TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

SAKU TASANEN

SALASANATIIVISTEIDEN MURTAMINEN

Kandidaatintyö

Tarkastaja: Tiina Schafeitel-Tähtinen

Jätetty tarkastettavaksi 09. joulukuuta 2018

TIIVISTELMÄ

SAKU TASANEN: Salasanatiivisteiden murtaminen
Tampereen teknillinen yliopisto
Kandidaatintutkielma, 21 sivua
Joulukuu 2018
Tietotekniikan koulutusohjelma
Pääaine: Ohjelmistotekniikka
Tarkastajat: Tiina Schafeitel-Tähtinen

Avainsanat: salasanatiiviste, bcrypt, PBKDF2, SHA-256, sanakirjahyökkäys, väsytyshyökkäys, tietoturva

Salasanoja ei koskaan pitäisi säilyttää selvätekstisinä. Sen sijaan pitäisi tallentaa niiden tiivisteet, jotka on laskettu kryptograafisella tiivistefunktiolla. Vaikkakin funktiot ovat suunniteltu yksisuuntaisiksi, on silti mahdollista selvittää alkuperäinen syöte. Tässä kandidaatintyössä perehdytään eri menetelmiin miten salasanatiivisteitä voidaan murtaa, miten tehokkaita ne ovat ja miten näiltä voidaan suojautua.

Tutkittaviksi murtamismenetelmiksi valittiin erilaisia väsytys- ja sanakirjahyökkäyksiä. Vertailu suoritettiin hyökkäämällä generoituja tietokantoja vastaan eri menetelmillä, joiden suoritusaikoja ja tehokkuutta, eli kuinka paljon salasanoja saadaan murrettua, vertailtiin.

Vertailussa selvisi, että sanakirjahyökkäykset olivat paljon tehokkaampia ja nopeampia suorittaa, kuin väsytyshyökkäykset. Käytetyllä tiivistefunktiolla ja laitteistolla oli kuitenkin suuri merkitys. Lisäksi sanakirjahyökkäyksen teho riippui suurilta osin käytetyistä sanalistoista. Tutkituista tiivistefunktioista parhaimman suojan tarjosi bcrypt.

SISÄLLYSLUETTELO

1. JOHDANTO

Tiivistefunktioita on jo pitkään käytetty salasanojen suojaamisessa. Ne muuttavat salasanan muotoon, josta alkuperäistä salasanaa ei pystytä päättelemään. Salasanatietokannan vuotaessa kolmannelle osapuolelle saadut salasanat ovat käyttökelvottomia, koska ne eivät ole selkokielisiä. Salasanojen tiivistäminen on kuitenkin osoittautunut vaikeaksi ja erilaisia hyökkäyksiä käyttämällä saatetaan saadaa tiivistetyistä salasanoista selvitettyä alkuperäinen.

Tiivistefunktioita on paljon erilaisia ja haasteena on valita näistä oikea. Pelkästään vahva tiivistefunktio ei kuitenkaan riitä, sillä käyttäjät ovat taipuvaisia valitsemaan hyökkääjän kannalta heikkoja salasanoja. Nämä seikat johtavat pahimmillaan siihen, että vuotanut salasanatietokanta saadaan lähes kokonaan muutettua takaisin selkokieliseksi.

Tässä työssä vertaillaan erilaisia tapoja, joilla hyökkääjä voi murtaa salasanatiivisteitä ja tutkitaan minkälaisiin tilanteisiin kukin menetelmä sopii. Tavoitteena on myös luoda katsaus minkälaisia tiivistefunktioita kannattaa valita ja miten käyttäjät voivat parantaa omaa tietoturvaansa.

Luvussa 2 tutustutaan siihen, minkälaisia salasanoja käyttäjät valitsevat. Luvussa 3 esitellään erilaisia tiivistefunktioita. Luku 4 käsittelee menetelmiä, joilla tiivisteitä voidaan murtaa. Luvussa 5 esitellään metodit, joiden avulla murtamismenetelmiä vertaillaan. Vertailun tulokset esitetään luvussa 6, luvussa 7 analysoidaan ne ja tulosten yhteenveto esitetään luvussa 8.

2. SALASANAN VALITSEMINEN

Tässä luvussa tutustutaan siihen, miten käyttäjät valitsevat salasanoja. On yleisesti tiedossa, että käyttäjät valitsevat heikkoja ja helposti muistettavia salasanoja. Lisäksi käyttäjillä on paljon tilejä verkkopalveluissa, mikä johtaa helposti salasanojen uusiokäyttöön. [1] [2]

2.1 Salasanan muodostus

Verkkopalvelut antavat käyttäjän valita salasanan täysin vapaasti, tai sitten ne toteuttavat jonkinlaisen salasanapolitiikan. Salasanapolitiikat ovat käytännössä sääntöjä, jotka salasanojen pitää toteuttaa. Säännöt saattavat vaihdella huomattavasti palveluiden välillä. Yleisiä sääntöjä ovat esimerkiksi seuraavat: salasanan tulee olla vähintään 8 merkkiä, sisältää vähintään yksi iso kirjain, vähintään yksi numero ja vähintään yksi erikoismerkki. Salasanapolitiikkojen tehokkuudesta on kuitenkin kiistelty [3].

Salasanapolitiikan tarkoitus on kasvattaa salasanan entropiaa. Mitä suurempi mahdollisten salasanakombinaatioiden määrä on, sitä suurempi entropia salasanalla on. Esimerkiksi numeron lisääminen salasanaan kasvattaa sen entropiaa, koska mahdollisten symbolien määrä kasvaa. Entropia lasketaan kaavalla [4, s.47]

$$H = log_2 N^L \tag{2.1}$$

missä N on mahdollisten symbolien määrä, L salasanassa olevien merkkien määrä ja H salasanan entropia bitteinä. Käytännössä siis entropia kertoo kaikkien mahdollisten kombinaatioiden määrän (N^L) logaritmisella asteikolla.

Entropia ei kuitenkaan anna todellista kuvaa salasanan vahvuudesta. Mittarin heikkous on sen yksinkertaisuus sillä se ei tiedä mitään oikeista sanoista. Esimerkiksi salasanan *Kirjakauppa* entropia on 49.4, jonka pitäisi olla hyvä. Sana löytyy kuitenkin sanakirjasta, joten salasanan arvaaminen on helppoa iteroimalla sanakirja läpi. Salasanan *XyNmLkttOjb* entropia on sama, vaikkakin se on huomattavasti vaikeampi arvata. Edellä mainittujen syiden takia kryptografikot ovat kehittäneet toisenlaisia tapoja laskea entropia. Näitä ei kuitenkaan käsitellä tässä työssä. [4, s.46]

2.2 Salasanatottumukset

Viimeisten vuosien aikana on valitettavasti tapahtunut valtava määrä tietomurtoja. Satoja miljoonia käyttäjätunnuksia ja salasanoja on päätynyt vääriin käsiin. [5] Tämä on mahdollistanut ihmisten salasanakäyttäytymisen arvioinnin. Tässä luvussa perehdytään muutamaan salasanatutkimukseen.

Tutkimuksessa [2] tutkittiin kolmea salasanakokoelmaa, jotka nimettiin seuraavasti: italialainen kokoelma, Microsoftin kokoelma ja suomalainen kokoelma. Yhteensä salasanoja oli hieman alle 60 000. Yksi kokoelma oli saatu koottua suomalaisille foorumeille tehdyistä tietomurroista.

Italialaisessa ja suomalaisessa kokoelmassa yli 50 % salasanoista koostui pelkästään pienistä alkukirjaimista. Microsoft-kokoelmassa 77 % salasanoista koostui pienistä alkukirjaimista, joiden perään oli laitettu jokin numerosarja. Valitettavasti suurimpaan osaan salasanoista oli laitettu vain numero 1 perään. Erot salasanojen valintaperusteissa johtuvat käytössä olevasta salasanapolitiikasta. [2, s.3]

Samanlaisia havaintoja tehtiin vielä laajemmassa tutkimuksessa [1], joka kattoi yli 500 000 käyttäjää. Pelkästään pieniä alkukirjaimia sisältäviä salasanoja käytettiin paljon, jos salasanapolitiikkaa ei ollut asetettu. Etenkin lyhyissä salasanoissa suosittiin pelkästään pieniä alkukirjaimia. Jos salasanapolitiikka vaati numeron salasanaan, se yleensä lisättiin sanan loppuun. Käyttäjien valitsemien salasanojen entropia oli keskimäärin 40.54.

Tutkimuksessa [6, s.10] analysoitiin kolme tietomurroissa paljastunutta salasanatietokantaa. Tietomurtojen kohteina olivat vuonna 2009 joulukuussa *rockyou.com*, vuonna 2009 helmikuussa *phpbb.com* ja vuonna 2012 kesäkuussa *linkedin.com*. Paljastuneita salasanoja oli yhteensä melkein 40 miljoonaa. Kaikissa tietokannoissa yleisin salasanan pituus oli 8 merkkiä. Yli 60 %:ssa salasanoista oli pelkkiä pieniä alkukirjaimia tai niiden perään oli lisätty numero.

Tutkimustuloksista kävi ilmi, että käyttäjät valitsevat liian lyhyitä salasanoja suhteessa hyökkääjän käytössä oleviin resursseihin. Käyttäjien valitsemissa salasanoissa pitäisi olla kattavampi yhdistelmä pieniä ja isoja alkukirjaimia, numeroita ja erikoismerkkejä. Lisäksi verkkopalvelujen sallima salasanan minimipituus on aivan liian lyhyt. [6, s.20]

3. TIIVISTEFUNKTIOT

Tiivistefunktiolla (hash function) tarkoitetaan funktiota, joka ottaa parametrinaan lähes mielivaltaisen pituisen viestin ja muodostaa siitä kiinteän mittaisen bittijonon, jota kutsutaan tiivisteeksi (hash). Tiivistefunktioita on olemassa paljon erilaisilla ominaisuuksilla, ja niitä hyödynnetään paljon kryptografiassa. [7, s.136]

Tiivistefunktiolla pitää olla tietyt ominaisuudet, jotta voidaan sanoa sen olevan kryptografisesti turvallinen. Ensimmäinen ominaisuus, joka turvallisella tiivistefunktiolla pitää olla, on törmäysten sietokyky (collision resistance). Tällä tarkoitetaan sitä, että pitäisi olla laskennallisesti lähes mahdotonta löytää kaksi eri syötettä, joiden tiivisteet ovat samoja. [8, s.6]

Tiivistefunktion pitää myös olla sellainen, että saadusta tiivisteestä ei voida päätellä alkuperäistä syötettä. Englanniksi tätä ominaisuutta kutsutaan nimellä *preimage resistance*. Lisäksi jos syöte ja tiiviste tiedetään, tulisi olla vaikea löytää toinen syöte, jolla on sama tiiviste (second preimage resistance). [8, s.7]

3.1 SHA-tiivistefunktiot

SHA (Secure Hash Algorithms) tiivistefunktiot on määritelty NIST:n julkaisemassa standardissa *Secure Hash Standard*. Algoritmeja voidaan käyttää viestin eheyden tarkistamiseen, sillä pienikin muutos alkuperäisessä viestissä tuottaa suurella todennäköisyydellä täysin erilaisen tiivisteen. Tätä ominaisuutta hyödynnetään muun muassa digitaalisessa allekirjoittamisessa, autentikointikoodeissa ja satunnaisnumeroiden generoinnissa. [9, s.3] Alla olevaan taulukkoon 3.1 on listattu standardin määrittelemät algoritmit.

Taulukko 3.1 *Standardin määrittelemät tiivistefunktiot [9].*

Algoritmi	Viestin koko (bit)	tiivisteen koko (bit)
SHA-1	$< 2^{64}$	160
SHA-224	$< 2^{64}$	224
SHA-256	$< 2^{64}$	256
SHA-384	$< 2^{128}$	384
SHA-512	$< 2^{128}$	512
SHA-512/224	$< 2^{128}$	224
SHA-512/256	$< 2^{128}$	256

Algoritmit eroavat toisistaan vahvuudeltaan ja tiivisteen koolta. Mitä uudempi algoritmi on kyseessä sekä mitä enemmän bittejä saadussa tiivisteessä on, sitä paremmin se kestää törmäyksiä ja muita hyökkäyksiä. [8, s.8]

Kaikista SHA-algoritmeista on olemassa HMAC (Hash Message Authentication Code) versio. Näissä funktioissa viestin lisäksi parametrina annetaan salaisuus. Tämän ominaisuuden ansioista viestin eheyden lisäksi voidaan todentaa myös viestin alkuperä. Vain salaisuuden tietävät osapuolet voivat luoda valideja viestejä. Funktiot nimetään lisäämällä käytetyn tiivistefunktion etuliitteeksi HMAC esim. HMAC-SHA-256. [10, s.117]

Kaikki SHA-algoritmit on suunniteltu nopeiksi eikä siten sovellu sellaisenaan salasanojen suojaamiseen. Esimerkiksi pieni neljän näytönohjaimen GPU-klusteri pystyy laskemaan SHA-512 agoritmin 1120 miljoonaa kertaa sekunnissa. Tällä nopeudella 8 merkkinen salasana saadaan murrettua alle 70 tunnissa. [11]

3.2 PBKDF2

PBKDF2 (Password-Based Key Derivation Function 2) ei varsinaisesti ole tiivistefunktio vaan avaimen johtamisfunktio (key derivation function). Se kuitenkin palauttaa suolatun tiivisteen, joten se sopii myös salasanojen tiivistämisfunktioksi. Funktion toiminta määritellään vuonna 2000 julkaistussa RFC 2898 -standardissa. Sitä käytetään muun muassa WPA-2 -salausprotokollassa ja TrueCrypt-ohjelmistossa. [10, s.106] Funktion formaatti on seuraava

JA = PBKDF2(Salasana, Suola, Iteraatiot, JApituus)

missä JA on johdettu avain (derived key), Iteraatiot kertovat iteraatioiden määrän ja JA-pituus määrittelee ulostulevan avaimen (JA) pituuden. [10, s.106]. Suola on merkkijono, jonka tarkoitus on estää sateenkaaritaulukoiden käyttö. Suolaamisesta kerrotaan lisää luvussa 4.4.

PBKDF2:ssa avain johdetaan ottamalla esimerkiksi jokin HMAC-algoritmi ja iteroimalla sitä useita kertoja. [12] Funktion vaatima laskenta-aika riippuu siis käytetystä algoritmista ja iteraatioiden määrästä. Iteraatioiden määrää kasvattamalla voidaan pitää laskenta-aika vakiona tekniikan kehittyessä. Toinen tärkeä huomio on se, että PBKDF2 on nopea laskea GPU:lla jos algoritmiksi on valittu jokin SHA-perheen HMAC algoritmi.

3.3 Bcrypt

Bcrypt on tiivistefunktio, joka on erityisesti suunniteltu salasanojen tiivistämiseen. Sen toiminta perustuu Blowfish-salakirjoitusjärjestelmään. Algoritmin on suunnitellut Niels Provos ja David Mazières ja se esiteltiin USENIX konferenssissa vuonna 1999. [13]

Tiivistefunktio saa syötteenä kierrosten määrän ja itse salasanan. Kierrosten määrä voidaan nähdä kustannuskertoimena, jolla voidaan nostaa tarvittavaa laskenta-aikaa. Jokaisen kierroksen lisäys tuplaa käytetyn ajan. Funktion paluuarvo on rakenteeltaan seuraava: etuliite $2a$, kierrosten määrä, käytetty suola ja itse tiiviste. [10]

Kuva 3.1 *Bcrypt tiivisteen rakenne. Perustuu lähteeseen [10].*

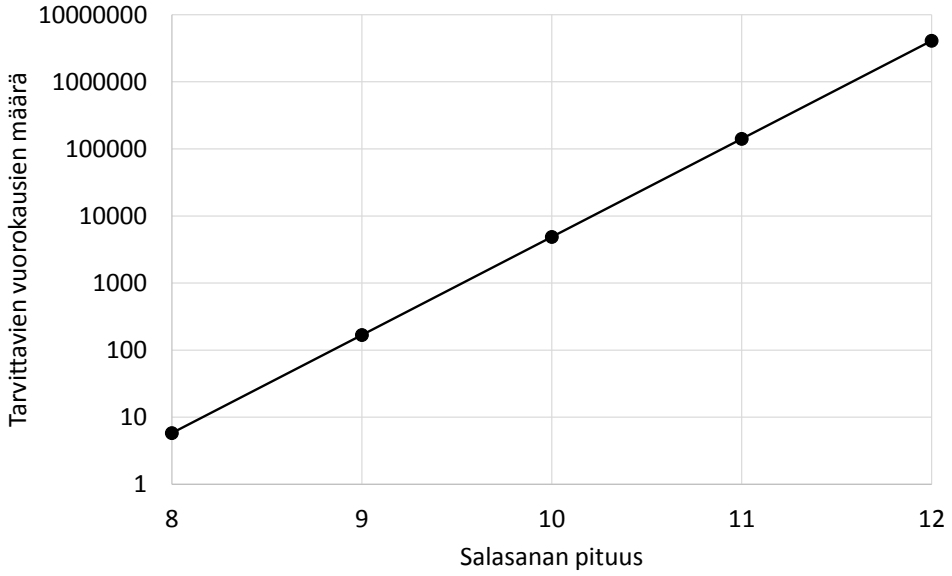
Myös bcryptin vahvuutena on se, että vaikka laskentateho kasvaa tekniikan kehittyessä, algoritmin teho ei heikkene. Aina kun laskentatehossa nähdään harppaus kustannuskerrointa kasvatetaan. Toinen bcryptin vahvuus on sen sietokyky GPU-pohjaista laskentaa vastaan. Algoritmi soveltuu heikosti GPU-klusterin laskettavaksi. [11]

4. MURTAMISMENETELMÄT

Vaikkakin tiivistefunktiot ovat suunniteltu yksisuuntaisiksi, on silti mahdollista löytää alkuperäinen syöte arvaamalla. Hyökkäyksen tehokkuus riippuu arvausten laadusta ja saatavilla olevasta laskentatehosta. Tässä luvussa esitellään eri tapoja murtaa tiiviste eli löytää alkuperäinen syöte.

4.1 Väsytyshyökkäys

Väsytyshyökkäyksessä (raakahyökkäys tai brute-force -hyökkäys) kokeillaan järjestelmällisesti kaikki mahdolliset salasanakombinaatiot läpi [10, s.131]. Hyökkääjä syöttää jokaisen salasanakombinaation tiivistefunktion läpi ja vertaa saatua tulosta tiivisteeseen, joka halutaan murtaa. Hyökkäyksen etuna on se, että sillä on mahdollista murtaa minkälainen salasana tahansa. Ongelmana on kuitenkin se, että mahdollisten salasanakombinaatioiden määrä nousee eksponentiaalisesti salasanan pituuden kasvaessa (kuva 4.1). Kombinaatioita saattaa olla niin paljon, että käytettävissä oleva aika ei riitä salasanan murtamiseen.



Kuva 4.1 *Murtamisaika salasanan pituuden funktiona (miljoona arvausta sekunnissa).*

Tarvittavien salasanakombinaatioiden määrää voidaan pienentää olettamalla, ettei käyttäjän salasanassa ole esimerkiksi isoja alkukirjaimia tai numeroita. Kuten kuvasta 4.1 nähdään, hyökkäys toimii vain salasanan ollessa riittävän lyhyt. Lisäksi hyökkäys vaatii riittävän määrän arvauksia sekunnissa ollakseen tehokas.

4.2 Sanakirjahyökkäys

Käyttäjät eivät yleensä valitse salasanoiksi satunnaisia merkkijonoja, vaan esimerkiksi jonkun sanan. Sanakirjahyökkäyksessä hyödynnetään tätä tietoa, ja käydään läpi järjestelmällisesti sanalistoja laskien sanoista tiivisteet. Sanalistojen ei ole pakko olla sanakirjasta otettuja sanoja, vaan voidaan hyödyntää myös listoja yleisimmin käytetyistä salasanoista.

Menetelmän etu verrattuna väsytyshyökkäykseen on se, että arvauksia tarvitsee tehdä huomattavasti vähemmän. Suomen kielen sanakirjasta löytyy yli 90 000 eri sanaa [14]. Arvauksia ei siis tarvita miljoonia tai miljardeja vaan huomattavasti pienempi määrä riittää. Lisäksi on mahdollista muodostaa huomattavasti tarkemmin rajattuja sanalistoja analysoimalla sitä kuinka käytetty sana on. Kombinaatioiden määrä kuitenkin kasvaa jos mukaan otetaan ulkomaankielisiä sanoja.

4.3 Sovellettu hyökkäys

Pelkästään sanojen läpikäynti ei aina riitä, vaan käyttäjä on saattanut valita salasanaksi jonkin tavallisen sanan variaation. Käyttäjä on saattanut lisätä esimerkiksi numeron tai kirjaimen sanan loppuun.

Hyökkääjä voi nyt hyödyntää sekä sanakirjahyökkäystä, että väsytyshyökkäystä. Hyökkääjä kokeilee systemaattisesti sanakirjan sanat läpi, ja arpoo jokaisen sanan loppuun n-määrän merkkejä. Tämä yhdistelmähyökkäys on erityisen tehokas, jos oletetaan, että käyttäjä on lisännyt vain numeron sanan perään. Tarvittavien kombinaatioiden määrä on tällöin sanakirjan koko kertaa kymmenen tai yhdeksän, jos nollaa ei oteta huomioon. Hyökkäystä voi tehostaa lisäämällä sanojen loppuun vain tunnettuja numerokombinaatioita kuten vuosilukuja tai toistamalla samaa numeroa. [15]

Kehittyneempi menetelmä on luoda lista sääntöjä, joiden perusteella luodaan sanasta variaatio. Salasanan murtamisohjelmat tukevat suuren määrän erilaisia sääntöjä ja niiden avulla on mahdollista toteuttaa sanasta lähes minkälainen variaatio tahansa. Esimerkki sanavariaatioita ovat: iso alkukirjain, kaikki kirjaimet isolla, jokainen kirjain kahteen kertaan, sana kahteen kertaan tai sana takaperin. [16]

Mahdollisia variaatioita on kuitenkin niin paljon, että kaikkia ei ehdi käymään läpi. Hyökkääjällä pitää olla käsitys minkälaisia salasanoja käyttäjät valitsevat. Salasanatottumuksia voi tutkia analysoimalla tietomurtoja samaan tapaan kuin luvussa 2.2 kerättiin taulukko yleisistä salasanatyypeistä. [16]

Hyökkääjä voi myös yhdistellä sanoja ja käydä läpi esimerkiksi kaikki kahden sanan kombinaatiot [17]. Tarvittavien arvausten määrä kuitenkin kasvaa eksponentiaalisesti, kun perättäisiä sanoja tulee lisää. Jos sanakirjan koko on 10 000, kahden sanan kombinaatioita on sata miljoonaa ja kolmen sanan kombinaatioita biljoona. Hyökkääjä tarvitsee siis erittäin paljon laskentatehoa tai hyvin tarkasti rajatun sanakirjan, jotta tämä hyökkäys on mahdollista toteuttaa järkevässä ajassa.

4.4 Sateenkaaritaulukot

Salasanasta laskettu tiiviste ei koskaan muutu. Joten riittää, että vain kerran lasketaan kaikista sanakirjan sanoista tiivisteet tai käydään väsytyshyökkäyksen avulla kaikki tarvittavat kombinaatiot läpi. Seuraavan kerran kun tuntematon tiiviste pitää murtaa, voidaan sitä verrata ennalta laskettuihin tiivisteisiin, joiden selväkielinen vastine tiedetään. Tällä tavoin voidaan hyvin nopeasti murtaa isoja salasanatietokantoja. Listoja tai tietokantoja, jotka sisältää ennalta laskettuja tiivisteitä sanotaan sateenkaaritaulukoiksi (rainbow table). [10, s.96]

Menetelmä on hyvin tehokas, joten siltä suojautuminen on tärkeää. Ennen tiivistefunktion suorittamista salasanaan lisätään pitkä satunnainen merkkijono, jota sanotaan suolaksi. On myös olemassa tiivistefunktioita, jotka ottavat parametrina suolan ja näiden käyttö on suositeltavaa. [10, s.96]

Suolaamisen teho perustuu siihen, että sateenkaaritaulukot ovat generoitu ilman sitä. Niissä olevat tiivisteet ovat täysin erilaisia, kuin ne missä on suola mukana. Suolan ollessa riittävän pitkä on ajallisesti mahdotonta generoida kaikkia eri suolakombinaatioita vastaavat sateenkaaritaulukot. Teho siis kasvaa, kun jokaiselle salasanalle generoidaan uniikki suola. Tämän jälkeen hyökkääjä joutuu yksitellen murtamaan tiivisteet eikä kokonaisen tietokannan murtaminen kerralla ole enää mahdollista.

Suolaaminen ei kuitenkaan vaikuta murtamisnopeuteen yhden tiivisteen tapauksessa. Tiiviste on lähes aina hyökkääjällä tiedossa, sitä ei tarvitse arvata. Suola on tallennettu salasanatietokantaan tai se on sisällytetty tiivisteeseen (esim. bcrypt). [10, s.97]

5. MURTAMISMENETELMIEN VERTAILU

Tässä työssä vertaillaan eri murtamismenetelmiä. Tavoitteena on tutkia mitkä menetelmät ovat mahdollista suorittaa järkevässä ajassa, minkälaisia salasanatyyppejä tai tiivisteitä vastaan hyökkäys toimii, ja mikä menetelmä vaikuttaa tehokkaimmalta. Vertailussa hyökätään kolmea itse generoitua salasanatietokantaa vastaan.

Generoidut tietokannat ovat kooltaan 10 000 salasanaa. Tiivistetyt salasanat ovat kaikissa tietokannoissa samat ja ne ovat generoitu sanalistan [14] pohjalta. Sanalistan koko oli 94 110 sanaa. Generointi tapahtui ottamalla listasta 10 000 satunnaista sanaa, suorittaen niille tarvittaessa muunnoksia. Puolet (50 %) sanoista otettiin sellaisenaan, 30 %:n perään lisättiin satunnainen numero 0-9 välillä ja lopulle 20 %:lle numeron lisäämisen lisäksi ensimmäinen kirjain muutettiin isoksi alkukirjaimeksi. Näillä salasanatyypeillä ja osuuksilla pyrittiin matkimaan oikeita tietokantoja, joita esiteltiin luvussa 2.2.

Kukin tietokanta suojattiin eri tiivistefunktiolla: SHA-256, PBKDF2-SHA256 ja bcrypt. Valintaperusteena käytettiin algoritmien erilaisuutta. SHA-256 on tunnetusti hyvin heikko salasanojen suojaamisessa, kun taas bcrypt ja PBKDF2 edustavat vahvempia algoritmeja. Bcryptin kustannuskertoimeksi valittiin 10, koska tämä on oletusarvo joissain kirjastoissa. PBKDF2:n iteraatioiden määräksi valittiin 280 000. Tällä iteraatiomäärällä laskentanopeus CPU:lla on lähes sama kuin valitussa bcryptissä.

Kaikki tulokset ovat laskettu kirjoittajan tietokoneella, jossa prosessorina on neliytiminen i7-4790K 4.0GHz ja GTX 970 näytönohjain. Hyökkäykset suoritettiin salasanan palautustyökalulla Hashcat ja käytössä ollut versio oli kirjoittamishetkellä uusin 5.0.

Tietokantoja vastaan suoritetaan väsytyshyökkäyksiä sekä sovellettuja sanakirjahyökkäyksiä. Väsytyshyökkäystä käytetään vain heikointa, SHA-256 suojattua tietokantaa vasten. Hyökkäyksestä käytetään kahta variaatiota. Ensimmäisessä oletetaan, että salasana on maksimissaan 8 merkkiä pitkä ja koostuu pelkästään pienistä alkukirjaimista. Toisessa variaatiossa kasvatetaan arvausten määrää olettamalla, että salasanan ensimmäinen kirjain on iso tai pieni ja viimeisenä on numero 0-9. Nämä hyökkäykset voidaan suorittaa Hashcatin komennoilla:

```
// Variaatio 1: Käy läpi kaikki 1-8 merkkiset salasanat
// 'alg' on ympäristömuuttuja joka määrittää tiivistefunktion:
// SHA256=1400, PBKDF2-HMAC-SHA256=10900, bcrypt=3200
# hashcat -a 3 -m $alg tiivisteet.txt ?l?l?l?l?l?l?l?l --increment


// Variaatio 2: Käy läpi kaikki 8 merkkiset salasanat, joissa
// ensimmäinen kirjain on iso tai pieni ja perään on lisätty numero
# hashcat -a 3 -m $alg tiivisteet.txt -2 ?l?u ?2?l?l?l?l?l?l?d
```

Sanakirjahyökkäyksestä suoritetaan neljä eri variaatiota. Ensimmäisessä hyökkäyksessä sanakirja iteroidaan sellaisenaan, ilman muutoksia sanoihin. Toisessa hyökkäyksessä sanakirjan sanoja muutetaan lisäämällä numero väliltä 0-9 sanan loppuun. Kolmas hyökkäys on lähes samanlainen, mutta numeron lisäksi ensimmäinen kirjain muutetaan isoksi alkukirjaimeksi. Viimeisessä hyökkäyksessä kokeillaan kaikki mahdolliset yhdyssanakombinaatiot. Hyökkäykset voidaan suorittaa Hashcatin komennoilla:

```
// Variaatio 3: Käydään läpi kaikki sanakirjan sanat
# hashcat -a 0 -m $alg tiivisteet.txt sanakirja.txt


// Variaatio 4: Käydään läpi kaikki 'sana+numero' kombinaatiot
# hashcat -a 6 -m $alg tiivisteet.txt sanakirja.txt ?d


// Variaatio 5: Käydään läpi kaikki 'Sana+numero' kombinaatiot
# hashcat -a 6 -m $alg tiivisteet.txt sanakirja1.txt ?d --rule-left="c"


// Variaatio 6: Käydään läpi kaikki 'sana+sana' kombinaatiot
# hashcat -a 1 -m $alg tiivisteet.txt sanakirja.txt sanakirja.txt
```

Hyökkäysmenetelmät ovat pyritty valitsemaan siten, että niiden avulla pystytään mahdolisimman tehokkaasti hyökkäämään kyseessä olevia tietokantoja vastaan. Tavoitteena on, että kombinaatioiden määrä ei kasva liian suureksi ja siten mahdottomaksi suorittaa.

6. TULOKSET

Tässä luvussa esitetään vertailussa saadut tulokset. Kutakin tietokantaa vastaan suoritettujen hyökkäysten tulokset esitetään omissa luvuissaan. Taulukkoon 6.1 on koottu tiivistelmä vertailussa käytetyistä hyökkäysvariaatioista. Tekstissä näihin hyökkäysvaraatioihin viitataan variaation numerolla, joka vastaa taulukossa esitettyä hyökkäysmenetelmää.

Taulukko 6.1 *Vertailuun valitut hyökkäysvariaatiot.*

Variaatio	Tyyppi	Kuvaus
1	Väsytyshyökkäys	Käydään läpi kaikki 1-8 merkin jonot.
2	Väsytyshyökkäys	Käydään läpi kaikki 8 merkin jonot, joissa ensimmäinen kirjain on iso tai pieni ja perään lisätty numero.
3	Suora sanakirjahyökkäys	Käydään läpi kaikki sanakirjan sanat sellaisenaan.
4	Sovellettu sanakirjahyökkäys	Käydään läpi kaikki sanakirjan sanat, lisäten perään numero väliltä 0-9.
5	Sovellettu sanakirjahyökkäys	Käydään läpi kaikki sanakirjan sanat, muutetaan ensimmäinen kirjain isoksi ja lisätään perään numero väliltä 0-9.
6	Kombinaatiohyökkäys	Käydään läpi kaikki kahden sanakirjan muodostamat yhdyssanakombinaatiot

Oletuksena on, että väsytyshyökkäys on huomattavasti hitaampi kuin mikään sanakirjahyökkäys. Poikkeuksena on kombinaatiohyökkäys, jonka laskenta-aika voi olla hyvinkin suuri isojen sanalistojen kanssa. Odotettavissa on myös se, että laskentanopeudet riippuvat suuresti käytetystä tiivistefunktiosta.

6.1 Tietokanta 1

Ensimmäistä tietokantaa vastaan sopii hyvin väsytyshyökkäys, koska yrityksiä voidaan tehdä valtava määrä. Myös kaikki neljä sanakirjahyökkäystä osoittautuivat erittäin tehokkaiksi ja kaikki mahdolliset kombinaatiot pystyttiin käymään läpi.

Ensimmäisenä suoritetaan hyökkäyksistä variaatio 1. Hyökkäyksen suorittaminen kesti 3 minuuttia 49 sekuntia ja suurin saavutettu tiivisteiden laskentanopeus oli 892.6 MH/s (miljoonaa tiivistettä sekunissa). Laskentanopeus vaihteli hyökkäyksen aikana suuresti, koska kombinaatioiden määrä ei ole kovin suuri suhteessa laskentatehoon esimerkiksi 5 merkin salasanoissa. Näytönohjaimella ei siis ole tarpeeksi työtä, eikä siten pysty maksimilaskentanopeuteen. 7-8 merkin salasanoissa kombinaatioita on jo niin paljon, että GPU toimi täydellä teholla. Hyökkäyksessä saatiin murrettua 649 tiivistettä.

Variaatiossa 2 ensimmäistä hyökkäystä laajennettiin siten, ensimmäinen kirjain voi olla iso tai pieni ja perään lisättiin numero. Hyökkäyksen suorittaminen kesti 2 minuuttia ja 56 sekuntia, jonka jälkeen murrettuja tiivisteitä oli yhteensä 1246. Laskentanopeus oli lähes sama kuin ensimmäisessä hyökkäyksessä 883.4 MH/s. Eri salasanakombinaatioita käytiin läpi 160 miljardia.

Tietokantaa vastaan suoritettiin vielä neljä sanakirjahyökkäystä. Ensimmäisenä variaatio 3, joka oli suora sanakirjahyökkäys. Tämä kasvatti murettujen tiivisteiden yhteismäärän 5263. Hyökkäys suoritettiin alle sekunnissa ja laskentanopeus oli 92540.0 kH/s.

Hyökkäysvariaatioiden 4 ja 5 avulla saatiin murrettua lähes kaikki loput tiivisteet. Variaation 4 jälkeen murrettuja tiivisteitä oli 8105 ja viimeisen hyökkäyksen jälkeen 9944/10000. Hashcat ilmoitti 4 variaatiossa laskentanopeudeksi 10008.6 kH/s ja 5 variaatiossa 14786.3 kH/s.

Vaikka lähes kaikki tiivisteet oltiin jo saatu murettua, suoritettiin vielä kombinaatiohyökkäys eli variaatio 6. Tarkoituksena oli testata kuinka nopeasti hyökkäys saadaan suoritettua. Suoritus kesti vain 10 sekuntia ja nopeudeksi saatiin 835,7 MH/s.

6.2 Tietokanta 2

Toinen tietokanta koostui PBKDF2-tiivisteistä, joiden kustannuskerroin oli 280 000. Hyökkäyksistä suoritettiin variaatiot 3, 4 ja 5. Väsytyshyökkäystä ei edes yritetty PBKDF2:n vaatimasta laskentatehosta johtuen. Esimerkiksi 8 merkin salasanan (pelkästään pieniä kirjaimia) murtaminen kestäisi yli 4 vuotta nopeudella 1500 H/s. Lisäksi tietokanta käyttää suolausta, joten kohdetiivisteet jouduttiin valitsemaan sattumanvaraisesti ja vain yhtä

tiivistettä murrettiin kerrallaan. Hyökkäysvariaatiota 6 ei yritetty johtuen suuresta kombinaatiomäärästä.

Laitteiston laskentatehon testauksessa näytönohjain saavutti nopeuden 1777 H/s ja prosessori 125 H/s. Odotusten mukaisesti näytönohjain voitti selvästi prosessorin. Tämä oli nähtävissä jo ensimmäistä tietokantaa vastaan suoritetuissa hyökkäyksissä, koska käytetty PBKDF2-tiivistefunktio käyttää sisäisesti HMAC-SHA256:ta.

Tällä laskentanopeudella variaatio 3:n suorittaminen oli helppoa. Hyökkäyksen suorittaminen kesti 53 sekuntia. Satunnaisesti valittua tiivistettä ei vielä tällä hyökkäyksellä saatu murrettua. Tuhannen tiivisten läpikäynti kestäisi hieman alta 15 tuntia ja koko tietokannan läpikäynti veisi aikaa hieman yli 6 päivää.

Variaation 4 suorittaminen kesti 8 minuuttia ja 50 sekuntia. Tiivistettä ei kuitenkaan vielä tällä hyökkäyksellä saatu murrettua. Variaatio 5:ssa kombinaatioiden määrä on sama, joten sen suorittaminen kokonaan kestää yhtä kauan. Hyökkäystä ehdittiin kuitenkin suorittaa vain 5 minuuttia 20 sekuntia, kun tiivisteen vastine löytyi. Näillä nopeuksilla tuhannen tiivisteen läpikäynti yhdellä variaatiolla veisi aikaa 147 tuntia.

6.3 Tietokanta 3

Kolmas tietokanta koostui bcrypt-tiivisteistä, joiden kustannuskerroin oli 10. Hyökkäyksistä suoritettiin variaatiot 3, 4 ja 5. Kuten tietokanta 2 tapauksessa väsytyshyökkäystä ei suoritettu, koska se olisi kestänyt vuosia. Hyökkäysvariaation 6 suorittaminen olisi myös kestänyt vuosia, joten sitä ei yritetty suorittaa.

Ensimmäisenä testattiin laitteiston laskentateho. Näytönohjain pystyi laskemaan 213 H/s ja prosessori 136 H/s. Tämä poikkesi hieman oletetusta, koska bcryptin pitäisi olla hitaampi näytönohjaimilla. Hyökkäyksissä ei kuitenkaan pystytty hyödyntämään molempia laitteita yhtäaikaa täydellä teholla, koska prosessorin kuormittuessa liikaa se ei ehdi tuottamaan tarpeeksi dataa näytönohjaimelle.

Tämä nähtiin siitä, kun variaatio 3 yritettiin ajaa käyttäen molempia yksiköitä. Saavutettu laskentanopeus oli vain 242 H/s, joka oli noin 30 % hitaampi kuin teoreettinen yhteenlaskettu nopeus 349 H/s. Hyökkäyksen aikana prosessorin kuorma oli keskimäärin lähes 100 %, mutta näytönohjaimen vain 50 %.

Pienestä laskentanopeudesta huolimatta hyökkäys oli onnistunut. Satunnaisesti valitun tiivisteen murtamiseen meni aikaa vain 4 minuuttia ja 23 sekuntia. Tiivisteen vastine löytyi, kun sanakirjasta oltiin käyty läpi 67 % (63456/94110) sanoista. Tämä tarkoittaa sitä, että

koko sanakirjan läpikäyntiin menee aikaa noin 6,5 minuuttia. Tunnissa pystyy siis yrittämään hieman yli yhdeksän eri tiivisteen murtamista. Vuorokaudessa vastaava määrä on 221. Koko tietokannan läpikäynti veisi aikaa noin 45 päivää.

Variaatioissa 4 ja 5 kombinaatioiden määrä oli niin suuri, että hyökkäyksiä ei suoritettu oikealla laitteistolla vaan tyydyttiin laskennalliseen tarkasteluun. Kombinaatioiden määrä oli 941 100, joten 242 H/s laskentanopeudella yhden hyökkäysvariaation suorittaminen olisi kestänyt hieman alta 65 minuuttia. Vuorokaudessa ehtisi yrittämään noin 24 salasanan murtamista. Koko tietokannan läpikäynti veisi noin 10 000 tuntia eli noin 416 päivää.

7. TULOSTEN ARVIOINTI

Olettamusten mukaisesti sanakirjahyökkäys oli paljon tehokkaampi kuin väsytyshyökkäys. Arvauksia piti suorittaa paljon vähemmän kuin sanakirjahyökkäyksessä. On kuitenkin hyvä muistaa, että väsytyshyökkäys on ainoa vaihtoehto, jos käyttäjä valitsee satunnaisista merkeistä generoidun salasanan. Toisin sanoen, jos käyttäjä haluaa varmasti vahvan salasanan, on syytä käyttää jotakin salasanojen hallintaohjelmaa (password manager), joka pystyy muistamaan ja generoimaan täysin satunnaisia salasanoja.

Generoidut tietokannat olivat aika yksinkertaisia ja hyökkääjän kannalta optimaalisia. Salasanat olivat helppo murtaa, koska ne olivat generoitu samasta sanakirjasta mitä käytettiin hyökätessä. Luvussa 2 käsitellyt tutkimukset kuitenkin osoittavat, että näin on myös oikeasti. Käyttäjät valitsevat salasanoiksi tavallisten sanojen variaatioita. Todenmukaisempia tuloksia saataisiin käyttämällä oikeita tietokantoja. Toki näiden vertailu on haastavaa, koska palveluilla on erilaisia salasanapolitiikkoja käytössä.

Sanakirjahyökkäyksissä käytetyt sanat ja niiden variaatiot olivat aika yksinkertaisia. Todellisissa hyökkäyksissä pitäisi ottaa mukaan myös paikkojen, maiden, tuotemerkkien ja erisnimien sanoja mukaan. Toisaalta käytetystä sanakirjasta voitaisiin pudottaa paljon sanoja pois. Tämä vaatisi kuitenkin laajempaa salasanatutkimusta. Esimerkiksi valitsevatko käyttäjät todennäköisemmin substantiivin kuin adjektiivin salasanaksi, tai käytetäänkö sanoista taivutettuja versioita. Lisäksi voisi tutkia, miten voitaisiin järkevästi muodostaa yhdyssana kombinaatioita, koska suoraan kahden sanakirjan yhdistäminen tuottaa valtavan määrän eri vaihtoehtoja.

Tietokanta 1 osoitti hyvin kuinka heikko SHA-256 algoritmi on, ja miksi sitä ei pitäisi ikinä käyttää tietokantojen suojaamisessa. Tietokantaa vastaan jopa väsytyshyökkäykset osoittautuivat mahdollisiksi. Näillä hyökkäyksillä ei kuitenkaan saatu kovin suurta osaa tietokannasta murrettua. Oikeita tietokantoja vastaan menetelmä olisi todennäköisesti tehokkaampi, koska käyttäjät valitsevat keskimäärin 8 merkkisen salasanan. Generoidussa tietokannassa keskipituus oli pitempi. Lisäksi suuri laskentanopeus mahdollistaa merkkimäärän kasvattamisen ainakin 10 merkkiin asti.

Tietokanta 2 oli suojattu huomattavasti paremmin kuin tietokanta 1. Tiivisteiden laskentanopeus oli alle 2000 H/s ja tiivisteet olivat suolattu. Kolme sanakirjahyökkäystä saatiin suoritettua hyvin lyhyessä ajassa. Nopein hyökkäys oli suora sanakirjahyökkäys, jonka suorittaminen kesti alle minuutin. Tällä nopeudella olisi voinut kokeilla paljon suurempiakin sanakirjoja. Toisaalta suolan käyttö rajoitti hyökkäyksen aina vain yhteen tiivisteeseen, joka teki koko sanakirjan läpikäynnin hyvin raskaaksi. Ilman suolaa koko tietokanta olisi murettu lähes kokonaan alle puolessa tunnissa. Tästä voidaan päätellä, että suolan käyttö on välttämätöntä salasanojen suojaamisessa ja on hyökkääjälle suurin hidaste.

Tuloksista kävi ilmi myös PBKDF2:n heikkous rinnakkaista GPU-laskentaa kohtaan. Laskentanopeus GPU:lla oli yli kymmenen kertaa suurempi kuin CPU:lla. Tämä on suuri etu hyökkääjälle, koska GPU-laskentatehoa on nykypäivänä paljon saatavilla. Iteraatioiden määrällä laskentanopeutta voi pienentää, mutta verkkopalveluissa tiivisteet yleensä lasketaan CPU:lla. Iteraatioiden määrä on siis vaikea nostaa niin suureksi, että hyökkäykset olisivat toteuttamiskelvottomia GPU:lla.

Jos kyseistä PBKDF2-funktiota käytettäisiin oikean tietokannan suojaamisessa olisi syytä käyttää suurempaa iteraatioiden määrää. Hyökkäykset yksittäisiä tiivisteitä kohtaan olivat liian nopeita. Näiden tulosten perusteella ei kuitenkaan voida sanoa mikä olisi sopiva iteraatioiden määrä. Kaiken kaikkiaan bcrypt vaikuttaa paljon vahvemmalta tiivistefunktiolta.

Tietokanta 3 oli ominaisuuksiltaan hyvin samanlainen kuin tietokanta 2. Suurimpana jarruna toimi suolaus ja funktion laskennallinen vaikeus. Yhden tiivisteen tapauksessa tulokset olivat kuitenkin erittäin positiivisia hyökkääjän kannalta. Kolme neljästä sanakirjahyökkäystä pystyttiin suorittamaan järkevässä ajassa. Usean tiivisteen tapauksessa laskenta-ajat kasvoivat useisiin päiviin, mutta pysyivät silti hyvin toteutettavissa.

Positiiviset tulokset hyökkääjän kannalta tarkoittavat kuitenkin sitä, että oikeissa tietokannoissa kustannuskertoimen pitäisi olla suurempi. Laskenta-ajan pitäisi kaksinkertaistua aina, kun kerroin nousee yhdellä. Tässä tutkimuksessa ei kuitenkaan otettu kantaa mikä on sopiva kerroin.

Käytössä ollut testijärjestelmä ei ollut kovin optimaalinen bcrypt-laskentaa varten ja kummastusta herätti, miksi näytönohjaimella suoritettu laskenta oli nopeampaa kuin CPU:lla. Tämä saattaa johtua siitä, että kirjoittajan CPU:ssa ei ollut kuin 4 ydintä. Jos laskentateho kasvaisi lineaarisesti jo 8 ytiminen prosessori olisi päässyt samoihin tuloksiin kuin näytönohjain.

Kaikkien saatujen tulosten yhteydessä on muistettava, että ne ovat laskettu vain yhden tietokoneen voimin. Lisäksi julkisissa tietomurroissa hyökkääjiä saattaa olla useita ja ny-

kyaikaiset pilvipalvelut avaavat täysin uusia mahdollisuuksia hyökkääjälle. Kenellä tahansa on mahdollisuus päästä käsiksi isoihin GPU/CPU-klustereihin mistäpäin maailmaa tahansa.

8. YHTEENVETO

Tässä työssä tutustuttiin ja vertailtiin erilaisia tiivisteiden murtamismenetelmiä. Vertailuun valittiin kaksi väsytyshyökkäysvariaatiota ja neljä erilaista sanakirjahyökkäystä. Vertailu suoritettiin hyökkäämällä kolmea itse generoitua salasanatietokantaa vastaan. Työn tavoitteena oli selvittää minkälaiset menetelmät ovat edes mahdollisia suorittaa järkevässä ajassa ja vertailla näiden tehokkuutta.

Suoritetussa vertailussa kävi ilmi, että heikosti suojattua tietokantaa vasten pystytään suorittamaan lähes minkälainen hyökkäys tahansa muutamassa minuutissa. Käytetyllä laitteistolla vahvempia tiivistefunktioita vastaan joudutaan tyytymään sanakirjahyökkäyksiin, joissa sanalistojen koko on luokkaa 100 000 sanaa. Lisäksi havaittiin, että hyökkäysten suurin hidaste ei ole tiivistefunktion vaatima laskentateho, vaan tietokannoissa käytössä oleva suola. Suola ei kuitenkaan hidastanut yksittäisiä tiivisteitä vastaan suoritettuja hyökkäyksiä.

Tutkituista menetelmistä suora sanakirjahyökkäys oli nopein suorittaa, ja se pystyttiin suorittamaan nopeasti jopa bcryptillä suojattua tietokantaa vastaan. Hyökkäyksen teho riippuu kuitenkin täysin käytetystä sanalistasta ja tiivistefunktiosta. Käsitellyistä tiivistefunktioista suojaksi kannattaisi valita bcrypt-tiivistefunktio, jonka kustannuskerroin on yli kymmenen. Nämä tulokset eivät kuitenkaan riitä kertomaan tarjoaako tämä yhdistelmä riittävästi suojaa.

Tavalliset käyttäjät voivat parantaa tietoturvaansa valitsemalla salasanansa siten, että ne eivät ole minkään tunnetun sanan muunnelmia. Tämä tekee salasanan muistamisesta vaikeaa, joten suositeltavaa olisi käyttää salasanojen hallintaohjelmaa generoimaan täysin satunnaisia merkkijonoja salasanoiksi. Tämä poistaa hyökkääjältä mahdollisuuden hyödyntää nopeita sanakirjahyökkäyksiä. Toinen vaihtoehto muodostaa vahva salasana, on liittää useita, vähintään kolme, tavallista sanaa peräkkäin. Yhdistelmän muistaminen on helppoa ja tekee sanakirjahyökkäyksen suorittamisen vaikeaksi, koska kombinaatioiden määrä kasvaa suureksi.

LÄHTEET

[1] Florencio D., Herley C., A large scale study of web password habits, ACM, November 2006.

[2] Dell'Amico M., Michiardi P., Roudier Y., Password strength: An empirical analysis, 2010 Proceedings IEEE INFOCOM, pp. 1–9, 2010.

[3] Cormac H., So long, and no thanks for the externalities: The rational rejection of security advice by users, in Proceedings of the 2009 Workshop on New Security Paradigms Workshop, ACM, 2009.

[4] National Institute of Standards and Technology, SP 800-63 - Electronic Authentication Guideline, 2004.

[5] Have I been pwned, verkkosivu, 2018. Saatavissa (viitattu: 22.11.2018): `https://haveibeenpwned.com/PwnedWebsites`.

[6] Yiannis C., Modern password cracking: A hands-on approach to creating an optimised and versatile attack, May 2013.

[7] Goldwasser S., Bellare M., Lecture Notes on Cryptography, Massachusetts Institute of Technology, 2008.

[8] Dang Q.H., SP 800-107. Recommendation for Applications Using Approved Hash Algorithms, National Institute of Standards & Technology, 2012.

[9] Department of Commerce and National Institute of Standards and Technology, Secure hash standard - SHS: Federal information processing standards publication 180-4, 2012.

[10] Buchanan W.J., Cryptography (River Publishers Series in Information Science and Technology), River Publishers, 2017, ISBN 8793379102.

[11] Aggarwal A., Chaphekar P., Mandrekar R., Cryptanalysis of bcrypt and SHA-512 using distributed processing over the cloud, International Journal of Computer Applications, 2015.

[12] Kaliski B., PKCS #5: Password-Based Cryptography Specification Version 2.0, RFC 2898, September 2000.

[13] Provos N., Mazières D., A future-adaptive password scheme, in Proceedings of the Annual Conference on USENIX Annual Technical Conference, USENIX Association, 1999.

[14] Kotimaisten kielten keskuksen nykysuomen sanalista, Kotus, verkkosivu 2018. Saatavissa (viitattu 23.11.2018): `http://kaino.kotus.fi/sanat/nykysuomi/`.

[15] Hybrid Attack, Hashcat, verkkosivu, 2018. Saatavissa (viitattu: 15.11.2018): `https://hashcat.net/wiki/doku.php?id=hybrid_attack`.

[16] Rule-based Attack, Hashcat, verkkosivu, 2018. Saatavissa (viitattu: 15.11.2018): `https://hashcat.net/wiki/doku.php?id=rule_based_attack`.

[17] Combinator Attack, Hashcat, verkkosivu, 2018. Saatavissa (viitattu 15.11.2018): `https://hashcat.net/wiki/doku.php?id=combinator_attack`.