kuvaan samalla avaimella tehty salaus muutti kaikki kuvat satunnaisen näköiseksi kohinaksi.

Algoritmi 3 Paranneltu salausalgoritmi

1: Luodaan L^2 kokoinen ruudukko, jolla muurahainen liikkuu.
2: Asetetaan muurahainen paikkaan (x_1, y_1, D_1) jossa x_1 ja y_1 ovat mielivaltaisia ruudukon koordinaatteja ja D_1 on mielivaltainen suunta.
3: Luodaan alkuvektori latomalla mielivaltaista dataa ruudukkoon.
4: Simuloidaan muurahaista K siirtoa.
5: Luetaan data ruudukosta ja lisätään se salatekstin loppuun.
6: Ladotaan ruudukkoon L^2 merkkiä salattavaa dataa.
7: Jos data loppui, lisätään mielivaltaisia täytearvoja lohkon loppuun ja kirjataan lohkon loppuun käytetty täytearvojen määrä
8: Suoritetaan datalle binäärinen XOR-operaatio siten, että operaatio tehdään jokaiselle tavulle käyttäen toisena operandina edellisen lohkon vastaavassa koordinaatissa ollutta arvoa.
9: Simuloidaan muurahaista K siirtoa.
10: Jos ollaan viimeisellä lohkolla, simuloidaan muurahaista niin pitkään, että se saavuttaa avaimessa määritellyn paikan `(x,y,D)`
11: Luetaan muuttunut data takaisin ruudukosta ja lisätään se salatekstin loppuun.
12: Jos salattavaa dataa on vielä jäljellä, palataan kohtaan 6. Muurahaisen nykyinen paikka on muurahaisen alkupaikka seuraavalla lohkolla.
13: Liitetään ylimääräinen siirtomäärä datan loppuun.

Algoritmi 4 Salauksen purkaminen: paranneltu algoritmi

1: Luodaan L^2 kokoinen ruudukko, jolla muurahainen liikkuu.

2: Asetetaan muurahainen paikkaan `(x,y,D)` jossa x ja y ja D ovat avaimessa määritelty muurahaisen loppupaikka.

3: Ladotaan ruudukkoon L^2 merkkiä purettavaa dataa.

4: Jos kyseessä on ensimmäinen purettava lohko, simuloidaan muurahaista datan lopussa ilmaistun ylimääräisen siirtomäärän verran taaksepäin.

5: Simuloidaan muurahaista K siirtoa taaksepäin.

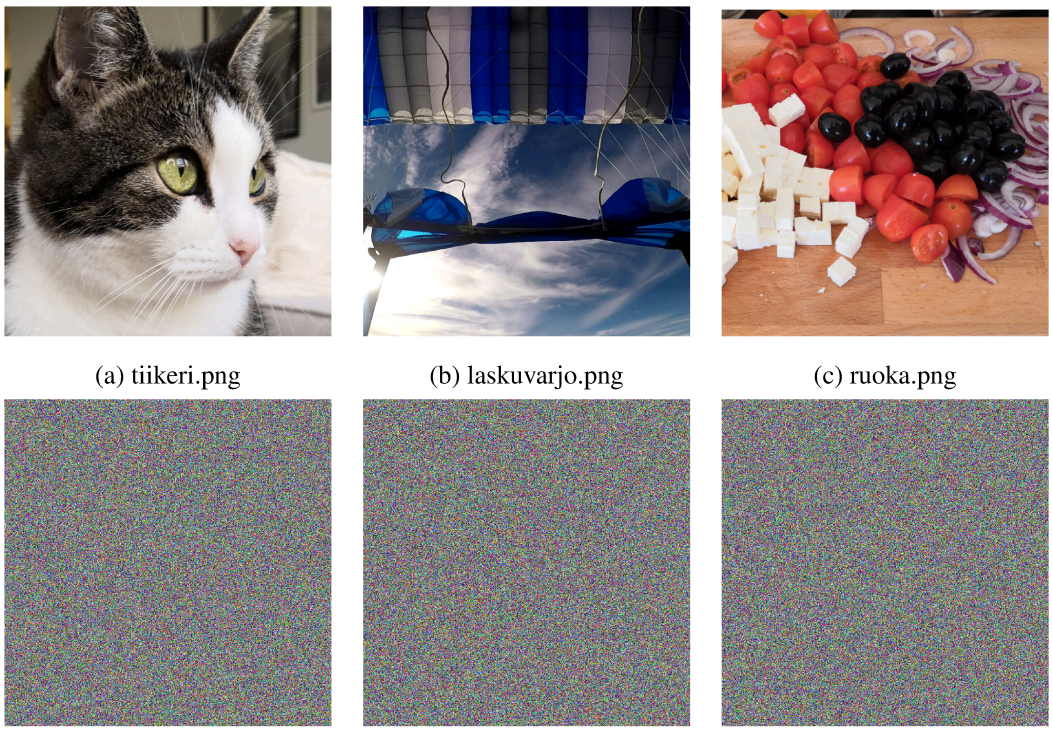
6: Suoritetaan datalle binäärinen XOR-operaatio siten, että operaatio tehdään jokaiselle tavulle käyttäen toisena operandina edellisen lohkon vastaavassa koordinaatissa ollutta arvoa.

7: Luetaan muuttunut data takaisin ruudukosta ja lisätään se selkotekstiin.

8: Jos purettavaa dataa on vielä jäljellä, palataan kohtaan 3. Muurahaisen nykyinen paikka on muurahaisen alkupaikka seuraavalla lohkolla.

9: Jos data loppui, luetaan selkotekstin lopusta ylimääräisten merkkien määrä ja poistetaan kyseinen määrä merkkejä datan lopusta.

10: Poistetaan alkuvektori selkotekstin alusta.



(a) tiikeri.png (b) laskuvarjo.png (c) ruoka.png

Kuva 6.4: Kolme esimerkkikuvaa ylärivillä ja niiden salatut versiot alarivillä

6.2.1 Siirtomäärän arviointi

Salausavaimeen valitaan siirtosääntöjen lisäksi erilaisia parametreja, kuten lohkokoko ja siirtojen määrä. Näiden valintaan täytyy kiinnittää erityistä huomiota, sillä kaikki yhdistelmät eivät ole välttämättä yhteensopivia. Vertaillaan esimerkkikuvia 6.5(b) ja 6.5(p). Erityisen pienellä lohkokoolla $L = 3$ jo $K = 500000$ siirtoa riittää tuottamaan visuaalisesti tarkasteltuna satunnaisen lopputuloksen, kun taas isolla lohkokoolla $L = 69$ kuvan värit ovat vaihtuneet vain hieman ja alkuperäisen kuvan piirteet ovat selkeästi havaittavissa. Syynä tähän on tehtyjen siirtojen kokonaismäärä. Salauksessa tehdään K siirtoa jokaiselle *lohkolle*, joten jos lohkon koko on pienempi, siirtoja tehdään ruutujen määrään suhteutettuna paljon enemmän. Siirtojen kokonaismäärä voidaan laskea kertomalla lohkojen määrä yhdellä lohkolla tehdyillä siirroilla. Määritetään P tarkoittamaan datan täydennettyä pituutta, jolloin siirtojen kokonaismäärä T saadaan laskettua lohkokoon sivun

pituudesta L ja siirtomäärästä K kaavalla:

$$T = \frac{P}{L^2} \cdot K \tag{6.1}$$

Näin laskemalla esimerkkikuvassa 6.5(b) siirtoja tehtiin yhteensä noin 45 miljardia. Jos halutaan laskea lohkokohtainen siirtomäärä tietylle kokonaissiirtomäärälle, voidaan kertoa molemmat puolet $\frac{P}{L^2}$:lla. Tästä saadaan

$$K = \frac{L^2 \cdot T}{P}$$

Jos lohkokoko L on 69, pienemmällä lohkokoolla tehdyn 45 miljardin siirtomäärän saavuttamiseksi tarvitaan noin 261 miljoonaa siirtoa per lohko, kun taas kuvan 6.5(l) esimerkissä vastaava luku oli 5 miljoonaa. Siirtomäärää ei siis voida määrittää mielivaltaisesti, vaan se täytyy suhteuttaa lohkokokoon.

Kaavan 6.1 avulla voidaan määrittää toinen kaava, jolla K on mahdollista muuttaa toiselle lohkokoolle. Määritetään alkuperäinen K_1 ja L_1, sekä tavoitesiirtomäärä K_2 ja tavoitelohkokoko L_2. Tästä saadaan

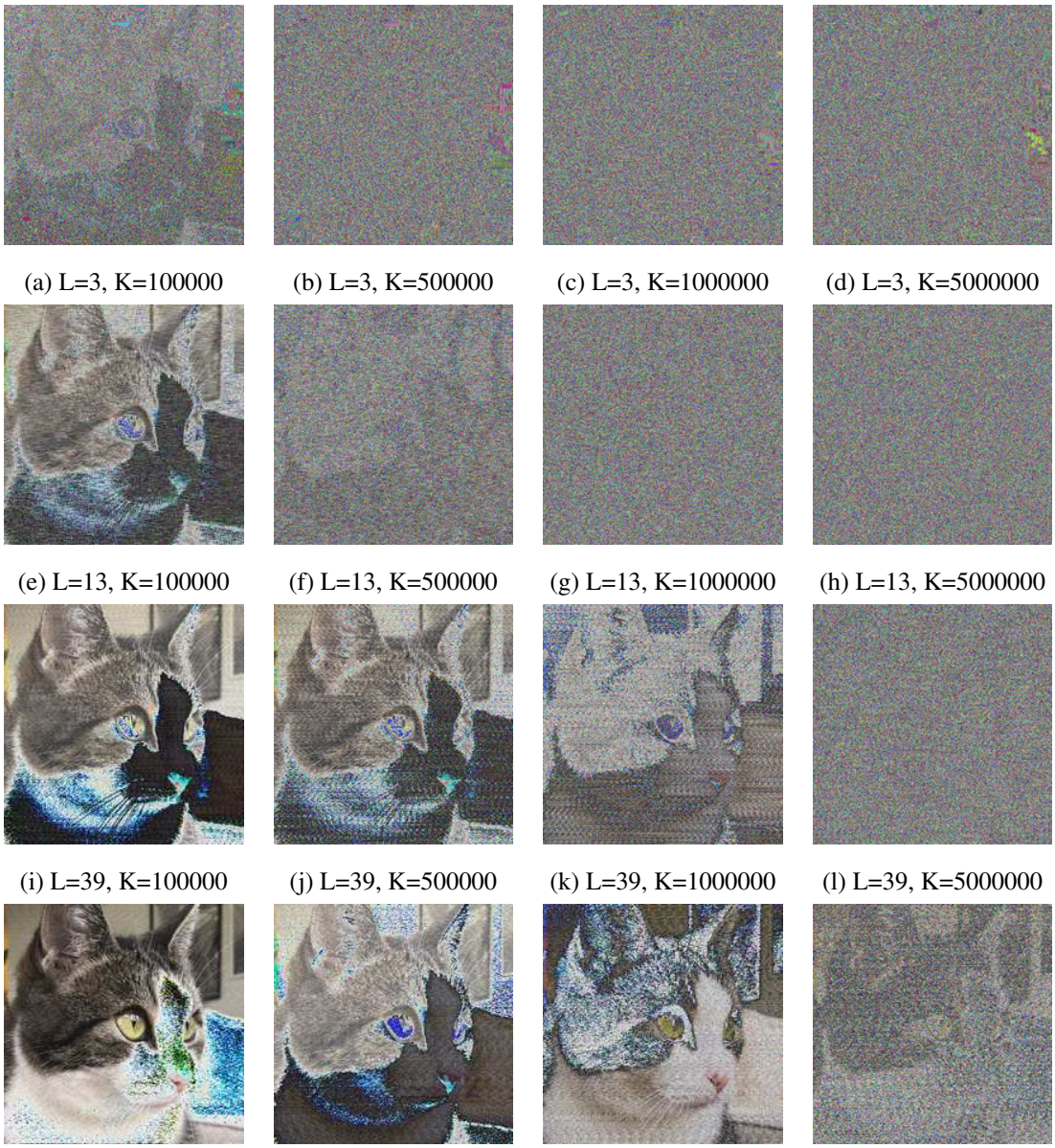
$$T = \frac{P}{L_1^2} \cdot K_1 = \frac{P}{L_2^2} \cdot K_2 \quad \text{(Jaetaan } \frac{1}{T}\text{:lla)}$$

$$\frac{1}{T} \cdot \frac{P}{L_1^2} \cdot K_1 = \frac{1}{T} \cdot \frac{P}{L_2^2} \cdot K_2 \quad (\frac{T}{P} \text{ on vakio, supistetaan})$$

$$\frac{1}{L_1^2} \cdot K_1 = \frac{1}{L_2^2} \cdot K_2 \quad \text{(Ratkaistaan } K_2\text{)}$$

$$K_2 = \frac{L_2^2 \cdot K_1}{L_1^2} \tag{6.2}$$

Kuvassa 6.6 on esitetty sama salaus neljällä eri lohkokoolla, mutta tällä kertaa siirtomäärä per lohko on laskettu kaavan 6.2 mukaan niin, että kuvien 6.6(a,b,c,d) kokonaissiirtomäärä on muunnettu jokaiselle lohkokoolle. Visuaalisesti tarkasteltuna siirtojen kokonaismäärä vaikuttaa lopputulokseen ja siirtomäärän muuntaminen lohkokokojen välillä annetulla kaavalla on antaa vertailukelpoisia tuloksia. Ensimmäisellä rivillä sekoittuminen on merkittävästi heikompaa muihin riveihin verrattuna. Tämä perustuu aliluvussa 4.5.3 esitettyyn havaintoon, jossa samankaltaiset alkutilat pysyvät lähellä toisiaan melko pitkään. Lohkokoolla $L = 3$ lohkolla on vain 9 eri arvoa, mikä vastaa kolmen kuvapisteen väriarvoja. Kolme peräkkäistä kuvapistettä voivat olla esimerkkikuvassa väriltään lähellä toisiaan, joten liian pienellä siirtomäärällä ne myös pysyvät suhteellisesti lähellä toisiaan. Ongelma voidaan välttää tekemällä huomattavasti enemmän siirtoja. Aiemmassa kuvassa 6.5(d) suurempi siirtomäärä on johtanut ensimmäiselläkin rivillä satunnaisempiin arvoihin.



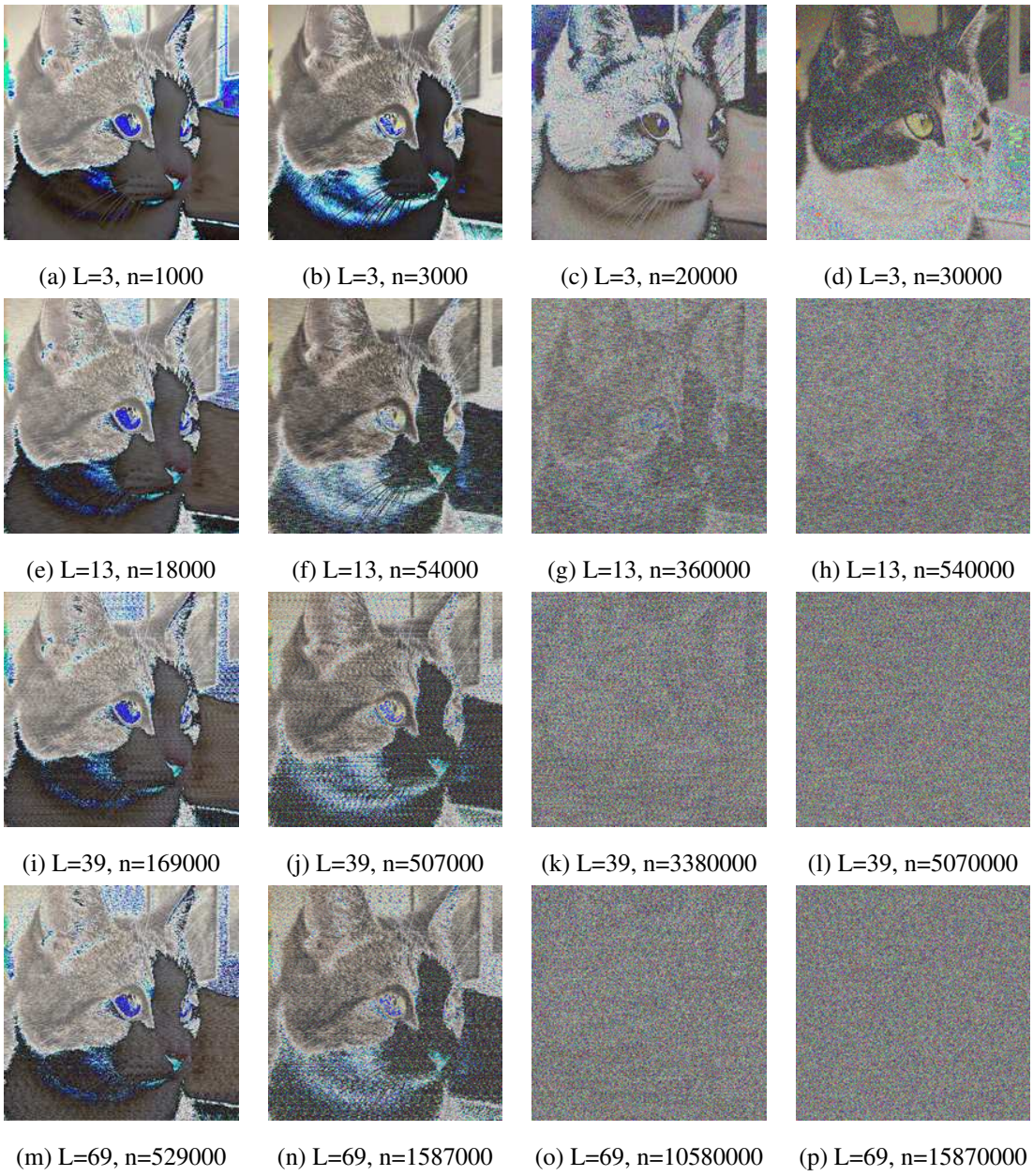
(a) L=3, K=100000 (b) L=3, K=500000 (c) L=3, K=1000000 (d) L=3, K=5000000

(e) L=13, K=100000 (f) L=13, K=500000 (g) L=13, K=1000000 (h) L=13, K=5000000

(i) L=39, K=100000 (j) L=39, K=500000 (k) L=39, K=1000000 (l) L=39, K=5000000

(m) L=69, K=100000 (n) L=69, K=500000 (o) L=69, K=1000000 (p) L=69, K=5000000

Kuva 6.5: Salauksen toiminta eri lohkokoolla L ja siirtomäärällä K.



(a) L=3, n=1000 (b) L=3, n=3000 (c) L=3, n=20000 (d) L=3, n=30000

(e) L=13, n=18000 (f) L=13, n=54000 (g) L=13, n=360000 (h) L=13, n=540000

(i) L=39, n=169000 (j) L=39, n=507000 (k) L=39, n=3380000 (l) L=39, n=5070000

(m) L=69, n=529000 (n) L=69, n=1587000 (o) L=69, n=10580000 (p) L=69, n=15870000

Kuva 6.6: Salauksen toiminta eri lohkokoolla L ja siirtomäärällä n. Lohkokoot kuvissa (a,b,c,d) on valittu manuaalisesti ja jälkimmäisen rivien siirtomäärä on laskettu ensimmäisen rivin sarakkeiden perusteella käyttäen kaavaa 6.2.

Tilastolliset menetelmät

Tarkastellaan kuvan laskuvarjo.png salausta ja valitaan lohkokooksi 39. Kuva salataan eri siirtomäärillä. Salauksen jälkeen valitaan 300 paria vierekkäisiä pikseleitä joko vaaka-

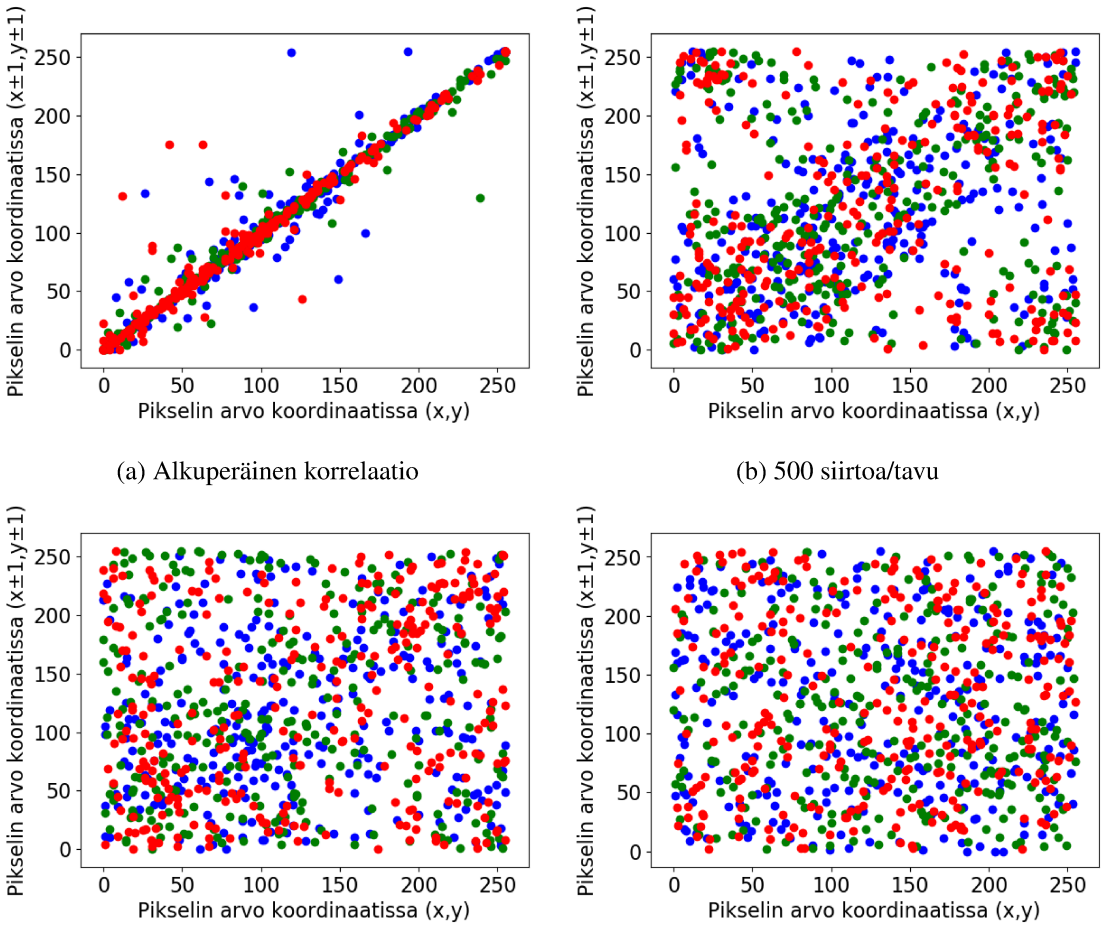
	Värikanavan korrelaatiot		
Siirtomäärä/ruutu	Vihreä	Sininen	Punainen
0	0.9840926927348125	0.9792461986761957	0.9798465525128055
100	0.39084495106992323	0.5328777935522508	0.6472466269154671
500	0.2380404487191325	0.12659958581261896	0.31923727379621303
1000	0.10988885503383365	0.05242644930843923	0.17079899979583885
2000	-0.0230900766320593	0.08373260499966055	-0.009599472258132485
3000	0.035313640428052066	0.04987633931694788	-0.04572253894716539
4000	0.10552701457553898	-0.11328634533686113	0.04291457637113758
5000	0.015395100103382383	-0.08573440019949663	-0.08610930287634792
6000	0.007311452326761753	0.04174756362050955	0.00875574324548402

Taulukko 6.1: Vierekkäisten pikselien korrelaatio salauksen jälkeen jokaiselle värikanavalle

suoraan, pystysuoraan tai vinottain. Valinta tehdään kullekin värikanavalle erikseen ja valituille pikseleille lasketaan Pearsonin korrelaatio. Taulukosta 6.1 käy ilmi, että salaamattomalle datalle (0 siirtoa) vierekkäisten pikselien korrelaatio on erittäin vahva, mutta korrelaatio putoaa nopeasti siirtomäärän kasvaessa. Taulukossa siirtomäärä on annettu lohkokoosta riippumattomasti siten, kuinka monta siirtoa tehdään jokaista selkotekstin tavua kohti. Esimerkiksi 100 siirtoa per tavu tarkoittaa, että lohkon siirtomäärä $K = 39^2 \cdot 100 = 152100$. Eri siirtomäärien sirontakuvioita on nähtävissä kuvissa 6.7(a-d). Tuhannellakin siirrolla jakauma on vielä hiukan ryhmittynyttä.

Histogrammit

Väridatan rakenteen voi esittää histogrammina, joka kuvaa kunkin värin intensiteettien määrät. Kuvista 6.8 nähdään eri siirtomäärien vaikutus salatun kuvan histogrammiin. Vielä 3000 siirron kohdalla histogrammissa voi havaita pienen kuopan keskellä, joka kertoo, etteivät intensiteetit ole jakautuneet tasaisesti. Saman voi huomata kuvasta 6.6(k) jossa on



(a) Alkuperäinen korrelaatio

(b) 500 siirtoa/tavu

(c) 1000 siirtoa/tavu

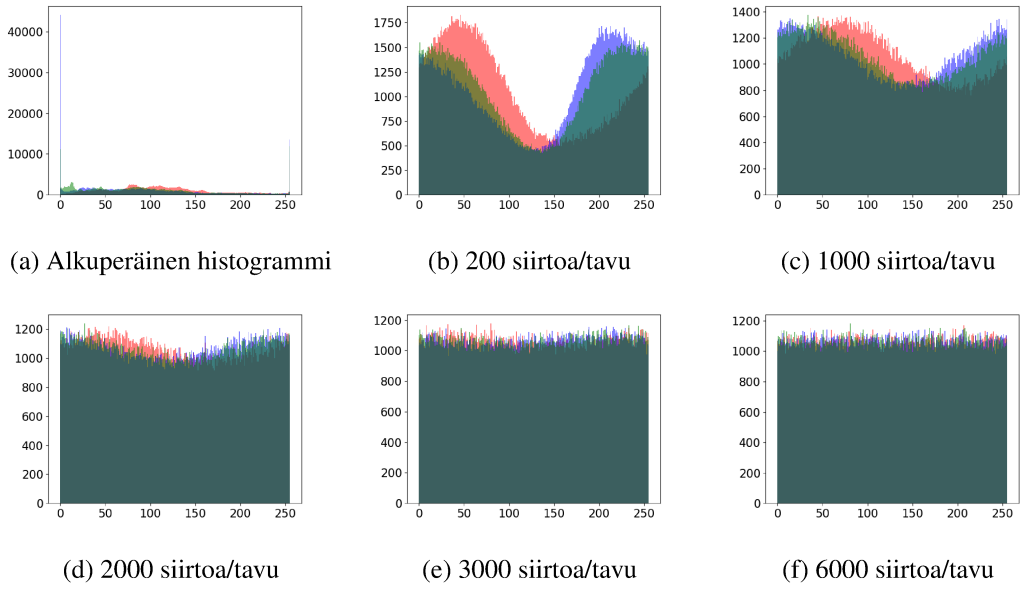
(d) 6000 siirtoa/tavu

Kuva 6.7: Kuvan laskuvarjo.png pikselikorrelaatioiden sirontakuviot eri siirtomäärien jälkeen

tehty noin 2200 siirtoa per tavu, mutta alkuperäisen kuvan piirteitä on vielä jonkin verran näkyvillä. Histogrammeista kuitenkin nähdään, että sinisten pikselien alkuperäinen intensiteetti on saatu jakautumaan tasaisesti, eikä histogrammissa mikään värikanava ole toistaan voimakkaampi.

Histogrammeista huomataan, että vierekkäisten pikselien korrelaatio ei riittänyt yksinään määrittelemään siirtomäärää. Korrelaatio ei juuri muuttunut enää 2000 siirron jälkeen, eli pikselit olivat sekoittuneet, mutta niiden suhteelliset määrät vaihtelivat vielä melko paljon. Tämä oli havaittavissa erityisesti kuvassa laskuvarjo.png, jossa suuri osa

datasta oli alun perinkin yhtä väriä.



(a) Alkuperäinen histogrammi (b) 200 siirtoa/tavu (c) 1000 siirtoa/tavu

(d) 2000 siirtoa/tavu (e) 3000 siirtoa/tavu (f) 6000 siirtoa/tavu

Kuva 6.8: Kuvan laskuvarjo.png histogrammit eri siirtomäärien jälkeen.

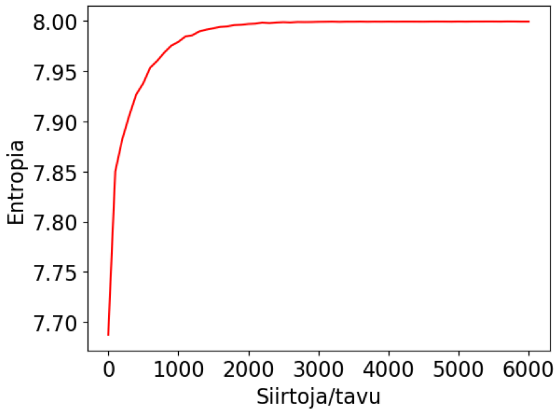
Informaatioteoria

Informaatioteoriassa entropia mittaa datan sisältämän informaation määrää. Esimerkiksi satunnaislukugeneraattori, joka tuottaa tasaisesti kaikkia lukuja, tuottaa maksimimäärän entropiaa. Entropia ei kuitenkaan ota kantaa datan järjestykseen, vaan sen jakaumaan. Lukuja 0-10 järjestyksessä jaksoittain tuottava generaattori antaa laskennallisesti suurimman määrän entropiaa, mutta datassa on silti toistoa ja järjestystä.

Jos tarkasteltavat arvot ovat 8 bittisiä, tasaisen jakauman entropia on 8. Entropia saadaan laskemalla

$$H = -\sum_{i=1}^{M} P_i \, log_2 \, P_i$$

Kuvassa 6.9 nähdään salattavan kuvan entropian kehitys eri siirtomäärillä. Suurin entropia 7.9993.. saavutetaan noin 3000 siirron jälkeen. Salattu data on siis jakautunut hyvin tasaisesti.



Kuva 6.9: Kuvan laskuvarjo.png entropia eri siirtomäärillä

Lopullinen siirtomäärä

Siirtomääräksi kannattaa valita ainakin 6000 siirtoa per tavu. Tosin pienellä lohkokoolla tämä ei välttämättä ole tarpeeksi, mutta erityisen pieniä lohkokokoja kannattaa joka tapauksessa välttää. Kuten aiemmin havaittiin kuvasta 6.6, salaus käyttäytyy hiukan eri tavalla pienellä lohkolla, vaikka muut parametrit olisivat samat.

6.2.2 Tilamuutoskaavion laajennus

Klassinen Langtonin muurahainen siirtää tilojaan aina järjestyksessä, kasvattaen tilaa yhdellä, kuten esimerkiksi kuvassa 4.1. Tilojen vaihtumisjärjestys voidaan satunnaistaa, kuten nähdään taulukosta 6.2. Tämä muuttaa muurahaisen käyttäytymistä. Siirtosäännölle, jonka pituus on s, tilojen vaihtuminen voidaan järjestää $s!$ erilaisella tavalla. Pitää kuitenkin huomioida, että $0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 0$ on sama kuin $1 \rightarrow 2 \rightarrow 3 \rightarrow 0 \rightarrow 1$ ja muut samassa syklissä kulkevat tilanmuutoskaaviot. Todellisuudessa erilaisia permutaatioita on syklisen permutaatiokaavan mukaan $(s-1)!$ Näin ollen säännöillä, joiden pituus on 4, tilat voidaan vaihtaa $(4-1)! = 6$ eri tavalla.

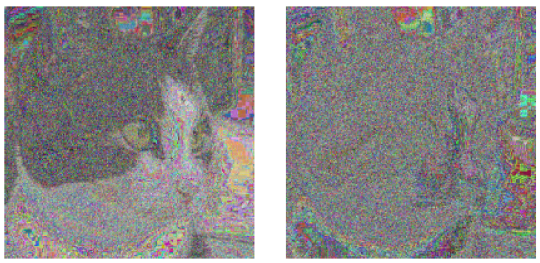
Ruudun tila	0	2	1	4	3
Tilan indeksi	0	1	2	3	4
Kääntymissuunta	L	R	L	R	R

Taulukko 6.2: Satunnainen tilojen vaihtumisjärjestys

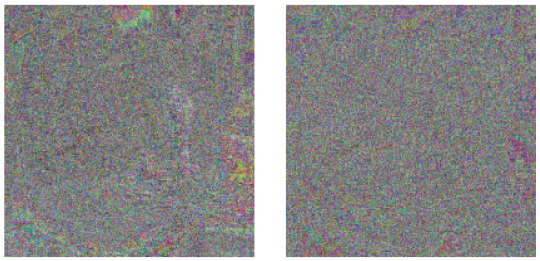
Jos tilanmuutoskaavio satunnaistetaan, avainavaruus kasvaa merkittävästi. Jokaiselle avaimelle saadaan $255!$ erilaista tilanmuutoskaaviota. Tällöin erilaisia avaimia olisi käytettävissä $(2^{256} - 2) \cdot 255!$.

Toinen saavutettu hyöty satunnaistamisesta on, että data näyttäisi menevän visuaalisesti sekaisin paljon nopeammin. Kuvassa 6.10 näkyy, miten huomattavasti pienemmät siirtomäärät sekoittavat datan paljon pienemmällä siirtomäärällä verrattuna kuvan 6.5 esimerkkeihin. Kuitenkin verrattain pienen siirtomäärän takia vierekkäiset arvot saattavat vaihtua samassa tahdissa, vaikka ne eivät vaihdukaan lineaarisesti ja ilman alkuvektoria sähköisen koodikirjan ongelma on myös näkyvillä erityisesti oikeassa laidassa.

Hyökkääjän näkökulmasta pienemmästä siirtomäärästä ei ole merkittävästi hyötyä. Vaikka siirtomäärä olisi tiedossa, hyökkääjä ei voi pyrkiä selvittämään salattua dataa siirtämällä jokaista ruutua muutaman tilan taaksepäin, sillä tilat eivät enää vaihdu lineaarisesti. Tämän version purkaminen väkisin vaatisi siis kaikkien erilaisten tilanmuutoskaavioiden kokeilemista.



(a) L=3, n=8 (b) L=3, n=28



(c) L=13, n=800 (d) L=13, n=2800

Kuva 6.10: Salauksen eteneminen eri lohkokoolla ja siirtomäärällä, käyttäen satunnaistettua tilanmuutoskaaviota.

6.3 Kryptoanalyysi

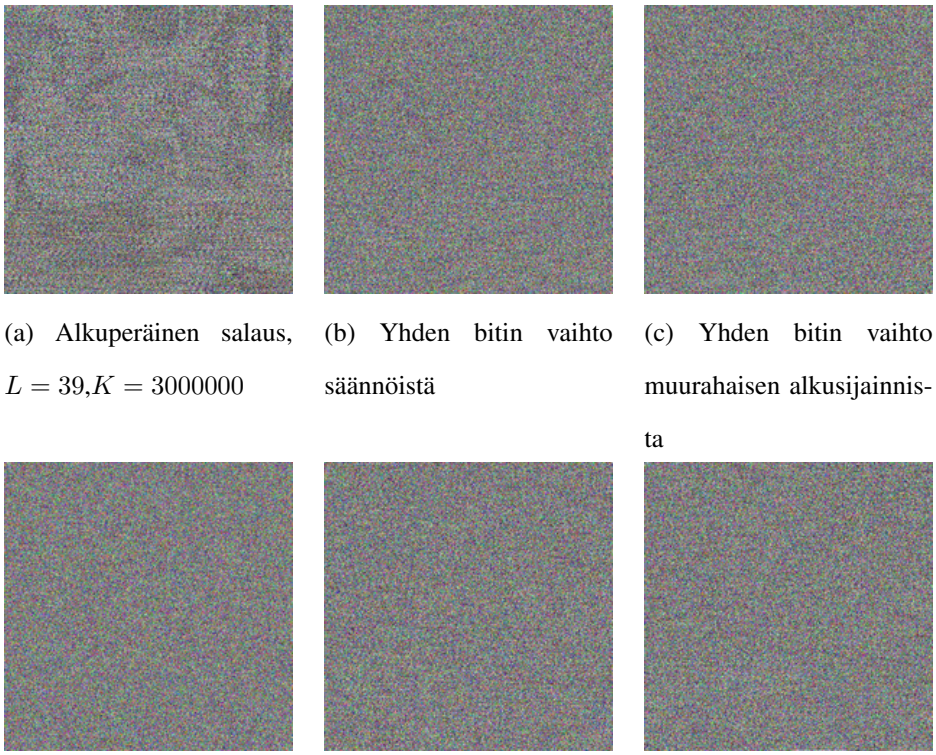
Tässä aliluvussa pohditaan tarkemmin salausmenetelmän luotettavuutta. Pohdinnassa on algoritmin versio, joka ei sisällä satunnaistettuja tilanmuutoskaavioita. Visualisoinneissa alkuvektoria ei ole käytetty, jolloin muurahaisen todelliset liikkeet näkyvät selkeämmin.

6.3.1 Purkaminen väärällä avaimella

Avaimen hyöty salauksen satunnaistajana pitää varmistaa. Ideaalitilanteessa pienikin muutos avaimessa saa aikaan kokonaan erilaisen salatekstin. Lisäksi väärän avaimen avulla purkamisen pitäisi vain sekoittaa data eri tavalla. Kuvassa 6.11(a) on alkuperäinen salaus. Tämän jälkeen avaimesta on vaihdettu ainoastaan yhtä arvoa, kuten muurahaisen alkusijainti tai -asento. Kuvassa 6.11(f) on syytä ottaa huomioon, että viimeinen lohko on todellisuudessa onnistuttu purkamaan lähes oikein. Viimeisellä lohkolla jäi tekemättä ai-

noastaan yksi siirto, jolloin data on jo suurimmaksi osaksi purettu. Tätä on melko hankalaa hahmottaa esitetystä kuvasta. Kuvista voidaan siis havaita, että data sekoittuu väärällä avaimella salattuna lisää, eikä johda esimerkiksi lähes tai osittain purettuun tietoon, paitsi viimeisessä esimerkissä. Esimerkkikuvissa vaikutus korostuu, sillä alkuperäinen salaus oli tehty riittämättömällä siirtomäärällä niin, että alkuperäisen kuvan hahmo on hiukan näkyvillä.

Periaatteessa voisi olla mahdollista, että väärällä avaimella purettaessa saadaan sattumanvaraisesti tietyn lohkon sisältö selville. Tästä ei kuitenkaan ole hyötyä muiden lohkojen purkamisessa, koska lohkot ovat ketjutettu toisiinsa. Jotta tietty lohko voidaan purkaa edellisen lohkon jälkeen, on kaikkien parametrien oltava täsmälleen oikein. Tästä käy ilmi, että salausavain aiheuttaa *sekaannusta*, eli kukin salatekstin bitti riippuu useasta avaimen kohdasta ja yhden bitin muuttaminen riittää sekoittamaan kuvan erilaiseksi.



(a) Alkuperäinen salaus, $L = 39$,$K = 3000000$

(b) Yhden bitin vaihto säännöistä

(c) Yhden bitin vaihto muurahaisen alkusijainnista

(d) $L = 13$

(e) Yhden bitin vaihto muurahaisen alkuasennosta

(f) Siirtomäärä yhden liian pieni

Kuva 6.11: Salauksen purkaminen väärällä avaimella. Avain on sama lukuun ottamatta kuvatekstissä mainittua muutosta.

6.3.2 Purkaminen väkisin

Koska avaimia on vain rajallinen määrä, salauksen purkaminen onnistuu varmasti, jos aikaa ja muita resursseja on käytettävissä riittävästi. Arvioidaan, montako erilaista avainta voidaan luoda asettamalla joitakin ehtoja.

Siirtosääntöjä on käytettävissä $2^{256} - 2$

Lohkon kooksi valitaan aiemmin käytetty 39.

Siirtomääräksi hyökkääjä olettaa tai tietää 5-25 miljoonaa.

Näiden lisäksi muurahainen voi olla jokaisessa ruudussa, jokaisessa asennossa, eli tässä tapauksessa $39^2 \cdot 4$ eri paikassa. Tästä saadaan

$$(2^{256} - 2) \cdot (25000000 - 5000000) \cdot 39^2 \cdot 4 \approx 1.41 \ldots \cdot 10^{88}$$

erilaista avainta. Vertailun vuoksi atomien määrä havaittavassa maailmankaikkeudessa on $\approx 10^{80}$ joten tämän salauksen purkaminen satunnaisesti kokeilemalla ei vaikuta todennäköiseltä

6.3.3 Avaimen päättely salatekstistä

Salattu data antaa jonkin verran tietoja lohkokoosta. Koska datan pituus on jaollinen tietyllä kokonaisluvulla, mahdolliset eri lohkokoot on mahdollista laskea datasta. Esimerkiksi, jos data on jaollinen 39^2:lla, lohkokoko on joko 39 tai 13, koska 39 on jaollinen myös 13:lla. Toisaalta esimerkiksi AES -salauksessa lohkokoko on aina 128 bittiä. Vaihtelevaa lohkokokoa voidaan siis pitää salauksen etuna, vaikka mahdollisia lohkokokoja olisikin vain muutama.

6.3.4 Herkkyys selkotekstin muutokselle

Aiemmin todettiin, että avaimen pienikin muutos avaimessa muuttaa salausta. Toinen toivottu ominaisuus on diffuusio, eli selkotekstin pienenkin muutoksen pitäisi muuttaa salatekstiä. Tässä muurahainen onnistuu ainoastaan, jos muutettu bitti on datan alkupäässä. Jos datan viimeisestä lohkosta muutetaan yksi bitti, kaikki kyseistä lohkoa ennen olevat lohkot salautuvat samalla tavalla. Vastaavasti dataa purkaessa, jos salatekstin yhtä bittiä muutetaan, vaikutus kohdistuu ainoastaan kyseisen lohkon jälkeen purettaviin lohkoihin.

Algoritmia voisi kehittää pidemmälle selvittämällä, voisiko algoritmiin kuulua useampia kierroksia. Tällöin viimeisen lohkon jälkeen lohko yhdistetään XOR-operaatiolla takaisin ensimmäiseen lohkoon ja tehdään uusi kierros. Näin yhden bitin vaihtaminen selkotekstin mistä tahansa kohdasta leviää koko datan alueelle. Samoin purkaessa yhden

bitin muuttaminen salatekstistä vaikuttaisi koko dataan. Tämä myös poistaisi kokonaan mm. Kumar *et al.* tutkimuksessa[53] olevan ongelman, jossa dataan tehty mielivaltainen muutos ei juurikaan hyödyttänyt salausta purkaessa. Useampien kierrosten käyttäminen salauksen aikana on hyvä ottaa käsittelyyn myöhemmissä tutkimuksissa.

6.4 Ongelmat kuvansalauksessa

Tutkielman esimerkeissä salattava data oli ensimmäistä esimerkkiä lukuun ottamatta kuvatiedostojen väriarvoja. Tällaista dataa salatessa on syytä huomioida, että salattu data voidaan visualisoida ja esimerkiksi salatusta datasta 6.6(g) alkuperäisen kuvan piirteet näkyvät vielä melko selkeästi. Yhden pikselin kolmen väriarvon siirtäminen muutamalla askeleella suuntaan tai toiseen vaihtaa pikselin sävyä. Vierekkäisten, samansävyisten pikselien muuttaminen muutamilla askeleilla, vaikkakin hiukan erilaisilla arvoilla saa pikselit silti näyttämään hyvin samankaltaisilta ja ihmissilmä kykenee hahmottamaan pikselien väliset suhteet. Jos kyseessä olisi tekstiä, alkuperäisen tekstin sisällöstä ei olisi mahdollista tehdä samaan tapaan suuntaa-antavia tulkintoja. Luultavasti liian pienellä siirtomäärällä salatun tekstinkin toistuvat rakenteet saattavat olla jossain määrin havaittavissa salatekstistä oikeanlaisella muunnoksella. Lisäksi kuviakin salatessa salaus kannattaa kohdistaa tiedostoon itseensä, eikä pelkästään väridataan. Näin hyökkääjä ei voi suoraa päätellä tiedostotyyppiä salatusta datasta.

6.5 Salausavaimen generointi

Salausavain on aliluvussa 6.2 esitetty kokoelma parametreja, joilla muurahaista siirrellään. Kuten esitettiin aliluvussa 6.2.1, kaikki parametrit eivät välttämättä anna yhtä vahvaa salausta. Siinä missä AES- tai RSA-avain voidaan valita luomalla satunnaisia bittejä, muurahaiselle tämä ei ole niin yksinkertaista. Säännöt voidaan valita satunnaisesti, mutta sen lisäksi tarvitaan siirtomäärä ja lohkokoko, jotka riippuvat toisistaan. Lisäksi muura-

haisen loppusijainti riippuu lohkokoosta. Jos avain generoidaan täysin satunnaisesti, ei voida sanoa ennalta, kuinka pitkään tietyn salauksen tekeminen kestää, tai onko salaus ylipäänsä turvallinen.

Luku 7

Johtopäätökset

Tutkielmassa esiteltiin tapa salata dataa Langtonin muurahaisen avulla ja pohdittiin erilaisten parametrien vaikutusta salauksen lopputulokseen. Kuten aiemmatkin tutkimustulokset, tämäkin soluautomaattiin perustuva salaus antoi lupaavia tuloksia.

Algoritmissa on useita hyviä ominaisuuksia. Se sopii mille tahansa datalle, eikä ole riippuvainen syötteen pituudesta. Algoritmissa ei myöskään esiinny säännöllisen koodikirjan ongelmaa ja yksittäisenkin lohkon sisällä olevat arvot sekoittuvat toisistaan riippumatta niin, ettei minkään lähde ja kohdearvon välillä ole yksikäsitteistä kuvausta. Salauksessa käytetään kerta-arvoja (engl. *nonce*), kuten muurahaisen simulaation alkupiste ja salauksen alkuvektori, jolloin sama data voidaan salata useita kertoja peräkkäin, eikä salatekstien välillä ole suoraa yhteyttä.

Algoritmin eräs puute on osittainen välinpitämättömyys selkotekstin ja salatekstin muutoksille. Jos yhtä bittiä muutetaan sopivasta kohdasta, osa datasta salautuu tai purkautuu täsmälleen samalla tavalla. Ratkaisuksi esitettiin useampia salauskierroksia, jolloin aiemmin salattu data vaikuttaisi koko lopputulokseen. Tätä mahdollisuutta ei kuitenkaan tutkittu vielä tarkemmin. Algoritmin muina haasteina voidaan pitää muurahaisen arvaamatonta käyttäytymistä, mikä hankaloittaa algoritmin matemaattista tarkastelua, josta voisi olla apua turvallisuuden todistamisessa. Eräänä ongelmana esitettiin myös, että tietyillä alkuarvoilla ja säännöillä muurahainen ei välttämättä saavuta haluttua paikkaa, jol-

loin algoritmi ei pääsisi loppuun saakka. Tätä ei kuitenkaan havaittu algoritmiin valituilla parametreilla ja algoritmista esitettiin myös variaatio, jossa tämä ei aiheuttaisi ongelmaa. Arvaamattomuudesta voi myös olla hyötyä, sillä se tekee muurahaisen liikkeiden ennustamisen hankalaksi myös hyökkääjälle.

Jatkotutkimus

Jos algoritmi halutaan ottaa käyttöön, tulisi päättää, millaisilla parametreilla muurahaista simuloidaan. Osa parametreista, kuten lohkon koko voidaan määrittää vakioiksi, mutta mahdollisimman monet parametrit voidaan myös jättää käyttäjän valittaviksi. Lohkon koko on melko helppoa päätellä datasta, joten valinnalla ei ole niin suurta merkitystä turvallisuuden kannalta. Lisäksi voidaan valita käytetäänkö klassista muurahaista, vai tilanmuutoskaavion suhteen satunnaistettua versiota, tai etukäteen valittua mahdollisimman epälineaarista tilanmuutoskaavioita. Jälkimmäisin olisi luultavasti paras, sillä lineaarisesti vaihtuvat tilat johtavat aliluvussa 4.5.3 esitettyyn riippuvuuteen vierekkäisten ruutujen suhteen. Käyttämällä epälineaarista tilanmuutoskaaviota data myös sekoittuu visuaalisesti huomattavan paljon nopeammin, kuten esitettiin kuvissa 6.10. Satunnainen tilanmuutoskaavio antaisi puolestaan lisää vaihtoehtoja avaimille, mutta toisaalta se johtaisi myös huonompien, lineaarisempien tilanmuutoskaavioiden käyttöön.

Nopeus

Algoritmi voidaan todeta varsin hitaaksi. Koska turvallisuus syntyy vasta melko pitkillä siirtomäärillä, algoritmi on melko hidas. Luvun 6.2.1 esimerkeissä kuvien salaamiseksi tehtiin tuhansia siirtoja jokaista tavua kohden, yhteensä jopa 45 miljardia siirtoa. Esimerkkitoteutus kykenee tekemään salausta noin 50 miljoonaa siirtoa sekunnissa Intel 6850K prosessorilla, joten kuvan salaus kestää useita minuutteja. Kovinkaan nopeaan viestienvaihtoon algoritmi ei siis sovellu.

7.1 Avoimet kysymykset

7.1.1 Systeemin tilojen saavutettavuus

Määritelmän 4.5.2 mukaan systeemi ei välttämättä saavuta kaikkia mahdollisia tiloja. Tämä oli havaittavissa vain lyhyillä säännöillä ruudukon ollessa pieni, mutta saattaa olla mahdollista, että vastaava tapahtuisi myös eri parametreilla. Tilojen määrän nostamisen jälkeen vastaavaa ilmiöitä ei enää löytynyt, mutta sellainen saattaa kuitenkin olla mahdollista. Jos näin on, voi olla mahdollista, että tiettyä dataa ei voida salata ollenkaan. Tämä voidaan todeta tunnetuksi ongelmaksi, joten algoritmia ei välttämättä kannata käyttää sellaisissa tilanteissa, joissa toimintavarmuus on oleellista.

Kuten aiemminkin mainittiin, tämä ongelma voidaan kiertää tallentamalla myös muurahaisen asento datan perään. Muurahainen käy varmasti kaikissa ruuduissa liikkumisensa aikana, joten salauksen lopussa voidaan vain simuloida muurahaista, kunnes se saapuu tiettyyn ruutuun ja tallentaa muurahaisen asento datan perään. Näin hyökkääjä tietää asennon, mutta kokeiltavana on edelleen L^2 eri sijaintia, joista salauksen purkaminen tulee aloittaa.

7.1.2 Satunnaiskävelyn saavuttamat tilat

Yrittäessä selvittää, montako erilaista tilaa on mahdollista saavuttaa $m \cdot m$ kokoisella ruudukolla, tehtiin satunnaiskävelyä $3 \cdot 3$ kokoisella ruudukolla. Aina siirtyessään ruutuun, ruudun tilaa korotettiin yhdellä ja otettiin modulo sallittujen tilojen määrällä. Tämän jälkeen siirryttiin satunnaiseen suuntaan joko vaakasuoraan, tai pystysuoraan. Reunojen yli siirryttyä hypätään vastakkaiselle puolelle. Tämä on käytännössä sama kuin muurahainen, mutta suunta valitaan satunnaisesti, sen sijaan että se valittaisiin muurahaisen alla olevan ruudun perusteella. Jokaisen siirron jälkeen tila tallennettiin, kunnes saavutettiin mielivaltainen luku. Tämän jälkeen tutkittiin tallennettua listaa ja todettiin, onko kaikki eri tilat käyty läpi. Havainto oli mielenkiintoinen. Ruudukolla tehty satunnaiskävely saavutti

kaikki eri tilat, jos yhden solun tilojen määrä on pariton. Jos taas ruudun tilojen määrä on parillinen ja teoreettinen koko ruudukon tilojen määrä on s, saavutettiin ainoastaan $s\frac{3}{16}$ tilaa. Testaus tehtiin ainoastaan $3 \cdot 3$ ruudukolla, sillä $4 \cdot 4$ ja sitä suuremmilla ruudukoilla laskenta kestäisi aivan liian pitkään. Aiemmin todettiin, että muurahainen ei ollut kokeiltaessa saavuttanut kaikkia mahdollisia tiloja. Avoimeksi kysymykseksi jää vielä selvittää, onko tässä kappaleessa käsitelty havainto oleellinen laskettaessa teoreettista tilojen määrää.

Lähdeluettelo

[1] Dennis Luciano ja Gordon Prichett. Cryptology: From caesar ciphers to public-key cryptosystems. *The College Mathematics Journal*, 18(1):2–17, 1987.

[2] Ronald L Rivest, Adi Shamir ja Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[3] Savita Mohurle ja Manisha Patil. A brief study of wannacry threat: Ransomware attack 2017. *International Journal of Advanced Research in Computer Science*, 8(5), 2017.

[4] J-S Coron. What is cryptography? *IEEE security & privacy*, 4(1):70–73, 2006.

[5] Kerckhoffs Auguste. La cryptographie militaire. *Journal des sciences militaires*, 9:538, 1883.

[6] Claude E Shannon. Communication theory of secrecy systems. *Bell system technical journal*, 28(4):656–715, 1949.

[7] Whitfield Diffie ja Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

[8] Claude Shannon. A Mathematical Theory of Cryptography. Memorandum MM 45-110-02, syyskuu 1945.

[9] Mihir Bellare ja Phillip Rogaway. The exact security of digital signatures-How to sign with RSA and Rabin. Teoksessa *International Conference on the Theory and Applications of Cryptographic Techniques*, ss. 399–416. Springer, 1996.

[10] Robert Zuccherato. Root certificate management system and method, tammikuu 16 2003. US Patent App. 09/906,504.

[11] Frank Stevenson. Cryptanalysis of contents scrambling system. 1999.

[12] Karen Scarfone, Wayne Jansen ja Miles Tracy. Guide to general server security. *NIST Special Publication*, 800(s 123), 2008.

[13] Jovan Dj Golić. Linear statistical weakness of alleged RC4 keystream generator. Teoksessa *International Conference on the Theory and Applications of Cryptographic Techniques*, ss. 226–238. Springer, 1997.

[14] Lars R Knudsen. Block Ciphers—a survey. Teoksessa *State of the Art in Applied Cryptography*, ss. 18–48. Springer, 1998.

[15] Marc Martinus Jacobus Stevens et al. *Attacks on hash functions and applications*. Mathematical Institute, Faculty of Science, Leiden University, 2012.

[16] Eli Biham ja Adi Shamir. *Differential cryptanalysis of the data encryption standard*. Springer Science & Business Media, 2012.

[17] Andrey Bogdanov, Dmitry Khovratovich ja Christian Rechberger. Biclique cryptanalysis of the full AES. Teoksessa *International Conference on the Theory and Application of Cryptology and Information Security*, ss. 344–371. Springer, 2011.

[18] Yuval Yarom ja Katrina Falkner. FLUSH+ RELOAD: a high resolution, low noise, L3 cache side-channel attack. Teoksessa *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, ss. 719–732. 2014.

[19] N Pathak, Apurva Pawar ja Balaji Patil. A survey on keylogger: A malicious attack. *International Jourcal of Advanced Research in Computer Engineering and Technology*, 2015.

[20] Jean-Jacques Quisquater ja François-Xavier Standaert. Exhaustive key search of the DES: Updates and refinements. *SHARCS 2005*, 2005.

[21] Jack Wells, Buddy Bland, Jeff Nichols et al. Announcing supercomputer summit. Tekninen raportti, Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States), 2016.

[22] A Ray Miller. The cryptographic mathematics of enigma. *Cryptologia*, 19(1):65–80, 1995.

[23] Kris Gaj ja Arkadiusz Orłowski. Facts and myths of enigma: Breaking stereotypes. Teoksessa *International Conference on the Theory and Applications of Cryptographic Techniques*, ss. 106–122. Springer, 2003.

[24] Michael Case. A beginners guide to the general number field sieve (2003).

[25] Joan Daemen ja Vincent Rijmen. The block cipher Rijndael. Teoksessa *International Conference on Smart Card Research and Advanced Applications*, ss. 277–284. Springer, 1998.

[26] Vincent Rijmen. Practical-Titled Attack on AES-128 Using Chosen-Text Relations. *IACR Cryptology ePrint Archive*, 2010:337, 2010.

[27] Jarkko Kari. Reversibility of 2D cellular automata is undecidable. *Physica D: Nonlinear Phenomena*, 45(1-3):379–385, 1990.

[28] Pratibha Sharma, Manoj Diwakar ja Niranjan Lal. Edge detection using Moore neighborhood. *International Journal of Computer Applications*, 61(3), 2013.

[29] Jarkko Kari. Reversible cellular automata. Teoksessa *International Conference on Developments in Language Theory*, ss. 57–68. Springer, 2005.

[30] Fred C Hennie. One-tape, off-line Turing machine computations. *Information and Control*, 8(6):553–578, 1965.

[31] B Jack Copeland. The church-turing thesis. 1997.

[32] Ozgur Yilmaz. Reservoir computing using cellular automata. *arXiv preprint arXiv:1410.0162*, 2014.

[33] Paul Rendell. Turing universality of the game of life. Teoksessa *Collision-based computing*, ss. 513–539. Springer, 2002.

[34] Turlough Neary ja Damien Woods. P-completeness of cellular automaton Rule 110. Teoksessa *International Colloquium on Automata, Languages, and Programming*, ss. 132–143. Springer, 2006.

[35] Carter Bays. Introduction to cellular automata and Conway's Game of Life. Teoksessa *Game of Life Cellular Automata*, ss. 1–7. Springer, 2010.

[36] Matthew Cook. Universality in elementary cellular automata. *Complex systems*, 15(1):1–40, 2004.

[37] Mark D Niemiec. Synthesis of Complex Life Objects from. *New Constructions in Cellular Automata*, s. 55, 2003.

[38] Christopher G Langton. Studying artificial life with cellular automata. *Physica D: Nonlinear Phenomena*, 22(1-3):120–149, 1986.

[39] Anahi Gajardo, Andre Moreira ja Eric Goles. Complexity of Langton's ant. *Discrete Applied Mathematics*, 117(1-3):41–50, 2002.

[40] Anahí Gajardo, Eric Goles ja Andrés Moreira. Generalized langton's ant: Dynamical behavior and complexity. Teoksessa *Annual Symposium on Theoretical Aspects of Computer Science*, ss. 259–270. Springer, 2001.

[41] Ian Stewart. The ultimate in anty-particles. *Scientific American*, 271(1):104–107, 1994.

[42] David Gale, James Propp, Scott Sutherland et al. Further travels with my ant. *arXiv preprint math/9501233*, 1995.

[43] Paul Dorbec ja Anahi Gajardo. Langton's flies. *Journal of Physics A: Mathematical and Theoretical*, 41(40):405101, 2008.

[44] Diego Maldonado, Anahí Gajardo, Benjamin Hellouin De Menibus et al. Nontrivial turmites are turing-universal. *arXiv preprint arXiv:1702.05547*, 2017.

[45] Majid Vafaei Jahan ja Faezeh Khosrojerdi. Text Encryption Based on Glider in the Game of Life. *International Journal of Information Science*, 6:20–27, 2016.

[46] Marco Tomassini ja Mathieu Perrenoud. Cryptography with cellular automata. *Applied Soft Computing*, 1(2):151–160, 2001.

[47] Franciszek Seredynski, Pascal Bouvry ja Albert Y Zomaya. Cellular automata computations and secret key cryptography. *Parallel Computing*, 30(5-6):753–766, 2004.

[48] Georges Marsaglia. DIEHARD: a battery of tests of randomness (1996). *See http://stat. fsu. edu/geo/diehard. html*, 17:1–6, 2015.

[49] George Marsaglia, Wai Wan Tsang et al. Some difficult-to-pass tests of randomness. *Journal of Statistical Software*, 7(3):1–9, 2002.

[50] Stephen Wolfram. Cryptography with cellular automata. Teoksessa *Conference on the Theory and Application of Cryptographic Techniques*, ss. 429–432. Springer, 1985.

[51] Willi Meier ja Othmar Staffelbach. Analysis of pseudo random sequences generated by cellular automata. Teoksessa *Workshop on the Theory and Application of of Cryptographic Techniques*, ss. 186–199. Springer, 1991.

[52] Xingyuan Wang ja Dapeng Luan. A novel image encryption algorithm using chaos and reversible cellular automata. *Communications in Nonlinear Science and Numerical Simulation*, 18(11):3075–3085, 2013.

[53] Manish Kumar, Sunil Kumar, Rajat Budhiraja et al. Intertwining logistic map and Cellular Automata based color image encryption model. Teoksessa *2016 international conference on computational techniques in information and communication technologies (ICCTICT)*, ss. 618–623. IEEE, 2016.

Liite A

Sääntö 1

Satunnaisesti generoitu esimerkkisääntö on seuraava:

```
RLLLLLLRRRLLRLRRRRRLLRLLLLRRLRRR
RLLRLRLRRLRLLRRLRRLLLLRRRRLLRLLL
LRLLLLRLRLLLLRLLLRLRRLLLRLRLLRRR
RRLLLLLRLLLLRLRRRLRLRLLLLLLRRLRL
LLLLRRRLRRRRRLLLLRRLLRRRRRRLLLRL
LRRLLRRLRRRLRLRLLRRRRRLRRRLLLLRR
LLRRLLLLRRLLRLRLLRRLLLRRRRLRLRLR
RRRRLLLLRLRLLRRLLRRRLRRRRRRLRRLR
```

Sääntö saadaan antamalla javan versiossa 11 satunnaisgeneraattorin siemeneksi 1 ja suorittamalla seuraava lähdekoodi

```
Random r = new Random(1);
String sääntö = "";
while (sääntö.length() < 256){
  sääntö += r.nextBoolean() ? "R" : "L";
}
```