Teemu Mäkelä

Salakirjoituksen ja obfuskoinnin vaikutus ohjelmakoodin suorituskykyyn

Tietotekniikan kandidaatintutkielma

22. heinäkuuta 2024

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Teemu Mäkelä

Yhteystiedot: tjmakela@student.jyu.fi

Ohjaaja: Jonne Itkonen

Työn nimi: Salakirjoituksen ja obfuskoinnin vaikutus ohjelmakoodin suorituskykyyn

Title in English: The Impact of Encryption and Obfuscation on Software Code Performance

Työ: Kandidaatintutkielma

Opintosuunta: Tieto- Ja Ohjelmistotekniikka

Sivumäärä: 23+0

Tiivistelmä: Salakirjoitus ja obfuskointi ovat keskeisiä tekniikoita ohjelmakoodin suojaamiseksi luvattomalta pääsyltä ja väärinkäytöltä. Näiden menetelmien avulla voidaan parantaa tietoturvaa, mutta ne voivat myös vaikuttaa ohjelmiston suorituskykyyn. Tämä tutkielma tarkastelee, miten salakirjoitus ja obfuskointi vaikuttavat ohjelmakoodin tehokkuuteen ja suoritusnopeuteen. Analyysissä käsitellään erilaisia salakirjoitusalgoritmeja ja obfuskointitekniikoita, ja arvioidaan niiden vaikutuksia käsittelyaikaan, muistinkäyttöön ja järjestelmän kokonais-suorituskykyyn. Lisäksi tutkielmassa tarkastellaan tietoturvan ja suorituskyvyn välistä tasapainoa ja annetaan näkemyksiä siitä, kuinka näitä tekniikoita voidaan optimoida eri ohjelmistoympäristöissä. Myös mahdolliset parannukset ja tulevaisuuden kehityssuunnat suorituskykyhaittojen minimoimiseksi ja vahvan tietoturvan ylläpitämiseksi ovat esillä.

Avainsanat: Enkryptio, Obfuskaatio, Salaus

Abstract: Encryption and obfuscation are crucial techniques for protecting software code from unauthorized access and misuse. While these methods enhance security, they can also impact the performance of the software. This study examines how encryption and obfuscation affect the efficiency and speed of software code execution. The analysis includes various encryption algorithms and obfuscation techniques, evaluating their influence on processing time, memory usage, and overall system performance. Additionally, the study explores the trade-offs between security and performance, providing insights into optimizing the use of

these techniques in different software environments. Potential improvements and future developments in minimizing performance overhead while maintaining robust security are also discussed.

Keywords: Encryption, Obfuscation

Termiluettelo

Algoritmi	Määrättyjen sääntöjen tai ohjeiden järjestetty sarja, joka määrittelee tietyn laskennallisen toiminnon suorituksen.
Dekoodaus ja purkaminen	Koodatun tiedon palauttaminen alkuperäiseen muotoonsa.
Lähdekoodi	Ohjelmiston tai tietokoodin alkuperäinen, ihmisen luettava muoto.
Merkkijono	Merkkien sarja, kuten sanat tai lauseet.
Salakirjoitus	Tiedon muuttaminen siten, että vain tietyn avaimen avulla se voidaan palauttaa alkuperäiseen muotoonsa.
Obfuskointi	Ohjelmakoodin muokkaaminen vaikeammin ymmärrettävään muotoon, jotta luvaton käyttö ja analysointi olisivat hankalampia.

Kuviot

Kuvio 1. Erilaisia kryptografisia algoritmeja ja niiden kulkua 4
Kuvio 2. Esimerkki hash-funktion transformoinnista 5
Kuvio 3. Ennen ja jälkeen control-flow obfuskaation (Kuvan lähde: PreEmptive. Käytetään luvalla) 7
Kuvio 4. Ennen ja jälkeen uudelleen nimeämisen (PreEmptive 2022) 8

Sisällys

1 JOHDANTO 1

2 SALAKIRJOITUKSEN PERUSTEET 2
2.1 Tausta 2
2.2 Salainen avainkryptografia 2
2.3 Julkinen avainkryptografia 3
2.4 Hash-funktio 4

3 OBFUSKAATION PERUSTEET 6
3.1 Tausta 6
3.2 Merkkijonon salaus 6
3.3 Control-flow obfuskaatio 7
3.4 Uudelleen nimeäminen 8

4 PÄÄTELMÄT JA YHTEENVETO 9

LÄHTEET 11

1 Johdanto

Digitalisaation edetessä ja teknologisten ratkaisujen monipuolistuessa tietoturvasta on tullut keskeinen huolenaihe ohjelmistokehityksessä. Ohjelmistojen lähdekoodin suojaaminen ulkopuolisilta hyökkäyksiltä ja luvattomalta pääsyltä on käynyt yhä tärkeämmäksi, erityisesti kun kyberuhkat ovat monimutkaisempia ja haitalliset toimijat yhä innovatiivisempia.

Salakirjoitus on tärkeä menetelmä ohjelmistojen suojaamiseksi, sillä se muuntaa tiedot siten, että ne voidaan palauttaa alkuperäiseen muotoonsa vain oikealla avaimella. Tämä suojaa herkkiä tietoja estäen niiden väärinkäytön ja lisäämällä ohjelmistojen tietoturvaa.

Obfuskointi puolestaan tekee ohjelmakoodista vaikeammin ymmärrettävää ilman, että sen toiminnallisuus muuttuu. Se voi sisältää tekniikoita kuten koodin muuttamista hämmentävämmäksi ja tarpeettomien komponenttien lisäämistä. Tämä lisää ohjelmakoodin vastustuskykyä haitallisille hyökkäyksille ja estää haitallisten ohjelmistojen analysointia tai muokkaamista.

Tässä kandidaatintutkielmassa keskitytään erityisesti siihen, miten salakirjoitus ja obfuskointi vaikuttavat ohjelmakoodin suorituskykyyn. Ensimmäisessä osassa tarkastellaan salakirjoituksen vaikutuksia ohjelmiston tehokkuuteen. Toisessa osassa analysoidaan obfuskoinnin vaikutusta ohjelmakoodin suorituskykyyn ja sen mahdollisia suorituskykyhaittoja.

Tutkielman tavoitteena on syventää ymmärrystä siitä, kuinka nämä suojaustekniikat vaikuttavat ohjelmakoodin suorituskykyyn ja selvittää, millaisia haasteita niiden käyttöön liittyy. Lisäksi arvioidaan, miten salakirjoitus- ja obfuskointimenetelmiä voidaan optimoida suorituskyvyn ja tietoturvan tasapainottamiseksi eri ohjelmistoympäristöissä. Käytännön näkökulmasta vertaillaan erilaisia salakirjoitus- ja obfuskointityökaluja ja arvioidaan niiden vaikutuksia ohjelmakoodin suorituskykyyn.

2 Salakirjoituksen perusteet

Tässä luvussa käydään perusteellisesti läpi salakirjoituksen perusteita ja keskeisiä käsitteitä.

2.1 Tausta

Salakirjoitusta käytetään laajalti tietojen suojaamiseen ja ohjelmakoodin piilottamiseen. Näiden käsitteiden ymmärtäminen on tärkeää tietoturvan asiantuntijoille ja ohjelmistokehittäjille, jotka pyrkivät suojaamaan arkaluonteisia tietoja ja estämään ei-toivottuja pääsyjä tietojärjestelmiin (Schneier 1996).

Salakirjoitus on menetelmä, jolla viesti muutetaan siten, että se on luettavissa vain niille, joilla on oikea salausavain. Tämä lisää merkittävästi tietojen turvallisuutta ja estää ulkopuolisia tahoja pääsemästä käsiksi herkkiin tietoihin. Salakirjoitusmenetelmiä on useita, kuten symmetrinen ja asymmetrinen salakirjoitus, jonka perusperiaatteet esittelivät Diffie ja Hellman (Diffie ja Hellman 1976). Symmetrinen salakirjoitus käyttää samaa avainta sekä salaamiseen että purkamiseen, kun taas asymmetrinen salakirjoitus käyttää kahta eri avainta – julkista ja yksityistä.

2.2 Salainen avainkryptografia

On useita tapoja luokitella kryptografisia algoritmeja, joista ensimmäinen on symmetrinen avainkryptografia (SAK) (Menezes, Oorschot ja Vanstone 1996). Lähettäjä käyttää avainta salaamaan selkokielisen viestin ja lähettää salakirjoitetun viestin vastaanottajalle. Vastaanottaja käyttää samaa avainta viestin purkamiseen ja alkuperäisen selkokielisen viestin palauttamiseen. Koska yhtä avainta käytetään molempiin toimintoihin, salainen avainkryptografiaa kutsutaan myös symmetriseksi salaukseksi (Shannon 1949). Tämän salausmuodon kanssa on ilmeistä, että avaimen on oltava tiedossa sekä lähettäjällä että vastaanottajalla; itse asiassa se on se salaisuus. Tämän lähestymistavan suurin vaikeus liittyy avaimen jakeluun koska sen kanssa pitää olla niin varovainen ettei se vuoda (Bellare ja Yee 2003).

Symmetrinen salakirjoitus, kuten salainen avainkryptografia, voi olla suorituskyvyltään no-

pea, koska sama avain käytetään sekä salaamiseen että purkamiseen (Simmons 1979). Tämä mahdollistaa tehokkaan käsittelyn erityisesti suurilla tietojoukoilla. Suurten tietojoukkojen käsittely voi kuitenkin aiheuttaa lisääntyneen laskennallisen kuormituksen, etenkin kun salakirjoituksen avainta vaihdetaan säännöllisesti. Tämä vaikuttaa ohjelman suorituskykyyn, mutta modernit salakirjoitusmenetelmät on optimoitu minimoidakseen haittoja (Benjamin Lynn 2004).

Symmetrisen avainkryptografian käytön vaikutus koodin suorituskykyyn riippuu käytetyn algoritmin tyypistä, avainkokoista ja toteutuksen tehokkuudesta. Yleisesti ottaen symmetriset algoritmit, kuten AES (Advanced Encryption Standard) voivat käsitellä suuria tietomääriä nopeasti ja tehokkaasti.

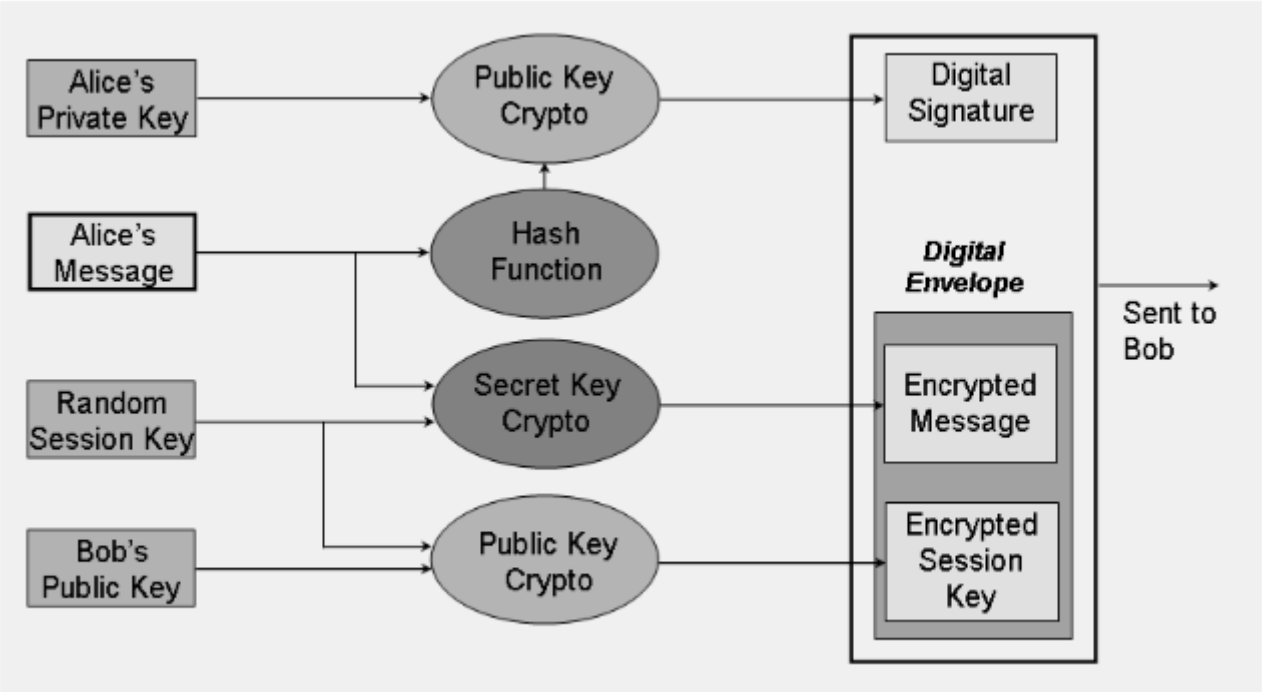
2.3 Julkinen avainkryptografia

Julkinen avainkryptografia (JAK), joka tunnetaan myös epäsymmetrisenä salauksena, käyttää kahta erillistä avainta: yhtä avainta salaukseen ja toista purkamiseen (M. E. Hellman 1978). Tämä menetelmä on keskeinen erityisesti tunnistautumisessa, kiistämättömyydessä ja avainten vaihtamisessa. JAK mahdollistaa turvallisen viestinnän suojaamattoman viestintäkanavan yli ilman tarvetta jakaa salaisuusavainta, mikä vähentää avaimen jakeluun liittyviä riskejä. JAK perustuu yksisuuntaisten funktioiden käyttöön; nämä matemaattiset funktiot ovat helppoja laskea, mutta niiden käänteisfunktion laskeminen on huomattavan vaikeaa (Rivest, Shamir ja Adleman 1978).

Yleinen salausmenetelmä julkisia avaimia varten on RSA jonka kehittivät Rivest, Shamir ja Adleman (Rivest, Shamir ja Adleman 1978). RSA-algoritmin perusta on kahden suuren satunnaisen alkuluvun käyttö avainten muodostamiseen. Näitä alkuperäisiä lukuja pidetään salassa, ja julkisessa avaimessa käytetään niiden tuloa. Perinteisillä tietokoneilla ei ole olemassa tehokasta ratkaisualgoritmia, joka voisi jakaa kokonaisluvun alkulukutekijöihinsä, mikä tekee RSA:sta turvallisen. Lisäksi RSA-menetelmää voidaan käyttää digitaalisessa allekirjoituksessa, mikä lisää sen monipuolisuutta ja käytettävyyttä eri sovelluksissa.

Vaikka asymmetrinen salakirjoitus, kuten julkisen avaimen kryptografia, tarjoaa korkeaa tietoturvaa, se voi aiheuttaa suuremman laskennallisen kuormituksen verrattuna symmetriseen

salakirjoitukseen (Maakar 2013). Tämä voi olla erityisen merkittävää, kun sitä käytetään laajasti suojattujen viestien vaihtoon, koska laskentatehtävät voivat olla resursseja kuluttavia. Käytännön toteutuksissa pyritään usein optimoimaan suorituskykyä, esimerkiksi käyttämällä hybridiratkaisuja, jotka yhdistävät symmetrisen ja asymmetrisen salakirjoituksen hyödyt. Tällaisissa ratkaisuissa julkisen avaimen kryptografiaa käytetään turvallisen salaisen avaimen vaihtoon, jonka jälkeen tietojen salaus ja purku tehdään tehokkaalla symmetrisellä salakirjoituksella. Tämä yhdistelmä mahdollistaa sekä vahvan tietoturvan että paremman suorituskyvyn (Schneier 1996).



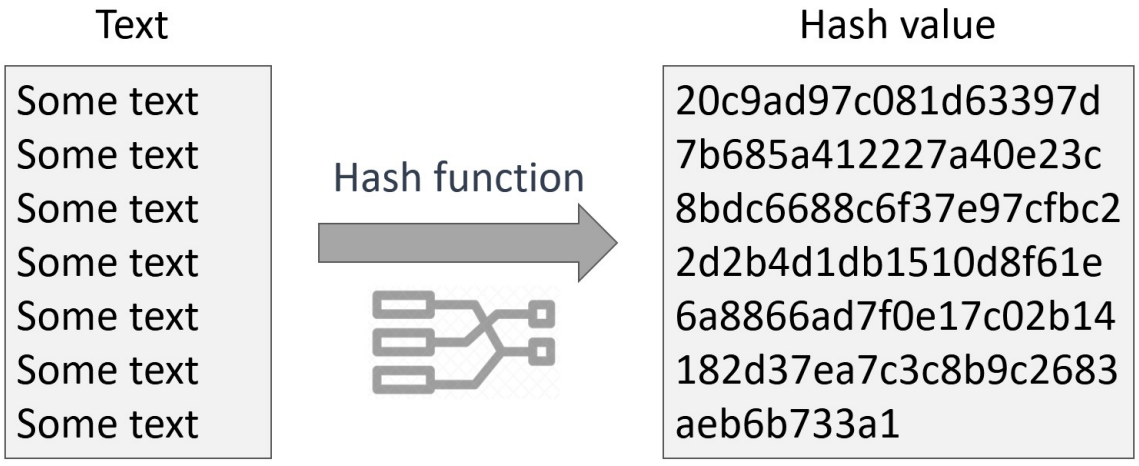
Kuvio 1: Erilaisia kryptografisia algoritmeja ja niiden kulkua

2.4 Hash-funktio

Hash-funktio on matemaattinen algoritmi, joka muuntaa syötteen kiinteän pituiseen tiivistearvoon, joka on uniikki syötteen perusteella. Yksisuuntaisen luonteensa vuoksi hash-funktiosta ei voi palauttaa alkuperäistä syötettä, mikä tekee siitä erinomaisen työkalun digitaalisten sormenjälkien luomiseen ja tietojen muutosten havaitsemiseen (Knuth 1997). Hash-funktioita

käytetään laajasti, esimerkiksi tiedostojen eheyden tarkistamisessa ja salasanojen suojaamisessa, joissa alkuperäiset salasanat hashataan ja tallennetaan tietokantaan niiden suojaksi. Yleisiä hash-funktioita ovat MD5, SHA-1 ja SHA-256, joista SHA-256 tarjoaa parasta turvallisuutta nykyaikaisissa sovelluksissa (National Institute of Standards and Technology 2012). Vaikka hash-funktiot tarjoavat tehokasta suojaa, niillä on myös rajoituksia, kuten mahdolliset kollisiot, joissa eri syötteet voivat tuottaa saman tiivistearvon. Näiden rajoitusten vuoksi on tärkeää käyttää vahvoja ja hyvin testattuja hash-funktioita, jotka tarjoavat riittävän turvallisuuden.

Hash-funktion suorituskyky on riippuvainen sen käytöstä (Gerard 1974). Esimerkiksi suurten tiedostojen tiivistäminen aiheuttaa enemmän laskennallista kuormitusta (Preneel 1994). Lisäksi, jos käytetään kryptografisia hash-funktioita, niiden tarkoituksena on olla laskennallisesti vaativia, mikä vaikuttaa suorituskykyyn, erityisesti suurilla datamäärillä (Han, Li ja Zeng 2012).



Kuvio 2: Esimerkki hash-funktion transformoinnista

3 Obfuskaation perusteet

Tässä luvussa selitetään perusteellisesti läpi obfuskaation perusteita ja keskeisiä käsitteitä.

3.1 Tausta

Obfuskaatio viittaa koodin tai datan muuntamiseen vaikeammin ymmärrettävään muotoon. Tämän tavoitteena on vaikeuttaa ohjelmiston analysointia ja ymmärtämistä, mikä tekee haitallisten hyökkääjien työn vaikeaksi (Barak ym. 2012). Obfuskaatio tarjoaa lisäkerroksen tietoturvaan, erityisesti tilanteissa, joissa on tärkeää suojata ohjelmakoodin rakennetta ja toimintaa. Tyypillisiä obfuskaatiomenetelmiä ovat koodin sekoittaminen, nimien muuttaminen ja ylimääräisen koodin lisääminen, jotka kaikki vaikeuttavat koodin ymmärtämistä ilman asianmukaista tulkintaa (Behera ja Bhaskari 2015).

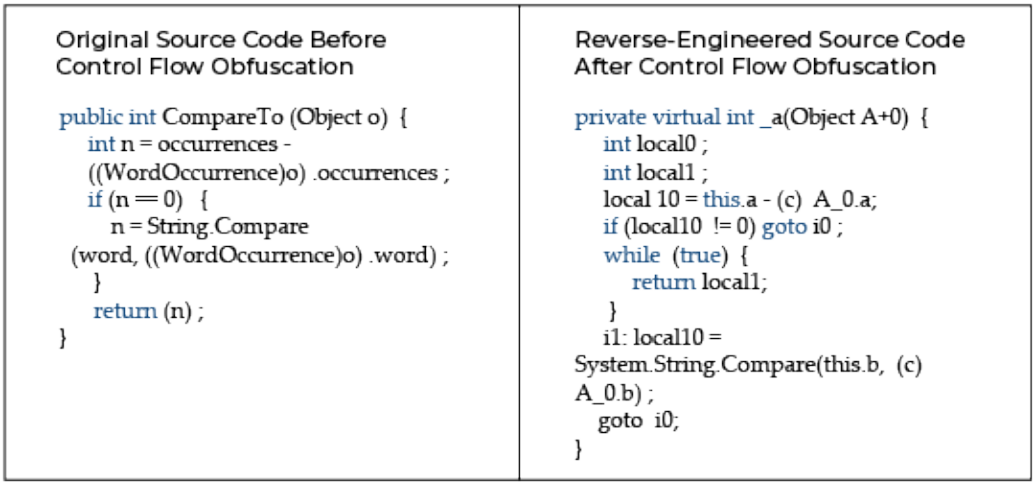
3.2 Merkkijonon salaus

Merkkijonon salaus on prosessi, jossa tekstimerkkijono muunnetaan koodatuksi tai salatuksi muodoksi tietoturvan parantamiseksi. Tässä tekniikassa alkuperäinen merkkijono, joka saattaa sisältää arkaluonteista tietoa kuten salasanoja tai muita luottamuksellisia tietoja, muutetaan muotoon, joka ei ole helposti luettavissa tai ymmärrettävissä ilman asianmukaista purkamismenetelmää tai salauksen avainta (Lynn, Prabhakaran ja Sahai 2004).

Merkkijonon obfuskaatiolla voi olla merkittävä vaikutus suorituskykyyn koska obfuskaatio lisää ylimääräistä koodia, joka täytyy suorittaa joka kerta, kun merkkijonoon viitataan. Tämä koodi on usein monimutkaista ja voi merkittävästi heikentää suorituskykyä. Tämä vaikutus on erityisen huomattava tilanteissa, joissa laskentatehoa vaativat ja usein toistuvat koodit tuottavat merkkijonoja. Näissä tapauksissa suorituskyvyn heikkeneminen voi olla voimakkaampaa, koska lisämenetelmien kutsuja tarvitaan obfuskoitujen merkkijonojen purkamiseen, mikä kuormittaa järjestelmän resursseja ja hidastaa kokonaissuoritusta.

3.3 Control-flow obfuskaatio

Control-flow obfuskaatio on tekniikka, jonka tarkoituksena on harhauttaa disassemblereita eli ohjelmistojen purkajia (Collberg, Thomborson ja Low 1997). Se lisää koodiin goto-lauseita, jotka lopulta suorittavat alkuperäisen ohjeiden sekvenssin, mutta kiertoteitse siten, että loogisen virtauksen seuraaminen vaikeutuu (Hohenberger ym. 2007). Tarkoituksena on tehdä alkuperäisen koodauksen palauttaminen niin vaativaksi, että hakkerit siirtyvät muihin, ehkä yksinkertaisempiin haasteisiin. Tämä erityinen tekniikka lisää pienen määrän koodia binääreihin ja aiheuttaa siten hieman ylimääräistä kuormitusta obfuskoituneille osille. (Collberg, Thomborson ja Low 1997)



Kuvio 3: Ennen ja jälkeen control-flow obfuskaation (Kuvan lähde: PreEmptive. Käytetään luvalla)

Control-flow obfuskaatio voi parantaa tietoturvaa vaikeuttamalla koodin purkamista. Kuitenkin negatiivisena puolena se lisää koodin monimutkaisuutta, mikä voi heijastua hienoisena suorituskyvyn laskuna (Balachandran ym. 2016). Lisäksi koodin optimointi voi tulla vaikeammaksi (Zhuang 2018).

3.4 Uudelleen nimeäminen

Uudelleennimeäminen on prosessi, jossa entiteettejä kuten luokkia, metodeita, kenttiä, annotaatioita ja paketteja nimetään uudelleen jotta bittikoodista tulee tiivimpää ja vaikeampaa lukea (Collberg, Thomborson ja Low 1997). Nämä entiteetit sisältävät runsaasti tietoa siitä, mitä ohjelmistosi yksittäiset komponentit tekevät. Nämä nimet säilytetään tuloksena syntyneessä bittikoodissa monista syistä. Ne ovat tärkeitä, jos haluat kutsua tietyn luokan päämetodia tai viitata metodiin toisessa projektissa (Shuaike Dong ym. 2018).

```
private void CalcPayroll(SpecialList employeeGroup) {
   while (employeeGroup.HasMore()) {
        employee = employeeGroup.GetNext(true);
        employee.UpdateSalary();
        DistributeCheck(employee);
    }
}
```

```
private void a(a b) {
    while (b.a()) {
        a = b.a(true);
        a.a();
        a(a);
    }
}
```

Kuvio 4: Ennen ja jälkeen uudelleen nimeämisen (PreEmptive 2022)

Uudelleennimeäminen parantaa tietoturvaa tehdessään koodista vaikeammin ymmärrettävän, mutta se lisää koodin monimutkaisuutta ja vaikeuttaa koodin ylläpitoa. Se vaikuttaa suorituskykyyn vähäisesti jos ollenkaan (Tang ym. 2009).

4 Päätelmät ja yhteenveto

Tässä tutkielmassa olemme tarkastelleet salakirjoituksen ja obfuskaation vaikutuksia koodin suorituskykyyn tietoturvan näkökulmasta. Käsiteltyämme näitä kaksi käytäntöä ja niiden roolia tietojen suojaamisessa sekä koodin piilottamisessa, voimme nyt tehdä johtopäätöksiä niiden vaikutuksesta koodin suorituskykyyn.

Salakirjoitus on välttämätön tietoturvatoimenpide, joka tuo mukanaan erilaisia vaikutuksia koodin suorituskykyyn ja sen osalta havaittiin, että vaikka se lisää merkittävää tietoturvaa, sillä on taipumus aiheuttaa pientä suorituskyvyn hidastumista (Kumar* ja Chaudhary 2016). Tämä turvallisuusmekanismi on suunniteltu suojaamaan herkkiä tietoja salaamalla ne, mutta samalla se voi tuoda mukanaan joitakin kompromisseja suorituskyvyn suhteen. Salakirjoituksen keskeinen piirre on laskennallinen kuormitus (Wright, Dave ja Zadok 2003. Salausprosessi vaatii resursseja, erityisesti kun käsitellään suuria datamääriä. Suorituskyky voi heikentyä merkittävästi, kun tietoa on salattava ja purettava säännöllisesti, erityisesti resurssivaativissa sovelluksissa, kuten suurissa tietokantajärjestelmissä. Salausprosessi vaatii huomattavia laskennallisia resursseja, mikä voi luoda pullonkauloja ja hidastaa järjestelmän yleistä suorituskykyä, erityisesti suuria tietomääriä käsiteltäessä. (Kumar* ja Chaudhary 2016). Salakirjoitus vaatii avaimia, ja avaimenhallinta voi itsessään aiheuttaa suorituskykyhaasteita. Avaimien generointi, jakaminen ja hallinta voivat lisätä laskennallista taakkaa. Erityisesti silloin, kun salausavaimet vaihdetaan säännöllisesti parantuneen tietoturvan vuoksi, avaimenhallinta voi tulla pullonkaulaksi suorituskyvylle. (Kumar* ja Chaudhary 2016). Salakirjoitus voi myös aiheuttaa haasteita koodin optimoinnille. Koska salattu data on tarkoituksella muutettu epäselväksi, perinteiset optimointimenetelmät eivät välttämättä toimi samalla tehokkuudella. Tämä vaikuttaa erityisesti algoritmien suorituskykyyn ja vaativiin laskentatehtäviin negatiivisesti (Kofahi, Al-Somani ja Al-Zamil 2003).

Koska obfuskointi perustuu koodin muokkaamiseen siten, että se muuttuu vaikeammin ymmärrettäväksi, se voi lisätä koodin monimutkaisuutta ja vaikuttaa haitallisesti suorituskykyyn. Erityisesti suurten ja monimutkaisten ohjelmistojen tapauksessa obfuskointi voi johtaa hieman hitaampaan suorituskykyyn (Zhuang 2018). Koodin obfuskointi voi vaikeuttaa sen kääntämistä ja optimointia, koska perinteiset kääntäjät olettavat selkeää ja ymmärrettävää

koodia, ja obfuskoidun koodin käsittely voi vaatia enemmän resursseja (Guelton ym. 2018). Tämä voi näkyä pidentyneinä kääntämisaikoina ja vähäisempinä optimointimahdollisuuksina.

Obfuskoidun koodin vaikeampi ymmärrettävyys saattaa hidastaa sen suorittamista, erityisesti laskentatehovetoisissa sovelluksissa (Sahin, Pollock ja Clause 2016). Lisäksi obfuskointi voi rajoittaa perinteisiä koodin optimointimenetelmiä, sillä monet optimointityökalut olettavat selkeää rakennetta ja ennakoitavissa olevia koodipolkuja. Obfuskoinnin käyttö voi vaikeuttaa näiden optimointityökalujen tehokasta toimintaa. Vaikka obfuskointi tuo mukanaan suorituskykyvaikutuksia, sen käyttö on perusteltua, kun korkea turvallisuustaso on välttämätöntä (Siyuan Dong ym. 2018).

Yhteenvetona voidaan todeta, että salakirjoitus ja varsinkin obfuskaatio vaikuttavat merkittävästi ohjelmien suorittamiseen riippuen kuinka paljon eri tekniikkoja käytetään. Salakirjoitus, vaikka parantaa merkittävästi tietoturvaa, saattaa tuoda mukanaan suorituskyvyn heikkenemistä laskennallisen kuormituksen vuoksi (Kumar* ja Chaudhary 2016). Obfuskaatio puolestaan voi monimutkaistaa koodin optimointia ja pidentää kääntämisaikoja, mikä vaikuttaa negatiivisesti ohjelman suorituskykyyn (Zhuang 2018). Näiden menetelmien käytössä on tärkeää löytää tasapaino turvallisuuden ja suorituskyvyn välillä. Kumpikin tekniikka tarjoaa arvokkaita suojausmekanismeja, mutta niiden tehokas käyttö edellyttää huolellista suunnittelua ja arviointia, jotta voidaan minimoida mahdolliset suorituskyvyn heikkenemiset samalla kun säilytetään korkeatasoinen tietoturva. Tulevaisuudessa olisi hyödyllistä kehittää uusia menetelmiä ja optimointitekniikoita, jotta ohjelmistot voivat toimia tehokkaasti ja turvallisesti.

Lähteet

Balachandran, Vivek, Sufatrio, Darell J.J. Tan ja Vrizlynn L.L. Thing. 2016. “Control flow obfuscation for Android applications”. *Computers Security* 61:72–93. ISSN: 0167-4048. https://doi.org/https://doi.org/10.1016/j.cose.2016.05.003. https://www.sciencedirect.com/science/article/pii/S0167404816300529.

Barak, Boaz. 2016. “Hopes, fears, and software obfuscation”. 59 (3): 88–96. https://doi.org/10.1145/2757276.

Barak, Boaz, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan ja Ke Yang. 2012. “On the (im)possibility of obfuscating programs”. *J. ACM* (New York, NY, USA) 59, numero 2 (toukokuu). ISSN: 0004-5411. https://doi.org/10.1145/2160158.2160159. https://doi.org/10.1145/2160158.2160159.

Behera, Chandan Kumar ja D. Lalitha Bhaskari. 2015. “Different Obfuscation Techniques for Code Protection”. Proceedings of the 4th International Conference on Eco-friendly Computing and Communication Systems, *Procedia Computer Science* 70:757–763. ISSN: 1877-0509. https://doi.org/https://doi.org/10.1016/j.procs.2015.10.114. https://www.sciencedirect.com/science/article/pii/S1877050915032780.

———. 2024. “Does Obfuscation Affect Code Performance?” Käyty 22 heinäkuu 2024, https://www.preemptive.com/blog/does-obfuscation-affect-code-performance/.

Bellare, Mihir ja Phillip Rogaway. 1995. “Optimal asymmetric encryption”. Teoksessa *Advances in Cryptology — EUROCRYPT’94,* toimittanut Alfredo De Santis, 92–111. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-540-44717-7.

Bellare, Mihir ja Bennet Yee. 2003. “Forward-Security in Private-Key Cryptography”. Teoksessa *Topics in Cryptology — CT-RSA 2003,* toimittanut Marc Joye, 1–18. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-540-36563-1.

Benjamin Lynn, Manoj Prabhakaran Amit Sahai. 2004. *Advances in Cryptology – EUROCRYPT 2004.* Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-24676-3_2.

Boicea, Alexandru, Florin Radulescu, Ciprian-Octavian Truica ja Cristina Costea. 2017. “Database Encryption Using Asymmetric Keys: A Case Study”. Teoksessa *2017 21st International Conference on Control Systems and Computer Science (CSCS),* 317–323. https://doi.org/10.1109/CSCS.2017.50.

Buchmann, Johannes A. kirjoittaja, Evangelos. kirjoittaja Karatsiolis ja Alexander. kirjoittaja Wiesmaier. 2013. *Introduction to Public Key Infrastructures.* 194. Berlin, Heidelberg: Springer Berlin Heidelberg. http://dx.doi.org/10.1007/978-3-642-40657-7.

Carlos Cid, Christian Rechberger. 2015. *Fast Software Encryption.* Springer Berlin, Heidelberg.

Chandra, Sourabh, Smita Paira, Sk Safikul Alam ja Goutam Sanyal. 2014. “A comparative survey of Symmetric and Asymmetric Key Cryptography”. Teoksessa *2014 International Conference on Electronics, Communication and Computational Engineering (ICECCE),* 83–93. https://doi.org/10.1109/ICECCE.2014.7086640.

Cimato, S., A. De Santis ja U. Ferraro Petrillo. 2005. “Overcoming the obfuscation of Java programs by identifier renaming”. *Journal of Systems and Software* 78 (1): 60–72. ISSN: 0164-1212. https://doi.org/https://doi.org/10.1016/j.jss.2004.11.019. https://www.sciencedirect.com/science/article/pii/S0164121204002286.

Collberg, Christian, Clark Thomborson ja Douglas Low. 1997. “A Taxonomy of Obfuscating Transformations”. *http://www.cs.auckland.ac.nz/staff-cgi-bin/mjd/csTRcgi.pl?serial* (tammikuu).

Damgård, Ivan Bjerre. 1990. “A Design Principle for Hash Functions”. Teoksessa *Advances in Cryptology — CRYPTO’ 89 Proceedings,* toimittanut Gilles Brassard, 416–427. New York, NY: Springer New York. ISBN: 978-0-387-34805-6.

Diffie, W. ja M. Hellman. 1976. “New directions in cryptography”. *IEEE Transactions on Information Theory* 22 (6): 644–654. https://doi.org/10.1109/TIT.1976.1055638.

Dong, Shuaike, Menghao Li, Wenrui Diao, Xiangyu Liu, Jian Liu, Zhou Li, Fenghao Xu, Kai Chen, XiaoFeng Wang ja Kehuan Zhang. 2018. “Understanding Android Obfuscation Techniques: A Large-Scale Investigation in the Wild”. Teoksessa *Security and Privacy in Communication Networks,* toimittanut Raheem Beyah, Bing Chang, Yingjiu Li ja Sencun Zhu, 172–192. Cham: Springer International Publishing. ISBN: 978-3-030-01701-9.

Dong, Siyuan, Setareh Rastegari, Jacob McConachie, Kai Zhang ja Guofei Yan. 2018. “Understanding android obfuscation techniques: a large-scale investigation in the wild”. Teoksessa *SecureComm 2018: International Conference on Security and Privacy in Communication Systems,* 254:172–192. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer, Cham. https://doi.org/10.1007/978-3-030-01701-9_10. https://link.springer.com/chapter/10.1007/978-3-030-01701-9_10.

Fussner, David. 2012. *The biblatex-chicago package: Style files for biblatex.* 30. heinäkuuta 2012. Viitattu 29. tammikuuta 2013. http://mirror.ctan.org/macros/latex/contrib/biblatex/doc/biblatex.pdf.

García, Daniel F. 2015. “Performance Evaluation of Advanced Encryption Standard Algorithm”. Teoksessa *2015 Second International Conference on Mathematics and Computers in Sciences and in Industry (MCSI),* 247–252. https://doi.org/10.1109/MCSI.2015.61.

Gauravaram, Praveen. 2012. “Security Analysis of salt||password Hashes”. Teoksessa *2012 International Conference on Advanced Computer Science Applications and Technologies (ACSAT),* 25–30. https://doi.org/10.1109/ACSAT.2012.49.

Gerard, Salton. 1974. *Information Storage and Retrieval Scientific Report.* Cornell Univ., Ithaca, N.Y. Dept. of Computer Science.

Guelton, Serge, Adrien Guinet, Pierrick Brunet, Juan Manuel Martinez, Fabien Dagnat ja Nicolas Szlifierski. 2018. “[Research Paper] Combining Obfuscation and Optimizations in the Real World”. Teoksessa *2018 IEEE 18th International Working Conference on Source Code Analysis and Manipulation (SCAM),* 24–33. https://doi.org/10.1109/SCAM.2018.00010.

Han, Demin, Renfa Li ja Juanli Zeng. 2012. “High-performance implementation of a new hash function on FPGA”. Teoksessa *2012 14th International Conference on Advanced Communication Technology (ICACT),* 217–220.

Hellman, M.E. 2002. “An overview of public key cryptography”. *IEEE Communications Magazine* 40 (5): 42–49. https://doi.org/10.1109/MCOM.2002.1006971.

Hellman, Martin E. 1978. *AN OVERVIEW OF PUBLIC KEY CRYPTOGRAPHY.* IEEE Communications Magazine.

Hofheinz, Dennis ja Eike Kiltz. 2008. “Programmable Hash Functions and Their Applications”. Teoksessa *Advances in Cryptology – CRYPTO 2008,* toimittanut David Wagner, 21–38. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-540-85174-5.

Hohenberger, Susan, Guy N. Rothblum, abhi shelat abhi ja Vinod Vaikuntanathan. 2007. “Securely Obfuscating Re-encryption”. Teoksessa *Theory of Cryptography,* toimittanut Salil P. Vadhan, 233–252. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-540-70936-7.

Kessler, Gary C. 2023. *The Handbook on Local Area Networksbook.* Auerbach.

Knuth, Donald E. 1997. *The Art of Computer Programming, Volume 3: Sorting and Searching.* Reading, MA: Addison-Wesley.

Kofahi, N.A., Turki Al-Somani ja Khalid Al-Zamil. 2003. “Performance evaluation of three encryption/decryption algorithms”. Teoksessa *2003 46th Midwest Symposium on Circuits and Systems,* nide 2, 790–793 Vol. 2. https://doi.org/10.1109/MWSCAS.2003.1562405.

Kumar*, Narander ja Priyanka Chaudhary. 2016. *Performance Evaluation of Encryption/-Decryption Mechanisms to Enhance Data Security.* Department of Computer Science, B. B. A. University (A Central University), Lucknow - 226025, Uttar Pradesh India;

Lamport, Leslie. 1994. *LaTeX: A Document Preparation System.* 2. painos. Reading, MA: Addison–Wesley.

Lehman, Philipp ym. 2012. *The biblatex Package: Programmable Bibliographies and Citations.* Elokuu. Viitattu 29. tammikuuta 2013. http://mirror.ctan.org/macros/latex/contrib/biblatex/doc/biblatex.pdf.

Lowry, Edward S. ja C. W. Medlock. 1969. “Object Code Optimization”. *Commun. ACM* (New York, NY, USA) 12, numero 1 (tammikuu): 13–22. ISSN: 0001-0782. https://doi.org/10.1145/362835.362838. https://doi.org/10.1145/362835.362838.

Lynn, Benjamin, Manoj Prabhakaran ja Amit Sahai. 2004. “Positive Results and Techniques for Obfuscation”. Teoksessa *Advances in Cryptology - EUROCRYPT 2004,* toimittanut Christian Cachin ja Jan L. Camenisch, 20–39. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-540-24676-3.

Maakar, Sunil. 2013. “A Performance Analysis of DES and RSA Cryptography” (marraskuu).

Mahajan, Prerna ja Abhishek Sachdeva. 2013. “A study of encryption algorithms AES, DES and RSA for security”. *Global Journal of Computer Science and Technology* 13 (15): 15–22.

Menezes, Alfred J., Paul C. van Oorschot ja Scott A. Vanstone. 1996. *Handbook of Applied Cryptography.* CRC Press.

National Institute of Standards and Technology. 2012. *Secure Hash Standard (SHS).* Tekninen raportti. NIST. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf.

Oram, Andy ja Greg Wilson, toimittaneet. 2011. *Making Software: What Really Works, and Why We Believe It.* O’Reilly.

Patashnik, Oren. 1988. BIBTEX*ing.* 8. helmikuuta 1988. Viitattu 29. tammikuuta 2013. http://mirror.ctan.org/biblio/bibtex/base/btxdoc.pdf.

Piper, Fred. 2002. “Cryptography”. Teoksessa *Encyclopedia of Software Engineering.* John Wiley Sons, Ltd. ISBN: 9780471028956. https://doi.org/https://doi.org/10.1002/0471028959.sof070. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/0471028959.sof070. https://onlinelibrary.wiley.com/doi/abs/10.1002/0471028959.sof070.

Prechelt, Lutz ja Marian Petre. 2011. “Credibility, or Why Should I Insist on Being Convinced”. Teoksessa Oram ja Wilson 2011, 17–34.

Preneel, Bart. 1994. “Cryptographic hash functions”. *European Transactions on Telecommunications* 5 (4): 431–448. https://doi.org/https://doi.org/10.1002/ett.4460050406. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/ett.4460050406. https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.4460050406.

Raigoza, Jaime ja Kapil Jituri. 2016. “Evaluating Performance of Symmetric Encryption Algorithms”. Teoksessa *2016 International Conference on Computational Science and Computational Intelligence (CSCI),* 1378–1379. https://doi.org/10.1109/CSCI.2016.0258.

Rajdeep Bhanot, Rahul Hans. 2015. *International Journal of Security and Its Applications.* 9. painos. IJSIA.

Rivest, R. L., A. Shamir ja L. Adleman. 1978. “A method for obtaining digital signatures and public-key cryptosystems”. *Commun. ACM* (New York, NY, USA) 21, numero 2 (helmikuu): 120–126. ISSN: 0001-0782. https://doi.org/10.1145/359340.359342. https://doi.org/10.1145/359340.359342.

Rodney Topor, Katsumi Tanaka. 1997. *Database Systems for Advanced Applications ’97: Proceedings of the Fifth International Conference on Database Systems for Advanced Applications.* World Scientific.

Sahin, Cansu, Lori Pollock ja James Clause. 2016. “How does code obfuscation impact energy usage?” *Journal of Software: Evolution and Process* 28 (7): 565–588. https://doi.org/10.1002/smr.1781. https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.1781.

Schneider, Johannes ja Thomas Locher. 2016. *Obfuscation using Encryption.* arXiv: 1612.03345 [cs.CR].

Schneier, Bruce. 1996. *Applied Cryptography: Protocols, Algorithms, and Source Code in C.* John Wiley & Sons.

Sebastian, Savio Antony, Saurabh Malgaonkar, Paulami Shah, Mudit Kapoor ja Tanay Parekhji. 2016. “A study review on code obfuscation”. Teoksessa *2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare (Startup Conclave),* 1–6. https://doi.org/10.1109/STARTUP.2016.7583913.

Shannon, Claude E. 1949. “Communication Theory of Secrecy Systems”. *Bell System Technical Journal* 28 (4): 656–715.

Simmons, Gustavus J. 1979. “Symmetric and Asymmetric Encryption”. *ACM Comput. Surv.* (New York, NY, USA) 11, numero 4 (joulukuu): 305–330. ISSN: 0360-0300. https://doi.org/10.1145/356789.356793. https://doi.org/10.1145/356789.356793.

Smart, Nigel P. 2016. *Cryptography Made Simple.* Springer Nature Switzerland AG 2016. https://doi-org.ezproxy.jyu.fi/10.1007/978-3-319-21936-3.

Tang, Zhangyong, Xiaojiang Chen, Dingyi Fang ja Feng Chen. 2009. “Research on Java Software Protection with the Obfuscation in Identifier Renaming”. Teoksessa *2009 Fourth International Conference on Innovative Computing, Information and Control (ICICIC),* 1067–1071. https://doi.org/10.1109/ICICIC.2009.312.

Tsai, M.H. 2006. “Three control flow obfuscation methods for Java software” [kielellä English]. *IEE Proceedings - Software* 153 (2): 80–86(6). ISSN: 1462-5970. https://digital-library.theiet.org/content/journals/10.1049/ip-sen_20050010.

Unicode Consortium. 2012. *The Unicode Standard, Version 6.2.0.* Viitattu 29. tammikuuta 2013. http://www.unicode.org/versions/Unicode6.2.0/.

Wright, C.P., J. Dave ja E. Zadok. 2003. “Cryptographic File Systems Performance: What You Don’ t Know Can Hurt You”. Teoksessa *Second IEEE International Security in Storage Workshop,* 47–47. https://doi.org/10.1109/SISW.2003.10005.

Zhuang, Yan. 2018. “The performance cost of software obfuscation for Android applications”. *Computers Security* 73:57–72. ISSN: 0167-4048. https://doi.org/https://doi.org/10.1016/j.cose.2017.10.004. https://www.sciencedirect.com/science/article/pii/S0167404817302092.