



## **Supertietokoneiden vaikutus salasanojen turvallisuuteen**

Tatu Heikkinen

Haaga-Helia ammattikorkeakoulu

Tradenomi, tietojenkäsittelyn koulutusohjelma

Tutkimusraportti

2023

## Tiivistelmä

<b>Tekijä(t)</b> Tatu Heikkinen
<b>Tutkinto</b> Tradenomi
<b>Raportin/Opinnäytetyön nimi</b> Tutkimusraportti
<b>Sivu- ja liitesivumäärä</b> 15 + 6
<p>Tiivistelmä edellytetään pääsääntöisesti vain opinnäytetöissä.</p>
<b>Asiasanat</b> Tärkeysjärjestyksessä 3–6 asiasanaa, jotka kuvaavat työn sisältöä parhaiten. Hyödynnä asiasanastoja <a href="http://finto.fi/fi/">http://finto.fi/fi/</a> ja <a href="https://annif.org/">https://annif.org/</a>

## Sisällys

1	Johdanto .....	1
2	Supertietokoneiden vaikutus salasanojen turvallisuuteen .....	3
2.1	Supertietokoneiden perusteet.....	3
2.2	Supertietokoneiden haasteet ja rajoitteet salasanojen murtamisessa .....	3
2.3	Laskentatehon kehitys.....	3
2.4	Salasanojen rooli.....	4
2.5	Salasanojen luominen .....	4
2.6	Salasanapolitiikat .....	5
2.7	Salasanojen tulevaisuus.....	7
2.8	Salasanalistat.....	7
3	Laskentatehoanalyysi kuluttajien komponenteilla ja supertietokoneilla .....	9
3.1	Tutkimusmenetelmä .....	9
3.2	Miten tutkimus toteutetaan? .....	9
3.3	Millaisia tutkimustulokset ovat? .....	9
3.4	Tutkimuksen toteutus ja replikointi.....	10
3.5	Tutkimustulokset .....	10
4	Johtopäätökset.....	12
4.1	Kuinka nopeasti supertietokoneet pystyvät murtamaan salasanoja brute-force- hyökkäyksillä? .....	12
4.2	Minkälainen potentiaali tietokoneilla on laskentatehon kehityksessä? .....	12
4.3	Miten pitkiä salasanojen tulisi olla, jotta niitä olisi epäkäytännöllistä murtaa brute-force salasanahyökkäyksillä? .....	12
4.4	Miten salausmenetelmiä tulisi muuttaa, jotta salasanojen murto olisi epäkäytännöllistä? .....	13
	Lähteet.....	14
	Liitteet .....	16

# 1 Johdanto

Aihepiirinä on supertietokoneiden kehittymisen vaikutus salasanojen turvallisuuteen. Salasanat ovat tärkeä osa tietoturvaa niin yritysten, kuin yksityishenkilöidenkin kannalta, mutta kvantti- sekä supertietokoneiden kehitys asettaa nykyiset salasanapolitiikat sekä -menetelmät tietoturvariskiuhan alle.

Aihetta kannattaa tutkia, koska se auttaa kartoittamaan riskejä, jotka tietokoneiden ja niiden laskentatehon nopeuden kehitys mahdollistaa. Tutkimusta voidaan myös hyödyntää kehittämään uusia ja parempia tapoja suojautua salasanahyökkäyksiä vastaan. Tutkimustuloksista voivat hyötyä tietoturva-asiantuntijat, organisaatiot ja yksityishenkilöt, jotka haluavat suojata tietoa perinteisiä salasanoja käyttämällä.

Tutkimuksessa pyritään löytämään vastaus alla lueteltuihin kysymyksiin:

- Kuinka nopeasti supertietokoneet pystyvät murtamaan salasanoja brute-force hyökkäyksillä?
- Minkälainen potentiaali tietokoneilla on laskentatehon kehityksessä?
- Miten pitkiä salasanojen tulisi olla, jotta niitä olisi epäkäytännöllistä murtaa brute-force salasanahyökkäyksillä?
- Miten salausmenetelmiä tulisi muuttaa, jotta salasanojen murto olisi epäkäytännöllistä?

Tutkimuskysymykset auttavat arvioimaan nykyisten tietoturvamenetelmien vahvuutta tulevaisuudessa, jotta kehitystä voitaisiin tehdä oikeaan suuntaan.

Tutkimuksessa keskitytään puhtaasti laskentatehon ja laskentamenetelmien kehittymisen vaikutuksista salasanojen vahvuuteen, eikä oteta huomioon tekoälyn tai salasanalistojen tuomaa hyötyä salasanojen murtamisessa.

Tutkimuksen kannalta keskeisiä käsitteitä ovat:

- **Brute-force hyökkäys** on hyökkäysmenetelmä, jossa kokeillaan kaikkia mahdollisia arvoja esimerkiksi salasanalle, kunnes oikea arvo ratkeaa (Zieniūtė 2022).
- **Hash-algoritmi** eli tiivistefunktio on funktio, jolla voidaan tuottaa kiinteän pituinen tiiviste mistä tahansa merkkijonosta. Hashit ovat useimmiten yksisuuntaisia, eli tietystä merkkijonosta saadaan aina sama tiiviste, mutta tiivistettä ei pysty muuttamaan takaisin alkuperäiseksi merkkijonoksi. Esimerkiksi käyttäjää luodessa salasanasta voidaan ottaa tiiviste ja tallentaa se palvelun tietokantaan ja kirjautuessa verrata annetun salasanan tiivistettä palvelun tietokannasta löytyvään tiivisteeseen. (Kyberturvallisuuskeskus 2020.)

- **Hashcat** on yksi tunnetuimmista salasananmurto-ohjelmista, jota käytetään esimerkiksi tietoturva-ammattilaisten toimesta (Javatpoint s.a). Hashcat on myös itsetunnustetusti maailman nopein salasananmurtotyökalu.
- **Laskentateho** tarkoittaa nopeutta millä tietojärjestelmä kykenee suorittamaan laskutoimituksia (Strickland s.a).
- **Salasana** on merkkijono, jolla rajoitetaan käyttäjien pääsyä tietokonejärjestelmiin, verkkopalveluihin tai tiedostoihin (Bacon 2021).
- **Salasanapolitiikka** on eri palveluissa määritetty joukko sääntöjä, mitkä salasanojen täytyy täyttää, jotta salasana on kelpollinen kyseisessä palvelussa (Odogwu 2021).
- **Supertietokone** on useista prosessori- ja näytönohjainytimistä rakennettu järjestelmä, jonka tarkoituksena on ratkaista haastavia ja monimutkaisia laskentatehtäviä (CSC s.a).

## **2 Supertietokoneiden vaikutus salasanojen turvallisuuteen**

### **2.1 Supertietokoneiden perusteet**

Supertietokoneet ovat sadoista tai tuhansista linkitetyistä tietokoneista kasattuja systeemejä, joiden tarkoituksena on ratkaista laskutoimituksia, jotka ovat mahdottomia tavallisille tietokoneille.

Supertietokoneita voidaan käyttää esimerkiksi simuloimaan ydinaseiden räjähdysä ja ilmaston muutosta tai tekoälyn luomiseen. Näin haastavien laskutoimitusten ratkaisemiseen tarvitaan suuret määrät laskentatehoa ja siksi supertietokoneet tarvitsevat satoja neliömetrejä tilaa sekä megavatteja energiaa, joka vastaa tuhansien kotitalouksien käyttämää energiamäärää. (Shankland 4.3.2020.)

### **2.2 Supertietokoneiden haasteet ja rajoitteet salasanojen murtamisessa**

Supertietokoneiden rakentaminen ja käyttäminen ei ole itsestäänselvyys, vaikka ne muistuttavatkin kuluttajille suunnattua laitteistoa. Ne rakennetaan samantapaisista tietokoneen komponenteista, mutta supertietokoneissa osia on paljon enemmän, jonka takia niiden rakentaminen vaatii suuren tilan sekä suuria investointeja. CSC:n Kajaanin datakeskukseen rakennetun supertietokoneen LUMI:n rakentamiseen tarvittiin tilaa lähes 300 neliometriä ja se painaa lähes 150 000 kiloa. Järjestelmään myös investoitiin yli 200 miljoonaa euroa ja siihen osallistui yhdeksän maata Suomen sekä EU:n jäsenmaiden ja komission yhteisyritys EuroHPC JU:n lisäksi. (CSC s.a.)

Supertietokoneiden tehtävänä on ratkaista monimutkaisia ongelmia, kuten ilmaston lämpenemisen vaikutusta elinolosuhteisiin tai lääkeaineiden kehityksen vaikutuksia pandemioiden vastaisessa taistelussa. Supertietokoneiden laskentatehoa voidaan käyttää myös tekoälyn sekä itseohjautuvien autojen ja laivojen opetuksessa. (CSC s.a.) Koska supertietokoneet ovat kehitetty monimutkaisten ongelmien ratkaisemiseen, ne eivät välttämättä pysty käyttämään koko potentiaaliaan yksinkertaisemmissa laskutoimituksissa, kuten merkkijonojen muuttamisessa hasheiksi ja niiden vertaamisessa tietokannasta löytyviin salasanaatiivisteisiin.

### **2.3 Laskentatehon kehitys**

Vuonna 1970 Gordon Moore ennusti tietokoneiden prosessorien nopeuden tuplaantuvan joka toinen vuosi ja viimeiset vuosikymmenet ovat todistaneet väitteen paikkansapitävyyttä. Kyseinen väite tunnetaan nykyään Mooren lakina ja sen mukaan tämännäpäiväisten prosessoreiden nopeus olisi noin 67 miljoonaa kertainen vuoden 1970 prosessoreihin verrattuna. (McCarthy 2.5.2017.)

Standard Performance Evaluation Corporationin tekemän vertailun mukaan vuosien 1985 ja 2010 välillä tietokoneiden tekemien liukulukuoperaatioiden nopeus on kasvanut yli 2000-kertaiseksi.

Mooren lain mukaan tällä aikavälillä kehityksen tulisi olla noin 4000-kertainen, joten olemme jääneet ennustuksesta kaksi vuotta jälkeen. Kehityksen hidastumiseen vaikuttaa se, että laskentatehon parantaminen ei ole taloudellisesta näkökulmasta järkevää. (National Academic Press 2011.)

Tietokoneiden nopeutta rajoittavat myös fyysiset tekijät sillä tiedon täytyy kulkea fyysisesti paikasta toiseen. Tiedonvälitys ei voi ylittää valonnopeutta, joten laskentaan osallistuvien osien koko ja etäisyys toisistaan, tiedon prosessointi sekä laskentaan käytettävät algoritmit vaikuttavat laskentanopeuteen. Komponenttien koko tulee olemaan kehityksen kohde, kunnes fyysiset rajat tulevat vastaan. Tämän vuoksi laskentaan käytettävien algoritmien kehittäminen on merkittävässä roolissa laskentatehon kannalta ja on mahdollista, että edistys tiedon prosessointimenetelmissä avaa uuden suunnan kehitykselle. (Scientific American 21.10.1999.)

MIT Computer Science & Artificial Intelligence Laboratory:n vuonna 2011 tekemän tutkimuksen mukaan vuosien 1960 ja 2011 välillä kohtuullista laskentatehoa vaativien ongelmien ratkaisuun käytetyistä algoritmeista noin puolet noudatti Mooren lakia vuodesta toiseen. Isoin hyöty algoritmien kehityksestä oli suurten datamassojen analysoimisessa. (Sherry & Thompson 21.9.2021.)

## **2.4 Salasanojen rooli**

Salasanoja käytetään rajoittamaan pääsyä tietokonejärjestelmiin, verkkopalveluihin tai tiedostoihin. Heikot eli helposti arvattavat salasanat voidaan murtaa tavallisilla tietokoneilla sekunneissa, joten niiden käyttäminen tunnistautumiseen tai tietojen salaamiseen ei ole suositeltavaa. Luvattoman pääsyn estämiseksi salasanojen tulisi olla monimutkaisia ja vaikeasti arvattavia, jotta tiedot eivät päätyisi väärin käsiin. Samojen salasanojen uudelleenkäyttö eri tarkoituksiin on tietoturvariski, koska salasanan selvittäminen ulkopuoliselle, voi hän yrittää saada pääsyn myös muihin käyttämiisi palveluihin tai tiedostoihin käyttämällä samaa salasanaa (Kyberturvallisuuskeskus 2023). Käyttäjätunnuksen selvittäminen on helpompaa, koska niitä ei kovinkaan usein salata mitenkään ja jo luotuja käyttäjätunnuksia on mahdollista löytää esimerkiksi kokeilemalla niitä eri palveluiden käyttäjänluontilomakkeessa, sillä palvelut ilmoittavat, jos käyttäjä on jo luotu samalla tunnuksella.

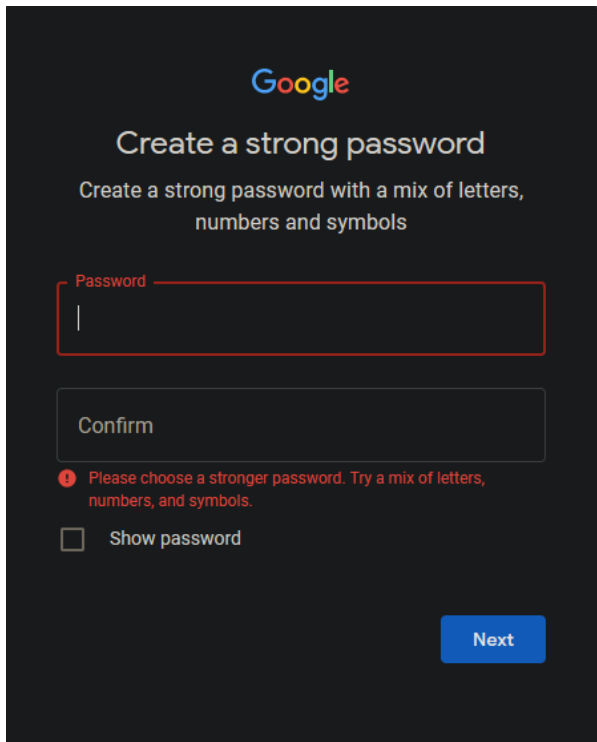
## **2.5 Salasanojen luominen**

Vahvojen salasanojen keksiminen voi olla vaikeaa, koska niiden tulisi täyttää erilaisia salasanapolitiikkoja ja olla helposti muistettavia. Tämän lisäksi salasanapolitiikat ovat vain minimivaatimuksia palveluiden käyttämiseksi. Pitkä salasana on aina parempi kuin lyhyt salasana, mutta niiden muistaminen on myös vaikeampaa. Siksi esimerkiksi kokonainen, helposti muistettava, mutta vaikeasti arvattava lause on hyvä salasana. Myös kirjoitusvirheet tai puhekielen

ilmaisut vahvistavat salasanaa, koska ne vaikeuttavat salasanalistojen käyttöä.  
(Kyberturvallisuuskeskus 2023.)

## 2.6 Salasanapolitiikat

Luodessa käyttäjää Googlen palveluihin salasanat täytyy olla aina vähintään kahdeksan merkkiä pitkä. Jos salasanaksi yrittää asettaa esimerkiksi "12345678" saa virheilmoituksen, jossa kehoitetaan käyttämään pieniä ja isoja kirjaimia sekä numeroita, mutta salasana voi kuitenkin koostua pelkästään numeroista, pienistä tai isoista kirjaimista. Monet helposti arvattavat salasanat kuten "salasana", "12341234" tai "PASSWORD" ovat estettyjä, mutta "tammikuu" ei. Google on kuitenkin estänyt kuukausien nimien käytön salasanana englanniksi, ruotsiksi ja jopa norjaksi, mutta ei suomeksi, vaikka suomen ja norjan väkiluvut ovat hyvin lähellä toisiaan. Tähän saattaa kuitenkin vaikuttaa se, että kuukausien nimet norjaksi ovat hyvin lähellä tai täysin samoja kuin englanniksi.



Kuva 1 Googlen salasanluontisäännöt (Google)

Facebookiin käyttäjää luodessa salasanat täytyy olla vähintään kuusi merkkiä pitkä ja virheilmoituksessa kehoitettiin käyttämään uniikkia sekä vaikeasti arvattavaa salasanaa. Muita rajoitteita salasanoille ei ollut, mutta yleisimmät salasanat kuten "password" ja "123456" olivat jälleen estetty. Käyttäjän luonti oli jälleen kerran mahdollista salasanat vähimmäispituuden täyttävillä suomenkielisillä kuukausien nimillä.



## Sign Up

It's quick and easy.

Please choose a more secure password. It should be longer than 6 characters, unique to you, and difficult for others to guess.




Birthday ?
 

Oct

5

2007

Gender ?
 

Female ☒

Male ☐

Custom ☐

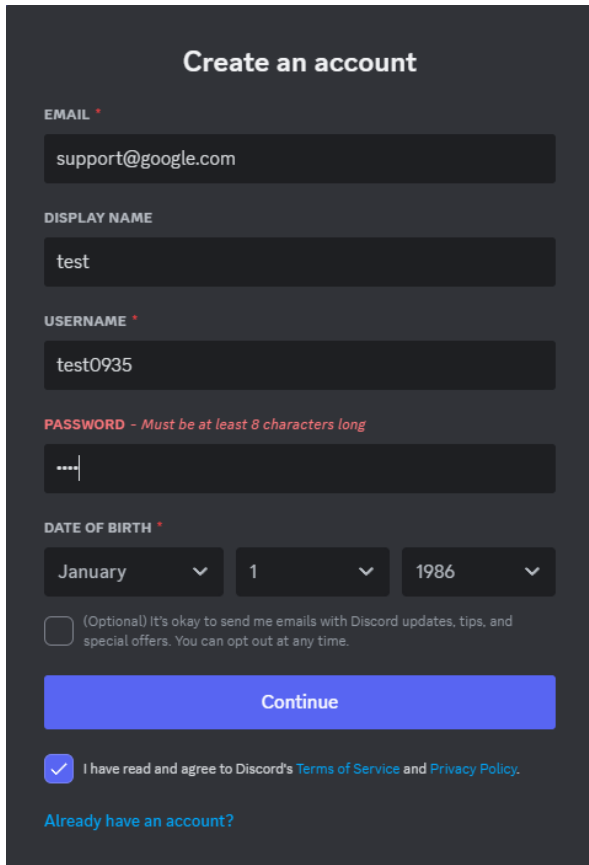
People who use our service may have uploaded your contact information to Facebook. [Learn more.](#)

By clicking Sign Up, you agree to our [Terms](#). Learn how we collect, use and share your data in our [Privacy Policy](#) and how we use cookies and similar technology in our [Cookies Policy](#). You may receive SMS Notifications from us and can opt out any time.

Sign Up

Kuva 2 Facebookin salasananluontisäännöt (Facebook)

Kokeilin luoda käyttäjää myös Discord -nimiseen palveluun, joka on kasvattanut suosiotaan tietokonepelaajien käyttämänä viestintäpalveluna. Salasanaa luodessa ainoana rajoitteena oli vähintään kahdeksan merkin pituus ja virheilmoituksena varoitettiin liian lyhyestä tai liian tavanomaisesta salasanasta. Myös Discordissa pelkästään pienistä kirjaimista koostuva kahdeksanmerkkinen salasana oli riittävän hyvä, mutta tällä kertaa suomenkieliset kuukaudet, kuten tammikuu olivat liian tavallisia sanoja käytettäväksi salasanana.



The image shows the 'Create an account' form on the Discord website. The form is dark-themed with white text. It includes fields for EMAIL (support@google.com), DISPLAY NAME (test), USERNAME (test0935), and PASSWORD (masked with dots). Below the password field is a checkbox for optional email updates. At the bottom, there is a 'Continue' button and a checkbox for agreeing to the Terms of Service and Privacy Policy. A link for 'Already have an account?' is also present.

Kuva 3 Discordin salasananluontisäännöt (Discord)

## 2.7 Salasanojen tulevaisuus

Useiden vahvojen salasanojen muistaminen on ihmiselle lähes mahdotonta, joten siihen on kehitetty muita tekniikoita. Yksi tekniikoista on salasananamanagerin käyttö. Salasanamanagerin käyttöön tarvittavat yhden pääsalasanat, jolla voit avata salatun tietokannan sekä lisätä, poistaa ja muuttaa tietokantaan tallennettuja käyttäjiä (Kyberturvallisuuskeskus 2023). Vaihtoehtoisia tunnistautumistapoja ovat esimerkiksi kaksi- sekä monivaiheinen tunnistautuminen, joissa salasanan lisäksi tunnistautumiseen vaaditaan esimerkiksi sormenjälki tai älypuhelimesta löytyvä tunnistin. Sormenjälkeä tai kasvojen piirteitä voidaan käyttää myös salasanan korvikkeena palveluissa ja laitteissa, joissa vaihtoehtona on biometrinen tunnistautuminen ilman, että käyttäjältä vaadittaisiin kaksivaiheista tunnistautumista. Tämä nopeuttaa esimerkiksi älylaitteiden lukituksen avaamista. (Bacon 2021.)

## 2.8 Salasanalistat

Salasanalistat ovat salasanojen murrossa käytettyjä tiedostoja, jotka pitävät sisällään suuren määrän mahdollisia salasanoja. Vuonna 2009 RockYou nimisestä palvelusta saatiin tietomurron yhteydessä varastettua tietokanta, joka sisälsi yli 32 miljoonan käyttäjän kirjautumistiedot

selväkielisenä tekstinä (Schofield 2009). Kyseisistä salasanoista luotiin varmaankin maailman tunnetuin salasanalista nimeltään rockyou.txt, jota käytetään salasanahyökkäyksissä niin tietoturva-ammattilaisten kuin rikollistenkin toimesta. Salasanalistat nopeuttavat salasanahyökkäyksiä huomattavasti, koska usein käytetyillä salasoilla saadaan murrettua enemmän käyttäjätunnuksia, kuin satunnaisesti luoduilla salasoilla.

Ihmisten luomissa salasoissa käytetään usein sanoja tai lukuja jotka on helppo muistaa. Helposti muistettavia ja salasoissa usein käytettyjä merkkijonoja ovat vuosiluvut kuten käyttäjän syntymävuosi tai vuosi jolloin salasana on luotu. Myös ihmisten ja urheilujoukkueiden nimet sekä kiro sanat ovat suosittuja salasoissa. Tietomurtoja seuraamalla vuonna 2023 Cybernewsin luoman listan mukaan kymmenen suosituimman salasanan joukosta löytyi "qwerty". (Masiliauskas 2023.)

Todennäköisyyslaskennan perusteella kuusikirjaimisia pienellä kirjoitettuja salanoja generoimalla vain yksi ihminen hieman yli 308 miljoonasta saisi salasanakseen "qwerty". Salasoissa saa kuitenkin käyttää isoja kirjaimia, numeroita sekä usein myös erikoismerkkejä ja salasanat voivat olla pidempiä kuin kuusi merkkiä pitkiä. Tämän perusteella kovinkaan monella ihmisellä ei kuuluisi olla samaa salasanaa, mutta todellisuus on kuitenkin toinen. On siis tehokkaampaa käyttää salasanalistoja salasanamurrossa, kuin kokeilla brute-force-menetelmällä kaikkia mahdollisia yhdistelmiä.

### **3 Laskentatehoanalyysi kuluttajien komponenteilla ja supertietokoneilla**

#### **3.1 Tutkimusmenetelmä**

Tutkimusmenetelmäksi valikoitui koejärjestely, koska salasanojen vahvuus pohjautuu niiden pituuden ja monimutkaisuuden suhteesta laskentatehoon ja tuloksia voidaan arvioida jakamalla kaikkien mahdollisten, tietyillä säännöillä luotujen, erilaisten merkkijonojen määrä nopeudella, millä tietokoneet pystyvät kokeilemaan samoilla säännöillä luotuja merkkijonoja.

Koejärjestelyn tarkoituksena ei ole antaa täysin tarkkoja vastauksia tutkimuskysymyksiin, vaan toimia viitteenä sekä vertailukohtana supertietokoneiden laskentateholle, jotta supertietokoneiden yltämään nopeuteen kykenevien tietojärjestelmien potentiaalia salasanojen murtamisessa olisi mahdollista arvioida mahdollisimman tarkasti.

#### **3.2 Miten tutkimus toteutetaan?**

Mittaan kuluttajille suunnattujen tietokoneiden näytönohjainten laskentatehoa simuloimalla brute-force-hyökkäyksiä ja vertaan sitä supertietokoneiden saavuttamaan laskentatehoon. Hyödynnän yleisessä käytössä olevien palveluiden, kuten Facebookin salasanaoperaatioitten mukaisia sääntöjä, joiden avulla arvioin nykyisten salasanojen vahvuutta tilanteessa, jossa eri toimijoilla on käytössään supertietokoneiden saavuttaman laskentatehon omaavia laitteita. Arvioin myös minimivaatimuksia salasanojen pituudelle samassa futuristisessa tilanteessa, jotta salasanojen murtaminen brute-force-hyökkäyksillä olisi ajankulutuksen takia epäkäytännöllistä.

Tämän kurssin aikana minulla ei ole mahdollisuutta päästä kokeilemaan supertietokoneita eikä niiden laskentatehoa, joten joudun turvautumaan niiden osalta internetistä löytyvään tietoon. Käytän internetistä löytyvää tietoa myös kuluttajille suunnattujen komponenttien osalta, jotta on mahdollista vertailla kuluttajille suunnattuja tietokoneita ja supertietokoneita sekä arvioida tietokoneiden laskentatehon kehitystä tulevaisuudessa.

#### **3.3 Millaisia tutkimustulokset ovat?**

Koejärjestelyn tulokset ovat eripituisia ajanjaksoja, joita merkkijonojen murtamiseen kuluu. Vertaan ajanjaksojen pituutta ja kokeiltujen merkkijonojen määrää sekä laitteiston laskentatehoa supertietokoneiden laskentatehoon ja arvioin millaisia tuloksia supertietokoneet saavuttaisivat samoissa testeissä.

Hashcatin antamien syötteiden mukana esitetyt kellonajat ovat pyöristetty tietokoneen mukaan käynnissä olleeseen sekuntiin, joten nopeus millä merkkijonoja on kokeiltu on suuntaa antava arvio. Tutkimuksen kannalta täysin tarkat tulokset ovat epäolennaisia, koska tietokoneiden

kehittyessä meillä on aikaa kehittää myös tietoturvamenetelmiä. Supertietokoneita ei myöskään olla optimoitu murtamaan salasanoja eikä tutkimuksessa käytetty laitteisto ole uudenveroista, jolla saattaa olla vaikutusta tutkimuksessa saatuihin tuloksiin.

### 3.4 Tutkimuksen toteutus ja replikointi

Tutkimus toteutettiin puhtaalla asennuksella Debian 11 käyttöjärjestelmästä tyhjälle kovalevylle. Tietokoneessa käytettiin Ryzen 5 3600 -prosessoria, NVIDIA GeForce RTX 3060 Ti -näytönohjainta, 64Gt DDR4 RAM-muistia ja Samsung 860 QVO SSD:tä kovalevynä. Käyttöjärjestelmään asennettiin NVIDIA 525.105.17 grafiikka-ajurit, NVIDIA CUDA rinnakkaislaskenta-alusta, Hashcat salasananmurtotyökalu sekä Micro tekstieditori.

Loin "passwordList.txt"-nimisen tiedoston, joka sisälsi kaikki mahdolliset vain pieniä kirjaimia a-z sisältävät kuusimerkkiset merkkijonot, jota käytin salasanojen murtamisessa. Yhdistelmiä oli 308 915 776. Yhdistelmien määrä voidaan tarkistaa laskemalla kaavalla  $26^6 = 308\,915\,776$ , koska erilaisia merkkejä on 26 ja salasanan pituus on kuusi merkkiä. Tämän jälkeen loin tietokantaa esittävän tiedoston, johon generoin viisi satunnaista kuusi pientä kirjainta a-z sisältävää salasanaa, jotka muutin hasheiksi käyttämällä MD5 tiivistealgoritmiä. Mursin tietokannan sisältämät hashit Hashcat-salasananmurtotyökalulla kokeilemalla kaikkia passwordList.txt:n sisältämiä merkkijonoja. Kokeilin lopuksi myös passwordList.txt:n ensimmäisestä sekä viimeisestä merkkijonosta tehtyjen tiivisteiden murtamista, jotta saisin käsityksen, kuinka paikkaansapitävinä tuloksia voidaan pitää. Tiivisteiden murrossa käytettiin passwordList.txt:n sisältämiä merkkijonoja aakkosjärjestyksessä, joten "aaaaaa":sta otetun tiivisteiden murtamisen pitäisi olla nopeampaa kuin minkään muun tutkimuksessa käytetyn merkkijonon ja samaa logiikkaa käyttämällä "zzzzzz":sta otetun tiivisteiden murtaminen pitäisi olla hitainta.

### 3.5 Tutkimustulokset

Tekemäni tietokannan murtamiseen kului noin 16 sekuntia. Tietokannan viimeisenä murrettu salasana oli "zhixln", joka oli oletettavaa, koska se oli tietokannan salasanoista aakkosjärjestyksessä viimeinen. Merkkijonoja käytiin läpi aakkosjärjestyksessä ja z on viimeinen tutkimuksessa käytetyistä 26:sta aakkosesta, zhixln:n järjestysnumeron voidaan siis arvioida olevan jonkin verran suurempi kuin  $25/26 \times 308\,915\,776$ . Hashcatin nopeus voidaan laskea kaavalla järjestysnumero/kulunut aika. Arvioitu nopeus millä Hashcat kokeili merkkijonoja ensimmäisessä testissä on  $308\,915\,776 \times (25/26) / 16 = 18\,564\,650$  merkkijonoa/sekunti.

PasswordList.txt:n viimeisestä merkkijonosta otetun tiivisteiden murtamiseen kului noin 18 sekuntia. Tässä testissä keskinopeus millä Hashcat kokeili eri merkkijonoja on  $308\,915\,776 / 18 = 17\,161\,987,6$ . Tutkimuksen kannalta on parempi, jos tulokset pyöristetään ylöspäin, jotta

supertietokoneiden potentiaalia salasanojen murrossa ei aliarvioitaisi. Hashcatin voidaan siis pyöristää kokeilevan noin 19 miljoonaa merkkijonoa sekunnissa NVIDIA GeForce RTX 3060 Ti -näytönohaimella.

## 4 Johtopäätökset

### 4.1 Kuinka nopeasti supertietokoneet pystyvät murtamaan salasanoja brute-force-hyökkäyksillä?

NVIDIA GeForce RTX 3060 Ti -näytönohjain kykenee suorittamaan 16,2 TFLOPS:a sekunnissa (Versus s.a). Kyseisellä näytönohjaimella pystyy tarkistamaan noin 19 miljoonan merkkijonon MD5-tiivisteet sekunnissa. 1 TFLOPS nopeus vastaa siis noin 1,2 miljoonan merkkijonon tiivisteiden lakemista sekunnissa. LUMI-supertietokone kykenee suorittamaan 375 PFLOPS:a sekunnissa (CSC s.a). Tämä vastaa 375 000 TFLOPS:a, joten LUMI-supertietokoneen voidaan arvioida laskemavan 450 miljardia MD5-tiivistettä sekunnissa.

### 4.2 Minkälainen potentiaali tietokoneilla on laskentatehon kehityksessä?

Mooren lain mukaan tietokoneiden nopeuden tulisi kaksinkertaistua jokatoinen vuosi. NVIDIA GeForce RTX 3060 Ti julkaistiin vuonna 2020 (NVIDIA 2020). Ero nopeudessa LUMI-supertietokoneeseen on noin 23 000 kertainen. Mooren lain mukaan tavalliset näytönohjaimet saavuttaisivat LUMI:n tai suuremman nopeuden vuonna 2050, koska  $2^{15} = 32\,768$  ja tämä on pienin kahden potenssi joka ylittää 23 000. Mooren lain perusteella jokainen potenssi vastaa kahta vuotta ja laskutoimituksen aloitusvuosi on 2020. Vuotta 2050 voidaan siis pitää ajankohtana, jolloin salasanapolitiikkojen tulisi olla tutkimuksen tulosten perusteella arvioidun käytännön mukaisia, jotta salasanamurroilta vältyttäisiin.

### 4.3 Miten pitkiä salasanojen tulisi olla, jotta niitä olisi epäkäytännöllistä murtaa brute-force salasanahyökkäyksillä?

Oletetaan, että hyökkääjällä on käytössään LUMI:n kaltainen järjestelmä, jolla hän yrittää murtaa salasanoja. Tilanteessa salasoissa käytettäviä merkkejä ovat pienet kirjaimet a-z, isot kirjaimet A-Z, numerot 0-9 sekä yksi erikoismerkki jokaista numeronäppäintä 0-9 kohden. Yhteensä erilaisia merkkejä on siis  $26+26+10+10 = 72$  merkkiä. Käytetään Mooren laista tuttua kahta vuotta ajanjaksona, jonka hyökkääjä saa tarkistaakseen kaikki mahdolliset merkkijonot. Jos hyökkääjä onnistuu tehtävässä salasana on epäturvallinen, muussa tapauksessa salasana on turvallinen. Kahdessa vuodessa hyökkääjä voi kokeilla  $2 \times 365 \times 24 \times 60 \times 60 \times 450\,000\,000\,000 = 28\,382\,400\,000\,000\,000\,000 = 2,838\,24 \times 10^{19}$  eri tiivistettä.

Tarpeeksi vahvan salasanan pituus voidaan ratkaista kun  $72^x > 2,9 \times 10^{19}$ , jossa x kuvaa salasanan pituutta.  $72^{11} = 2,69... \times 10^{20}$  ja on samalla pienin 72 potenssi, joka täyttää ehdon. Tämä tarkoittaa, että salasanojen tulisi olla vähintään 11 merkkiä pitkiä, jotta niitä ei pystyisi varmuudella murtamaan kahdessa vuodessa LUMI:n kaltaisella järjestelmällä. Kaikista

mahdollisista 11-merkkisistä salasanoista kyettäisiin kuitenkin murtamaan noin kymmenesosa. 11 merkin pituutta voidaan kuitenkin pitää turvallisena lähitulevaisuudessa, koska nykyisillä supertietokoneilla omaavilla toimijoilla ei ole selkeää motiivia käyttää kahta vuotta yksityishenkilön salasanan murtamiseen.

#### **4.4 Miten salausmenetelmiä tulisi muuttaa, jotta salasanojen murto olisi epäkäytännöllistä?**

Salusmenetelmistä ja tiivistealgoritmeista voidaan tehdä monimutkaisempia sekä hitaampia, jotta eri merkkijonoista tiivisteiden laskemiseen kuluu enemmän aikaa ja salasanojen murtaminen hidastuu. Tämä kuitenkin kuluttaisi enemmän resursseja myös palveluissa, joihin salasuilla yritetään kirjautua. Tärkeimpänä, helpoimpana sekä merkittävimpana muutoksena eri palveluiden salasanapolitiikkoja tulisi muuttaa, jotta salasanojen murto olisi epäkäytännöllistä. Jos salasanat pakotettaisiin esimerkiksi 12 merkin pituisiksi, jopa supertietokoneilla olisi käytännössä mahdotonta murtaa eri salasuja.

Kaikkien tulisi myös käyttää salasanamanageria, jolloin käyttäjän tarvitsee muistaa vain yksi salasu, jolla hän voi avata muiden palveluiden käyttäjätunnuksen sekä salasanan sisältävän tietokannan. Salasanamanagerin avulla voitaisiin välttyä samojen salasanojen uudelleenkäytöltä mikä parantaisi myös muiden käyttäjien tietoturvaa, koska salasanalistailla olisi enemmän satunnaisesti luotuja salasuja.

Yksi vaihtoehto tiedon turvaamiselle on monivaiheinen tunnistautuminen, joka yhdessä salasanamanagerin sekä vahvojen salasanojen käytön kanssa tekisi käyttäjien murtamisesta lähes mahdotonta yksittäisiä henkilöitä lukuunottamatta. Vahvan salasanan paljastuessa hyökkääjän pitäisi saada esimerkiksi kohteen matkapuhelin haltuunsa, joka tekisi kiinnijäämisestä paljon todennäköisempää sekä antaisi kohteelle aikaa varautua hyökkäykseen esimerkiksi, jos salasanan vuoto havaitaan.

#### **Oppimiskokemukset**

Ennen tutkimuksen aloittamista minulla oli käsitys millaisia menetelmiä salasanojen salaamiseen käytetään, sekä millaiset salasananluontikäytänteet ovat hyviä. Olen seurannut aihetta joitakin vuosia ja joillakin koulun kursseilla ollaan käyty aihetta läpi. En kuitenkaan aikaisemmin ollut tutustunut salasuohjeistuksien todelliseen paikkansapitävyyteen, vaan toimin varman päälle luodessani salasuja. Kehotin myös tuttaviani käyttämään jopa 20-merkkisiä salasuja, mutta ilman salasanamanagerin käyttöä tällainen on lähestulkoon mahdotonta. Tutkimuksen valmistuttua minulle kuitenkin selvisi, että jo 11-merkkisen salasanan murtamiseen kuluisi supertietokoneilta noin kaksi vuotta, joten esimerkiksi monimutkaisen 12-merkkisen salasanan pitäisi olla turvallinen ainakin lähitulevaisuudessa.



## Lähteet

Bacon, M. 2021. What is password? TechTarget. Luettavissa:

<https://www.techtarget.com/searchsecurity/definition/password>. Luettu: 20.9.2023.

Javatpoint s.a. What is Hashcat. Luettavissa: <https://www.javatpoint.com/what-is-hashcat>. Luettu: 20.10.2023.

Kyberturvallisuuskeskus 14.1.2020. SHA-1-tiivistefunktio on lopullisesti murrettu. Luettavissa: <https://www.kyberturvallisuuskeskus.fi/fi/ajankohtaista/sha-1-tiivistefunktio-lopullisesti-murrettu>. Luettu 20.9.2023.

Kyberturvallisuuskeskus 13.7.2023. Pidempi parempi – Näin teet hyvän salasanan. Luettavissa: <https://www.kyberturvallisuuskeskus.fi/fi/ajankohtaista/ohjeet-ja-oppaat/pidempi-parempi-nain-teet-hyvan-salasanan>. Luettu: 20.9.2023.

LUMI: Euroopan tehokkain supertietokone s.a. CSC. Luettavissa: <https://csc.fi/lumi>. Luettu: 20.9.2023.

Masiliauskas, P. 20.4.2023. Most common passwords: latest 2023 statistics. Cybernews. Luettavissa: <https://cybernews.com/best-password-managers/most-common-passwords/>. Luettu: 20.10.2023.

McCarthy, P. 2.5.2017. Infographic: The growth of computer processing power. OFFGRID. Luettavissa: <https://www.offgridweb.com/preparation/infographic-the-growth-of-computer-processing-power/>. Luettu 5.10.2023.

National Academic Press 2011. The Future of Computing Performance: Game Over or Next Level? Luettavissa: <https://nap.nationalacademies.org/read/12980/chapter/10>. Luettu: 5.10.2023.

NVIDIA 2020. GeForce RTX 3060 Ti Out Now: Faster Than RTX 2080 SUPER, Starting At \$399. Luettavissa: <https://www.nvidia.com/en-us/geforce/news/geforce-rtx-3060-ti-out-december-2/>. Luettu: 20.10.2023.

Odogwu, C. 25.12.2021. What Is a Password Policy and Why Is It Important? Make Use Of. Luettavissa: <https://www.makeuseof.com/what-is-a-password-policy/>. Luettu 5.10.2023.

Schofield, J. 15.12.2009. 32.6m passwords may have been compromised in RockYou hack. The Guardian. Luettavissa: <https://www.theguardian.com/technology/blog/2009/dec/15/rockyou-hacked-passwords>. Luettu 20.10.2023.

Scientific American 21.10.1999. Computers are becoming faster and faster, but their speed is still limited by the physical restrictions of an electron moving through matter. What technologies are emerging to break through this speed barrier? Luettavissa:

<https://www.scientificamerican.com/article/computers-are-becoming-fa/>. Luettu 5.10.2023.

Shankland, S. 4.3.2020. El Capitan supercomputer to blow past rivals, with 2 quintillion calculations per second. CNET. Luettavissa: <https://www.cnet.com/tech/computing/el-capitan-supercomputer-blow-past-rivals-with-2-quintillion-calculations-per-second/>. Luettu: 12.9.2023.

Sherry, Y & Thompson, N. 20.9.2021. IEEE Xplore. Luettavissa: <https://ieeexplore.ieee.org/document/9540991/authors#authors>. Luettu: 5.10.2023.

Strickland, J. s.a. What is computing power? Howstuffworks. Luettavissa: <https://computer.howstuffworks.com/computing-power.htm>. Luettu: 19.10.2023.

Versus s.a. Nvidia GeForce RTX 3060 Ti review: specs and price. Luettavissa: <https://versus.com/en/nvidia-geforce-rtx-3060-ti>. Luettu 20.10.2023.

Zieniūtė, U. 7.6.2022. Mikä on brute force -hyökkäys eli väsytyshyökkäys? NordVPN. Luettavissa <https://nordvpn.com/fi/blog/vasytyshyokkays/>. Luettu: 19.10.2023.

## Liitteet

### Liite 1. Dokumentaatio koejärjestelyn suorituksesta

#### Instructions on how I tested the speed of brute-forcing passwords with Hashcat and how to replicate the research

This research was made on Debian 11. Hardware included NVIDIA GeForce RTX 3060 Ti GPU, Ryzen 5 3600 CPU and Samsung 860 QVO SSD. This hardware has been in use for multiple years, and it might influence the test results.

#### Installing prerequisites

First, I made a fresh installation of Debian 11 on my spare SSD. I considered using Kali Linux or Debian 11 on a virtual machine but decided the results would be more or at least as accurate as if done with a real computer. Instructions for installing Debian 11 on a virtual machine or on hardware can be found From Tero Karvinen's [blog](#).

After operating system installation was ready, I installed a few tools to make the research possible and my job easier. I installed [Hashcat](#) which is self-proclaimed the fastest password cracking tool out there and Micro which is my preferred text editor.

```
$ sudo apt-get update
$ sudo apt-get -y install hashcat micro
```

I used instructions found from [Debian's wiki](#) and installed NVIDIA graphics drivers.

I also needed to install CUDA computing platform on Linux to be able to use GPU for password cracking. I followed two installation guides made by NVIDIA. The installation guides can be found from [here](#) and [here](#).

Commands I used to install CUDA computing platform are listed below and are ONLY a reference for me to use if I ever need to install CUDA again. There might be some commands that are unnecessary and possibly harmful on your system. Explanations for these commands and instructions can be found from the official CUDA installation guides.

```
$ lspci | grep -i nvidia
$ wget
https://developer.download.nvidia.com/compute/cuda/12.3.0/local_installers/cuda-
repo-debian11-12-3-local_12.3.0-545.23.06-1_amd64.deb
$ sudo dpkg -i cuda-repo-debian11-12-3-local_12.3.0-545.23.06-1_amd64.deb
$ sudo cp /var/cuda-repo-debian11-12-3-local/cuda-*-keyring.gpg
/usr/share/keyrings/
$ sudo add-apt-repository contrib
$ sudo apt-get update
$ sudo apt-get -y install cuda-toolkit-12-3
$ sudo apt-get install -y cuda-drivers
$ sudo apt-get install nvidia-libopencl1 ocl-icd-libopencl1
$ sudo apt-get upgrade
$ sudo apt-get -y install cuda-toolkit-12-3
$ sudo apt-get install -y cuda-drivers
```

```
$ uname -m && cat /etc/*release
$ gcc --version
$ uname -r
$ sudo apt-get install linux-headers-5.10.0-16-amd64
$ sudo add-apt-repository contrib
$ wget
https://developer.download.nvidia.com/compute/cuda/repos/debian11/x86_64/cuda-
keyring_1.1-1_all.deb
$ sudo dpkg -i cuda-keyring_1.1-1_all.deb
$ sudo apt-get update
$ sudo apt-get -y install cuda
$ sudo reboot
```

### Creating a dictionary for the attack

Next step was to make a new directory where I can store every file needed for simulating the attack

```
$ cd
$ mkdir research
$ cd research/
```

`Cd` is used to move between directories and by default it goes to your home directory in Linux. Your home directory is basically the same as "C:\Users\user" on Windows. `Mkdir` makes a new directory. With `cd research` I moved to a directory called "research".

I made a new python file "generatePasswordList.py" which I could use to generate a list with every combination of six lowercase alphabets from a-z. I used the code found from [Stackoverflow](https://stackoverflow.com).

```
$ micro generatePasswordList.py
$ cat generatePasswordList.py
import itertools

def foo(i):
    yield from itertools.product(*([i] * 6))

for x in foo('abcdefghijklmnopqrstuvwxyz'):
    print(''.join(x))
```

I chose to use combinations of only six lowercase letters because that filled the requirements to Facebook's account registration and making files with all possible combinations of predefined letters takes a lot of space on a hard drive and this was the easiest method to get reliable results. Also results with more complicated passwords and password policies can be easily calculated very accurately since the speed of brute-forcing is nearly the same in longer periods of time. Time to crack more complicated or longer passwords can be predicted by comparing the amount of all possible combinations for a password fulfilling requirements in the first password policy to the amount of all possible combinations for a password following the rules of the second password policy.

```
$ time python3 generatePasswordList.py > passwordList.txt

real    1m44,425s
user    1m43,241s
sys     0m1,096s
```

The command above ran `generatePasswordList.py` and stored the results - every combination of six lowercase letter combinations - to a file named `passwordList.txt`. The script took one minute and 44 seconds to run.

In Facebook's case password must contain at least six characters. The amount of six lowercase letters (a-z) is  $26 \times 26 \times 26 \times 26 \times 26 \times 26 = 26^6 = 308,915,776$ . This number should be the same as the number of different "words" in `passwordList.txt` and can be confirmed by using 'wc' which stands for word count.

```
$ wc -l passwordList.txt
308915776 passwordList.txt
```

There are more than 300 million different combinations of six lowercase letters and generating that list took almost two minutes, generating all possible seven-character combinations would take 26 times longer and the file would also be 26 times longer and bigger. Brute-forcing a seven-character long password would also take 26 times longer. Math makes testing different password policy cases unnecessary because of how time-consuming cracking passwords can be and how much resources on a computer it uses.

### Creating a "database" to store hashed passwords

Next step was to come up with different six letter strings which represent possible passwords for Facebook users. I used online [password generator](https://passwordsgenerator.net) where I defined passwords to contain exactly six characters and every character should be a lower-cased letter.

https://passwordsgenerator.net

Password Generator Plus<sup>3.0</sup> Old Version

Password Length:

Include Numbers: ☐ ( e.g. 123456 )

Include Lowercase Characters: ☒ ( e.g. abcdefgh )

Include Uppercase Characters: ☐ ( e.g. ABCDEFGH )

Begin With A Letter: ☐ ( don't begin with a number or symbol )

Include Symbols: ☐ [ ! " # \$ % & ' ( ) \* + , - . / : ; < = > ? @ [ \ ^ \_ { | } ~ ]

No Similar Characters: ☐ ( don't use characters like i, l, 1, O, 0, etc. )

No Duplicate Characters: ☐ ( don't use the same character more than once )

No Sequential Characters: ☐ ( don't use sequential characters, e.g. abc, 789 )

Auto Generate On The First Call: ☐ ( generate passwords automatically when you open this page )

Quantity:

Save My Preference: ☐ ( save all the settings above in cookies )

Generate( V2 ) Generate( V1 ) Copy the 1st Line Copy ALL

Your New Passwords:

- dfppck
- tyzpx
- fznvtp
- phqrrf
- zhixln

Usually, passwords are stored as hashes in a database, and they can't be reversed to be plain text. Password cracking is done by hashing potential passwords and comparing the generated hash to

every hash in the database, if a match is found the string that generated said hash is the password. This is also done while logging in to different services. Your input in the password field is run through a hashing algorithm and if it matches the hash stored with the account in the database you are trying to log in to, you will be logged in.

I created a database (a text file on my computer) which stored the generated passwords hashed with md5 hashing algorithm. To come up with this command I found a helpful thread from [AskUbuntu](#).

```
$ printf '%s' "dfppck" | md5sum | cut -d ' ' -f 1 | tee >> hashes.txt.ntds
$ printf '%s' "tyzexp" | md5sum | cut -d ' ' -f 1 | tee >> hashes.txt.ntds
$ printf '%s' "fznvtp" | md5sum | cut -d ' ' -f 1 | tee >> hashes.txt.ntds
$ printf '%s' "phqrrf" | md5sum | cut -d ' ' -f 1 | tee >> hashes.txt.ntds
$ printf '%s' "zhixln" | md5sum | cut -d ' ' -f 1 | tee >> hashes.txt.ntds
```

After creating the database called "hashes.txt.ntds" I printed the contents (hashes) to the terminal with `cat`. The hashes are in the same order as the passwords have been hashed, one password or hash per row (dfppck is the first hash, zhixln is the fifth hash).

```
$ cat hashes.txt.ntds
e6a85b22dfd21b9c0eed5591abf470bf
3a80eb627647123cb8c5036652ac5923
c11d2d77801b790fc1cf62b9d9833df6
d82f08920c21a0881c1e8fd2663b3fd3
a9c1e5c83902588a1e3b3f0caff77c63
```

### Brute-force attack

The passwords in the database were ready to be cracked with Hashcat. To be exact, the attack I used is called a dictionary attack but because the word list I used contains every combination of six lowercase letters, it does the same job as an actual brute-force attack would. At the time of simulating the attack, I couldn't figure out how to use the brute-force attack-mode with Hashcat. I used Tero Karvinen's [article](#) and Infinite Logins' YouTube [video](#) as references on how to use Hashcat.

```
$ hashcat -m 0 hashes.txt.ntds passwordList.txt -o solved.txt
...
Started: Fri Oct 20 03:28:43 2023
Stopped: Fri Oct 20 03:28:59 2023
```

`Hashca` is used to start the program, `-m 0` is the mode it tries to crack the passwords with, in this case it is md5, `hashes.txt.ntds` is the database, `passwordList.txt` is the dictionary where Hashcat gets the strings that will be hashed and compared to the database, `-o solved.txt` outputs the cracked passwords to a text file called "solved.txt" in the order they are cracked. The other rows are taken from the bottom of the output of Hashcat and they are the only other necessary rows for this research because they represent how long it took to crack those passwords, 16 seconds.

I used `cat` to print the contents of "solved.txt" to the terminal and from the output we can see they are cracked in the alphabetical order. This made me wonder how much the results would lie because Hashcat didn't need to go through all the combinations before it cracked all the passwords in my database.

```
$ cat solved.txt
e6a85b22dfd21b9c0eed5591abf470bf:dfppck
c11d2d77801b790fc1cf62b9d9833df6:fznvtp
```

```
d82f08920c21a0881c1e8fd2663b3fd3:phqrrf
3a80eb627647123cb8c5036652ac5923:tyzepx
a9c1e5c83902588a1e3b3f0caff77c63:zhixln
```

I decided to create hashes for "aaaaaa" and "zzzzzz" because they are the first and the last possible passwords with the requirements used in this research in alphabetical order.

### Testing the accuracy of the results

This time I didn't store the hashes in a file but printed them to the terminal and used the printed hashes themselves instead of a database file in Hashcat to be cracked.

```
$ printf '%s' "aaaaaa" | md5sum | cut -d ' ' -f 1
0b4e7a0e5fe84ad35fb5f95b9ceeac79

$ hashcat -m 0 '0b4e7a0e5fe84ad35fb5f95b9ceeac79' passwordList.txt -o solved.txt
...
Started: Fri Oct 20 03:33:11 2023
Stopped: Fri Oct 20 03:33:12 2023
```

Cracking "aaaaaa" was instantaneous even though the output says it took a second. This was expected because it was the first string in "passwordList.txt".

```
$ printf '%s' "zzzzzz" | md5sum | cut -d ' ' -f 1
453e41d218e071ccfb2d1c99ce23906a

$ hashcat -m 0 '453e41d218e071ccfb2d1c99ce23906a' passwordList.txt -o solved.txt
...
Started: Fri Oct 20 03:33:31 2023
Stopped: Fri Oct 20 03:33:49 2023
```

Cracking "zzzzzz" took 18 seconds and in that time Hashcat tried over 308 million different passwords before it cracked it. Hashcat tried 308,915,776 different passwords in 18 (or under) which equals to about 17 million passwords every second.

### Sources:

<https://askubuntu.com/questions/53846/how-to-get-the-md5-hash-of-a-string-directly-in-the-terminal>

[https://developer.nvidia.com/cuda-downloads?target\\_os=Linux&target\\_arch=x86\\_64&Distribution=Debian&target\\_version=11&target\\_type=deb\\_local](https://developer.nvidia.com/cuda-downloads?target_os=Linux&target_arch=x86_64&Distribution=Debian&target_version=11&target_type=deb_local)

<https://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html#debian>

<https://hashcat.net/hashcat/>

<https://passwordsgenerator.net/>

<https://stackoverflow.com/questions/45990454/generating-all-possible-combinations-of-characters-in-a-string>

<https://terokarvinen.com/2021/install-debian-on-virtualbox/?fromSearch=debian>

<https://terokarvinen.com/2022/cracking-passwords-with-hashcat/>

<https://terokarvinen.com/2023/eettinen-hakkerointi-2023/>

<https://wiki.debian.org/NvidiaGraphicsDrivers>

<https://www.youtube.com/watch?v=Ndm5t4sy8o0>