SAVONIA

OPINNÄYTETYÖ - AMMATTIKORKEAKOULUTUTKINTO
TEKNIIKAN JA LIIKENTEEN ALA

MEDIKAALILAITTEEN TIEDOSTOJÄRJESTELMÄN SALAUS

TEKIJÄ: Jari Kinnunen

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Tutkinto-ohjelma Sähkö- ja automaatiotekniikan tutkinto-ohjelma	
Työn tekijä Jari Kinnunen	
Työn nimi Medikaalilaitteen tiedostojärjestelmän salaus	
Päiväys 2.5.2021	Sivumäärä/Liitteet 48
Toimeksiantaja/Yhteistyökumppani Bittium Biosignals Oy	

Tiivistelmä

Mukana kannettavan medikaalilaitteen SD-kortti on tärkeää suojata salauksella, jotta voidaan estää pääsy sille tallennettuihin tietoihin, mikäli laite häviää tai joutuu vääriin käsiin. Opinnäytetyön tavoitteena oli tutkia keinoja tällaisen sulautetun medikaalilaitteen SD-kortin tiedostojärjestelmän salaukselle.

Työssä käytiin läpi salausta yleisesti, sekä tutkittiin erilaisia salausmenetelmiä ja algoritmeja. Asymmetrisistä salausmenetelmistä käytiin läpi muutama esimerkki, jonka jälkeen keskityttiin enemmän symmetrisiin lohkosalausmenetelmiin ja niiden toimintatiloihin. Symmetrisistä lohkosalausmenetelmistä ja niiden tiloista keskityttiin tarkemmin laitteen ja tutkimuksen perusteella potentiaalisimpiin vaihtoehtoihin.

Opinnäytetyön tutkimuksen tuloksena löytyi sopiva vaihtoehto laitteen SD-kortin salaukselle, sekä saatiin tarkempaa yleistietoa tästä salausmenetelmästä ja sen algoritmin toteutuksesta. Sopivaksi todettua algoritmia ei sellaisenaan testattu laitteen firmwaressa, vaan tehtiin yksinkertaisempi XOR-salaustesti, joka todisti osittain algoritmin konseptin.

Avainsanat
Tiedostojärjestelmä, salaus, sulautettu laite

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport	
Degree Programme Degree Programme in Electrical and Automation Engineering	
Author Jari Kinnunen	
Title of Thesis File System Encryption on Medical Device SD Card	
Date May 2, 2021	Pages/Appendices 48
Client Organisation /Partner Bittium Biosignals Oy	

Abstract

For a portable medical device, it is important that the SD card of the device is protected with an encryption so that the information on the card is not compromised in case of theft or lost device. The objective of this thesis was to explore means of encrypting the SD card file system of this kind of embedded medical device.

The encryption in general was studied as well as various encryption methods and algorithms were researched. Asymmetric encryption was reviewed with a couple of examples followed by a detailed study on symmetric block cipher methods and their modes of operation. Concerning the symmetric block ciphers and their modes, the study focused more on the most potential options based on the device and research.

As a result of the research, a suitable solution for the encryption of the SD card was found. Also, more specific information was gathered about this encryption method and the ways to implement the algorithm. The found algorithm was not tested as it is on the official firmware of the device, but a simpler XOR encryption test was performed, which partially proved the concept of the algorithm.

Keywords
File system, encryption, embedded device

ESIPUHE

Haluaisin kiittää Bittium Biosignals Oy:tä mielenkiintoisesta opinnäytetyön aiheesta, sekä erityisesti yrityksen firmware tiimin loistavasta työyhteisöstä ja tuesta. Opinnäytetyön aikana pääsin tutustumaan syvemmin oman alani yrityksen toimintaan ja työskentelytapoihin. Työn aikana pääsin hyödyntämään merkittävästi jo opiskelun aikana saatuja taitoja sekä kasvattamaan niitä ammatillisesti oikeaan suuntaan.

Suuret kiitokset myös perheelleni, ystävilleni, avopuolisolleni ja hänen perheelleen mittaamattoman arvokkaasta tuesta tällä koko opintojeni matkalla.

Kuopiossa (2.5.2021)

Jari Kinnunen

SISÄLTÖ

1 JOHDANTO

Viime vuosien aikana sulautettujen laitteiden ja niiden järjestelmien sekä sovellusten turvallisuus, sen huomioiminen ja kehittäminen on ollut kasvussa ja yhä tärkeämpää. Tämän opinnäytetyön tarkoitus on tutkia, vertailla ja löytää mahdollisuuksia sulautetun medikaalilaitteen ja sen sd-kortin tiedostojärjestelmän salaukselle, sekä lopulta mahdollisuuksien mukaan testata toteutusta. Opinnäytetyön toimeksiantaja on Bittium Biosignals Oy Kuopiossa ja tämä opinnäytetyön aihe syntyi yrityksen Firmware osaston toimesta liittyen heidän laitteiden ominaisuuksiin. Medikaalilaitetta käytetään ihmisen biosignaalien mittaamiseen. Lähtökohtaisesti tällaisen medikaalilaitteen SD-kortilla olevan datan (Data at rest) tietoturvalle on kaksi vaihtoehtoa: fyysisesti estää pääsy laitteen SD-kortille tai tekemällä SD-kortin data lähes mahdottomaksi tulkita ymmärrettävästi salauksen avulla. Koska kyseessä on mukana kannettava medikaalilaite, jota potilas itse käyttää biosignaalien mittaamiseen, ei ensimmäinen vaihtoehto ole käytännössä mahdollista toteuttaa. Riskinä siis on esimerkiksi se, että tällaisen medikaalilaitteen voi purkaa ja täten pääsy sen sisällä olevalle SD-kortille on suhteellisen helppoa. Tiedostojärjestelmän salauksella pystytään suojelemaan laitteen SD-kortille kerättyä tietoa ja dataa laitteen joutuessa vääriin käsiin. Salaus suojaisi laitteella olevaa dataa myös esimerkiksi siinä tapauksessa, että ulkopuolinen taho haluaisi manipuloida kerättyä tietoa. Haasteena ovat sulautettuna järjestelmänä medikaalilaitteen rajalliset resurssit sekä sen tarkoitus pysyä mahdollisimman energiatehokkaana virrankulutuksen ja akun käyttöiän suhteen.

1.1 Bittium

Bittium on luotettavien ja turvallisten viestintä- ja liitettävyysratkaisujen kehittämiseen erikoistunut yritys. Sen tytäryhtiö Bittium Biosignals Oy on terveydenhuollon teknologian tuotteita ja palveluita biosignaalien mittaamiseen tarjoava yritys Kuopiossa. Heidän medikaalilaitteet ja palvelut keskittyvät ihmisen biosignaalien mittaamiseen kardiologian, neurologian, kuntoutuksen, työterveyden ja urheilulääketieteen osa-alueilla. Bittiumin kardiologian laitteet ja ohjelmistot käsittävät esimerkiksi holtermittaukset, sydämen telemetrian ja sydämen kuntoutuksen. EEG puolelta löytyy ratkaisuina esimerkiksi TMS-EEG ja fMRI-EEG ensiavun ja tehohoidon tarpeisiin. Bittium Oyj on listattuna NASDAQ Helsingissä ja sen vuoden 2019 liikevaihto oli 75,2 miljoonaa euroa ja liikevoitto 6,3 miljoonaa euroa. (Bittium, 2021.)

1.2 Lyhenteet ja määritelmät

ADC	Analog to Digital Converter, AD-muunnin
AES	Advanced Encryption Standard
ANSI	American National Standards Institute
Biosignaali	Esimerkiksi ihmisestä mitattava biosähköinen signaali, joita ovat esimerkiksi EEG ja EKG (Jauhiainen, 2009).
CDC	Communications Device Class (USB)
CPU	Central Processing Unit
DAC	Digital to Analog Converter, DA-muunnin
FIPS	U.S. Federal Information Processing Standards
ISO	International Organization for Standardization
Data at rest	Dataa, joka on tietokoneen/laitteen fyysisellä massamuistilla missä tahansa digitaalisessa muodossa.
DARP	Data at rest protection
EKG	Elektrokardiografia eli sydänfilmi. Perustuu sydämen sähköisen toiminnan eli sen supistumista säätelevien heikkojen sähköimpulssien mittaamiseen. (Pertti Mustajoki, 2008).
EEG	Elektroenkefalografia eli aivosähkökäyrätutkimus, joka mittaa aivojen sähköistä toimintaa (Aivosähkökäyrätutkimus EEG, 2019).
FAT	File Allocation Table
Firmware	Laitteen/laitteiston ohjelmisto tai sen osa, joka ohjaa laitteen perustoimintoja sekä sisältää tarvittavat käskyt kommunikointiin laitteen muiden komponenttien kanssa (Christensson, 2006).
HID	Human Interface Device Class (USB)
Kardiologia	Lääketieteen osa-alue, joka tutkii sydämen ja verenkiertoelimistön toimintaa ja sairauksia, kuten rytmihäiriöt ja sydämen vajaatoiminta (Kardiologia, 2020).
LBA	Logical Block Addressing
MAI	Media Access Interface

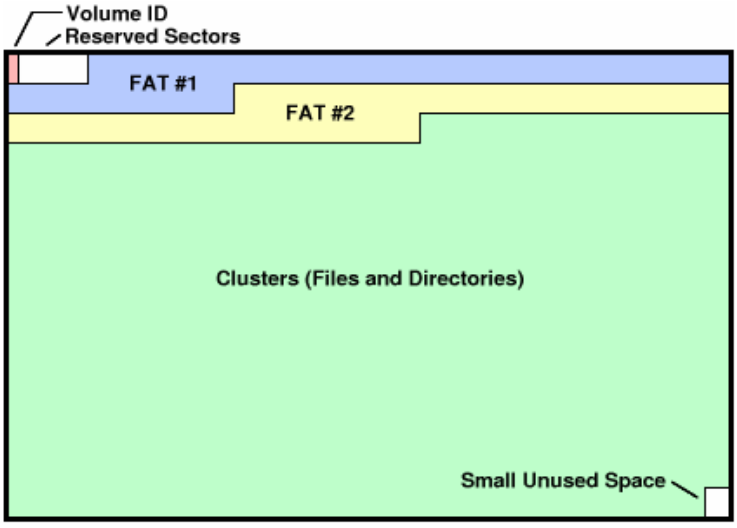
MIPS-vuosi	Salauksessa usein käytettävä yksikkö, joka viittaa tietokoneen, joka toimii miljoonalla operaatiolla sekunnissa (1 MIPS), yhden vuoden aikana suorittamaan työhön (Kaliski, 2005).
MSC	Mass Storage Device Class (USB)
Neurologia	Lääketieteen osa-alue, johon kuuluu hermoston eli aivojen, selkäytimen ja ääreishermoston sekä lihassairauksien tutkiminen, hoito ja kuntoutus (Orton).
NSA	National Security Agency, kansallinen turvallisuusvirasto
RAM	Random Access Memory, keskusmuisti
RISC	Reduced Instruction Set Computer
ROM	Read-only-memory
SPI	Serial Peripheral Interface
Sulautettu järjestelmä	Tiettyyn tarkoitukseen ohjelmoitu laite tai laitteisto, jota ohjaa yleensä mikroprosessori tai mikrokontrolleri (omni sci, 2020).
Watchdog	Elektroninen tai ohjelmallinen ajastin, jota käytetään toimintahäiriöiden havaitsemiseen ja palauttamiseen esimerkiksi sulautetuissa järjestelmissä (Barr, 2001).
Bitwise XOR	Bittikuviolle suoritettava operaatio, jossa suoritetaan ehdoton tai -operaatio kullekin vastaavien bittien parille. Operaatio tuottaa 1 vain silloin kun toinen biteistä on 1, muuten 0.

2 TAUSTAA

2.1 FAT ja exFAT tiedostojärjestelmät

File Allocation Table eli FAT, on Microsoftin vuonna 1977 kehittämä tiedostojärjestelmä ja siitä edelleen kehitettyjä versioita, kuten FAT32, käytetään vieläkin usein esimerkiksi muistikorteissa. FAT on saanut nimensä taulukosta lohkonumeroita, joka sijaitsee tiedostojärjestelmän alkupäässä ja näitä on usein kaksi kopiota luotettavuussyistä. FAT tiedostojärjestelmän etuja ovat sen yksinkertaisuus ja suhteellisen helppo implementointi.

Levy, joka on alustettu FAT tiedostojärjestelmällä, koostuu klustereista eli tilanvarausyksiköistä, jotka koostuvat yhdestä tai useammasta 512 tavun mittaisesta lohkosta, riippuen levyn koosta. Koska yksi varausyksikkö tai lohko on pienin tiedoston tallentamiseen varattava tila, yhden tavun mittainen tiedosto vie koko klusterin kokoisen tilan. Ensimmäinen sektori tiedostojärjestelmässä on varattu käynnistyslohkolle (MBR, Master Boot Record), joka sisältää tiedon lohkokoosta, FAT taulukoiden määrästä ja koosta, sekä juurihakemiston sijainnista ja koosta. Tämän jälkeen sijaitsevat FAT taulukot, jonka indeksiin talletettu luku on seuraavan tiedostolle varatun lohkon numero ja jokainen indeksi taulukossa vastaa yksi yhteen tiedostojärjestelmän lohkoa. Tilanvaraustaulukoita seuraa levyn päähakemisto, joka sisältää kustakin tiedostosta tiedoston nimen, koon, luomisajan ja päivämäärän, attribuutit ja tiedoston ensimmäisen lohkon numeron. Näiden alueiden jälkeen sijaitsee levyn varsinainen tiedostoalue, jonka alkuosoite on aina klusteri kaksi. Kuvassa 1 on yksinkertaisesti kuvattuna FAT tiedostojärjestelmän asettelu, eli alussa käynnistyslohko ja joitakin varattuja sektoreita, sitten FAT taulukot ja näiden jälkeen itse data, joka on järjestetty klustereihin.



KUVA 1. FAT tiedostojärjestelmän asettelu (Stoffregen, 2005)

Datan säilyvyyden ja eheyden kannalta FAT taulukon päivittäminen on tärkeää, mutta myös hidasta. Hidasta siitä tekee esimerkiksi sen, että kovalevyn lukupäiden on kuljettava aina takaisin aseman

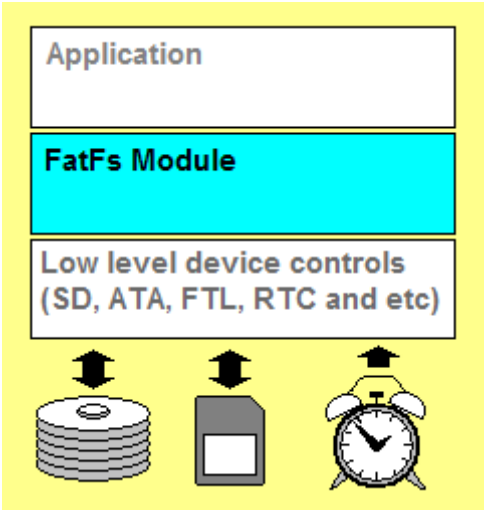
loogiseen nolla uraan, kun taulukkoa päivitetään. (Microsoft, 2020; Stoffregen, 2005; Wikipedia, 2020.)

FAT:n ja FAT32:n seuraaja exFAT eli extended File Allocation Table, kuuluu myös FAT tiedostojärjestelmä perheeseen ja sen suunnittelun tavoitteita ovat olleet FAT tiedostojärjestelmien yksinkertaisuuden säilyttäminen, sallien suuret tiedostot ja tallennuslaitteet, sekä sisällyttää laajennettavuus tulevia innovaatioita varten. Se on yksi standardeista tiedostojärjestelmistä irrotettaville tiedostovälineille.

ExFat tiedostojärjestelmä mahdollistaa klustereiden koon aina 32MB asti ja se käyttää 64 bittiä kertoakseen tiedoston koon. Tämä mahdollistaa tuen isommille tiedostoille kuin esimerkiksi FAT:lla (4 GB) ja suuremmille tallennuslaitteille. (Microsoft, 2019.)

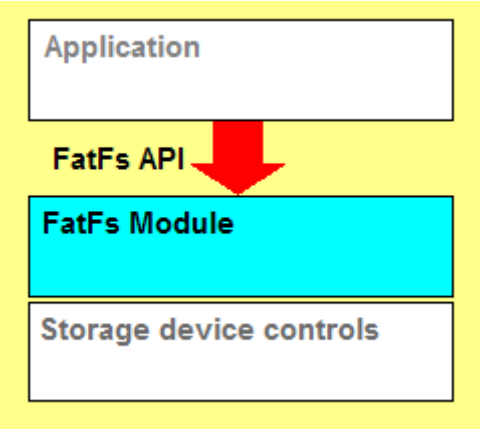
2.2 FatFs tiedostojärjestelmä

Laitteen tiedostojärjestelmänä toimii FatFs, joka on geneerinen FAT/exFAT tiedostojärjestelmämoduuli ja se on kirjoitettu noudattaen ANSI C (C89) C-ohjelmointikielen standardia. FatFs on tarkoitettu pienille sulautetuille laitteille/järjestelmille, joissa käytettävät resurssit ovat rajalliset. Se on myös täysin erotettu levyn I/O (input/output) kerroksesta, joka mahdollistaa sen toiminnan riippumatta alustasta, jolla sitä käytetään (KUVA 2).



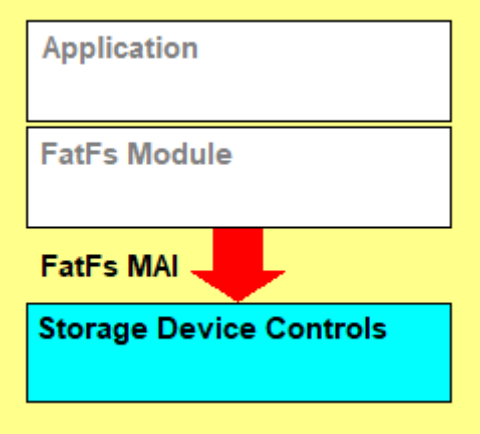
KUVA 2. FatFs moduuli (ChaN, 2020).

Ohjelmien toteuttamiselle FatFs sisältää tarvittavat tiedostojärjestelmä-funktiot tiedostojen käytölle, lukemiselle, hallitsemiselle sekä levyn osioinnille ja systeemin konfiguroinnille. Näin FatFs tarjoaa täydellisen API (Application Programming Interface) kerroksen toteutettavalle sovellukselle (KUVA 3).



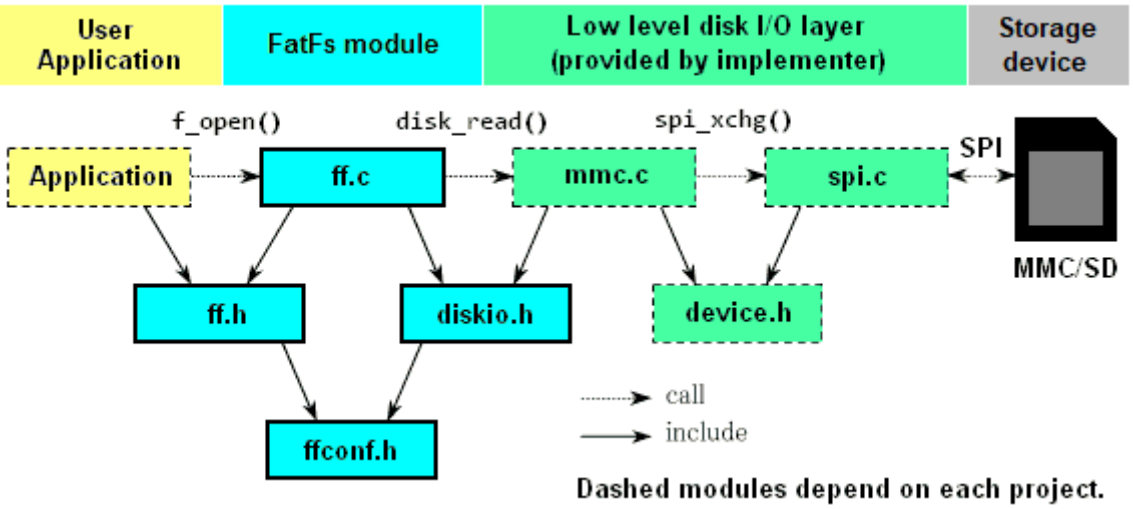
KUVA 3. FatFs API (ChaN, 2020).

FatFs:n ollessa irtonainen mistään alustasta tai tallennusmediasta, on se omana tiedostojärjestelmä kerroksenaan kokonaan erotettu fyysisistä laitteista, kuten muistikorteista ja kiintolevyistä. Massamuistilaitteen kontrollointimoduuli ei ole missään muotoa osa FatFs moduulia, joten sen käyttäjän on huolehdittava itse tämän toteutuksesta. FatFs kontrolloi massamuistilaitteita yksinkertaisilla median käyttöliittymä-funktioilla (Media Access Interface, MAI) (KUVA 4). Ainoa vaatimus on, että kyseessä on kiinteä kokoisissa lohkoissa luettu/kirjoitettu lohkolaite, johon on pääsy levyn I/O MAI-funktioilla. (ChaN, 2020.)



KUVA 4. FatFs MAI (ChaN, 2020).

Kuvassa 5 on esimerkkinä tyypillinen FatFs:n konfiguraatio sulautetussa järjestelmässä. Kuvassa näkyy eri kerrosten riippuvuudet, esimerkiksi eri c-kielisten tiedostojen ja otsikkotiedostojen (.h-pääte) kautta ja koko ketju sovelluksesta (User Application) FatFs moduulin kautta alemman kerroksen levyn I/O kerrokseen ja massamuistilaitteelle asti.



KUVA 5. Tyypillinen FatFs konfiguraatio sulautetussa järjestelmässä (ChaN, 2020).

Taulukossa 1 on listattuna FatFs:n asettamia rajoituksia toteutuksen kannalta.

TAULUKKO 1. FatFs rajoitukset (ChaN, 2020).

Tiedostojärjestelmän tyyppi	FAT, FAT32(rev0.0) ja exFAT(rev1.0)
Loogisten asemien lukumäärä	10 loogiseen asemaan asti
Sektorin koko	512, 1024, 2048 ja 4096 tavua
Minimi loogisen aseman koko	128 sektoria
Maksimi loogisen aseman koko	2^{32} – 1 sektoria 32-bittisessä LBA:ssa, virtuaalisesti rajoittamaton 64-bittisessä LBA:ssa exFAT:lla
Maksimi tiedostokoko	2^{32} – 1 tavua FAT loogisella asemalla, virtuaalisesti rajoittamaton exFAT loogisella asemalla
Klusteri koko	128 sektoriin asti FAT loogisella asemalla ja 16 MB asti exFAT loogisella asemalla

2.3 Laitteen mikroprosessori

Laitteessa mikrokontrollerina toimii Texas Instruments:n valmistama MSP430, jonka arkkitehtuuri on suunniteltu mahdollisimman energiatehokkaaksi ja virrankulutukseltaan optimoiduksi esimerkiksi kannettavia mittaussovelluksia varten. Suorittimena MSP430:ssa on 16-bittinen RISC arkkitehtuurin CPU. Taulukossa 2 on listattuna MSP430:n perusominaisuuksia.

TAULUKKO 2. MSP430 perusominaisuuksia

Kellotaajuus	20MHz
Flash muisti	512 KB
SRAM-muisti	64 KB
ADC	12-bittinen
2x DAC	12-bittinen
I/O pinnit	74
Käyttöjännite	3.6V – 1.8V
Rekisterit	16-bittiset
UART	3
USB	Kyllä
I2C	3
SPI	6

MPS430:n matalan virrankulutuksen ja tätä kautta sen sopiminen tällaisia ominaisuuksia vaativiin sulautettuihin laitteisiin, mahdollistaa suurelta osin sen arkkitehtuuriin suunnitellut eri pienitehoiset tilat. Näitä tiloja ovat esimerkiksi:

- Aktiivinen tila (Active mode, AM): kaikki systeemin kellot ovat aktiviisia: 295 uA/MHz 8 mega-hertsissä, 3.0 V, flash ohjelman suorittaminen
- Valmiustila (Standby mode, LPM3): Watchdog ajastin, toimitusvalvoja toiminnassa, täydellinen RAM-muistin säilytys, nopea herätys: 2.0 uA 2.2 V:ssa, 2.2 uA 3.0 V:ssa
- Sammunut, reaaliaikakello-tila (Shutdown, real-time clock (RTC) mode (LPM3.5)): Sammutus tila, reaaliaikakello aktiivinen: 1.1 uA 3.0 V:ssa
- Sammunut tila (Shutdown mode (LPM4.5)): 0.45 uA 3.0 V:ssa

Digitaalisesti kontrolloitu oskillaattori (DCO) mahdollistaa laitteen heräämisen näistä pienitehoisista tiloista aktiiviseen tilaan jopa 3 mikrosekunnissa. (Texas Instruments, 2020.)

2.3.1 MSP430 USB API

Laitteen USB toiminnallisuudet ja niiden kehittäminen on toteutettu käyttämällä Texas Instruments:n tarjoamaa valmista MSP430 USB API pakettia. Paketti tarjoaa kaikki tarvittavat ohjelmointirajapinnat USB toimintojen toteuttamiseen, kuten MSC, HID ja CDC luokat.

USB-laitteen tiedot tallennetaan sen ROM-muistin segmentteihin. Nämä segmentit ovat kuvaajia (engl. descriptor), joita käytetään laitteen tunnistamiseen kuuluvaksi johonkin tiettyyn määrään USB-luokkia. (USB Implementers Forum, 2001.)

USB MSC luokka on joukko protokollia, jotka on määritellyt USB Implementers Forum. Ne mahdollistavat USB-laitteen kommunikoinnin isäntä (engl. host) laitteen (esimerkiksi tietokone) kanssa, kuten tiedostojen siirtämisen laitteen ja isännän välillä. Isännälle USB-laite näkyy ulkoisena kovalevynä. (USB Implementers Forum, 2010.)

USB HID on osa USB-spesifikaatiota, joka määrittelee laiteluokan ihmisen käyttämille liitäntälaitteille, joita ovat esimerkiksi näppäimistö ja hiiri. HID luokan määrittely mahdollistaa uusien USB-pohjaisten laitteiden ja sovellusten suunnittelun ilman tarvetta kehittää mukautettuja laiteohjaimia. HID laite käyttää tietojen noutamiseen ja reitittämiseen omaa HID-luokan ajuria (engl. driver). Laitteen raportti kuvaajassa (engl. Report descriptor) on tiedot laitteen data protokollasta ja datan tyypistä. Raportti kuvaaja ladataan ja parsitaan HID-ohjaimen toimesta heti kun laite on kytketty ja tunnistettu. (USB Implementers Forum, 2001.)

2.4 Laitteen reaaliaikajärjestelmä

Laitteessa reaaliaikakäyttöjärjestelmänä on FreeRTOS, joka on Real Time Engineers Ltd:n kehittämä avoimen lähdekoodin reaaliaikakäyttöjärjestelmä sulautetuille laitteille. FreeRTOS on ilmainen ja tuettu, sekä vapaasti käytettävissä kaupallisissa tuotteissa ja sovelluksissa ilman mitään vaatimusta lähdekoodien julkaisuun. (The FreeRTOS Kernel.)

Käyttöjärjestelmän tyyppi määritellään esimerkiksi sen mukaan, miten käyttöjärjestelmässä hoidetaan eri tehtävien ja ohjelmien suoritusten järjestys. Käyttöjärjestelmän vuoronnuksen eli skeduloinnin (engl. scheduling) vaihtaessa nopeasti kunkin ohjelman välillä, syntyy illuusio ohjelmien samanaikaisesta suorituksesta, vaikka todellisuudessa jokaisella prosessorin ytimellä voi olla vain yksi suoritussäie kerrallaan tietyssä ajankohdassa.

Reaaliaikakäyttöjärjestelmässä (RTOS) skedulointi on suunniteltu niin, että se tarjoaa ennustettavissa olevan, deterministisen aikarajan noudattamisen. Sulautetuilla järjestelmillä on usein reaaliaikaiset vaatimukset eli järjestelmän on vastattava tiettyyn tapahtumaan tiukasti määritellyn ajan kuluessa. Tämä takuu reaaliaikaisten tiukkojen vaatimusten täyttämisestä voidaan antaa vain, jos käyttöjärjestelmän skedulointi on suunniteltu niin, että se on ennustettavissa ja siten determinististä. Tavallisesti reaaliaikainen ja deterministinen skedulointi saavutetaan niin, että käyttäjän annetaan määritellä jokaiselle suoritettavalle säikeelle oma prioriteetti. Näin järjestelmän skeduleri (engl. scheduler) tietää asetetun prioriteetin avulla minkä suoritettavan säikeen se suorittaa seuraavaksi. FreeRTOS:ssa suoritettavaa säiettä kutsutaan tehtäväksi (engl. task).

FreeRTOS on microkernel tyyppinen RTOS, joka koostuu yksinkertaisimmillaan vain kolmesta C-kielisestä tiedostosta eli se sisältää vain tärkeimmät ydin osat reaaliaikalaskennan toiminnallisuuksille, tehtävien väliselle viestinnälle ja tarvittavat synkronointi primitiivit. FreeRTOS ei siis sisällä esimerkiksi mitään laiteohjaimia, tiedostojärjestelmää tai protokollapinoja.

FreeRTOS ohjelmakoodi koostuu pääosin kolmesta eri alueesta: tehtävistä, kommunikaatiosta ja laitteistoon liittyvästä käyttöliittymästä. Tehtävä on käyttäjän määrittelemä funktio, jolle on määritetty tietty prioriteettitaso. Laitteistosta riippumattomat ja riippuvat koodit ovat FreeRTOS:ssa sisällytettynä ja se tukee useita kääntäjiä (esimerkiksi GCC, IAR) ja prosessoriarkkitehtuureja (ARM, PIC, 8051, x86).

Tehtävät käyttävät jonoja (engl. queue) datan jakamiseen ja keskenään kommunikointiin. Semaforeja ja mutexeja käytetään kriittisten resurssien hallintaan ja vuorottamiseen. Semafori on signaali tehtävältä toiselle, kun taas mutexia käytetään yhteisen resurssin suojelemiseksi, jotta vain yhdellä tehtävällä on pääsy tiettyyn resurssiin kerrallaan. (FreeRTOS.)

3 SALAUSALGORITMIT JA MENETELMÄT

Salaukseen, sen tehoon ja sen menetelmiin liittyy yleisesti salausavain, joka on tiedossa vain salattuun viestintään liittyvillä osapuolilla. Salauksessa puhutaan yleisesti ”selkeästä tekstistä” (engl. *plain text*), joka voi olla esimerkiksi lähetettävästä viestistä laiteohjelmiston päivitykseen, levyllä olevaa dataa tai mitä tahansa muuta, joka voidaan esittää bittivirtana. Salausprosessi muuntaa tämän tekstin salaustekstiksi (engl. *ciphertext*) ja salauksen purku takaisin selväksi tekstiksi. Salausmenetelmät voidaan jakaa kahteen kategoriaan, joita ovat symmetriset ja asymmetriset menetelmät. Asymmetrisessä eli julkisen avaimen salausmenetelmässä käytetään kahta eri, julkista ja yksityistä avainta salaukseen ja sen purkamiseen. Symmetrisessä menetelmässä käytetään samaa yksityistä avainta sekä viestin salaukseen, että salauksen purkuun. (Liikenne- ja viestintäministeriö, 2018).

Symmetriset salausmenetelmät perustuvat tietokoneen avulla tapahtuvaan bittien sekoittamiseen, kun taas asymmetriset enemmän matemaattiseen laskentaan. Salausmenetelmät jakaantuvat vielä yleisesti kahteen pääluokkaan, eli jonosalaukseen (engl. *stream cipher*) ja lohkosalaukseen (engl. *block cipher*). Lohkosalausta käytetään sekä symmetrisissä, että epäsymmetrisissä salausalgoritmeissa yleisesti enemmän. Lohkosalauksessa teksti jaetaan tietynmittaisiin lohkoihin, kuten 128 bittiä ja sitten salataan lohko kerrallaan. (Liikenne- ja viestintäministeriö, 2018).

Salausmenetelmiin liittyy edellä olevien ominaisuuksien lisäksi vielä tarkempia ominaisuuksia ja käsitteitä, joita ovat esimerkiksi sessio, avaintenvaihto (engl. *key exchange*) ja sessioavain (engl. *session key*). Sessio on lyhytkestoinen ajanjakso, jonka aikana kommunikoivat osapuolet vaihtavat viestejä ja sessioavain on avain, joka on voimassa vain yhden session ajan. Avaintenvaihto on prosessi, jonka aikana sovitaan yhteisestä avaimesta, jolla varsinainen viestintä tai data salataan. (Liikenne- ja viestintäministeriö, 2018).

4 ASYMMETRISET SALAUSMENETELMÄT

4.1 RSA

RSA (Rivest-Shamir-Adleman) on kehittäjiensä sukunimien mukaan nimetty asymmetrinen salausalgoritmi. Ron Rivest, Adi Shamir ja Leonard Adleman kuvasivat algoritmin julkisesti jo vuonna 1977 ja se on yksi vanhimmista salausalgoritmeista. RSA avaimet liittyvät toisiinsa niin, että salaus toteutetaan käyttämällä julkista avainta, jolloin se voidaan purkaa vain käyttämällä siihen liittyvää yksityistä avainta, jota ei voida käytännössä johtaa siihen liittyvästä julkisesta avaimesta. Tämä menettely käy myös toisinpäin, eli salataan yksityisellä avaimelle ja puretaan julkisella avaimella, jolloin yksityisen avaimen haltijalla on mahdollisuus esimerkiksi allekirjoittaa viestinsä niin, että julkisen avaimen haltijat voivat varmistua salatun viestin lähettäjästä. (R.L. Rivest; Wikipedia, 2021.)

RSA algoritmin tekniikka ja turvallisuus perustuu ajatukseen ja olettamukseen, että on vaikeaa jakaa tekijöihinsä kahden suuren alkuluvun tuloa. Julkinen avain koostuu kahdesta luvusta, joista toinen on kahden alkuluvun tulo. Yksityinen avain on myös johdettu näistä kahdesta samasta alkuluvusta. (Townsend, 2019; RSA Algorithm in Cryptography, 2021.)

Avainten luominen RSA algoritmille tapahtuu seuraavasti:

1. Valitaan kaksi erillistä alkulukua p ja q, jotka pidetään salassa ja lasketaan niiden tulo N, $N=p*q$. Lukua N käytetään tietynlaisena vakiona eli moduluksena julkiselle ja yksityiselle avaimelle. Luvun N pituus bitteinä on avaimen koko.
2. Valitaan kokonaisluku e, joka täyttää ehdot $1 < e < N$, sekä jolle $(p - 1)(q - 1)$ on suhteellinen alkuluku, eli ne ovat keskenään jaottomat.
3. Luvut N ja e muodostavat RSA:n julkisen avaimen.
4. Yksityinen avain d lasketaan luvuista p, q ja e. Matemaattinen riippuvuus lukujen välillä on

$$ed = 1 \, mod \, (p - 1)(q - 1) \qquad (1)$$

5. Tämä kaava (1) on euklidisen algoritmin peruskaava, jolle p ja q syötetään parametreinä.

Salaus tapahtuu niin, että on esimerkiksi jokin teksti P, jolla on jokin sovittu numeerinen arvo, n ja e ovat julkinen avain. Tästä saadaan $C = P^e \, (mod \, n)$, jossa C on saatu salausteksti. Salauksen purkaminen onnistuu vain vastaanottajan tiedossa olevalla yksityisen avaimen indeksillä d, eli $D = C^d \, (mod \, n)$.

RSA salauksen huonoja puolia sulautetussa järjestelmässä on, että se on suorituskyvyltään heikko, kun sen käyttämien avainten pituudet kasvavat liian suuriksi. Jotta RSA:lla saavutettaisiin sama suojaustaso kuin esimerkiksi AES-256 (256-bittinen avain) sessiossa, jossa avaimen hallintaan käytetään 512-bittistä elliptisen käyrän sessioavainta, olisi RSA toteutuksella käytettävä jopa 15360-bittisiä avaimia (KUVA 6). Tämä vaatisi laskennallisesti liikaa sulautetulta järjestelmältä. RSA on paljon käytössä esimerkiksi selaimissa ja VPN yhteyksissä, sekä yleisesti se sopii paremmin sellaiseen tilanteeseen, että meillä on fyysisesti tai jopa maantieteellisesti kaksi eri päätepistettä. (Townsend, 2019; Maletsky, 2020.)

Security Bits	Symmetric Encryption Algorithm	Minimum Size (Bits) of Public Keys	
		RSA	ECC
80	Skipjack	1024	160
112	3DES	2048	224
128	AES-128	3072	256
192	AES-192	7680	384
256	AES-256	15360	512

KUVA 6. Salauksen tason vertailua eri algoritmien avainten koko yhdistelmiin (Maletsky, 2020)

4.2 ECC

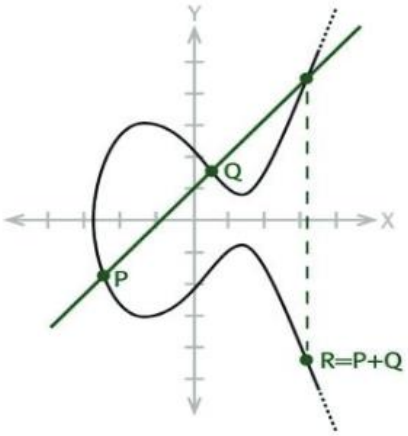
ECC (engl. *Elliptic-curve cryptography*) salaus perustuu matemaattiseen ongelmaan nimeltä ECDLP (engl. *elliptic curve discrete logarithm problem*) eli diskreetin logaritmin ongelma elliptisillä käyrillä. Tekniikan esittelivät ensimmäisinä erikseen Neal Koblitz ja Victor Miller vuonna 1985. (Vivek Kapoor, 2008.) Elliptinen käyrä määritellään yhtälöllä

$$y^2 + xy = x^3 + ax + b \qquad (2)$$

Tätä yhtälöä voidaan käyttää myös muodossa

$$y^2 = x^3 + ax + b \qquad (3)$$

Ratkaiseva ominaisuus elliptisellä käyrällä on se, että voidaan määrittää sääntö kahden käyrällä olevan pisteen yhteenlaskulle, josta saadaan kolmas piste, joka on myös käyrällä. Kuvassa 7 *P + Q = R* saadaan, kun piirretään jänne *P:n* ja *Q:n* läpi, jolloin tämä jänne lävistää elliptisen käyrän myös kolmannessa pisteessä. Tämä piste on *-R* ja *R* saadaan, kun *-R* peilataan x-akselin suhteen.



KUVA 7. Elliptinen käyrä, jossa *P + Q = R* (Vivek Kapoor, 2008)

Jotta yhteenlasku voidaan määritellä hyvin kahdelle pisteelle, on sisällytettävä vielä ylimääräinen nollapiste *0*, joka ei täytä elliptisen käyrän yhtälöä. Tämä 0 oletetaan olevan käyrän piste. Käyrän järjestys on käyrän erillisten pisteiden määrä mukaan lukien nollapiste. (Vivek Kapoor, 2008.) Nämä

koordinaatit valitaan äärellisestä kunnasta (engl. *finite field, Galois field*). Äärellinen kunta on matematiikassa kunta, jonka alkioiden lukumäärä on äärellinen. Kertalukua *q*, eli kunnan alkioiden lukumäärää, merkitään *GF(q)*. (Äärellinen kunta, 2016; Vivek Kapoor, 2008.)

Esimerkkinä salaukselle, henkilöt A ja B sopivat julkisesta elliptisestä käyrästä ja kiinteästä käyrän pisteestä *F*. A valitsee yksityisen satunnaisen kokonaisluvun A_k, joka on hänen yksityinen avaimensa, sekä julkiseksi avaimeksi käyrän pisteen $A_p = A_kF$. Henkilö B tekee samat toimenpiteet. Henkilö A haluaa lähettää salatun viestin henkilölle B. Henkilö A voi laskea A_kB_p ja käyttää saatua tulosta yksityisenä avaimena esimerkiksi lohkosalausmenetelmässä. B voi laskea saman numeron laskemalla B_kA_p, sillä

$$B_k A_p = B_k * (A_k F) = A_k * (B_k F) = A_k B_p \qquad (4)$$

Salauksen vahvuus perustuu siihen olettamukseen, että on vaikeaa laskea *k*, kun *F* ja *kF* on annettu. Hyvän turvallisuuden takaamiseksi, käyrä ja kiinteä piste *F* valitaan niin, että käyrän kiinteän pisteen *F* järjestys on suuri alkuluku, joka määritetään käyrän järjestyksestä, joka taas saadaan Schoofin algoritmista (engl. *Schoof's Algorithm*). Schoofin algoritmi on tehokas algoritmi elliptisten käyrien laskemiseen äärellisistä kunnista (Schoof's Algorithm, 2019). Nykytietämyksen mukaan, jos kiinteän pisteen järjestys *F* on *n*-bittinen alkuluku, kestää *k*:n laskeminen *kF*:stä ja *F*:stä karkeasti arvioiden $2^{n/2}$ operaatiota. (Vivek Kapoor, 2008.)

Elliptisten käyrien ja sitä kautta ECC:n käyttämisestä salausavainten laskemiseen tekee houkuttelevaksi avainten huomattavasti pienempi koko verrattuna esimerkiksi RSA menetelmään, saavuttaen kuitenkin sama suojauksen taso. Tietokoneiden suorituskyvyn ja sitä kautta laskentatehojen kasvaessa, säilyttääkseen saman suojaustason, RSA:lla toteutetun salausavainparin pituus eli bittien määrä kasvaa huomattavasti nopeammin kuin ECC:llä generoidun avainparin pituus (TAULUKKO 3). Menezes ja Jurisic kertovat tutkimuksessaan *Elliptic Curves and Cryptography, Strong digital signature algorithms* (Aleksander Jurisic, 1997), että RSA systeemissä riittävä salauksen taso saavutetaan 1024-bittisellä moduluksella, kun taas ECC:ssä riittää 160-bittinen modulus.

ECC on nopeampi ja vaatii vähemmän muistitilaa, kuin esimerkiksi vastaava RSA systeemi. Tämä tarkoittaa, että se on huomattavasti käytännöllisempi rajallisten resurssien systeemeissä, kuten sulautetut järjestelmät, missä nopeat operaatiot ovat välttämättömiä. Elliptisiä käyriä käytetään esimerkiksi logaritmipohjaisissa protokollissa, kuten ECDSA (engl. *Elliptic Curve Digital Signature Algorithm*) ja ECDH (engl. *Elliptic Curve Diffie-Helman*).

TAULUKKO 3. RSA ja ECC vertailua (Vivek Kapoor, 2008)

Murtamiseen kuluva aika (MIPS-vuosina)	RSA avaimen koko (bitteinä)	ECC avaimen koko (bitteinä)
10^4	512	106
10^8	768	132
10^{11}	1024	160
10^{20}	2048	210
10^{78}	21000	600

5 SYMMETRISET SALAUSMENETELMÄT

5.1 DES

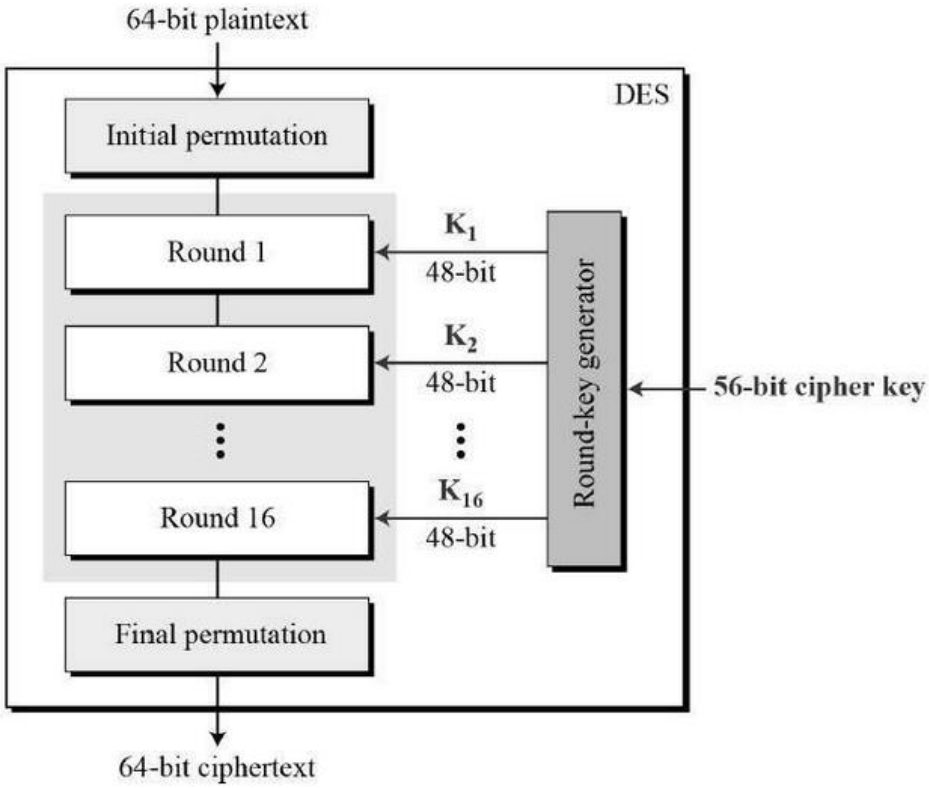
DES (engl. *Data Encryption Standard*) on symmetrisen avaimen lohkosalaus algoritmi, joka kehitettiin jo 1970-luvun alkupäässä IBM:n toimesta ja myöhemmin se oli Yhdysvalloissa liittovaltion standardiksi 1976 valittu salausmenetelmä. ANSI standardissa DES tunnetaan nimellä DEA (engl. *Data Encryption Algorithm*) ja ISO standardoinnissa nimellä DEA-1.

5.1.1 DES algoritmi

DES perustuu Feistelin lohkosalausjärjestelmään (engl. *Feistel Cipher*). Feistel Cipher on suunnittelumalli, josta johdetaan monia erilaisia lohkosalauksia. Feistel rakenteessa salaus koostuu useasta tekstinkäsittelykierroksesta, joista jokainen koostuu ”substituutio” vaiheesta, jota seuraa permutointivaihe. (Feistel Block Cipher, 2021.) DES käyttää 16 kierroksista Feistel rakennetta ja sen lohkokoko on 64-bittiä. Efektiivisen avaimen koko on kuitenkin vain 56 bittiä, koska kahdeksaa bittiä käytetään pariteetti tarkistuksiin. (Data Encryption Standard, 2021.)

Sen sijaan, että käytettäisiin koko salausavainta jokaisen kierroksen aikana, salausavaimesta johdetaan kierroksesta riippuva avain (engl. *a subkey*). Kaikki nämä aliavaimet liittyvät kuitenkin alkuperäiseen salausavaimeen, mutta jokainen kierros käyttää eri avainta (KUVA 8). DES salausprosessia voidaan käyttää neljässä eri tilassa, salaamalla jokainen lohko erikseen tai tekemällä jokaisesta salatusta lohkosta riippuvainen kaikkiin edellisiin lohkoihin. Salauksen purku tapahtuu päinvastaisessa järjestyksessä salaukseen verrattuna, eli avainten käyttöprosessi käydään vain käänteisessä järjestyksessä läpi. (Cobb, 2014; Data Encryption Standard, 2021.)

DES-algoritmia ja sen 56-bittistä salausavainta ei ole pidetty turvallisena enää 1990-luvun puolivälin jälkeen, mutta sen rooli on ollut erittäin merkittävä salauksen edistämisessä. Vuonna 1997 NIST (*National Institute of Standards and Technology*) järjestikin aloitteen DES standardin seuraajalle. Virallisesti DES vedettiin pois vuoden 2005 toukokuussa. Kuitenkin DES salaukseen perustuvaa 3DES algoritmia, joka käyttää kolmea iterointia DES algoritmista, käytetään vielä laajalti. (Cobb, 2014.)



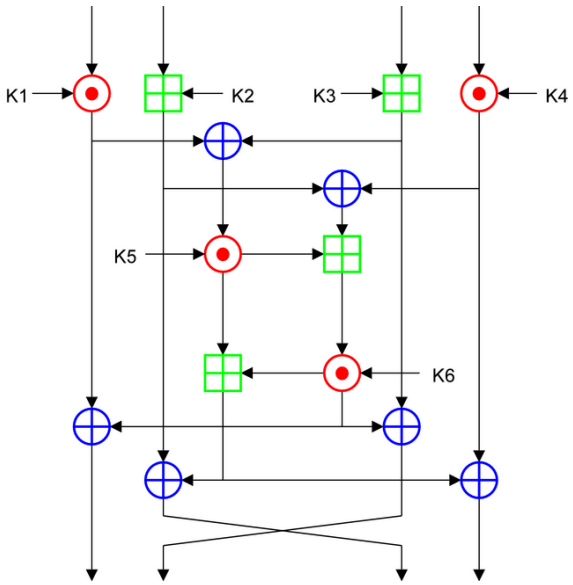
KUVA 8. DES rakenne (Data Encryption Standard, 2021)

5.2 IDEA

IDEA (engl. *International Data Encryption Algorithm*) on symmetrisen avaimen lohkosalaus algoritmi, jonka kehittivät James Massey ja Xuejia Lai ETH Zurichissa ja se esiteltiin ensimmäisen kerran vuonna 1991. IDEA algoritmi on hieman uudistettu versio aikaisemmasta PES (engl. *Proposed Encryption Standard*) salauksesta ja se oli tarkoitettu DES-standardin korvaajaksi. Viimeisimmät IDEA patentit päättyivät vuonna 2012, joten se on täysin vapaasti käytettävissä. (International Data Encryption Algorithm, 2021.)

5.2.1 IDEA algoritmi

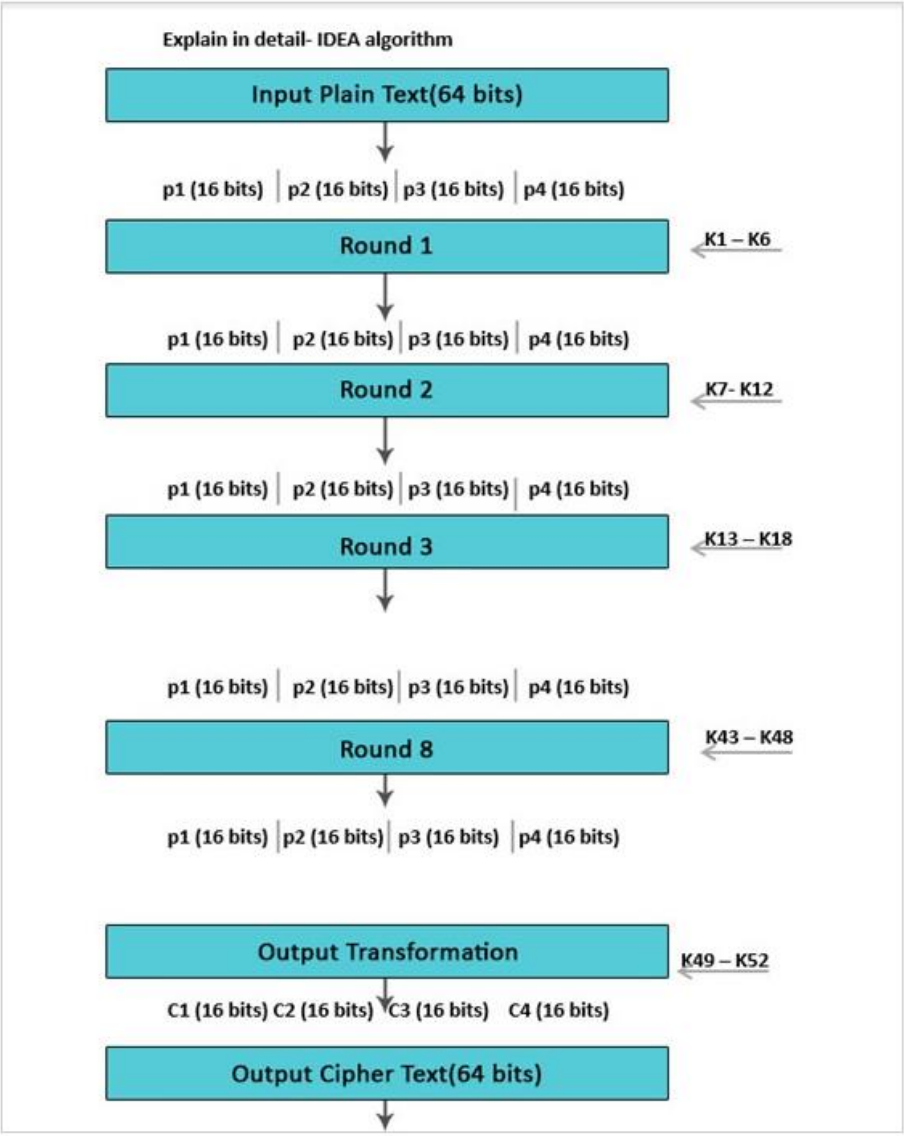
IDEA algoritmi operoi 64-bittisillä lohkoilla käyttäen 128-bittistä avainta ja se koostuu kahdeksasta identtisestä muunnos kierroksesta, sekä viimeisestä ulostulon tuottavasta puolikkaasta kierroksesta. Algoritmin turvallisuus johtuu eri algebrallisten ryhmien operaatioiden lomittamisesta: modulaarinen summaus ja kertolasku, sekä bittikohtainen XOR-operaatio (engl. bitwise *exclusive-OR operation*). Yhtä algoritmin kierrosta esittävässä kuvassa 9, sininen ympyröity plus merkki on XOR-operaatio ja vihreä neliö modulus summa 2^{16}. Punainen ympyröity piste kuvaa kertolasku modulusta $2^{16} + 1$, missä kaikki tulon nolla merkit (0x0000) ovat 2^{16} ja 2^{16} tulkitaan nolla merkiksi ulostulossa.



KUVA 9. IDEA algoritmin salaus kierros (International Data Encryption Algorithm, 2021)

Kuvassa 10 on esitettynä IDEA algoritmin lohkokaavio. 64-bittinen teksti lohko jaetaan neljään 16-bittiseen osaan p1, p2, p3 ja p4. Nämä osat toimivat tuloina algoritmin ensimmäiselle kierrokselle. Kierroksia algoritmissa on edellä mainittu kahdeksan ja avain koostuu 128 bitistä. Jokaisella kierroksella syntyy kuusi aliavainta. Jokainen aliavain sisältää 16 bittiä ja kaikki nämä aliavaimet käytetään neljään tulolohkoon p1-p4. Viimeisenä tuloksena on neljä 16 bittistä lohkoa salattua tekstiä C1-C4, jotka yhdistetään yhdeksi 64-bittiseksi salatuksi lohkoksi.

Salauksen purku on toiminnallisesti samanlainen kuin itse salaus, mutta kierrosten avainten järjestys on käänteinen ja parittomien kierrosten aliavaimet ovat ylösalaisin. Esimerkkinä K1-K4 korvataan käänteisillä K49-K52 kullekin ryhmäoperaatiolle, K5 ja K6 kullekin ryhmälle tulisi korvata K47:lla ja K48:lla salauksen purkua varten.



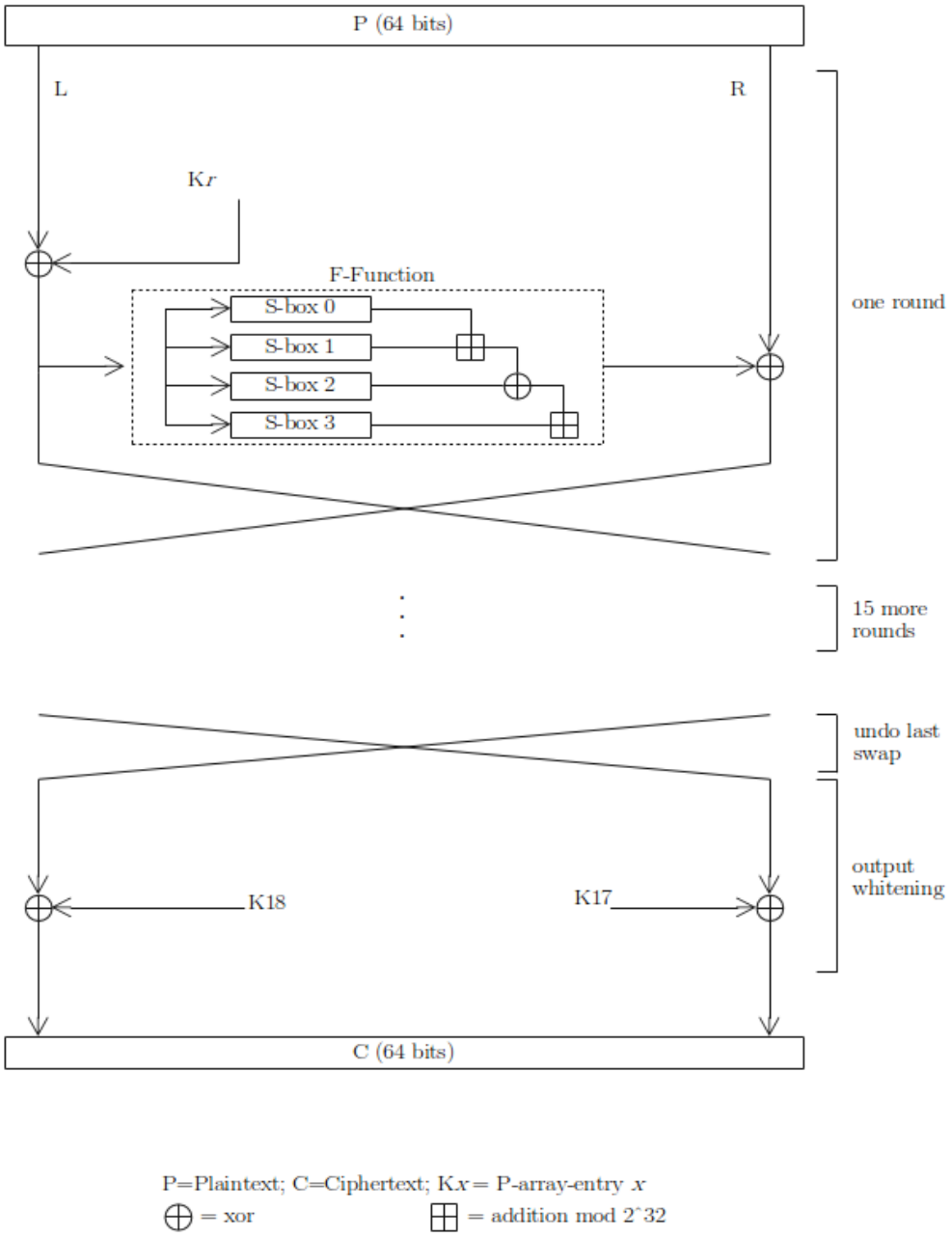
KUVA 10. IDEA lohkokaavio (Pedamkar, 2020)

5.3 Blowfish

Blowfish on symmetrisen avaimen lohkosalaus algoritmi, jonka kehittäjä on yhdysvaltalainen Bruce Schneier. Schneier kehitti algoritmin vuonna 1993 ja sen oli tarkoitus olla yleiskäyttöinen ja potentiaalinen korvaaja vanhentuvalle DES algoritmille, koska tuohon aikaan useat muut mallit olivat patentoituja, kaupallisia tai muuten valtion salaisuuksia.

5.3.1 Blowfish algoritmi

DES algoritmin tapaan Blowfish perustuu myös Feistel salaus- malliin, 64-bittiseen lohkokokoon ja sen avaimen pituus vaihtelee 32 bitistä 448 bittiin. Blowfish käyttää salaukseen 16 kierroksista Feistel mallia ja suuria avaimesta riippuvia S-laatikoita. S-laatikko eli substituutio laatikko (engl, *S-box, substitution box*) on salauksessa symmetristen avainalgoritmien peruskomponentti, joka suorittaa substituution eli korvaamisen. Niitä käytetään peittämään avaimen ja salatun tekstin välinen riippuvuussuhde lohkosalauksissa. (S-box, 2021.)



Kuva 11. Blowfish algoritmin Feistel rakenne (Blowfish (cipher), 2021)

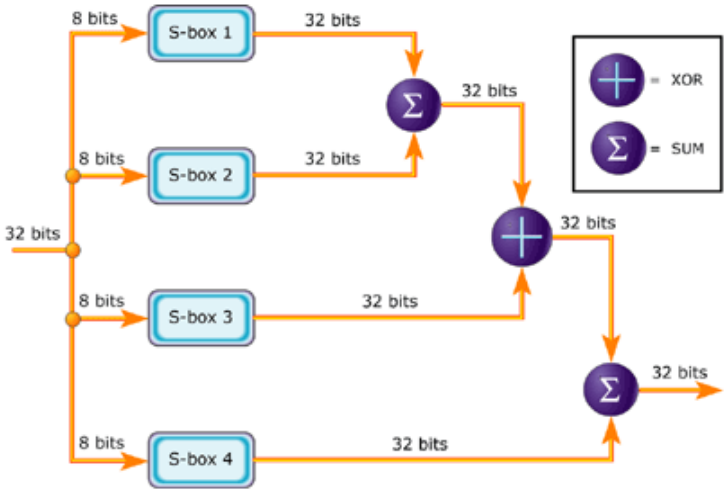
Algoritmi koostuu kahdesta osasta: avaimen laajennusosasta ja datan salauksesta. Avaimen laajennus muuttaa maksimissaan 448 bittisen avaimen lukuisiin aliavain ryhmiin, maksimissaan yhteensä 4168 tavua. Kuvassa 11 on kuvattu Blowfish algoritmin rakennetta ja salauksen rutiinia. Jokainen kierros sisältää salausavain riippuvaisen permutaation, sekä avain- ja data riippuvaisen substituution. Kaikki operaatiot ovat XOR ja yhteenlasku operaatioita 32-bittisille osille (jokainen viiva kuvassa 11). Ainoat lisäoperaatiot kierrosta kohti ovat datan haku neljä indeksisestä taulukosta.

Blowfish algoritmin aliavaimet on laskettava ennen kuin suoritetaan datan salausta tai salauksen purkua. Kuvassa 11 näkyvä Kr on P-taulukko, joka koostuu 18:sta 32-bittisestä aliavaimesta, jonka jälkeen on neljä S-laatikkoa (S0-S3) 256-bittisillä tuloilla. Aliavaimet lasketaan Blowfish algoritmilla ja aluksi alustetaan P-taulukko ja S-laatikot heksadesimaaliluvuilla, jotka on johdettu Pii:n lukuarvoista.

Tämän jälkeen suoritetaan XOR-operaatio ensimmäiselle 32 bitille avaimesta P1:n kanssa, toiselle 32 bitille P2:sen kanssa ja tätä jatketaan niin kauan, että koko P-taulukko on käyty läpi XOR-operaatioilla avaimen kaikille biteille. Sitten suoritetaan salaus pelkkiä nollia sisältävälle 64-bittiselle lohkolle, josta saatu salattu teksti korvaa P1 ja P2 ja salataan muokatuilla aliavaimilla taas nollalohko. Sitten taas P3 ja P4 korvataan edellisen kohdan ulostulolla. Tätä jatketaan niin kauan, että koko P-taulukko on korvattu, jonka jälkeen tehdään vielä sama prosessi S-laatikoille. Kaikkiaan vaaditaan 521 iteraatiota, jotta saadaan luotua kaikki tarvittavat aliavaimet, mutta nämä avaimet voidaan tallentaa sovelluksessa, jotta koko aliavainten luontiprosessia ei tarvitse käydä läpi useaan kertaan. (Schneier, 1994; Blowfish (cipher), 2021.)

Kuvan 11 jokainen kierros *r* koostuu neljästä toiminnosta:

1. Suoritetaan XOR-operaatio vasemmalle puoliskolle L (32-bittiä) dataa kierroksen *r* P-taulukon alkiolla.
2. XOR-operaatiosta saatu data viedään F-funktion tuloksi. F funktio jakaa 32-bittisen tulon neljään 8-bittiseen osioon ja nämä osiot toimivat tuloina S-laatikoille. S-laatikot tuottavat 8-bittisistä tuloista 32-bittiset lähdöt summaamalla ja XOR-operaatiolla (KUVA 12).



KUVA 12. Blowfish F-funktio (Gatliff, 2003)

3. Tehdään XOR operaatio F-funktiosta saadulle tulokselle oikean puolen datan R kanssa.
4. Vaihdetaan L:n ja R:n puolia.

Salauksen purku on samanlainen prosessi kuin itse salaus, mutta P1, P2, …, P18 käytetään päinvastaisessa järjestyksessä. Blowfish toteutus vaatii noin 5 kilotavua muistia ja 32-bittisessä prosessorissa voi salata tai purkaa 64-bittisen tekstin noin 12 kellosyklissä. Tätä pidemmät tekstit nostavat laskenta-aikaa lineaarisesti eli 128-bittinen teksti vie noin 2x12 kellosykliä. (Gatliff, 2003.)

5.4 Twofish

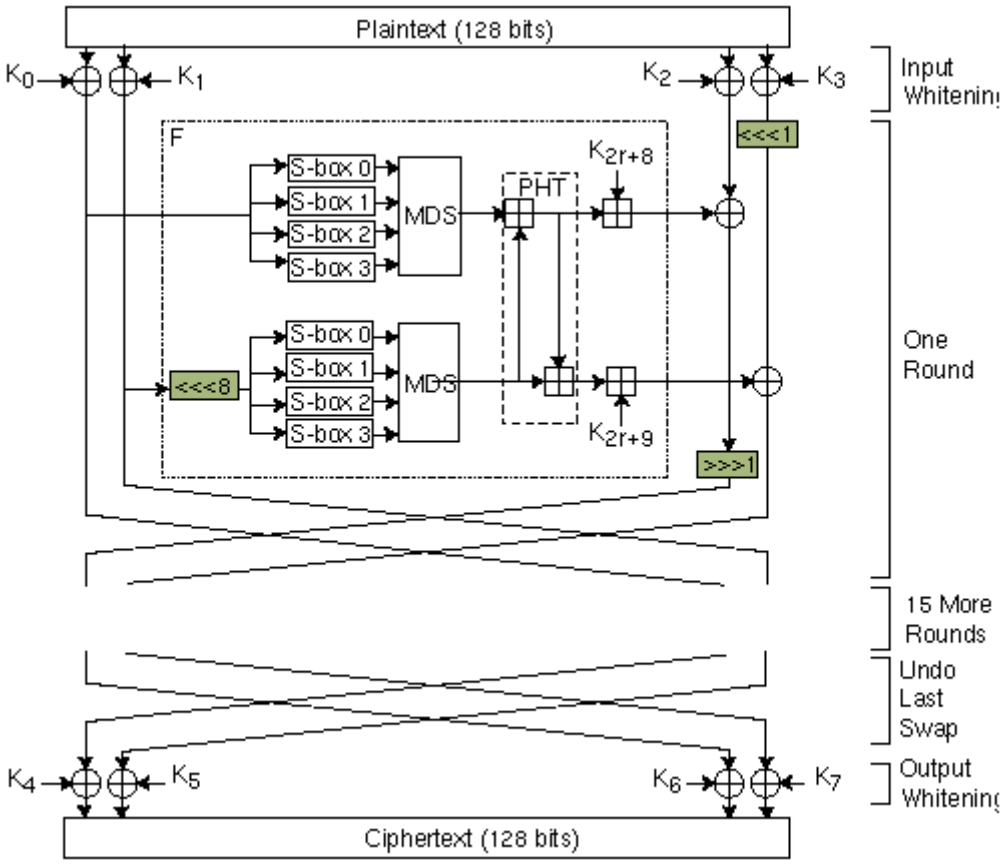
Twofish on jatkokehitetty edellä esitellystä Bruce Schneierin Blowfish algoritmista. Schneierin lisäksi sen suunnittelijoina toimivat myös John Kelsey, Doug Whiting, David Wagner, Chriss Hall ja Niels Ferguson ja algoritmi esiteltiin vuonna 1998. Twofish oli yksi finalisteista NIST:n 1997-2000 järjestämässä Advanced Encryption Standard kilpailussa, missä etsittiin seuraajaa DES-algoritmille. Twofish algoritmi ei lopulta tullut valituksi finalistien joukosta standardisointiin.

Twofish suunniteltiin mahdollisimman nopeaksi, joustavaksi ja kuitenkin turvalliseksi. Se ei sisällä radikaaleja uusia ideoita tai suunnitteluelementtejä, vaan se on tehty optimoimalla ja käyttäen ominaisuuksia, jotka ovat jo hyväksi sekä toimivaksi todettuja. Twofish on täysin ilmainen eikä sille ole missään vaiheessa ollut tarkoitus hakea patenttia, lisenssejä tai tekijänoikeuksia. Twofish algoritmia voidaan käyttää joustavuutensa ansiosta esimerkiksi verkkosovelluksissa, joissa avaimia vaihdetaan usein, sekä sovelluksissa, joissa RAM- ja ROM-muistia on vähän tai ei ollenkaan. Twofish on todettu nopeaksi myös esimerkiksi 32- ja 8-bittisillä suorittimilla, kuten sirukorteissa ja sulautetuissa järjestelmissä, sekä laitteistotasolla. (Schneier, The Twofish Encryption Algorithm, 1998.)

5.4.1 Twofish algoritmi

Twofish on myös symmetrinen lohkosalaus algoritmi, joten se käyttää yhtä avainta salaukseen ja salauksen purkamiseen. Sen salauksen lohkokoko on 128 bittiä ja sen salausavaimen koko on maksimissaan 256 bittiä. Kuten DES ja edeltäjänsä Blowfish, Twofish salaus perustuu myös Feistel verkosto -malliin, joka tarkoittaa, että jokaisella kierroksella puolet tekstilohkosta syötetään F-funktion läpi ja tehdään XOR-operaatio lohkosta toisen puolen kanssa (KUVA 13).

Kuvassa 13 on kuvattuna Twofish algoritmin toiminta, jossa jokaisella kierroksella kaksi 32-bittistä tekstiä toimii F-funktiolle tuloina (kaksi pystyviivaa vasemmalla). Jokainen teksti jaetaan neljään tavuun ja nämä syötetään neljän salausavaimesta riippuvan S-laatikon läpi. S-laatikoiden ulostulot, eli neljä tavua yhdistetään käyttämällä MDS matriisia (engl. *Maximum Distance Separable matrix*) 32-bittiseksi tekstiksi. MDS matriisi on matriisi, joka edustaa funktiota, jolla on tietyt diffuusio ominaisuudet (MDS matrix, 2021). Diffuusio tässä yhteydessä tarkoittaa, että kun muutetaan yhtä bittiä selkotekstistä, niin tilastollisesti puolet salatun tekstin biteistä muuttuvat ja tämä pätee myös toisinpäin. Diffuusion tarkoitus on peittää salatun tekstin ja selkotekstin yhteyttä toisiinsa. (Confusion and diffusion, 2021.) MDS operaation jälkeen nämä kaksi 32-bittistä tekstiä yhdistetään PHT:llä (engl. *Pseudo-Hadamard Transform*), lisätään kahden kierroksen aliavaimeen ja suoritetaan XOR-operaatio oikean puolen tekstin kanssa. PHT on bittijonon käänteinen muunnos ja sen tarkoitus on tuottaa salauksen diffuusio. Kuvasta 13 nähdään, että Twofish algoritmi käyttää myös kahta operaatiota *Input whitening* ja *Output whitening*, joissa lisä aliavaimilla suoritetaan XOR-operaatio tekstilohkolle ennen algoritmin ensimmäistä kierrosta ja viimeisen kierroksen jälkeen. (Schneier, The Twofish Encryption Algorithm, 1998.)

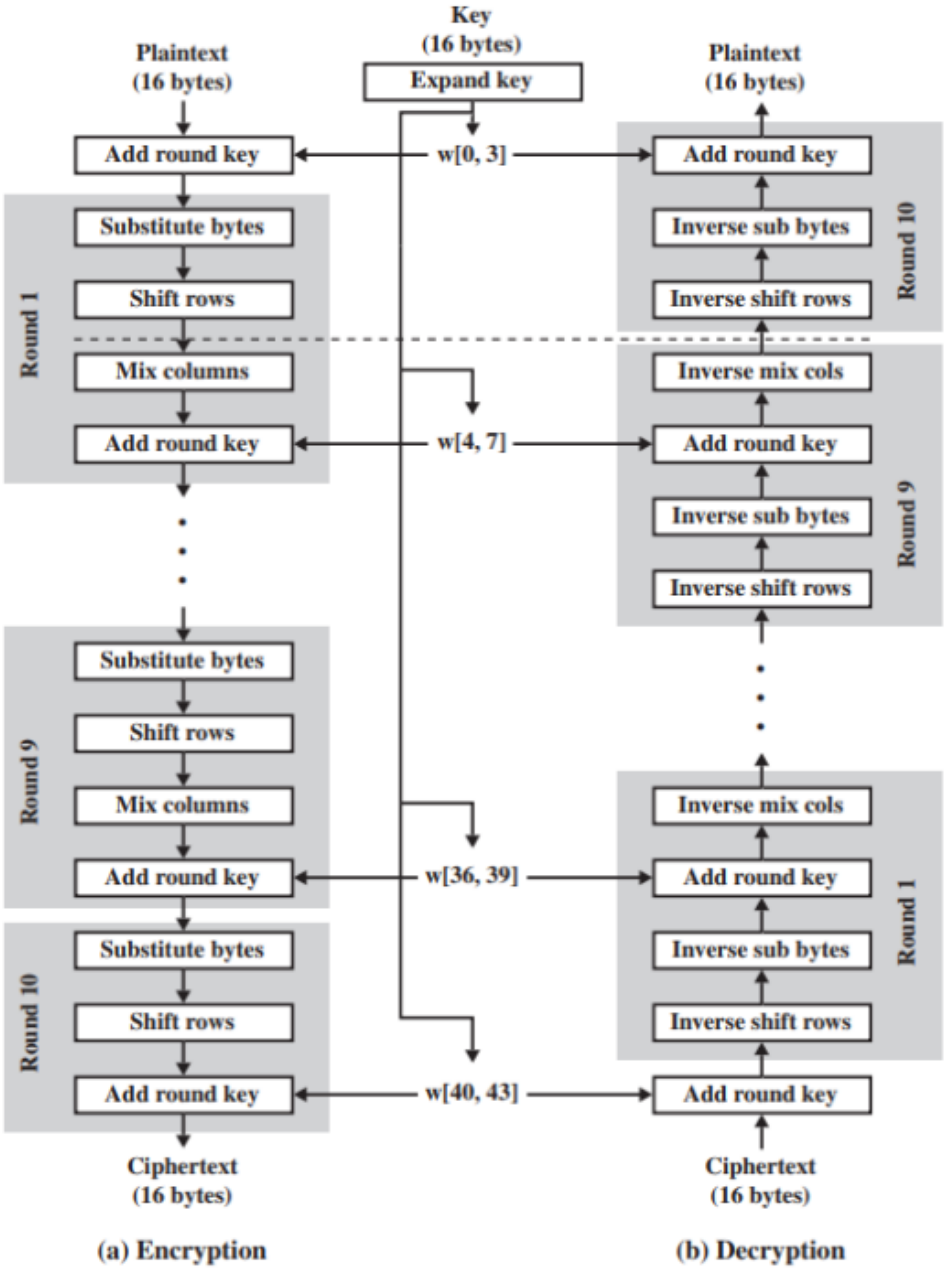


KUVA 13. Twofish algoritmi (Schneier, The Twofish Encryption Algorithm, 1998)

5.5 AES

NIST:n järjestämän kilpailun, joka kesti vuodesta 1997 vuoteen 2000 asti, DES salausalgoritmin korvaamiseksi uudella symmetrisellä lohkosalauksella AES:lla, valikoitui lopulta 15 erilaisen esitetyn salausmallin joukosta 2. lokakuuta 2000 kahden belgialaisen, Vincent Rijmen ja Joan Daemen, suunnittelema Rijndael lohkosalaus algoritmi seuraavaksi tulevaksi AES-algoritmiksi. Kilpailuun osallistuvien algoritmien oli kaikkien oltava lohkoilla toimivia salauksia, tuettuna lohkokokona 128 bittiä ja avaimen kokona 128, 192 ja 256-bittiä. 11. marraskuuta vuonna 2001 NIST ilmoitti, että Rijndael algoritmiin perustuva seuraava AES on hyväksytty FIPS PUB 197 -standardiksi. AES sisältyy ISO/IEC 18033-3 -standardiin ja se tuli voimaan Yhdysvaltain kauppaministerin hyväksynnän jälkeen Yhdysvaltojen liittohallituksen standardina 26. toukokuuta 2002. AES on myös ensimmäinen ja ainoa NSA:n hyväksymä julkisesti saatavilla oleva salaus käytettäväksi huippusalaista tietoa varten. (NIST, 2001; Advanced Encryption Standard Process, 2020.)

5.5.1 AES algoritmi



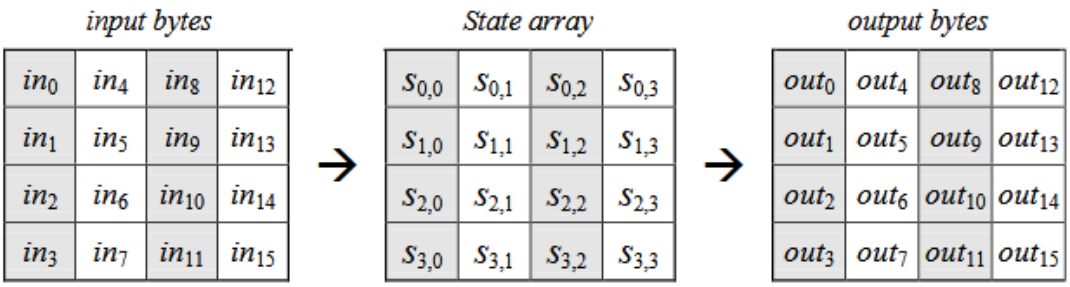
KUVA 14. AES rakenne (Stallings, 2014, s. 136)

AES salauksen taso ilmoitetaan yleensä salausavaimen koon perusteella, joita on kolme: AES-128, AES-192 ja AES-256, jossa numero kertoo käytettävän salausavaimen koon bitteinä. Nämä kaikki operoivat salattavaa tekstiä 128-bitin lohkoissa. Toisin kuin edeltäjänsä DES algoritmi, AES salaus ei perustu Feistel-malliin vaan substituutio-permutaatio verkon, SPN (engl. *Substitution-permutation network*), suunnitteluperiaatteeseen. SPN on lohkosalaus algoritmeissa käytetty joukko linkitettyjä matemaattisia operaatioita, esimerkiksi XOR-operaatioita ja bittien siirtämistä (engl. *Bitwise rotation*). Salausavain ja selkoteksti toimivat SPN:n tuloina. AES avaimen koko määrää salaukseen käytettävien algoritmin kierrosten määrän (TAULUKKO 4). (NIST, 2001.)

TAULUKKO 4. AES avaimen koot ja kierrokset

Käytettävä AES	Avaimen koko	Kierrokset
AES-128	128 bittiä	10
AES-192	192 bittiä	12
AES-256	256 bittiä	14

AES algoritmin toiminnot suoritetaan sisäisesti kaksiulotteisella taulukolla tavuja, jota kutsutaan nimellä State (tila). State koostuu neljästä rivistä tavuja, joista jokainen sisältää *Nb* määrän tavuja, missä *Nb* on lohkon koko jaettuna 32:lla. AES standardissa *Nb* = 4. State taulukossa, merkitään *s*, jokaisella yksittäisellä tavulla on kaksi indeksiä, rivi numero *r*, joka on välillä $0 \le r < 4$ ja sarake numero *c* välillä $0 \le c < Nb$. Näin jokaiseen yksittäiseen tavuun State:ssa voidaan viitata $s_{r,c}$, tai *s[r,c]* (KUVA 14). (NIST, 2001.)

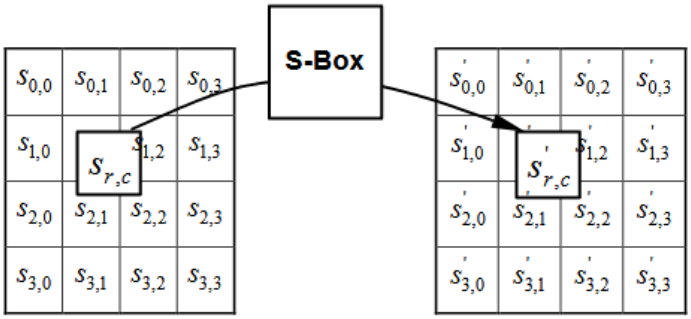


KUVA 15. State taulukko, tulo ja lähtö (NIST, 2001)

AES algoritmissa avaimen kokoa merkitään *Nk* = 4, 6 tai 8, eli 32-bittisten sanojen määrä (sarakkeiden määrä) salausavaimessa. Algoritmin kierrosten määrää merkitään *Nr*, esimerkiksi *Nr* = 10, kun *Nk* = 4 (TAULUKKO 4). Salauksen alussa kuvan 14 mukaisesti, tulon taulukon tavut (*input bytes*) kopioidaan State-taulukkoon (*State array*). Salausoperaatiot suoritetaan State-taulukossa, jonka jälkeen lopulliset arvot kopioidaan lähtötaulukkoon (*output bytes*). State taulukon jokaisen sarakkeen neljä tavua muodostavat 32-bittisiä sanoja, missä rivin numero *r* on indeksi neljään tavuun jokaisessa sanassa. State voidaan siis tulkita yksiulotteisena taulukkona 32 bittisiä sanoja (sarakkeita), *w*, jossa sarakkeen numero *c* on indeksi tähän taulukkoon, esimerkiksi kuvassa 14: $w_0 = s_{0,0}\ s_{1,0}\ s_{2,0}\ s_{3,0}$, $w_1 = s_{0,1}\ s_{1,1}\ s_{2,1}\ s_{3,1}$ jne. (NIST, 2001.)

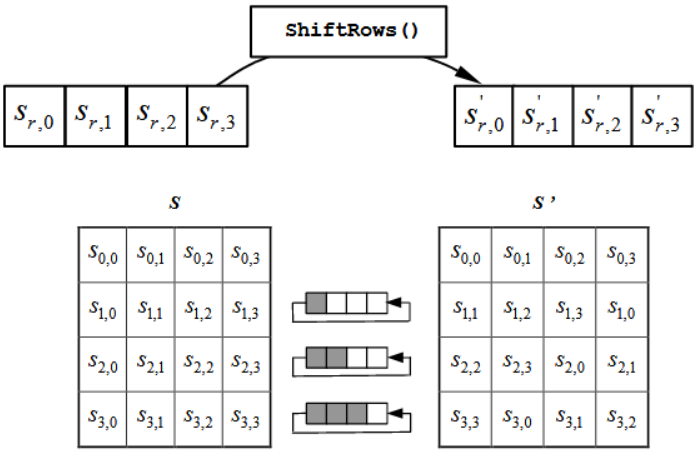
AES algoritmi käyttää salaukseen ja sen purkamiseen kierroksia, jotka koostuvat neljästä eri tavuihin liittyvästä muunnosvaiheesta:

1. Substituutio vaihe (engl. *SubBytes transformation*) eli Staten jokaisen tavun korvaus käyttäen S-laatikkoa, jolla muodostetaan epälineaarinen salaus (KUVA 16). Substituutio vaiheen käyttämä S-laatikko on heksadesimaali muodossa kuvassa 20. Esimerkkinä, jos State taulukossa tavun $s_{1,1}$, arvona on {35}, niin silloin substituutio eli korvaava arvo määritellään indeksillä kolme olevan rivin ja indeksillä viisi olevan sarakkeen risteämiskohdassa olevalla arvolla eli $s'_{1,1} = 96$ (KUVA 20).



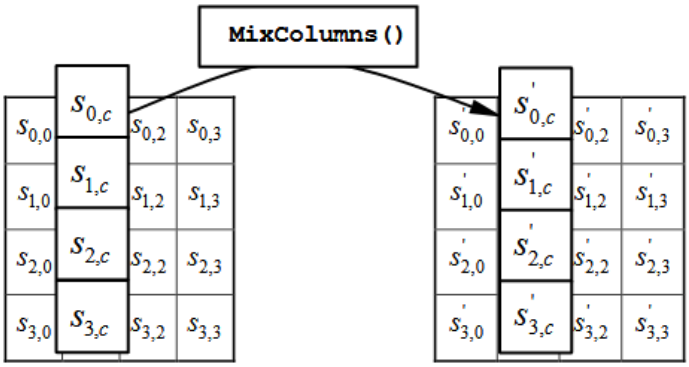
KUVA 16. SubBytes-operaatio (NIST, 2001)

2. Rivien siirtäminen (engl. *Shiftrows transformation*), jossa State taulukon rivejä siirretään eri standardin määräämillä offset-arvoilla. Esimerkiksi ensimmäistä riviä ei siirretä, mutta toisen rivin tavuja siirretään kertaalleen vasemmalle (yksi tavu), kolmatta kaksi kertaa vasemmalle ja niin edelleen (KUVA 17).



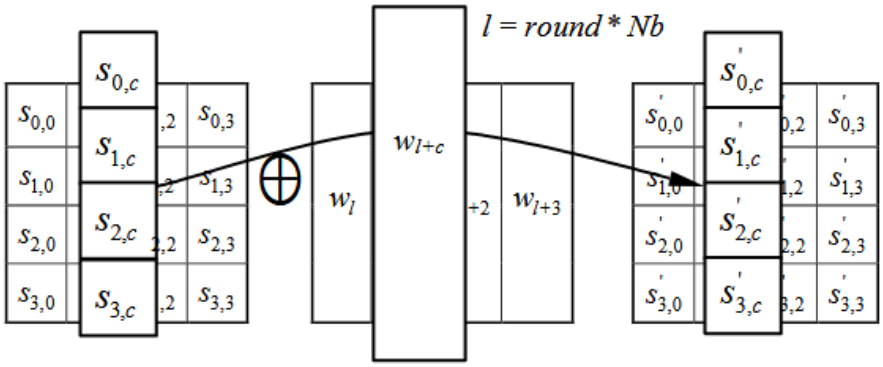
KUVA 17. ShifRows-operaatio (NIST, 2001)

3. Sarakkeiden sekoitus vaihe (engl. *MixColums transformation*), joka operoi Statessa sarake-sarakkeelta, yhdistäen tilan pystysarakkeiden neljä tavua käännettävän lineaarimuunnoksen avulla (KUVA 18).



KUVA 18. MixColumns-operaatio (NIST, 2001)

4. Kierroksen avaimen summaus (engl. *AddRoundKey transformation*), jossa suoritetaan jokaiseen Staten sarakkeeseen bittitason XOR-operaatio kierrosavaimen kanssa (KUVA 19). Tämä operaatio suoritetaan ensimmäisellä kerralla lohkon ja salausavaimen välillä ja myöhemmin AES-välituloksen eli algoritmin operaatioiden välisen 128-bittisen tuloksen ja kierrosavaimen välillä. (NIST, 2001.)



KUVA 19. AddRoundKey-operaatio (NIST, 2001)

Algoritmin ensimmäisellä kierroksella suoritetaan pelkästään kierrosavaimen summaus eli AddRoundKey-funktio. Avaimenlaajennus operaatio (engl. *Key Expansion routine*) on lohkosalausavaimeen *K* liittyvä AES algoritmin rutiini, joka tuottaa edelleen säännöt (engl. *key schedule*) minkä perusteella salaukseen käytettävät kierrosavaimet lasketaan. Avainlaajennus tuottaa yhteensä *Nb (Nr + 1)* määrän sanoja, eli algoritmi vaatii aluksi *Nb* määrän sanoja ja jokainen kierros *Nr* vaatii *Nb* sanan määrän avaindataa. Tuloksena syntyy lineaarinen taulukko 4-tavuisia sanoja. Algoritmin viimeinen kierros on muuten samanlainen kuin edeltävät kierrokset mutta pois jää lineaarimuunnos sarakkeiden sekoituksesta (KUVA 14). (NIST, 2001.)

Salattu lohko puretaan salaamattomaksi tekstiksi käyttämällä käänteisiä versioita salaukseen käytetyistä operaatioista ja suorittamalla ne käänteisessä järjestyksessä (engl. *Inverse Cipher*). Salauksen

purkamiseen käytettävät yksittäiset muunnosfunktiot, jotka operoivat Statea ovat *InvShfitRows(), InvSubBytes(), InvMixColumns()* ja *AddRoundKey()*. (NIST, 2001.)

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

KUVA 20. S-laatikko tavulle xy heksadesimaali muodossa (NIST, 2001)

6 LOHKOSALAUSTILAT

Aiemmin kuvatut lohkosalausmenetelmät soveltuvat itsessään vain salaamaan ja purkamaan tekstiä yksittäisinä, kiinteinä, vakiokokoisina lohkoina. Lohkosalaustila (engl. *Block cipher mode of operation*) mahdollistaa käytettävän salausalgoritmin laajentamisen niin, että yhden lohkon salausta ja purkamista voidaan käyttää toistuvasti vakiokokoisia lohkoja suurempiin tietomääriin.

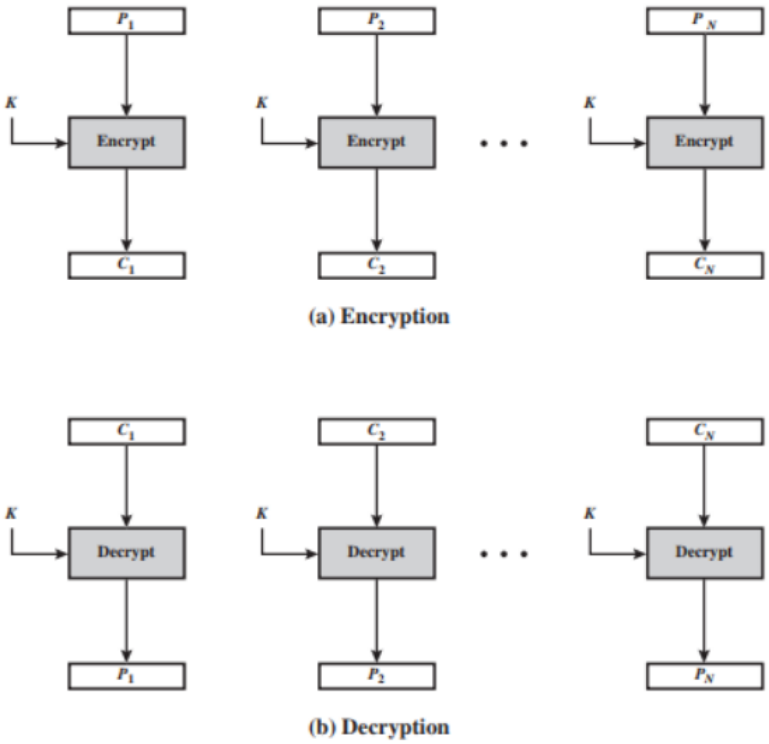
Vanhimmat lohkosalaustilat, kuten ECB, CBC, OFB ja CFB, ovat jo vuodelta 1981 ja ne määriteltiin NIST FIPS 81 julkaisussa nimellä *DES Modes of Operation*. Vuonna 2001, kun AES tuli kuvioihin standardi lohkosalausalgoritmina, NIST lisäsi hyväksyttyjen standardisoitujen lohkosalaustilojen listaan CTR-tilan ja myöhemmin vuonna 2010 XTS-AES:n. (Block cipher mode of operation, 2021.)

6.1 ECB

ECB (engl. *Electronic Code Book*) on näistä tiloista yksinkertaisin, jossa tekstiä käsitellään yksi lohko kerrallaan ja jokainen lohko salataan aina samalla avaimella. Mikäli selkotekstin viimeinen lohko jää vajaaksi, voidaan se ”täyttää” (engl. *padding*) lisäämällä bittien määrää ennen salausta.

Tässä lohkosalaustilassa sama selkokielinen tekstilohko tuottaa aina saman salatun tekstilohkon. ECB käyttö on ideaalisinta vähäiselle määrälle dataa, kuten salausavaimelle. Pidemmille teksteille ECB:tä ei pidetä hyvin turvallisena vaihtoehtona, koska jos teksti on hyvin jäsenneltyä, esimerkiksi teksti alkaa aina samanlaisilla kentillä, voi hyökkääjä löytää salatusta tekstistä samankaltaisuuksia. Kuvassa 21 on ECB lohkosalaustilan lohkokaavio, jossa selkoteksti koostuu b-bittisistä lohkoista P_1,

P_2,…,P_N ja vastaavasti salatut lohkot C_1,…,C_N ja salaukseen käytettävä salausavain K. (Stallings, 2014, ss. 180-181.)



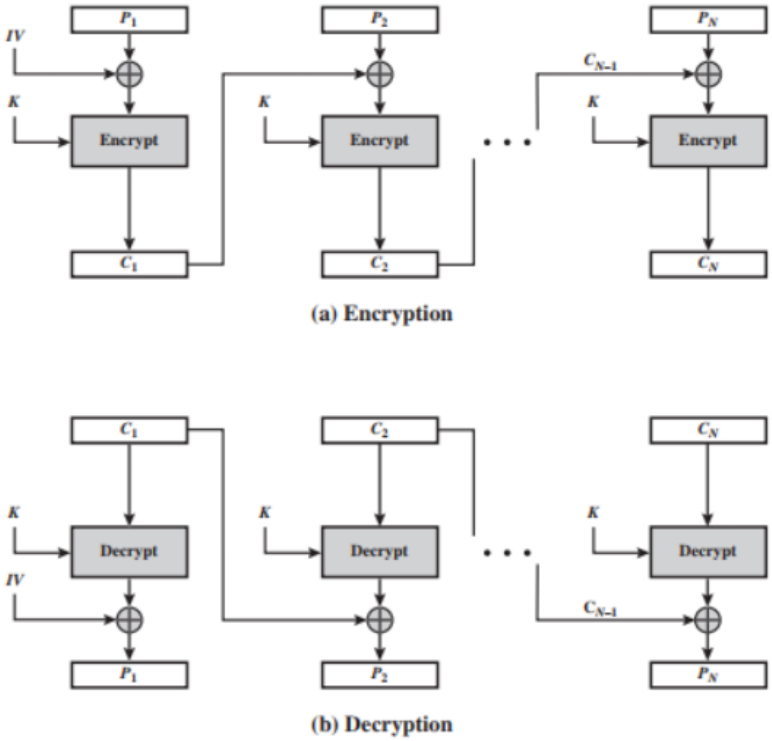
KUVA 21. ECB-lohkosalaustila (Stallings, 2014, s. 182)

6.2 CBC

CBC (engl. *Cipher Block Chaining Mode*) tilassa (KUVA 22) salausalgoritmiin syötetään nykyinen selkoteksti XOR-operaatiolla edellisen salatun tekstilohkon kanssa, jolloin saavutetaan ECB:hen verrattuna turvallisempi salaus, kun jokaisella salausfunktion tulolla ei ole suoraa yhteyttä selkotekstin lohkoon. Samaa salausavainta käytetään jokaiseen lohkoon. Kuten ECB:ssä, myös CBC:ssä tarvitsee vajaat lohkot täyttää kokonaiseksi.

Jotta saadaan ensimmäinen salattu tekstilohko, on käytettävä alustusvektoria IV (engl. *initialization vector, IV*), jolle tehdään XOR-operaatio ensimmäisen selkotekstilohkon kanssa. Salauksessa alustusvektori on tulo salausprimitiiviin, jota käytetään alkutilan aikaansaamiseksi. IV on tyypillisesti oltava satunnainen tai vain arvaamaton tai ainutlaatuinen. IV on datalohko, joka on samankokoinen kuin salattu tekstilohko. IV on oltava tiedossa salaukseen liittyvillä osapuolilla, mutta tuntematon mahdolliselle kolmannelle, eli ulkopuoliselle osapuolelle. IV lohkon luomiseen voidaan käyttää esimerkiksi satunnaislukugeneraattoria.

Salauksen purkamiseksi, jokainen salattu lohko syötetään salauksenpurku algoritmin läpi (KUVA 22), jonka tulos XOR:taan edellisen salatun tekstilohkon kanssa, jotta saadaan selkotekstin lohko. (Stallings, 2014, ss. 183-184.)



KUVA 22. CBC-lohkosalaustila (Stallings, 2014, s. 183)

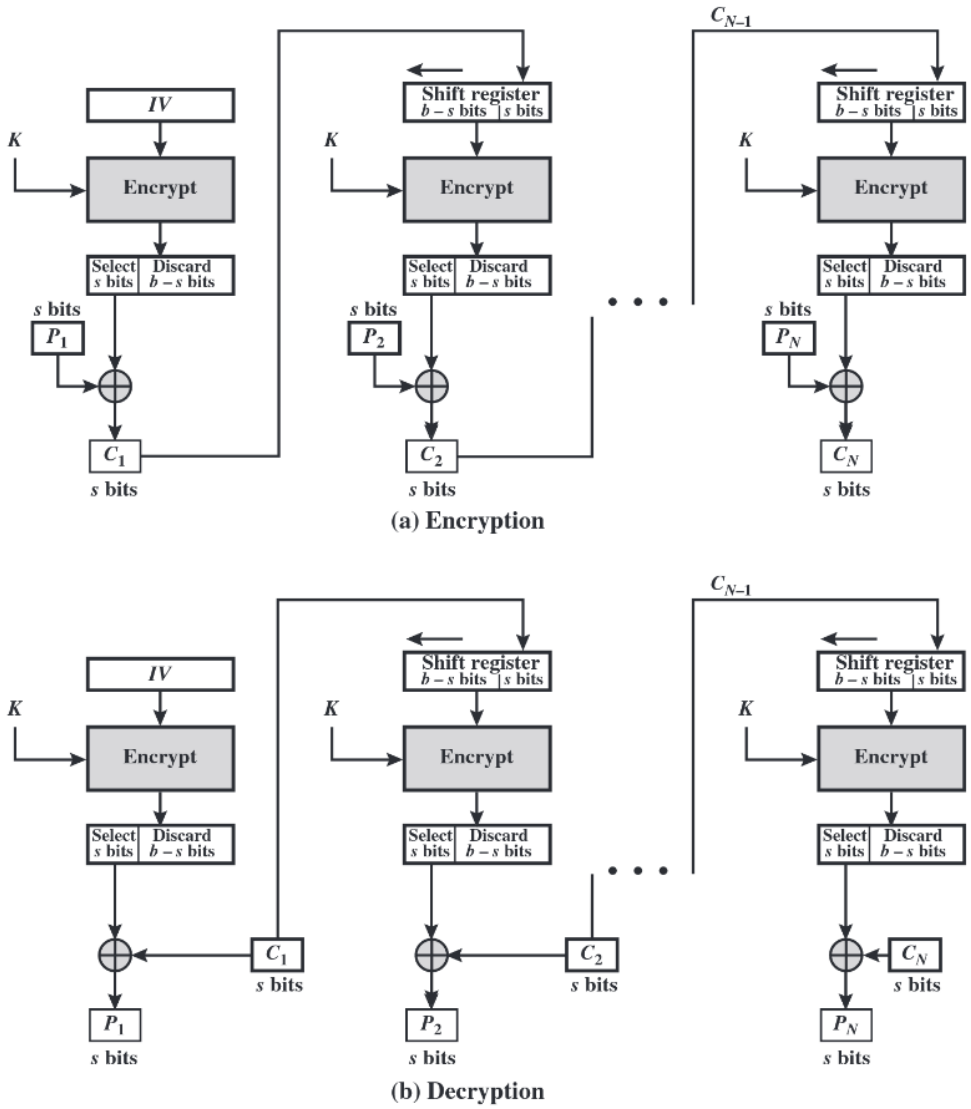
6.3 CFB

Tietyt lohkosalaustilat, kuten seuraavaksi esiteltävät CFB (engl. *Cipher Feedback Mode*), OFB (engl, *Output Feedback Mode*) ja CTR (engl. *Counter Mode*) pystyvät periaatteessa muuntamaan lohkosalauksen jonosalaus tyyppiseksi. Jonosalaus eliminoi tarpeen muuttaa selkoteksti tietyn mittaisiin kokonaisluvun lohkoihin ja jonosalaus pystyy myös toimimaan reaaliajassa.

Kuvassa 23 on esitettynä CFB-lohkosalaustilan lohkokaavio, jossa lähetettävä yksikkö on *s*-bittiä (yleensä 8 bittiä). Kuten CBC:ssä, selkotekstin yksiköt ovat sidottuna toisiinsa niin, että salattu teksti mistä tahansa selkotekstiyksiköstä on kaikkien edellisten selkotekstien funktio. Tässä tapauksessa, toisin kuin jakamalla selkoteksti lohkoihin b-bittejä, se jaetaan segmentteihin s-bittejä.

Salausfunktion tulo on b-bittinen vuororekisteri (engl. *shift register*) joka asetetaan aluksi johonkin alustusvektoriin IV. Salausfunktion ulostulon vasemmanpuoleiset, eli eniten merkitsevät s-bitit (engl. *most significant bit, MSB*) XOR:taan selkotekstin ensimmäisen segmentin P_1 kanssa, josta saadaan ensimmäinen salatun tekstin yksikkö C_1, joka sitten lähetetään. Lisäksi vuororekisterin sisältöä siirretään vasemmalle s-bittien määrä ja C_1 asetetaan oikeanpuoleisimmaksi, eli vähiten merkitseviksi (engl. *least significant bit, LSB*) s-biteiksi rekisterissä. Tätä jatketaan, kunnes selkotekstin kaikki yksiköt on käyty läpi ja salattu.

Salauksen purkamiseksi käytetään tätä samaa teemaa, poikkeuksena vastaanotettu salattu yksikkö XOR:taan salausfunktion ulostulon kanssa, selkotekstin yksikön saamiseksi. Vaikka CFB voidaan tulkita tietynlaisena jonosalauksena, ei se kuitenkaan ole toiminnaltaan täysin samanlainen tyypillisen jonosalauksen kanssa. (Stallings, 2014, ss. 185-187.)



KUVA 23. CFB-lohkosalaustila (Stallings, 2014, s. 186)

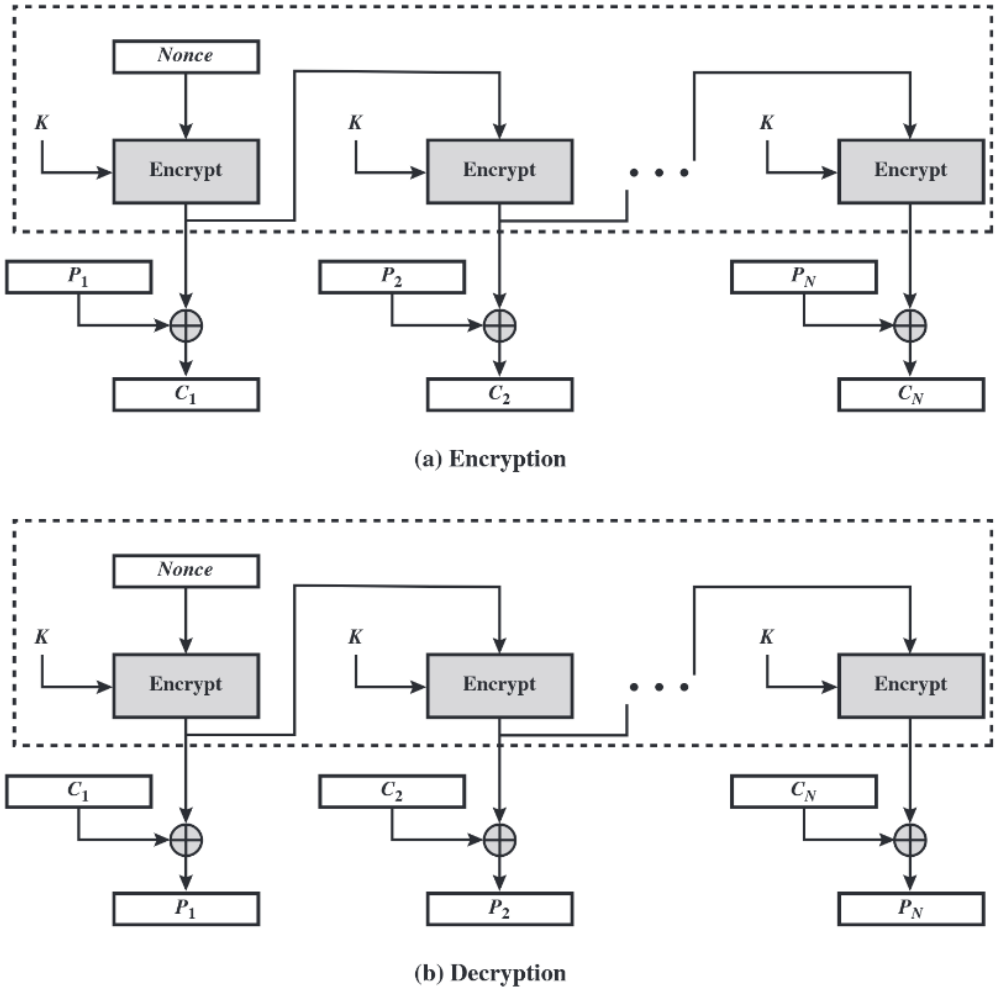
6.4 OFB

OFB on rakenteeltaan hyvin samanlainen CFB:n kanssa. OFB lohkosalaustilassa salausfunktion ulostulo toimii takaisinsyöttönä seuraavan selkotekstin lohkon salaukseen (KUVA 24). XOR-yksikön ulostulo syötetään takaisin seuraavan lohkon salauksen tuloksi. Toinen eroavaisuus on, että OFB tila operoi täysillä lohkoilla selkotekstiä ja salattua tekstiä, eikä s-bittisillä alijoukoilla, kuten CFB.

Kuten CBC ja CFB, OFB tila vaatii myös alustusvektorin IV. OFB:n tapauksessa IV:n on oltava *nonce*, eli sen on oltava uniikki jokaiselle suoritettavalle salauksen operaatiolle. Syy tähän on, että salattujen ulostulolohkojen sarja O_i riippuu ainoastaan avaimesta ja IV:stä eikä ollenkaan selkotekstistä.

OFB metodin yhtenä etuna on, että bitti virheet lähetyksessä eivät monistu, eli jos C_1:ssä tapahtuu bittivirhe, vain palautuva arvo P_1 kärsii tästä, eikä myöhemmät selkoteksti yksiköt korruptoidu tämän seurauksena. OFB:n rakenne on tyypillinen jonosalaus, koska salaus tuottaa bittien jonon alkuperäisen arvon ja avaimen funktiona, sekä tämä bittijono XOR:taan selkotekstin bittien kanssa. Kuvassa

24 olevat katkoviivaiset laatikot kuvaavat sitä, että tuotettava bittien jono joka XOR:taan selkotekstin kanssa on itsessään erillään selkotekstistä. (Stallings, 2014, ss. 187-189)



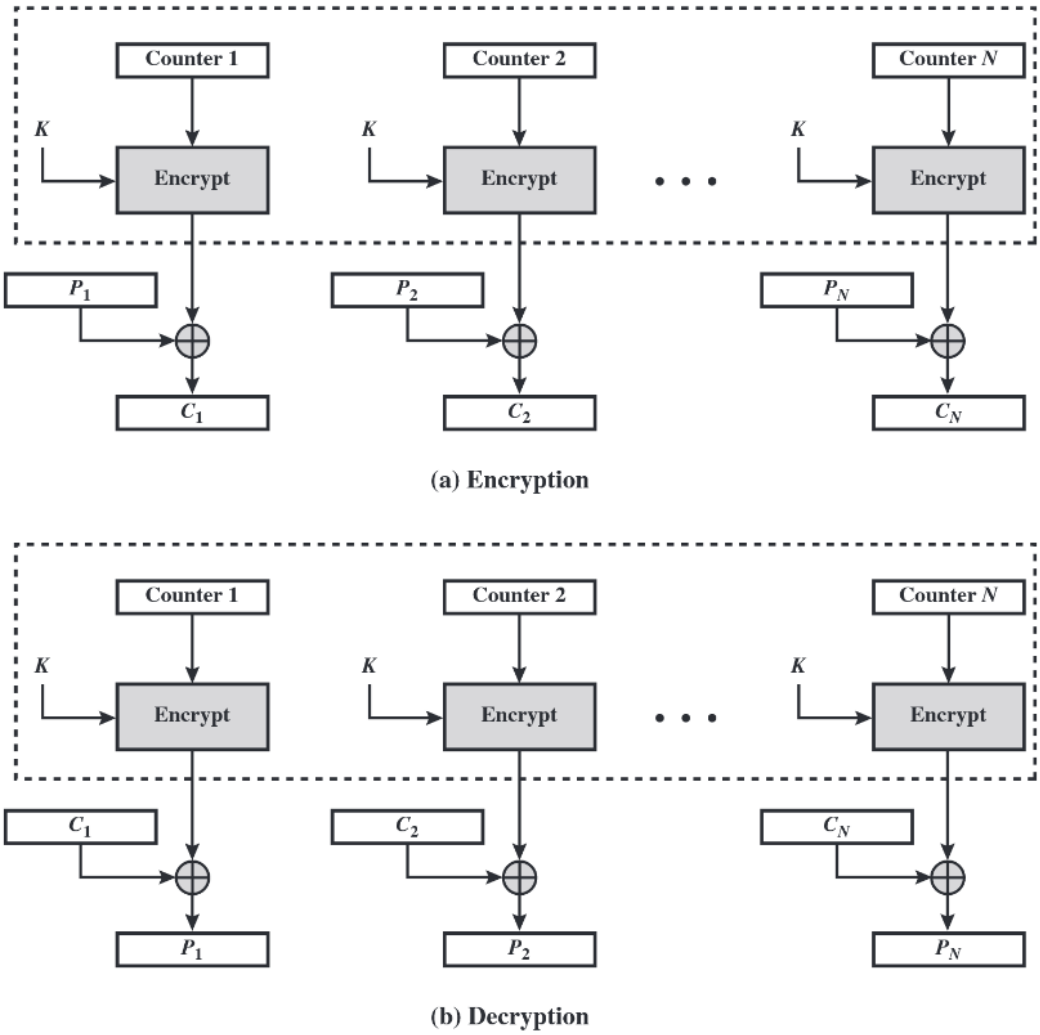
KUVA 24. OFB-lohkosalaustila (Stallings, 2014, s. 188)

6.5 CTR

CTR tilassa (KUVA 25) käytetään laskuria (engl. *counter*), joka tyypillisesti alustetaan johonkin arvoon ja kasvatetaan aina yhdellä jokaisella myöhemmällä lohkolla (*modulo* 2^b, missä b on lohkon koko). Laskurin arvo on siis selkotekstin lohkon koko ja laskurin arvon on oltava erilainen jokaiselle selkotekstille mikä salataan. Salauksessa, laskuri salataan ja sitten XOR:taan selkotekstin lohkon kanssa, josta saadaan salattu tekstilohko.

Salauksen purkamisessa käytetään samaa laskurin arvojen sarjaa, jokainen laskuri XOR:taan salatun tekstilohkon kanssa, jotta saadaan palautettua vastaava selkotekstin lohko. Tämä tarkoittaa, että alkuperäinen laskurin arvo on oltava tiedossa salauksen purkua varten.

Viimeiselle selkotekstin lohkolle, joka voi olla osittainen lohko u biteistä, eniten merkitsevät u bitit viimeisimmästä ulostulon lohkosta käytetään XOR-operaatioon ja loput $b - u$ bitit heitetään pois. Toisin kuin ECB, CBC ja CFB tiloissa, CTR:ssä ei tarvitse täyttää vajaata lohkoa johtuen CTR tilan rakenteesta.



KUVA 25. CTR-lohkosalaustila (Stallings, 2014, s. 190)

6.6 XTS-AES

XTS-AES on IEEE standardi, *IEEE Std 1618-2007*, jonka kehittivät IEEE Security in Storage Working Group (P1619). Standardi määrittelee salausmetodin data tallennettuna sektoriin -tyyppisille laitteille, missä uhkamalliin (engl. *Threat model*) kuuluu, että hyökkääjällä on pääsy tallennettuun dataan.

XTS-AES malli perustuu muokattavan tai säädettävän lohkosalauksen malliin (engl. *Tweakable block cipher*). Muokattavassa lohkosalausmallissa on kolme tuloa: selkoteksti *P*, symmetrinen avain *K* ja säätö (engl. *a tweak*) *T*, jotka tuottavat salatun tekstin *C*. Säätö *T* arvo on pidettävä salassa. Avaimen tarkoitus on tietysti tarjota turvallisuus, mutta säädön tarkoitus on tarjota vaihtelevuus. Tämä tarkoittaa, että eri säätöarvojen käyttö, selkotekstin ja avaimen pysyessä samana, tuottaa erilaisia tuloksia. (Stallings, 2014, ss. 191-192.)

Kuvassa 26 on tyypillisen säädettävän lohkosalauksen lohkokaavio. Salaus voidaan määritellä:

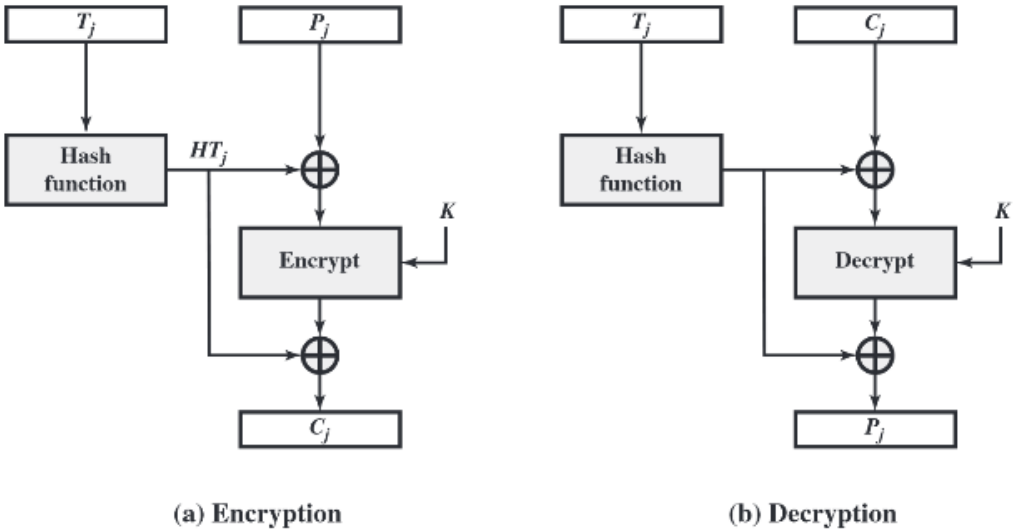
$$C = H(T) \oplus E(K, H(T) \oplus P)$$

Missä *H* on hajautusalgoritmi (engl. *hash function*). Salauksen purkamiselle voidaan käyttää samaa rakennetta, missä selkotekstiä käytetään tulona ja funktiona kuvan 26 ”Decryption” rakennetta. Jotta voidaan todistaa, että tämä toimii, voidaan kirjoittaa

$$H(T) \oplus C = E(K, H(T) \oplus P)$$

$$D[K, H(T) \oplus C] = H(T) \oplus P$$

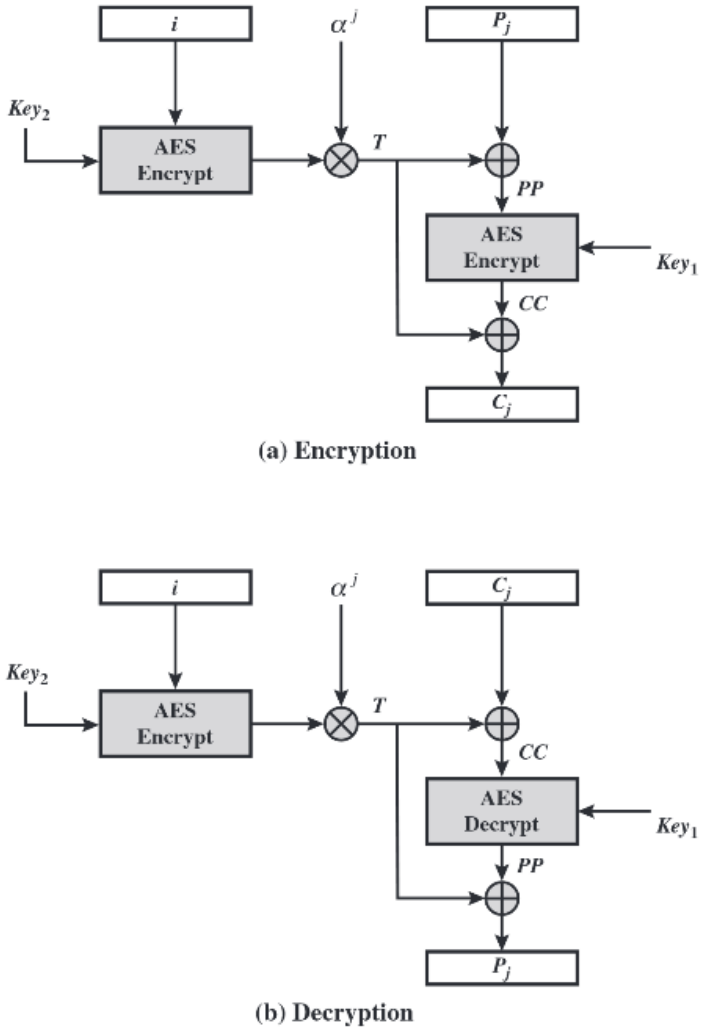
$$H(T) \oplus D(K, H(T) \oplus C) = P$$



KUVA 26. Säädettävä lohkosalaus (Stallings, 2014, s. 193)

Pohjimmiltaan käytetään ECB tilaa, mutta jokaiselle lohkolle säätöä *T* muutetaan. Näin päästään yli ECB:n turvallisuuteen liittyvästä heikkoudesta, joka on, että kaksi salausta samasta lohkosta tuottaa saman salatun tekstilohkon. (Stallings, 2014, ss. 192-193.)

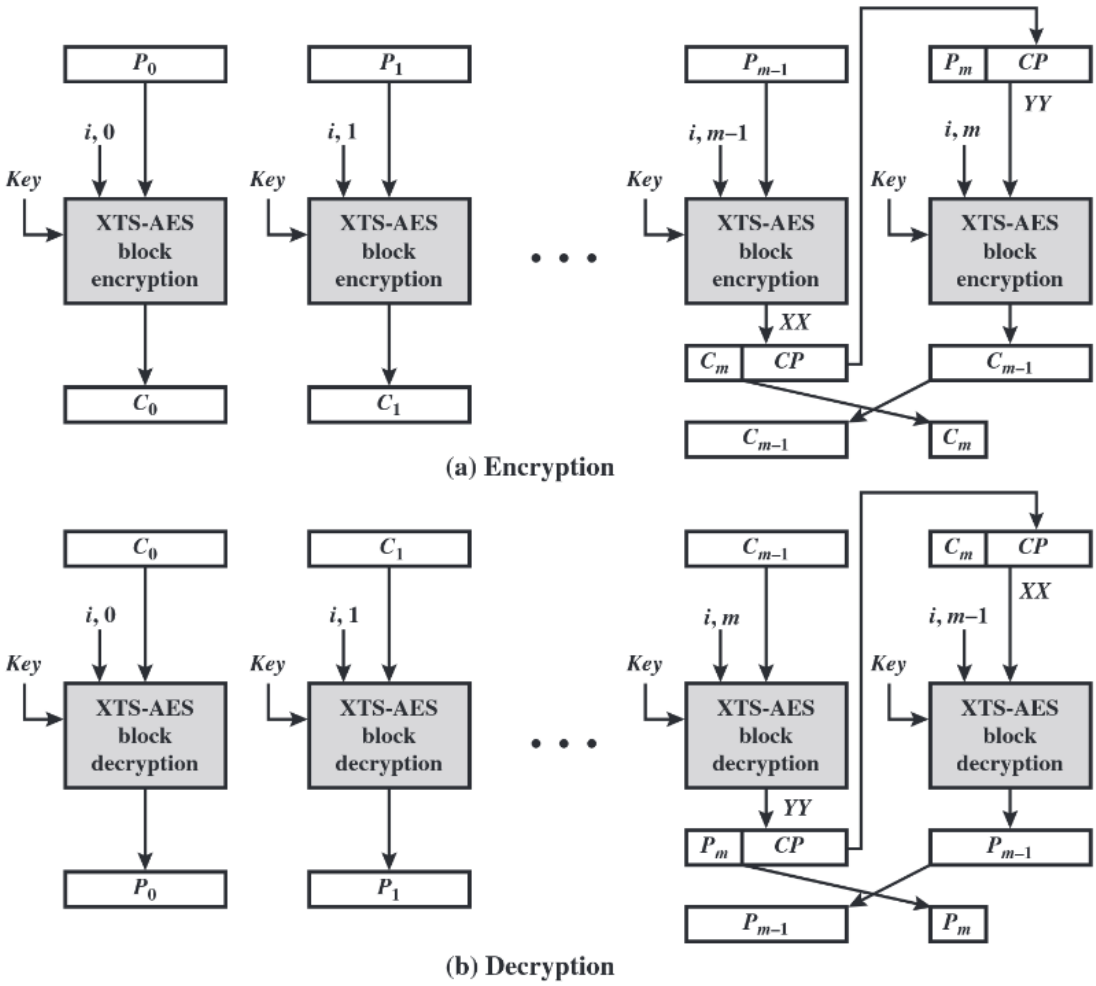
6.6.1 Operaatiot sektorilla



KUVA 27. XTS-AES operaatiot yhdelle lohkolle (Stallings, 2014, s. 195)

Sektorin tai datayksikön selkoteksti jaetaan 128 bitin kokoisiin lohkoihin. Lohkot nimetään P_0, P_1,…,P_m. Viimeinen lohko voi olla tyhjä (engl. *null*) tai voi sisältää 1-127 bittiä. XTS-AES algoritmin tulo koostuu siis m määrästä 128-bittisiä lohkoja ja mahdollisesti viimeisestä osittaisesta lohkosta.

Salauksessa ja salauksen purkamisessa, jokaista lohkoa käsitellään erikseen ja salataan/puretaan kuvan 27 mukaisesti. Ainoa poikkeus tapahtuu silloin, kun viimeisessä lohkossa on vähemmän kuin 128 bittiä. Tässä tapauksessa kaksi viimeistä lohkoa salataan/puretaan käyttäen ”salaustekstin varastamisen” -tekniikkaa (engl. *ciphertext-stealing*) lohkon täyttämisen sijasta. Tämä tekniikka näkyy kuvassa 28, jossa P_{m-1} on viimeinen täysi selkotekstin lohko ja P_m on viimeinen selkotekstin lohko, joka koostuu biteistä s välillä $1 \leq s \leq 127$. C_{m-1} on viimeinen täysi salattu tekstilohko ja C_m on viimeinen salattu tekstilohko, joka sisältää s määrän bittejä. ”Salaustekstin varastaminen” -nimitys tulee siitä, kun viimeisen lohkon prosessointi ”varastaa” väliaikaisesti salattua tekstiä viimeiseltä lohkolta täyttääkseen viimeisen salatun lohkon. (Stallings, 2014, s. 196.)



KUVA 28. XTS-AES lohkosalaustila (Stallings, 2014, s. 197)

7 LAITTEEN SD-KORTIN SALAUS

Tässä työssä esitettyjen sekä tutkittujen salausmenetelmien ja algoritmien joukosta potentiaalisimmaksi vaihtoehdoksi tällaisen sulautetun laitteen kannalta nousi *Advanced Encryption Standard* eli AES. Ennen varsinaisia teknisiä ominaisuuksia ja vaatimuksia, on mielestäni AES verrattuna näihin muihin mainittuihin menetelmiin jo yleisesti eri tasolla. Se on pitkän valintaprosessin läpi käynyt standardoitu algoritmi, sekä laajasti käytössä arkaluonteisten valtiontietojen salauksesta rajoitettujen resurssien ympäristöihin kuten sulautetut järjestelmät. Se voidaan myös toteuttaa sujuvasti sekä laitteisto-, että ohjelmistotasolla ja tarjoaa edellä esiteltyjen standardi lohkosalaustilojen kautta useampia vaihtoehtoja salauksen tekemiselle.

7.1 Salauksen taso

Laite kirjoittaa ja lukee dataa SD-kortilta 512-tavun sektoreissa. Liittyen datan kirjoittamiseen ja lukemiseen SD-kortin kanssa, laitteen firmware koostuu useasta eri ohjelmallisesta kerroksesta (engl. *software layers*). Näitä kerroksia ovat esimerkiksi tiedostojärjestelmä FatFs, joka sisältää tarvittavat

operaatiot tiedostojärjestelmän toiminnallisuuksille. FatFs-kerros kommunikoi siitä alemman kerroksen kanssa, joka hoitaa lopulta datan kirjoittamisen/lukemisen mikroprosessorin rekistereiden kautta SD-kortille. Perinteistä SPI-väylää käytetään datan siirtämiseen massamuistin kanssa, sekä myös DMA-tekniikkaa (engl. *Direct Memory Access*) eli oikosiirtoa, joka mahdollistaa nopean pääsyn suoraan muistipaikkoihin ilman tarvetta kuljettaa dataa suorittimen kautta ja näin kopioitavaa tietoa ei tarvitse käsitellä siirron aikana.

Tällaiselle laitteelle, jonka massamuistina toimii SD-kortti, salaus kortilla olevalle datalle (data at rest) voidaan suorittaa useammalla eri tavalla. Salaus voidaan tehdä tiedostotasolla, jolloin jokainen kirjoitettava tiedosto salattaisiin erikseen tai sitten esimerkiksi tiedostojärjestelmän tasolla, jossa data salataan levyn sektoritasolla. Yksittäisten tiedostojen salaaminen olisi todennäköisesti liian hidasta, sekä se ei tarjoa riittävää turvaa siinä tapauksessa, että laitteen voi purkaa ja lukea itse SD-kortin sisältöä erikseen ilman laitteen liittämistä PC:hen. Sektoritason salaus on näistä sopiva välimuoto, joka tarjoaa turvan myös siinä tapauksessa, että hyökkääjällä on fyysisesti pääsy laitteen SD-kortille. Sektorien salauksen purku onnistuisi vain siinä tapauksessa, kun käyttäjä yhdistää laitteen PC:hen ja syöttää laitteelle asetetun salasanan. Koska laitteen sd-kortille kirjoittaminen perustuu kiinteä kokoisiin sektoreihin, saadaan tähän sovitettua suoraan symmetrinen lohkosalaus AES sopivalla lohkosalaustilalla.

7.2 Laitteen vaatimukset

Sektoritasolla ja AES:lla salaamiseen on valittava laitteen, sen rajoitusten ja toiminnallisuuksien perusteella sopiva lohkosalaustila. On huomioitava, että lukuun ottamatta ECB ja XTS lohkosalaustiloja, kaikki muut tilat hyödyntävät jollakin tavalla tietynlaista ”takaisinkytkentää” (engl. *feedback*), jossa seuraavan selkotekstilohkon salaamiseen tarvitaan tietoa edellisen lohkon tuloksesta.

Lohkosalaustilat, jotka muutetaan luvussa 6 esitetyllä tavalla jonosalauksen tyyppisiksi, esimerkiksi CFB, OFB ja CTR, eivät ole periaatteessa järkeviä vaihtoehtoja, koska jonosalauksessa vaaditaan niiden turvallisuuden säilymiseksi, että samaa alkutilaa ei käytetä kahdesti. Joka tässä tapauksessa tapahtuisi, jos sektori päivitetään eri datalla. Tämä vaatisi salaukselta sen, että tallennettaisiin erikseen erilliset alkutilat jokaiselle sektorille ja tällä tavalla menisi paljon tilaa hukkaan. ECB lohkosalaustilaa ei ole järkevää turvallisuuden kannalta käyttää useammalle lohkolle, johtuen ECB:n ominaisuudesta, jossa sama selkokielinen tekstilohko tuottaa aina saman salatun tekstilohkon.

Yksi suuri rajoittava tekijä on se, että käytössä on hajasaantimuistia, jolloin ei voida olettaa sellaisissa lohkosalaustiloissa, joissa seuraavan selkotekstilohkon salaamiseen tarvitaan edellisen salatun lohkon dataa, että sitä on käytettävissä. Standardi lohkosalaustiloista XTS-AES on yleisesti paljon käytetty lohkosalaustila, kun salataan hajasaantimuistissa olevaa dataa. Edellä esitetyllä tavalla siinä jokaista lohkoa käsitellään erikseen ja se on suunniteltu juuri ympäristöön, jossa dataa tallennetaan sektoreittain.

7.3 AES instruction set

Käytettäessä AES algoritmia salaukseen tällaisen mukana kannettavan laitteen kaltaisessa rajallisten resurssien ympäristössä on myös se etu, että useat mikroprosessori valmistajat tarjoavat tietynlaista AES käskykantaa tai käskyjoukkoa (engl. *AES instruction set*) sisällytettynä itse mikroprosessorin arkkitehtuuriin. Yleisesti puhutaan myös ”salauskiihdyttimestä” (engl. *encryption accelerator*). Tämä mahdollistaa esimerkiksi entistä nopeamman AES:ia käyttävän sovellustoteutuksen toiminnan. Tämä kertoo myös paljon AES algoritmin suosiosta.

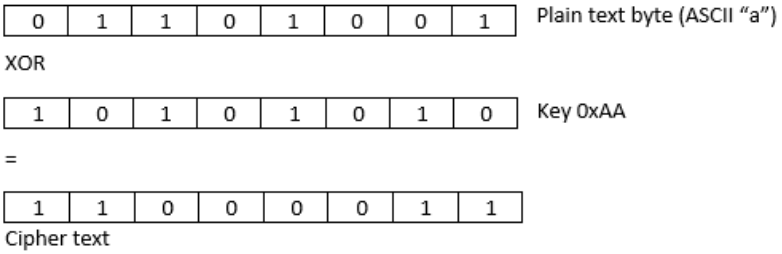
Käytettäessä AES algoritmia mikroprosessoriin sisällytetyn AES käskyjoukon kautta, parantaa tämä myös algoritmin turvallisuutta esimerkiksi ”*side-channel attack*” -hyökkäystä vastaan verrattuna pelkkään sovellustoteutukseen AES algoritmista. Side-channel attack on sellainen hyökkäys, jossa hyödynnetään salausta käytettävästä systeemistä saatavaa tietoa, eikä niinkään heikkouksia itse salausalgoritmista. Nämä hyökkääjän tarkkailemat tiedot voivat olla esimerkiksi laitteen virrankulutus, elektromagneettiset vuodot tai laitteen järjestelmään liittyvät ajoitukset. (Side-channel attack, 2021.)

Mikroprosessorien joukossa esimerkiksi ARM-arkkitehtuurin useat prosessorit (esimerkiksi ARM Cortex-A53 ja A57) sisältävät tämän AES käskykannan. Valitettavasti tähän työhön liittyvässä medikaalilaitteen MSP430 mikroprosessorin versiossa ei tällaista salauskiihdytintä ole. Osittain tästä johtuen tehtiin työn testauksen osalta päätös, että itse AES algoritmia ei tulla testaamaan laitteen firmwaressa sellaisenaan, vaan käytetään yksinkertaisempaa luvussa 8 kuvattua tapaa salauskonseptin testaukseen.

8 TESTAUS

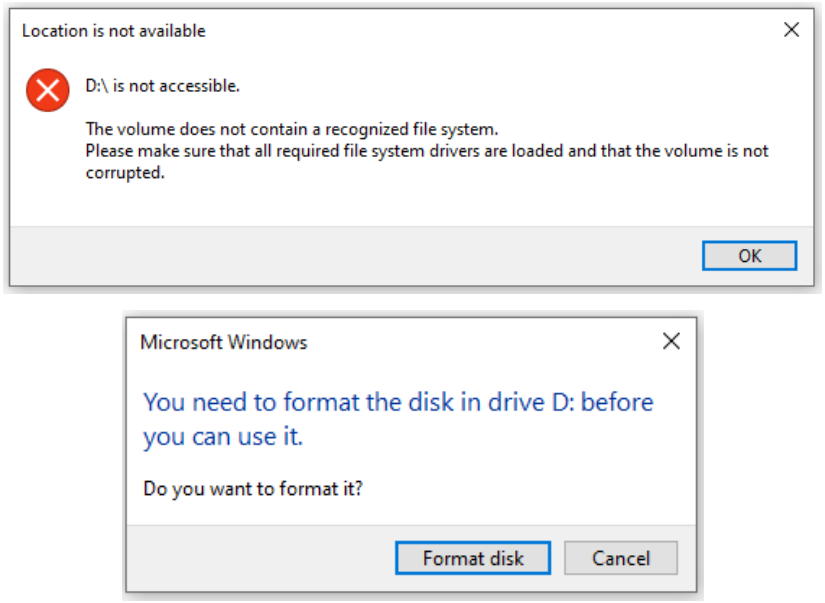
Osittain tämän työn osalta käytettävissä ja testattavissa olleen medikaalilaitteen mikroprosessori version rajallisuuksien, sekä tiukan aikataulun vuoksi, ei AES salausta sellaisenaan laitteessa testattu. Sen sijaan, jotta työ ei jäisi vain kirjallisuus ja tutkimustieto katsaukseksi, vaan saataisiin myös konkreettista tietoa salauksen sopivuudesta laitteelle, päätettiin laitteelle toteuttaa yksinkertaisempaan XOR-operaatioon perustuva salaus. Tämän salausmenetelmän testi tehtiin itse laitteen firmwareen ja se toteutettiin niin, että se todistaisi riittävällä tasolla mahdollisen oikean AES salauksen konseptin, joka tarkoittaa, että laitteen SD-kortille menevää dataa on mahdollista salata sektori kerrallaan.

XOR-operaatioon perustuva salaus toteutettiin ennen alinta SD-korttia käyttävää ohjelmistokerrosta. Alimmalle kerrokselle lähetettävät kiinteän 512 tavun kokoiset datalohkot salataan XOR-operaatiolla, eli ne käydään esimerkiksi läpi tavu tavulta ja XOR:taan avaimen, esimerkiksi heksaluku 0xAA, kanssa (KUVA 29).



KUVA 29. XOR-operaatio salaus

Ensimmäisenä SD-kortti formatoidaan sisäisesti, kun laite resetoidaan ensimmäisen kerran käyttämällä Fatfs:n f_mkfs() -funktiota. Tämä formatointi tapahtuu XOR-operaatioiden ollessa päällä, jolloin pelkästään laite osaa lukea formatoitua SD-korttia, koska funktioihin, joita laite käyttää kortin lukemiseen, on myös toteutettuna XOR-operaatio takaisinpäin. Tämä tarkoittaa sitä, että kun laite yhdistetään tietokoneeseen ja käyttäjä syöttää laitteen kortille asetetun salasanan, purkaa laite datan XOR-operaatioiden kautta taas tulkittavaksi USB:n kautta tietokoneelle, jolloin windowsin päässä näkyy laitteen massamuisti normaalisti. Siinä tapauksessa, että laitteesta irrotetaan SD-kortti ja yritetään lukea pelkästään korttia syöttämällä kortti tietokoneen SD-korttipaikkaan, ilmoittaa windows, että kortti vaatii formatoinnin koska se ei pysty tulkitsemaan sen sisältöä (KUVA 30). Tämä siksi, että tässä tapauksessa välistä puuttuu laitteen suorittama XOR-operaatio takaisinpäin, koska tätä ei suoriteta, kun laite ei ole kytkettynä USB:n kautta tietokoneeseen.



KUVA 30. Pelkän salatun SD-kortin lukeminen

9 YHTEENVETO JA POHDINTA

Opinnäytetyön tavoitteena oli tutkia mahdollisuuksia ja vaihtoehtoja pienen, käyttäjän mukana kannettavan medikaalilaitteen SD-kortin tiedostojärjestelmän salaukselle. Tutkimuksen syynä ja taustana on uhkamalli, jossa mahdollinen ulkopuolinen taho/ihminen voi purkaa laitteen, jolloin laitteen SD-kortille ja sillä olevalle datalle on vapaa pääsy. Tiedostojärjestelmän salauksella pystytään tässä tapauksessa suojelemaan laitteella olevia tietoja, jolloin tiedostojärjestelmän salauksen purku tapahtuu vain siinä tapauksessa, kun käyttäjä yhdistää laitteen USB:n kautta tietokoneeseen ja syöttää sille asetetun salasanan.

Opinnäytetyön tutkimuksen tuloksena löytyi potentiaalinen vaihtoehto tällaisen tiedostojärjestelmän tason salauksen toteuttamiselle tämänkaltaisen sulautetun laitteen rajallisten resurssien ympäristöön. Potentiaalisimpana salausmenetelmänä päädyttiin AES eli Advanced Encryption Standard -algoritmiin, joka on paljon käytössä oleva, standardi algoritmi, josta löytyy sopiva lohkosalaustila data tallennettuna sektoriin -tyyppisille laitteille. AES algoritmia ei kuitenkaan toteutettu sellaisenaan tämänhetkiseen laitteeseen, johtuen tällä hetkellä laitteessa olevasta mikroprosessorin versiosta ja aikataulullisista haasteista. Sen sijaan päätettiin toteuttaa yksinkertaisempi XOR-operaatioon perustuva salaustesti, joka todistaa tulevaisuutta varten potentiaalisen AES algoritmin toteutuksen konseptin, eli että salaus on mahdollista SD-kortin sektoritasolla laitteeseen toteuttaa.

Mahdollisia laitteen tulevia versioita varten, joissa on esimerkiksi käytössä tehokkaampi, AES salauskiihdyttimen sisältävä mikroprosessori, työ ja sen lopussa tehty XOR-salauksen testi, antoi alustavaa tietoa, miten salauksen voisi laitteeseen implementoida.

Työ antoi myös sen tekijälle paljon tietoa eri salausmenetelmistä ja algoritmeista. Lopputestauksena tehty XOR-operaatioon perustuva salaus oli haastava, koska laitteen firmware koostuu hyvin monesta ohjelmallisesta kerroksesta ja sinne tällaisen uuden ominaisuuden toteuttaminen ei ollut yksinkertaista, rikkomatta olemassa olevaa koodia. Lopputestauksen toimivuus kuitenkin todisti lohkosalauksen konseptin, joka on täysin toteutettavissa laitteelle.

LÄHTEET

Advanced Encryption Standard Process. (27. 11. 2020). Haettu 4. 4. 2021 osoitteesta Wikipedia: https://en.wikipedia.org/wiki/Advanced_Encryption_Standard_process

Aivosähkökäyrätutkimus EEG. (24. 9. 2019). Haettu 20. 2. 2021 osoitteesta Terveyskylä: https://www.terveyskyla.fi/tutkimukseen/eri-tutkimuksia/yleisimm%C3%A4t-kuvantamistutkimukset/aivos%C3%A4hk%C3%B6k%C3%A4yr%C3%A4-eeg

Aleksander Jurisic, A. J. (4. 1997). *Strong digital signature algorithms*. Haettu 26. 3. 2021 osoitteesta Elliptic Curves and Cryptography: http://dns.uls.cl/~ej/daa_08/Algoritmos/books/book10/9704b/9704b.htm#rt2

Barr, M. (1. 10. 2001). *Introduction to Watchdog Timers*. Haettu 6. 3. 2021 osoitteesta embedded: https://www.embedded.com/introduction-to-watchdog-timers/

Bittium. (2021). Haettu 20. 2. 2021 osoitteesta BIttium: https://www.bittium.com/

Block cipher mode of operation. (3. 3. 2021). Haettu 11. 4. 2021 osoitteesta Wikipedia: https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

Blowfish (cipher). (28. 1. 2021). Haettu 28. 3. 2021 osoitteesta Wikipedia: https://en.wikipedia.org/wiki/Blowfish_(cipher)

ChaN. (5. 12. 2020). *FatFs - Generic FAT Filesystem Module*. Haettu 6. 3. 2021 osoitteesta Electronic Lives Mfg.: http://elm-chan.org/fsw/ff/00index_e.html

ChaN. (5. 12. 2020). *FatFs Module Application Note*. Haettu 6. 3. 2021 osoitteesta Electronic Lives Mfg.: http://elm-chan.org/fsw/ff/doc/appnote.html

Christensson, P. (2006). *Firmware definition*. Haettu 20. 2. 2021 osoitteesta TechTerms: https://techterms.com/definition/firmware

Cobb, M. (11. 2014). *TechTarget*. Haettu 27. 3. 2021 osoitteesta Data Encryption Standard (DES): https://searchsecurity.techtarget.com/definition/Data-Encryption-Standard

Confusion and diffusion. (11. 11. 2021). Haettu 3. 4. 2021 osoitteesta Wikipedia: https://en.wikipedia.org/wiki/Confusion_and_diffusion

Data Encryption Standard. (2021). Haettu 27. 3. 2021 osoitteesta Tutorialspoint: https://www.tutorialspoint.com/cryptography/data_encryption_standard.htm

Feistel Block Cipher. (2021). Haettu 27. 3. 2021 osoitteesta Tutorialspoint: https://www.tutorialspoint.com/cryptography/feistel_block_cipher.htm

FreeRTOS. (ei pvm). Haettu 7. 3. 2021 osoitteesta What is An RTOS?: https://www.freertos.org/about-RTOS.html

Gatliff, B. (15. 7. 2003). *Encrypting data with the Blowfish algorithm*. Haettu 28. 3. 2021 osoitteesta Embedded: https://www.embedded.com/encrypting-data-with-the-blowfish-algorithm/

International Data Encryption Algorithm. (29. 1. 2021). Haettu 27. 3. 2021 osoitteesta Wikipedia: https://en.wikipedia.org/wiki/International_Data_Encryption_Algorithm#cite_note-2

Jauhiainen, J. (2009). *T291004 Lääketieteelliset mittauslaitteet: Biopotentiaalit ja niiden mittaus.* Haettu 21. 2. 2021 osoitteesta OAMK: http://www.oamk.fi/~jjauhiai/opetus/mittalaitteet/biopotentiaali.pdf

Kaliski, B. (2005). *MIPS-Year*. Haettu 26. 3. 2021 osoitteesta Encyclopedia of Cryptography and Security: https://link.springer.com/referenceworkentry/10.1007%2F0-387-23483-7_255

Kardiologia. (16. 11. 2020). Haettu 20. 2. 2021 osoitteesta Wikipedia: https://fi.wikipedia.org/wiki/Kardiologia

Liikenne- ja viestintäministeriö. (2. 2018). *Sähköisen viestinnän salaus- ja suojausmenetelmät.* Haettu 21. 3. 2021 osoitteesta Liikenne- ja viestintäministeriö: https://julkaisut.valtioneuvosto.fi/bitstream/handle/10024/160614/LVM_02_2018_Sahkoisen_viestinnan%20salaus_ja_suojaus.pdf

Maletsky, K. (2020). *RSA vs. ECC Comparison for Embedded Systems.* Haettu 21. 3. 2021 osoitteesta Microchip: https://www.microchip.com/content/dam/mchp/documents/SCBU/ProductDocuments/SupportingCollateral/00003442A.pdf

MDS matrix. (18. 2. 2021). Haettu 3. 4. 2021 osoitteesta Wikipedia: https://en.wikipedia.org/wiki/MDS_matrix

Microsoft. (27. 8. 2019). *exFAT file system specification*. Haettu 4. 3. 2021 osoitteesta Microsoft: https://docs.microsoft.com/en-us/windows/win32/fileio/exfat-specification

Microsoft. (12. 7. 2020). *Overview of FAT, HPFS, and NTFS File Systems*. Haettu 4. 3. 2021 osoitteesta Microsoft: https://docs.microsoft.com/en-us/troubleshoot/windows-client/backup-and-storage/fat-hpfs-and-ntfs-file-systems

NIST. (26. 11. 2001). Publication 197. *FIPS*. Haettu 4. 4. 2021 osoitteesta Announcing the ADVANCED ENCRYPTION STANDARD (AES): https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf

omni sci. (2020). *Embedded System Definition*. Haettu 21. 2. 2021 osoitteesta omni sci: https://www.omnisci.com/technical-glossary/embedded-systems

Orton. (ei pvm). *Neurologia*. Haettu 20. 2. 2021 osoitteesta Orton: https://www.orton.fi/palvelut/hoitopalvelut/neurologia/

Pedamkar, P. (2020). *IDEA Algorithm*. Haettu 27. 3. 2021 osoitteesta EDUCBA: https://www.educba.com/idea-algorithm/

Pertti Mustajoki, J. K. (9. 7. 2008). *EKG (sydänfilmi), Laboratoriotutkimusten tulkinta*. Haettu 20. 2. 2021 osoitteesta Terveyskirjasto: https://www.terveyskirjasto.fi/terveyskirjasto/tk.koti?p_artikkeli=snk03210

R.L. Rivest, A. S. (ei pvm). *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.* Haettu 18. 3. 2021 osoitteesta http://people.csail.mit.edu/rivest/Rsapaper.pdf

RSA Algorithm in Cryptography. (5. 1. 2021). Haettu 19. 3. 2021 osoitteesta GeeksforGeeks: https://www.geeksforgeeks.org/rsa-algorithm-cryptography/

S-box. (27. 1. 2021). Haettu 28. 3. 2021 osoitteesta Wikipedia: https://en.wikipedia.org/wiki/S-box

Schneier, B. (1994). *Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)*. Haettu 28. 3. 2021 osoitteesta Schneier on Security: https://www.schneier.com/academic/archives/1994/09/description_of_a_new.html

Schneier, B. (12. 1998). *The Twofish Encryption Algorithm*. Haettu 2. 4. 2021 osoitteesta Schneier on Security: https://www.schneier.com/academic/archives/1998/12/the_twofish_encrypti.html

Schoof's Algorithm. (22. 12. 2019). Haettu 26. 3. 2021. osoitteesta Wikipedia: https://en.wikipedia.org/wiki/Schoof%27s_algorithm

Side-channel attack. (13. 4. 2021). Haettu 17. 4. 2021 osoitteesta Wikipedia: https://en.wikipedia.org/wiki/Side-channel_attack

Stallings, W. (2014). *Cryptography and Network Security: Principles and Practice (6th Edition).* Pearson Education Inc. Haettu 10. 4. 2021

Stoffregen, P. (24. 2. 2005). *Understanding FAT32 Filesystems*. Haettu 7. 3. 2021 osoitteesta PJRC: https://www.pjrc.com/tech/8051/ide/fat32.html

Svec, C. (ei pvm). *FreeRTOS*. Haettu 8. 3. 2021 osoitteesta The Architecture of Open Source Applications: https://www.aosabook.org/en/freertos.html

Texas Instruments. (9. 2020). *MSP430F5659.* Haettu 6. 3. 2021 osoitteesta Texas Instruments: https://www.ti.com/lit/ds/symlink/msp430f5659.pdf?ts=1615030426688&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FMSP430F5659

The FreeRTOS Kernel. (ei pvm). Haettu 7. 3. 2021 osoitteesta FreeRTOS: https://www.freertos.org/RTOS.html

Townsend, P. (25. 3. 2019). *RSA VS AES ENCRYPTION - A PRIMER*. Haettu 19. 3. 2021 osoitteesta TOWNSEND SECURITY DATA PRIVACY BLOG: https://info.townsendsecurity.com/rsa-vs-aes-encryption-a-primer

USB Implementers Forum. (27. 5. 2001). *Device Class Definition for HID 1.11.* Haettu 11. 3. 2021 osoitteesta USB-IF: https://www.usb.org/sites/default/files/hid1_11.pdf

USB Implementers Forum. (19. 2. 2010). *Mass Storage Class Specification Overview 1.4 .* Haettu 11. 3. 2021 osoitteesta USB-IF: https://usb.org/sites/default/files/Mass_Storage_Specification_Overview_v1.4_2-19-2010.pdf

Vivek Kapoor, V. S. (20. 5. 2008). Haettu 25. 3. 2021 osoitteesta Elliptic Curve Cryptography: https://dl.acm.org/doi/pdf/10.1145/1386853.1378356

Wikipedia. (26. 3. 2020). *FAT*. Haettu 4. 3. 2021 osoitteesta Wikipedia: https://fi.wikipedia.org/wiki/FAT

Wikipedia. (12. 3. 2021). *RSA (cryptosystem)*. Haettu 19. 3. 2021 osoitteesta Wikipedia: https://en.wikipedia.org/wiki/RSA_(cryptosystem)

Äärellinen kunta. (10. 5. 2016). Haettu 26. 3. 2021 osoitteesta Wikipedia: https://fi.wikipedia.org/wiki/%C3%84%C3%A4rellinen_kunta