



Kandidatuppsats

IT-forensik och informationssäkerhet, 180 HP

Hur långt lösenord behöver du för att vara säker?

- Undersökning av effektiviteten hos password cracking-program

Digital forensik, 15 HP

Halmstad 2025-06-13

Filip von Platen Orwén, Martin Larsson

Hur långt lösenord behöver du för att vara säker?

- Undersökning av effektiviteten hos password cracking-program

Filip von Platen Orwén

filorw20@student.hh.se

Martin Larsson

martil21@student.hh.se

Handledare: Eric Järpe

eric.jarpe@hh.se

Examinator: Urban Bilstrup

urban.bilstrup@hh.se

Akademien för informationsteknologi (ITE)

Högskolan i Halmstad

Sammanfattning

Lösenord är idag den primära lösningen för cybersäkerhet kopplat till digitala konton på internet. Med en sådan dominerande roll är det viktigare än någonsin att användare väljer säkra lösenord för att motverka dataintrång och identitetskapning. Samtidigt kan lösenord även vara sårbara mot modern programvara som kan användas för att få fram lösenord, så kallade password cracking-program, vilka idag är väldigt kraftfulla och sofistikerade. Säkerheten hos ett lösenord står ofta i kontrast med dess användarvänlighet, då användaren måste komma ihåg invecklade lösenord om denne inte använder sig av lösenordshanterare. Denna studie ger, med stöd av undersökningar av idag frekvent använda password crackning-program, rekommendationer och vägledning i exakt hur avancerade lösenord en användare behöver ha för att de ska vara resistenta mot moderna password cracking-program. Studien har genomfört ett antal experiment för att testa prestandan hos password cracking-programmen *Hashcat* och *John the Ripper* och hur de påverkas av lösenords längd och komplexitet. Även hur hashning och saltning av lösenorden påverkar programmets prestanda granskades. Tillsammans med existerande forskning ligger experimentens resultat till grund för förbättrade rekommendationer kring val av säkra lösenord.

Nyckelord:

Lösenordsäkerhet, Password cracking, Hashning och saltning, Hashcat, John the Ripper

Förord

Vi vill rikta ett stort tack till vår handledare, Eric Järpe, för värdefull vägledning och konstruktiv kritik genom hela processen.

Tack även till Anas Ayman Abdl Muti och Carlos Karat för era värdefulla åsikter om vårt arbete från alla opponeringar vi haft sedan start.

Innehållsförteckning

1	Introduktion	1
1.1	Bakgrund.....	2
1.2	Problembeskrivning	3
1.3	Syfte	3
1.4	Frågeställningar	3
1.4.1	Problematisering av frågeställningar.....	4
1.5	Avgränsningar.....	4
2	Metod.....	7
2.1	Teoretisk bakgrund	7
2.2	Experimentuppställning	8
2.3	Datainsamling	12
2.4	Verktyg och programvara	13
2.5	Positionering av metoden.....	14
2.6	Problematisering av metoden.....	15
2.7	Förtjänst	16
2.8	Etiska överväganden	16
3	Resultat	19
3.1	Hashcat.....	20
3.1.1	Hybridattacker.....	20
	Hybridattacker – Siffror.....	21
	Hybridattacker – Specialtecken	23
3.1.2	Ordboksattacker	26
3.2	John the Ripper	27
3.2.1	Hybridattacker.....	27
	Hybridattacker – Siffror.....	27
	Hybridattacker – Specialtecken	29
3.2.2	Ordboksattacker	31
4	Diskussion	33
4.1	Hur påverkar lösenordets längd dess motståndskraft?.....	33
4.2	Hur påverkar lösenordets komplexitet motståndskraften?.....	35
4.3	Hur påverkar hashing och saltning lösenordets resistens?.....	36

4.4 Hur balanseras användarvänlighet och säkerhet vid skapandet av säkra lösenord?	38
4.5 Hur långt och komplext behöver ett lösenord vara, och vilka faktorer påverkar dess säkerhet?	39
5 Slutsatser.....	43
5.1 Framtida arbeten	43
6 Referenser	45

1 Introduktion

Är ditt lösenord verkligen helt säkert mot angripare?

Lösenord är idag en etablerad lösning för autentisering och identifiering på nätet. I stort sett alla tjänster där en användare kan registrera ett konto så hänvisas denne till att välja ett lösenord för att begränsa åtkomsten till kontot, och utgör därav den huvudsakliga säkerhetsaspekten. Även när ytterligare metoder som tvåfaktorsautentisering och sensorer som läser av biometriska egenskaper på en människa, som fingeravtryck eller ansiktsgenkänning, har tillkommit så förblir användandet av lösenord fortfarande det mest dominerande för att upprätthålla säkerheten för digitala konton (Kanta m.fl., 2023) (Kanta m.fl., 2021) (Zimmermann & Gerber, 2020). Med en sådan viktig roll är det nödvändigt att användare är noggranna med hur de väljer sina lösenord. Vanliga rekommendationer är att välja långa, unika lösenord och helst med speciella tecken. Det uppmanas även att använda sig av lösenordshanterare och lösenordsgeneratorer för att få fram starka lösenord. Utöver det så ska användare undvika att använda enklare och vanligt förekommande ord och fraser vilka är lätta att gissa, exempelvis "12345" eller "password", som lösenord och att aldrig återanvända lösenord till flera konton eller tjänster (Šimonélytė, 2023) ("Har mitt lösenord läckt", n.d.). Vissa lösenordshanteringsprogram har även funktioner som meddelar användaren när dess användaruppgifter har påträffats i en databasläcka, så att den drabbade användaren kan välja ett nytt lösenord och undvika att någon obehörig får tillgång till ens konto (Google, n.d.).

Följs inte dessa rekommendationer kan det leda till allvarliga konsekvenser. Tidigare forskning visar att svaga och återanvända lösenord ofta spelar en avgörande roll till att angripare får åtkomst till digitala konton. En empirisk studie av Weber m.fl. (2008) visar på det faktum att användare ofta väljer lösenord som är lätta att gissa, vilket medför signifikanta säkerhetsrisker. Även Güven m.fl. (2022) understryker problemet med svaga lösenord och visar, genom en omfattande analys av verkliga dataintrång, att förutsägbara mönster och återanvändning av lösenord är vanligt förekommande. Detta beteende ökar risken för att angripare ska lyckas knäcka lösenorden och få tillgång till känslig information, vilket belyser ett behov av både tekniska åtgärder och förändrat användarbeteende.

Problemet med sårbara lösenord sträcker sig dock längre än det rent tekniska, då det har tydliga juridiska konsekvenser. Europeiska dataskyddsförordningen (GDPR) kräver att organisationer vidtar lämpliga säkerhetsåtgärder. Enligt Artikel 83 i GDPR kan intrång till följd av brutna lösenord som exponerar personuppgifter kan leda till sanktionsavgifter (EU 2016/679).

1.1 Bakgrund

Lösenord har använts som en metod för autentisering och identifiering i tusentals år. Ett av de tidigaste exemplen kommer från Bibelns Domarbok (12:6), där ordet "shibboleth" användes för att avslöja fiender från Efraims stam. Deras annorlunda uttal av ordet avslöjade deras identitet ("Domarboken 12", 2017). Händelsen tros ha inträffat under den så kallade Domartiden, cirka 1400–1100 f.Kr., och betraktas som ett av de första dokumenterade fallen av lösenordsanvändning i västerländsk historia (Lennon, 2015).

I det antika Rom användes lösenord på ett systematiskt sätt inom militären (Mani & Adedayo, 2024). Romerska soldater använde så kallade tesseræ, små trä- eller lertavlor med lösenord eller order ingraverat. Dessa delades ut varje kväll av särskilda officerare, och varje vaktenhet signerade tavlan för att bekräfta mottagandet. Tesseræ användes inte bara för att sprida lösenord, utan även som ett sätt att kommunicera viktiga militära instruktioner och säkerställa att endast behörig personal hade tillgång till specifika områden. Denna form av lösenordsanvändning förekom i den romerska armén under perioden cirka 100 f.Kr. till 400 e.Kr., då arméns organisation var som mest utvecklad (Wasson, 2021) (Schmitt, 2024).

Lösenord förekommer även i berättelser och folktro. I sagan Ali Baba och de fyrtio rövare, populariserad av Antoine Galland på 1700-talet, används frasen "Sesam, öppna dig" för att komma åt en dold skatt i en grotta (Chraïbi, 2004) (Waksman, 2013). Historien illustrerar vikten av att minnas ett lösenord. När Ali Babas bror glömmer frasen blir han inlåst och inte lyckas ta sig ut.

Under 1920-talets förbudstid i USA blev lösenord åter aktuella då illegala barer, så kallade svartklubbar, använde kodord för att hålla verksamheten hemlig från myndigheterna ("Passwords have a long history", n.d.) ("The Speakeasies of the 1920s", n.d.).

Även militära styrkor har fortsatt att använda lösenord. Under andra världskriget, vid D-dagen år 1944, använde amerikanska fallskärmstrupper ett så kallat "challenge and response-system" för att identifiera vän från fiende. En soldat ropade då "Flash", och det korrekta svaret var "Thunder", vilket bekräftade att personen tillhörde de allierade. Kodorden byttes regelbundet för att hålla dem säkra ("Passwords have a long history n.d.).

När fler användarsystem började utvecklas på 1960-talet, introducerade Fernando Corbató vid MIT ett av de första datorbaserade lösenordssystemen. I det så kallade Compatible Time-Sharing System (CTSS) tilldelades varje användare ett eget lösenord för att skydda sina filer och resurser. I det delade datorsystemet tilldelades varje användare ett eget lösenord för att skydda sina filer. Dock ledde detta snabbt till den första dokumenterade "lösenordshackningen", då forskaren Allan Scherr lyckades få tillgång till andra användares lösenord för att utöka sin datortid. Händelsen visade redan då hur sårbara digitala autentiseringssystem kunde vara om inte skyddsåtgärder vidtogs ("Passwords have a long history n.d.).

Från dessa militära och historiska exempel har lösenord fortsatt vara en central del av autentisering. I takt med att datorer och digitala system blivit en självklar del av vardagen, har lösenord övergått till att bli första försvarslinjen för användares digitala identitet. Men i dagens digitala miljö utmanas lösenordssystem ständigt av kraftfulla attacker och programvara som utnyttjar svagheter i både teknik och användarbeteende. Program som *Hashcat* och *John the Ripper* kan snabbt knäcka lösenord, särskilt om de är enkla, återanvända eller saknar tillräcklig längd och variation. För att förstå hur ett lösenord ska utformas för att stå emot dessa verktyg krävs en fördjupad analys av både programmets kapacitet och de tekniska mekanismer så som hashning och saltning, som används för att skydda lösenord.

1.2 Problembeskrivning

Att använda långa och unika lösenord gör det svårare för en angripare att kunna gissa rätt lösenord, men även att kunna komma åt ett lösenord med hjälp av password cracking-program (Kanta m.fl., 2023). Denna studies fokus ligger på att undersöka hur effektiva dessa verktyg är mot lösenord som följer vedertagna säkerhetsrekommendationer. Genom att undersöka detta ger denna studie ytterligare rekommendationer och vägledning i exakt hur avancerade lösenord en användare behöver ha för att fortfarande vara säker. Säkerhet står ofta i kontrast till flexibilitet och användarvänlighet. Därför finns det ett tydligt behov av att etablera en nivå för hur komplexa lösenord en genomsnittlig användare behöver ha, och fortfarande kunna ha ett lösenord som går att både komma ihåg och enkelt kan skriva in vid inloggning.

1.3 Syfte

I takt med utvecklingen av cybersäkerhetshot och framsteg inom password cracking-metoder, har frågor kring lösenordssäkerhet blivit alltmer aktuella (Alkhwaja m.fl., 2023). Syftet med denna studie var att undersöka hur långt och komplext ett lösenord behöver vara för att stå emot moderna password cracking-metoder, samt vilka faktorer som påverkar lösenordets säkerhet. Genom att analysera hur lösenordets längd, komplexitet, hashning och saltning påverkar motståndskraften mot attacker, låg fokus på att förstå de mekanismer och mönster som gör vissa lösenord mer sårbara än andra. Målet med studien var att hitta en balans mellan användarvänlighet och säkerhet för lösenorden.

1.4 Frågeställningar

Huvudfråga

Hur långt och komplext behöver ett lösenord vara för att motstå moderna password cracking-metoder, och vilka faktorer påverkar dess säkerhet?

Delfrågor

1. Hur påverkar lösenordets längd dess motståndskraft mot moderna password cracking-metoder?
2. Hur påverkar lösenordets komplexitet dess motståndskraft mot moderna password cracking-metoder?
3. Hur påverkar hashingalgoritmer och saltning lösenordets motståndskraft mot password cracking-metoder?
4. Hur balanserar man användarvänlighet och säkerhet vid skapandet av säkra lösenord?

1.4.1 Problematisering av frågeställningar

Studiens huvudfråga hade som syfte att täcka in både de bakomliggande faktorerna kring vad som utgör ett säkert lösenord, och även formulera en mer konkret fråga gällande vad som är ett säkert lösenord, vilken skulle ge ett tydligt och användbart svar.

Huvudfrågeställningen delades upp i delfrågor för att dels göra huvudfrågan enklare att besvara, men också för att kunna redogöra mer detaljerat för de olika aspekterna som täcks av huvudfrågan.

Även om delfrågorna hade som syfte att göra avgränsningar i vad som togs upp så fanns risken att svaren skulle bli alltför omfattande. Framför allt den sista frågan berörde breda forskningsområden, vilket kunde göra den krångligare att besvara inom ramen för studien. Samtidigt var huvudfrågeställningen mångfasetterad och tog hänsyn till en rad olika faktorer för att den skulle ge ett välgrundat och användbart svar. Därför blev några delfrågor bredare än andra.

Det grundläggande resonemanget för arbetet var att lösenordssäkerheten baseras på lösenordets motståndskraft mot password cracking-program (Kanta m.fl., 2023). Därför låg fokus i denna studie på just hur prestandan för sådana program påverkades av hur väl lösenorden strukturerats utifrån säkerhetsrekommendationer. Utifrån prestandan kunde sedan slutsatser dras, vilka sedan användes för att ge svar på huvudfrågeställningen och dess delfrågor.

1.5 Avgränsningar

Nämnda password cracking-program som *Hashcat* och *John the Ripper* var inte de enda existerande verktygen som används vid knäckning av lösenord. Liknande program som *Cain and Abel*, *Ophcrack*, *Brutus* och *THC Hydra* fanns också tillgängliga. Denna studie avgränsades dock till att endast undersöka *John the Ripper* och *Hashcat* eftersom tidsbegränsningen för arbetet endast gjorde det möjligt att undersöka ett fåtal program. Dessutom var *Hashcat* och *John the Ripper* de mest frekvent förekommande programmen

som nämndes när de bästa och mest använda password cracking-programmen efterfrågades på internet.

John the Ripper fanns tillgängligt i en gratisversion och en premiumversion som kostade pengar. Detta arbete använde sig av gratisversionen.

Hashningalgoritmer som användes vid kryptering av lösenorden begränsades till bcrypt, osaltad SHA-256 och saltad SHA-256 i denna studie. Dessa valdes på grund av dess breda användning vid kryptering av lösenord (Al-Aboosi m.fl., 2022) (Yao m.fl., 2024) (Mellberg Granat & Gustavsson, 2017).

För arbetets experiment sattes en tidsbegränsning för hur länge programmen fick genomföra en attack. Närmare bestämt begränsades varje attack till att få pågå i maximalt en timmes tid. Detta gjordes mestadels på grund av den begränsade tiden till att genomföra experimentet och arbetets ramar. Samtidigt var även syftet att undersöka hur mycket en angripare kan åstadkomma med de undersökta programmen under en begränsad tid med en vanlig dator. På så sätt är studiens resultat applicerbara i en bredare kontext.

2 Metod

I denna studie användes en kombination av experimentell analys och litteraturoversikt för att undersöka hur moderna password cracking-verktyg hanterade olika typer av lösenord. Metoden omfattar en teoretisk bakgrund, experimentuppställning, datainsamling, verktyg och programvara, positionering, problematisering, förtjänst och etiska överväganden.

2.1 Teoretisk bakgrund

Idag är lösenordsbaserad autentisering den vanligaste metoden för att skydda känslig information och identifiera användare i moderna datasystem (Shouling Ji m.fl., 2017) (Gafni m.fl., 2017). Trots detta kvarstår stora sårbarheter i lösenordssystem, där dataläckor och framsteg inom password cracking-tekniker visar att många lösenord är lätta att knäcka. Verktyg som *John the Ripper* och *Hashcat* kan snabbt analysera och knäcka lösenord som är korta, förutsägbara eller saknar komplexitet (Shi m.fl., 2021) (Shouling Ji m.fl., 2017). Dessa utmaningar visar på ett behov av att förstå vilka faktorer som bidrar till ett säkert lösenord och hur moderna cracking-verktyg påverkas av olika säkerhetsåtgärder.

Forskning visar att majoriteten av användare väljer lösenord baserat på hur lätta de är att komma ihåg och inte baserat på hur säkert lösenordet är (Weber m.fl., 2008). Användning av personlig information som t.ex. födelsedatum i lösenord försvagar lösenordet ytterligare då det blir lättare för en angripare att knäcka (Gafni m.fl., 2017).

Tydligt är också att hashingalgoritmer och saltning av hashvärden spelar en avgörande roll i att skydda lösenord. Vid korrekt implementering av hashingalgoritmer samt saltning av de så ökar säkerheten för lösenordssystem avsevärt genom att göra det svårare för angripare att använda sig av offline-attacker för att knäcka lösenord (Shi m.fl., 2021) (Weber m.fl., 2008). Felaktig eller saknad av implementering av saltning kan leda till stora säkerhetsproblem (Tatlı, 2015).

Hashingalgoritmer är matematiska funktioner som genererar ett hashvärde för ett en textsträng (Hornby, 2021). Vid förvaring av lösenord hashas de med hashingalgoritmer. Att förvara lösenord i klartext betecknas inom IT-säkerhetsbranschen som en dödssynd eftersom de då blir osäkra om en lösenordsdatabas skulle läcka och publiceras på det publika internet. Om lösenorden har förvarats hashade är det fortfarande svårt för en angripare att få fram rätt lösenord, då de först måste kringgå hashningen av lösenordet. (ScienceDirect, 2016).

De flesta password cracking-program, inklusive *Hashcat* och *John the Ripper*, erbjuder möjligheten genomföra så kallade "brute-force-attacker". I en sådan attack utnyttjar programmen en mekanism som vanligtvis används av inloggningssidor när användare besöker att skriva in sitt lösenord. Eftersom lösenord inte lagras i klartext tar inloggningssidan det av användaren inskrivna lösenordet, hashar det och jämför hashvärdet med hashvärdet på

det lösenord som finns lagrat i användarens uppgifter. Matchar hashvärdet på det inskrivna lösenordet med det lagrade så indikerar det att användaren har angett rätt lösenord och ges åtkomst till sitt konto. Att göra så här fungerar eftersom en hashingalgoritm alltid genererar ett unikt hashvärde för en textsträng. Undantaget för denna mekanism är att det finns en risk att samma hashvärde ges till två olika textsträngar, vilket kallas kollision (ScienceDirect, 2016). Sannolikheten för kollision är däremot extremt liten hos välanvända hashingalgoritmer som bcrypt och SHA-256 (Yao m.fl., 2024).

När ett password cracking-program genomför en brute-force-attack ges det ett eller flera hashvärden. Därefter genererar det gissningar av lösenord som sedan hashas. Hashvärdet för lösenordsgissningarna jämförs sedan med hashvärdet för lösenordshasharna som programmet instruerades att lösa. Finner programmet att en gissning matchar ett givet hashvärde innebär det att programmet fått fram korrekt lösenord (ScienceDirect, 2016).

Hashvärden som genereras av hashingalgoritmerna bcrypt och SHA-256 har alltid en bestämd mängd tecken, de är deterministiska. Oavsett hur lång den ursprungliga textsträngen var så kommer alla hashvärden ha samma antal tecken. När ett password cracking-program försöker knäcka ett hashat lösenord genom brute-force går det igenom och jämför en rad förslag, som börjar med ett litet antal tecken som succesivt ökar. Således är det inte primärt själva hashingalgoritmen som avgör säkerheten hos ett hashat lösenord, utan även hur många tecken lösenordet består av spelar en avgörande roll (ScienceDirect, 2016).

Ett sätt att åtgärda problemet med deterministiska hashvärden är att lägga till en slumpmässig mängd tecken, kallat "salt", i hashvärdet. Detta kallas saltning och gör att varje hashvärde blir unikt, även om det utgår från samma ursprungliga textsträng (Marchetti & Bodily, 2022).

Utöver tekniska aspekter som hashning och saltning påverkar även mänskliga faktorer lösenordens säkerhet. Forskning visar att användare ofta prioriterar enkelhet och återanvänder lösenord eller väljer förutsägbara lösenord baserade på personlig information, vilket ökar sårbarheten för attacker (Gafni m.fl., 2017) (Weber m.fl., 2008). Dessa faktorer gör att balansen mellan säkerhet och användarvänlighet blir en avgörande utmaning inom lösenordsskydd.

2.2 Experimentuppställning

Studien bestod av en experimentell analys av password cracking-verktyg som *Hashcat* och *John the Ripper*, där data om vilka lösenord som programmen hade lätt att knäcka, samt vilka de hade svårt att knäcka, analyserades. Först testades hur framgångsrikt dessa verktyg kunde knäcka lösenord med olika längd, och som använde sig av både siffror och specialtecken. Analysen av saltning och hashning utfördes genom att lösenord hashades med några av de vanligaste algoritmerna så som SHA-256 (både saltad och osaltad) samt

bcrypt (Al-Aboosi m.fl., 2022) (Yao m.fl., 2024) (Mellberg Granat & Gustavsson, 2017). För lösenorden som hashades med bcrypt saltades lösenorden automatiskt av algoritmen, medan för SHA-256 behövde lösenorden saltas manuellt innan de hashades (Eum m.fl., 2023) (Grigutyte, 2023). Därefter genomfördes försök att knäcka de hashade lösenorden med hjälp av password cracking-verktygen, i syfte att undersöka hur effektiv respektive hashingalgoritm var. Resultaten från de hashade lösenorden jämfördes sedan med resultaten för lösenord som även saltats, för att ge en inblick i hur saltning påverkade säkerheten. Olika password cracking-verktyg användes för att analysera hur snabbt olika typer av lösenord kunde knäckas.

Inför experimentet gjordes en kategorisering av de undersökta lösenord som användes. För att kunna avgöra lösenordens kvalitet delades de in i tre kategorier, här kallat nivåer. Alla tre nivåer finns visualiserade i Tabell 1. Nedan följer en genomgående beskrivning av respektive nivå.

”Nivå 1 – Osäkert” blev benämningen för den första kategorin, där lösenord som inte följde några säkerhetskriterier placerades in. Lösenorden i denna kategori skulle även följa allmänt associerade riskfaktorer såsom att endast använda små bokstäver och ingen användning av varken specialtecken, siffror eller ovanliga ord eller fraser. Orden som användes i lösenordet skulle även ha personlig koppling till användaren. Slutligen fick lösenorden i denna kategori inte bestå av fler än åtta tecken.

Mittenkategorin, ”Nivå 2 – Säkrare” representerade lösenord som skapats av användare med större medvetenhet kring säkerhet jämfört med Nivå 1. Här skulle lösenorden uppfylla någon form av säkerhetsrekommendation. Detta kunde vara att användaren bytt ut en bokstav mot en siffra eller ett specialtecken, använt sig av siffror eller specialtecken eller ha använt stor bokstav någonstans i lösenordet. Gällande teckenlängd sattes en gräns från åtta till tolv tecken för denna kategori.

Den högsta nivån, ”Nivå 3 – Säkert”, innefattade de lösenord som följde flera säkerhetsrekommendationer, och därför kunde klassificeras som de säkraste lösenorden. I Nivå 3 skulle lösenorden innehålla flera siffror eller specialtecken, bestå av både stora och små bokstäver och bokstäver skulle ha ersatts med specialtecken eller siffror. Innefattade lösenord skulle även bestå av ord och ordkombinationer som inte tydligt kunde återfinnas i lösenordslistor, främst den Rockyou-lista som användes i experimentet. Här inkluderas även lösenord som bar tydliga spår av att vara genererade av lösenordsgeneratorer, dock var det inte ett krav att alla lösenord i denna kategori behövde vara genererade på detta sätt.

Teckenlängden för varje nivå baserades på ett flertal källor som utgjorde grunden för arbetet. I en informationssida från NordVPN lägger Šimonelyte (2023) fram att användare inte bör välja lösenord som är kortare än tio tecken för att det ska vara säkert. Zimmermann & Gerber (2020) rekommenderade i en artikel publicerad i den vetenskapliga tidskriften ”International Journal of Human-Computer Studies” att säkra lösenord inte borde vara kortare än åtta tecken långt. Denna rekommendation hänvisade de i sin tur till riktlinjer från National Institute of Standards and Technology (NIST). Ytterligare en källa, en

informationssida från Darktrace (n.d.) hävdade att ett säkert lösenord skulle omfatta åtminstone åtta tecken. Alla dessa vägdes samman för att avgöra en lämplig teckenlängd för respektive nivå.

Tabell 1: Lista över studiens tre kategorier för lösenordsklassning och deras respektive kriterier.

<p>Nivå 1 – Osäkert</p> <ul style="list-style-type: none">• Endast små bokstäver• Inga specialtecken, siffror eller ovanliga ord eller fraser• Ord med personlig koppling till användaren• Upp till åtta tecken långt
<p>Nivå 2 – Säkrare</p> <ul style="list-style-type: none">• Följer någon säkerhetsrekommendation<ul style="list-style-type: none">○ Byter ut en bokstav mot en siffra eller ett specialtecken○ Läger in stor bokstav någonstans• Från åtta till tolv tecken långt
<p>Nivå 3 – Säkert</p> <ul style="list-style-type: none">• Använder flera säkerhetsrekommendationer<ul style="list-style-type: none">○ Använder sig av flera siffror eller specialtecken○ Kombination av stora och små bokstäver○ Byter ut bokstäver mot specialtecken eller siffror○ Innehåller ovanliga ord eller ovanliga kombinationer av ord○ Vara tydligt genererat av en lösenordshanterare• Minst tretton tecken långt

I båda programmen kördes samma sorters attacker. De utvalda attackerna som genomfördes i denna studies experiment var ordboksattack (dictionary attack) och hybridattack (hybrid attack). I experimentet gjordes båda sorters attacker mot en lösenordsdatabas vilken hade hashats med någon av tre ovannämnda hashingalgoritmer. Båda programmen ställdes in att utgå från databasfilen RockYou.txt som redan fanns förinstallerad i Kali Linux. De konfigurerades även till att köras utifrån samma parametrar gällande siffror och specialtecken. Hybridattackerna kördes i två olika former. En där programmen instruerades att lägga till siffror i knäckningsprocessen, och en där specialtecken lades till. För dessa hybridattacker startade antalet specificerade siffror eller specialtecken på ett, och stegrades sedan succesivt tills maxgränsen på fem tecken.

Motiveringen bakom valet av just ordboksattack och hybridattack var främst att dessa vanligtvis går snabbt att genomföra, framförallt jämfört med en brute-force-attack. Precis som för en verklig angripare var utgångspunkten i denna studie att få fram så många

lösenord som möjligt på kortast möjliga tid. Därför uteslöts att genomföra brute-force-attacker. Istället togs beslutet att ordboksattacker och hybridattacker skulle utföras.

I en ordboksattack går password cracking-programmet igenom en given lista över kända lösenord och söker efter matchningar mellan den givna listan och listan av okända lösenord den blivit tilldelad att knäcka. Främst är denna sorts attack effektiv på lösenord som följer förutsägbara mönster eller består av vanligt förekommande ord och fraser (Alkhwaja m.fl., 2023). En hybridattack är en sorts kombinationsattack där angriparen använder sig av flera olika attackmetoder som kompletterar varandra till en större attack. Vanligtvis är det en ordboksattack som kombineras med någon form av brute-force-attack (Hashcat.net, n.d.b). I denna studie bestod hybridattackerna av en ordboksattack som kombinerades med en maskattack, vilket är en sofistikerad form av brute-force där password cracking-programmet instrueras att lägga till specificerade tecken i sina försök att hitta rätt lösenord (Hashcat.net, n.d.c). När hybridattackerna genomfördes i denna studies experiment kördes tio olika exekveringar per hashingalgoritm. Fem där programmen instruerades att lägga till ett suffix på en till fem siffror och fem där programmet instruerades att lägga till ett suffix på en till fem specialtecken.

Till följd av tidsbegränsningen för detta arbete togs beslutet att låta varje experimentomgång ta maximalt en timme. Efter att en timme från start hade passerat avbröts körningen av programmet och dess resultat plockades ut, oavsett hur långt programmet hade kommit i exekveringen. Ingen extra omgång gjordes om programmen blev klara med en exekvering innan en timme hade passerat, utan resultaten plockades ur och körningen räknades som avslutad.

Som utgångspunkt för programmen används databasfilen RockYou.txt. Anledningen till valet av just RockYou.txt är att det är en etablerad lista över lösenord som väldigt ofta används vid tester för cybersäkerhet. Den ursprungliga versionen av RockYou-filen innehåller över 32 000 000 lösenord, och på grund av dess ursprung är många av lösenorden i filen representativa för hur genomsnittliga användare väljer lösenord än idag (Veras m.fl., 2021) (Lurey, 2023). Just RockYou-filen som fanns förinstallerad i Kali Linux innehöll ungefär 14 000 000 lösenord, så den var en förkortad version av ursprungsfilen. På den gjordes ett slumpmässigt urval på 100 000 lösenord för denna studie.

Utöver databasfilen RockYou, som användes för ordlistattacker, användes även en lösenordslista från projektet SecLists för att säkerställa att experimentet utgick från vanligt använda lösenord. SecLists är en etablerad och fritt tillgänglig samling ordlistor som används inom cybersäkerhet, bland annat vid penetrationstester och lösenordsanalyser. Samlingen innehåller bland annat lösenord, användarnamn, URL:er, känsliga datamodeller och andra säkerhetsrelaterade listor (Wilson, 2025). I detta arbete begränsades användningen strikt till endast lösenordslistor utan någon koppling till någon personlig information.

Utifrån lösenordslistan i SecLists som innehöll 10 000 000 lösenord så utfördes ett slumpmässigt urval på vilka 100 000 lösenord som användes i experimentet. Det

slumpmässiga urvalet genomfördes med terminalen i Linux som slumpmässigt plockade 100 000 lösenord och skrev de till en fil. Denna fil användes sedan i experimentet.

Utöver de praktiska experimenten studerades även redan existerande litteratur som täckte lösenordssäkerhet och cracking-tekniker. Till slut vägdes resultaten från experimenten ihop med existerande litteratur för att kunna ge välgrundade slutsatser.

Experimentets resultat redovisas med hjälp av tabeller och diagram i arbetets resultatkapitel. Baserat på resultaten har sedan osäkerheten för resultaten beräknats. Mer exakt har standardfel beräknats, och detta har sedan använts för att visualisera resultatens osäkerhet i form av felstaplar (error bars) i diagrammen i resultatkapitlet. För att beräkna felstaplarna användes ett konfidensintervall på 95%.

För beräkning av standardfel (SE) har följande formler använts:

$$\hat{\pi}_i = \frac{n_i}{N}$$
$$SE_i = \sqrt{\frac{\hat{\pi}_i(1 - \hat{\pi}_i)}{N}}$$

N representerar i detta fall summan av alla 100 000 lösenord som fanns i den ursprungliga lösenordsdatabasen som sedan hashades med olika hashingalgoritmer.

n_i är antalet knäckta lösenord från en specifik attack.

$\hat{\pi}_i$ är här andelen knäckta lösenord, av totala antalet lösenord i databasen, från en specifik attack.

För beräkning av konfidensintervallet (CI) för felstaplarna (error bars) användes nedanstående formel:

$$CI = \hat{\pi}_i \pm 1.96 SE_i$$

2.3 Datainsamling

Datainsamlingen i denna studie utfördes genom en kombination av praktiska experiment och analys av befintlig litteratur. Praktiska data genererades genom att skapa en lösenordsdatabas bestående av 100 000 lösenord. Denna databas hashades sedan med var och en av hashingalgoritmerna. Dessa lösenord konstruerades för att uppvisa varierande komplexitet, med syftet att analysera hur olika password cracking-verktyg hanterade olika typer av lösenord. Lösenorden delades in i kategorier baserat på längd, användning av siffror och specialtecken samt generell struktur. På så sätt möjliggjordes en mer detaljerad analys av hur specifika egenskaper påverkade verktygens förmåga att knäcka dem.

Under experimentomgångarna samlades kvantitativa data in, såsom antalet knäckta lösenord och den totala körningstiden. Denna data användes sedan för att jämföra resultaten

mellan olika verktyg, attacktyper och hashningsalgoritmer. Testerna utfördes under kontrollerade förhållanden där varje försök tilläts pågå i maximalt en timme, för att säkerställa jämförbarhet mellan försöken.

Parallellt med de praktiska testerna genomfördes även en litteraturöversikt. Denna hade som syfte att identifiera tidigare forskning inom området lösenordssäkerhet, med särskilt fokus på password cracking-tekniker, användarbeteende samt tekniska skyddsåtgärder som saltning och val av hashningsalgoritm. Genom att kombinera empiriska resultat med teoretiska insikter skapades ett starkare underlag för analys och slutsatser. Litteraturen bidrog även med kontext för att tolka resultaten samt identifiera mönster och trender som påverkade lösenordens sårbarhet i praktiken.

2.4 Verktyg och programvara

För att genomföra denna studie användes tre huvudsakliga tekniska verktyg och programvaror: *John the Ripper*, *Hashcat* samt operativsystemet Kali Linux. Dessa verktyg valdes på grund av deras utbredda användning inom penetrationstester, cybersäkerhetsutbildning och praktiskt säkerhetsarbete.

John the Ripper är ett open source-verktyg för lösenordsanalys och återställning (Openwall, n.d.), och användes i denna studie för att genomföra både ordboksattacker och hybridattacker mot hashade lösenord. Det är särskilt känt för sin flexibilitet och sitt stöd för en mängd olika hashformat. Den version som användes i detta arbete var den kostnadsfria versionen, vilket innebär att vissa funktioner från den kommersiella versionen inte var tillgängliga. Trots detta bedömdes verktyget vara tillräckligt kraftfullt för studiens syfte.

Hashcat är ett annat kraftfullt verktyg för lösenordsåterställning (Hashcat.net, n.d.a) och användes i denna studie parallellt med *John the Ripper* för att möjliggöra en jämförande analys mellan två vanliga cracking-verktyg. *Hashcat* är känt för sin höga prestanda och möjlighet att använda både CPU och GPU för att knäcka lösenord. Verktyget erbjuder dessutom flera olika attacker, inklusive ordboksattacker och hybridattacker, vilket gjorde det särskilt lämpligt för att testa olika lösenordstyper under kontrollerade förhållanden

Kali Linux användes som operativsystem under hela experimentet. Det är en Debian-baserad Linux-distribution som är särskilt framtagen för penetrationstestning och forensiska analyser (Kali, 2024). Kali Linux erbjuder ett brett utbud av förinstallerade säkerhetsverktyg, där både *John the Ripper* och *Hashcat* ingår. Systemet möjliggjorde en standardiserad testmiljö, vilket ökade experimentens jämförbarhet och tillförlitlighet. Kali Linux kördes på en virtuell maskin, nämligen Oracle VM VirtualBox. Anledningen till att det beslutades att experimenten skulle köras på en virtuell maskin är för att det skulle gå att garantera att förutsättningarna blev samma även på olika datorer. Förmågan att kunna återställa en virtuell maskin till ett tidigare skick var dessutom lättare än att återställa en konfigurerad dator.

Experimenten genomfördes på två datorer med olika komponenter och prestanda. Främst gjordes detta för att kunna genomföra experimenten inom den planerade tidsramen för arbetet. För att säkerställa att denna skillnad inte skulle påverka resultaten av experimentet konfigurerades de virtuella maskiner som körde Kali Linux på datorerna till samma grundinställningar. I stort var redan de virtuella maskinerna identiskt konfigurerade. Det som behövde justeras var arbetsminnet (RAM-minnet), som ställdes till 8 GB, och antalet processorkärnor som maskinen kunde använda, vilket begränsades till två kärnor. På så sätt hade programvaran på båda datorerna samma förutsättningar att arbeta utifrån.

Valet av dessa verktyg grundades inte bara i deras tekniska kapacitet, utan också i deras frekventa förekomst i verkliga attacker. Genom användning av samma verktyg som både angripare och säkerhetstestare använder, kunde resultaten i denna studie ges en större praktisk relevans.

2.5 Positionering av metoden

Metoden vald för denna studie är en experimentell analys i kombination med en litteraturoversikt. Metoden tillämpar både kvantitativa och kvalitativa moment, men betraktas främst som kvantitativ eftersom experimentet genererar en stor mängd mätbara datapunkter som sedan analyseras för att dra slutsatser. Ett flertal experiment genomfördes där olika program för password cracking samt hashningalgoritmer undersökes. Experimenten utfördes på en mängd lösenord med varierande säkerhetsnivå och hashfunktioner för att möjliggöra en bred analys av metodernas effektivitet.

Metoden består också av kvalitativa inslag, då den kompletterades med en litteraturoversikt som grundade sig på tidigare forskning inom området. Denna teoretiska förankring bidrog till djupare förståelse av resultaten och deras kontext.

Eftersom syftet med studien var att undersöka hur moderna password cracking-program påverkades av lösenordskomplexitet, bedömdes en experimentell metod som mest relevant för att uppnå tillförlitliga resultat. Alternativa metoder som enkätundersökningar och intervjuer, bedömdes mindre lämpliga, då de inte förväntades kunna generera den typ av empiriska data som krävdes för att analysera programmets effektivitet vid password cracking. En del tidigare arbeten som ligger till grund för denna studie och som har undersökt lösenord genomförde experiment som metod. Därför är samma val av metod lämplig för detta arbete.

Väldigt få tidigare gjorda studier på området fokuserar på specifika password cracking-program, som *John the Ripper*. De flesta studier som detta arbete grundar sig på undersöker hur användarbeteende och lösenordshantering påverkar säkerheten. Även vilka faktorer som spelar in när en användare väljer ett lösenord tas upp av några artiklar som ligger till grund för detta arbete. En av de uppbackande studierna som undersökt *John the Ripper*, en studie gjord av Marchetti & Bodily (2022), uppmärksammade att framtida studier bör fortsätta granska programmet och utvärdera dess effektivitet. Särskilt hur väl det klarar av

att knäcka andra hashningalgoritmer än MD5 och SHA-256. På grund av denna brist på omfattande forskning på sådana verktyg är det högst aktuellt att undersöka detta område genom experiment uppbäckade av litteratur kring lösenordssäkerhet och dokumentation för programmen. Samma brist på undersökningar gäller även för programmet *Hashcat*.

Denna studie undersökte sannolikheten att olika lösenord knäcktes på kort tid, en timme, med verktygen *Hashcat* och *John the Ripper*. Det bidrar dels med hur effektiva verktygen är på att knäcka lösenord på kort tid, men även en jämförelse mellan verktygen som visar på vilket som är mest effektivt beroende på hur lösenordet är lagrat.

2.6 Problematisering av metoden

Att utföra experiment med password cracking-program på detta sätt är en tidskrävande process, särskilt om man försöker genomföra en brute-force-attack där programmet ska gissa rätt lösenord genom att testa alla möjliga kombinationer. Särskilt för långa lösenord kan detta ta lång tid för en vanlig dator att lösa. Dock kan en sådan utdragen process kringgås genom att försöka med en annan sorts attack, exempelvis en ordboksattack som kan ge snabbare resultat. Med program som *John the Ripper* erbjuds möjligheten att försöka knäcka lösenord genom en rad olika metoder. Att nyttja detta faktum speglar även hur en verklig angripare skulle genomföra en lösenordsattack, vilket gav experimentens resultat ökad trovärdighet.

I detta arbete undersöks endast två password cracking-program, *John the Ripper* och *Hashcat*, och då endast gratisversionen av den förstnämnda. Även om dessa två är bland de mest populära och vanligast förekommande programmen som dyker upp i lösenordsknäckningssammanhang så finns det skillnader mellan dem och andra frekvent använda program. I grunden är det olika program som fungerar på olika sätt och alla har inte samma funktioner. Dessutom är några av programmen baserade på endast ett operativsystem. Exempelvis programmet *Brutus* fungerar endast till Windows (Darktrace, n.d.) (Poston, 2025). Därför går det att argumentera för att resultatet och efterföljande slutsatser i detta arbete inte går att applicera på alla övriga sorters password cracking-verktyg. En invändning till detta är att vidden för detta arbete och dess frågeställningar gör att slutsatserna blir tillräckligt allmängiltiga så att de går att applicera för andra verktyg, åtminstone till viss del. Dessutom är det arbetes syfte att ge rekommendationer och vägledning i att välja säkra lösenord baserat på prestandan hos specifika password cracking-program. Att genomföra liknande undersökningar med samma metod för andra program, och på så sätt bygga vidare på detta arbete, är något för framtida studier.

RockYou-filen i Kali Linux innehöll ett mindre antal lösenord jämfört den ursprungliga RockYou-filen som finns allmänt tillgänglig på internet. Detta kan påverka pålitligheten

och hur representativa resultaten av denna studie är för verkliga angripare, som möjligtvis skulle föredra att använda den större versionen av RockYou-filen.

På grund av studiens begränsningar kommer antalet testrundor för experimentet som är möjliga att genomföra inom tidsspannet vara otillräckliga för att kunna dra brett tillämpbara slutsatser utifrån. Däremot är verktygen som används i experimentet är så pass frekvent förekommande att man fortfarande kan dra tillräckligt användbara slutsatser baserat på deras resultat.

2.7 Förtjänst

Trots vissa begränsningar har studien flera styrkor som gör att resultaten kan anses relevanta inom området lösenordssäkerhet. Genom att kombinera praktiska experiment med en teoretisk genomgång av tidigare forskning har det varit möjligt att undersöka hur olika typer av lösenord påverkas av moderna password cracking-verktyg. Detta upplägg har bidragit till att resultaten fått en praktisk tillämpning, exempelvis som stöd för att förstå hur angripare kan gå tillväga och vilka typer av lösenord som är särskilt sårbara.

En central förtjänst i studien är att experimenten genomförts med verktyg som ofta förekommer vid riktiga attacker, såsom *Hashcat* och *John the Ripper*. Dessa används inte enbart av angripare utan även inom säkerhetstestning, vilket ökar relevansen i de slutsatser som dragits. Genom användning av båda verktygen har det dessutom varit möjligt att göra en jämförelse mellan dem, vilket inte är vanligt förekommande i tidigare liknande arbeten.

För att spegla realistiska scenarier användes ordboksattacker och hybridattacker med tillägg av siffror och specialtecken. Detta gav ett mer verklighetsnära perspektiv på hur attacker faktiskt går till, snarare än att enbart fokusera på teoretiska angrepp.

2.8 Etiska överväganden

I detta arbete användes lösenord från verkliga lösenordsläckor, vilket innebär att den data som analyserats har tillhört verkliga användare. För att säkerställa att studien genomfördes på ett etiskt försvarbart sätt så har endast publikt tillgängliga lösenordsläckor använts och inga försök har gjorts för att koppla lösenorden till specifika användare eller konton.

Dessutom är listorna som använts etablerade och används frekvent inom penetrationstester och lösenordskontroller. Lösenordlistorna hanteras enbart i forsknings- och utbildningssyfte, med målet att belysa svagheter i lösenordshantering och därigenom till förbättrad säkerhetsmedvetenhet.

Värt att notera är att innehållet och insikterna som detta arbete ger kan användas i oetiska syften av individer med ont uppsåt. De nya rekommendationerna som arbetet resulterar i

kan ses som nya utmaningar att knäcka för de som skapar och använder sig av hacking-verktyg för att göra sådana mer effektiva och resistent mot IT-säkerhetsåtgärder. Väl medvetna om denna risk är det samtidigt viktigt att poängtera betydelsen i att undersöka password cracking-program i syfte att identifiera deras tillkortakommanden och därav kunna ge vägledning i hur användare kan välja lösenord som är motståndskraftiga mot sådana program. I slutändan är meningen med studien att öka medvetenheten kring risker relaterade till användningen av svaga lösenord.

3 Resultat

I denna del redovisas resultaten av alla experimentomgångar som gjorts i studiens experimentella del. Kapitlet har delats upp enligt vilka två program som användes, vilken sorts attack som gjordes med programmen samt vilken hashingalgoritm som användes för att kryptera lösenordsdatabasen som användes i den omgången. För varje experimentomgång redovisas mängden lösenord som programmet lyckades knäcka och tiden det tog programmet att köra klart omgången. Om däremot programmet beräknades behöva köras längre än en timme redovisas resultaten av en timmes körning.

3.1 Hashcat

Hashcat är ett av programmen som användes i experimentet i denna studie. Det kördes mot lösenordsdatabaser som hade hashats med de valda algoritmerna, vilka var bcrypt, osaltad SHA-256 och saltad SHA-256.

Tabell 2. Tabellen visar hur många av lösenorden som knäcktes med någon av de genomförda attackmetoderna i *Hashcat* (ordboksattack och hybrid med siffror eller specialtecken). Dubletter har filtrerats bort. Tabellen redovisar lösenordens teckenlängd, det totala antalet lösenord för varje längd, hur många som knäcktes samt motsvarande andel i procent.

Teckenlängd	Total mängd lösenord	Knäckta lösenord	Andel knäckta (%)
3	55	14	25,45
4	1 057	233	22,04
5	3 493	1 068	30,58
6	16 773	5 626	33,54
7	16 869	3 066	18,18
8	29 811	4 107	13,78
9	10 087	1 172	11,62
10	7 445	621	8,34
11	4 467	207	4,63
12	3 259	110	3,38
13	2 438	28	1,15
14	1 440	19	1,32
15–38	2 806	17	0,61

3.1.1 Hybridattacker

Den första sortens attack som genomfördes var hybridattack. Dessa genomfördes mot hashingalgoritmerna bcrypt, osaltad och saltad SHA-256. Två sorters hybridattacker genomfördes. För den första sorten lade programmet till siffror i knäckningsprocessen, och i den andra tillades specialtecken. Programmet började med att lägga till endast en siffra eller specialtecken i knäckningen, och därefter stegrades antalet per körning. Maxantalet siffror eller specialtecken som kördes var fem.

Hybridattacker – Siffror

Hybridattacker med adderade siffror kördes mot olika versioner av lösenordsdatabasen, där variationen låg i vilken typ av hashingalgoritm som hade applicerats på den. Då antalet siffror stegrades från en till fem siffror per hashingalgoritm blev det totalt femton hybridattacker som gjordes med siffror i *Hashcat*.

För den lösenordsdatabas som hashades med *bcrypt* genomfördes fem separata hybridattacker med *Hashcat*. Vid den första attacken lades en siffra till för varje ord i ordlistan. Vid den andra lades två siffror till. Adderandet stegrades tills fem siffror hade lagts till. Den sista attacken som genomfördes hade ställts in på att lägga till fem siffror. Samtliga attacker kördes i en timmes tid vardera.

Vid den första attacken, där en siffra adderades i knäckningen, resulterade körningen i att 0 av 100 000 lösenord kunde knäckas. Programmet kördes under en timmes tid utan något träffresultat. Sökresultatet blev likadant för resterande attacker där två, tre, fyra och fem siffror tillades.

Näst på tur var den databas som var hashad med osaltad SHA-256. Även här utfördes fem separata hybridattacker; där siffror adderades successivt som tillägg till ordlistan i knäckningsprocessen. Samtliga resultat från attackerna med *Hashcat* mot den osaltade SHA-256-databasen finns uppställt i Figur 1.

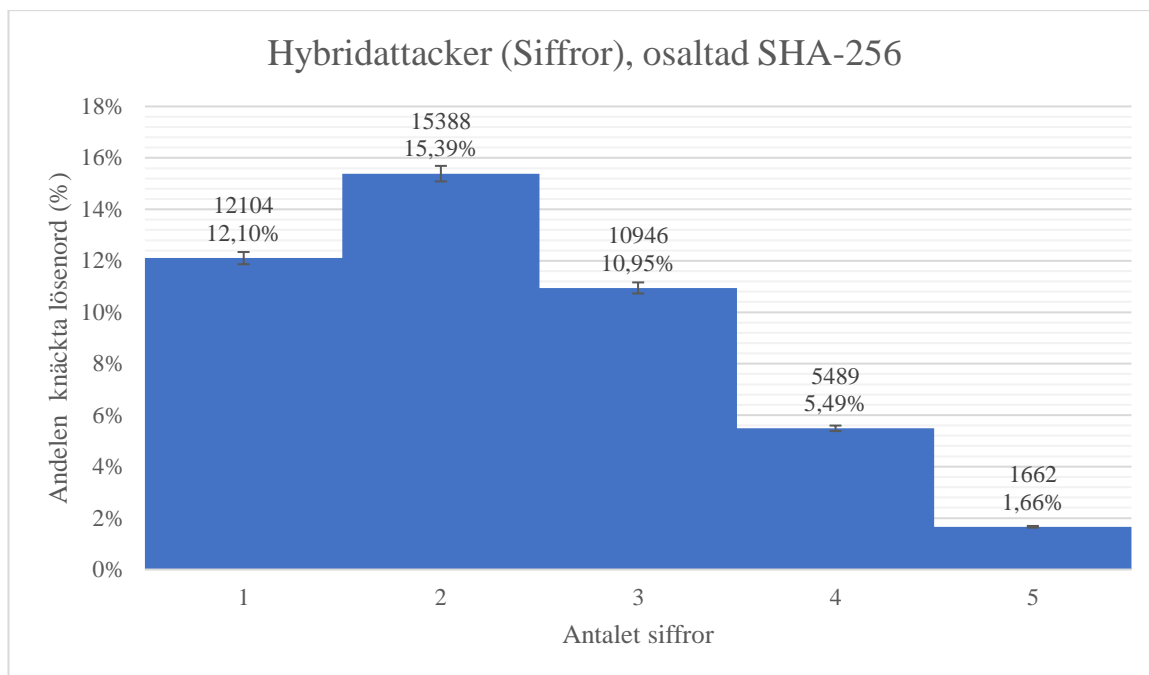
När *Hashcat* kördes med en siffra som tillägg, lyckades det knäcka 12 104 av de 100 000 hashade lösenorden. Attacken blev klar efter att 51 sekunder hade passerat.

Vid den andra körningen, där istället två siffror användes, blev resultatet att totalt 15 388 lösenord av 100 000 knäcktes efter samma tidsintervall som tidigare körning. Denna attack blev klar efter fem minuter och elva sekunder hade passerat.

För den tredje körningen lyckades *Hashcat* knäcka 10 946 av 100 000 lösenord efter att 51 minuter hade passerat. Här adderades tre siffror till knäckningen.

Från den fjärde körningen, där fyra siffror adderades i knäckningen, lyckades *Hashcat* få fram 5 489 av 100 000 lösenord efter en timmes körning.

Slutligen, från den femte och sista körningen där fem siffror lades till i knäckningen så fick *Hashcat* fram 1 662 av 100 000 lösenord ur den hashade databasen. Detta efter att ha körts i en timme.



Figur 1: Histogram för antalet och andelen knäckta lösenord för samtliga hybridattacker med siffror gjorda med Hashcat mot databasen som hashats med osaltad SHA-256. Osäkerheten visas med felstaplar

Avslutningsvis utfördes fem hybridattacker mot databasen som var hashad med saltad SHA-256. *Hashcat* konfigurerades först att lägga till en siffra, därefter två siffror till orden i ordlistan fram till fem siffror på samma sätt som i tidigare försök. I Figur 2 finns alla resultat från denna typ av hybridattack med *Hashcat* illustrerat.

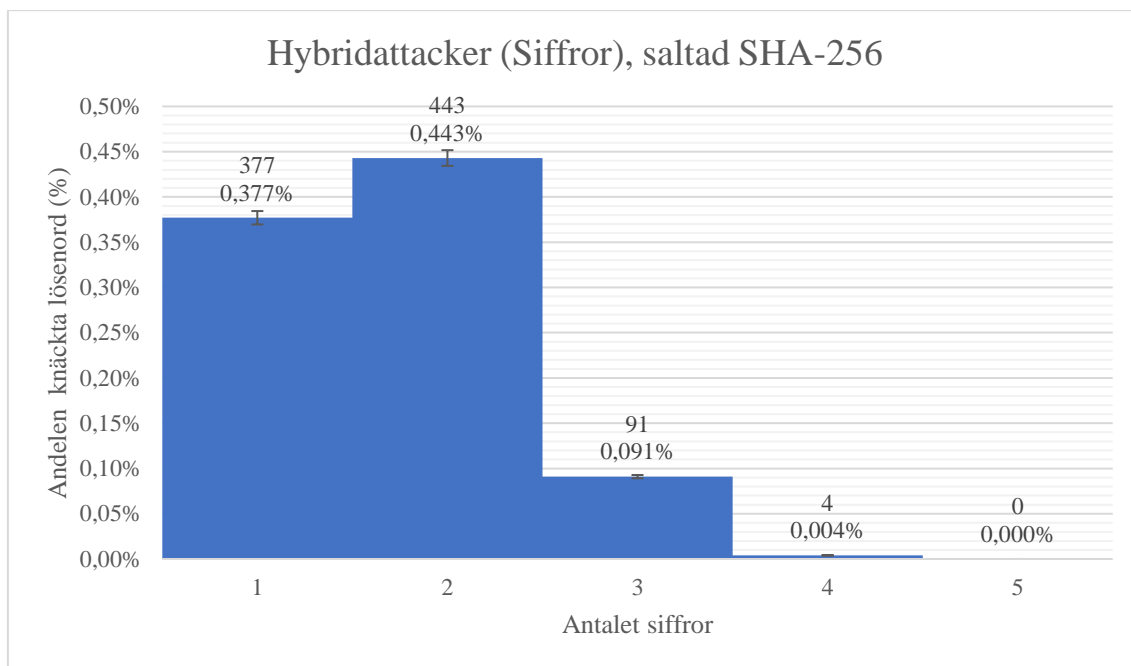
För den första experimentkörningen, där en siffra användes, blev antalet knäckta lösenord 377 av 100 000. Även denna körning hade en varaktighet på en timme.

I den andra körningen, där två siffror användes, lyckades programmet knäcka 443 lösenord av 100 000. Resultatet uppnåddes efter en timmes körning.

Efter att ha kört den tredje experimentkörningen blev resultatet att 91 av 100 000 lösenord hittades. Precis som övriga omgångar kördes denna i en timme. Vid denna körning lades tre siffror till av programmet i knäckningen.

Den näst sista körningen genomfördes med att fyra siffror adderades till knäckningen. Efter att ha körts i en timme blev resultatet att *Hashcat* lyckades få fram 4 av 100 000 lösenord från den hashade databasen.

För den sista körningen, i vilken fem siffror lades till i knäckningen, blev utfallet att inga lösenord knäcktes av *Hashcat*. Detta resultat framkom efter att programmet körts i en timme.



Figur 2: Histogram för antalet och andelen knäckta lösenord för samtliga hybridattacker med siffror gjorda med Hashcat mot databasen som hashats med saltad SHA-256. Osäkerheten visas med felstaplar.

Hybridattacker – Specialtecken

Utöver hybridattacker med siffror genomfördes även hybridattacker där enbart specialtecken adderades till knäckningen. Precis som med hybridattackerna med siffror kördes dessa attacker mot samma lösenordsdatabas som hade hashats med hashingalgoritmerna bcrypt, osaltad SHA-256 och saltad SHA-256 vardera. Eftersom fem attacker kördes mot varje hashing algoritm, där antalet tillagda specialtecken stegvis ökade från en till fem, genomfördes totalt femton hybridattacker med specialtecken i Hashcat.

Den första av hashingalgoritmerna var bcrypt. På den version av lösenordsdatabasen som hashats med bcrypt genomfördes totalt fem hybridattacker. Vid varje attack ökade antalet specialtecken som lades till i knäckningsprocessen. Antalet började med en siffra och stegrades gradvis till fem siffror. Alla dessa hybridattacker exekverades i en timme. Om programmet blev klar med en attack innan en timme hade passerat anges det.

För samtliga av hybridattackerna med adderade specialtecken gjorda mot den bcrypt-hashade databasen blev resultatet detsamma. Hashcat lyckades inte få fram några lösenord från bcrypt från någon av attackerna med specialtecken. Samtliga attacker mot bcrypt kördes i en timme vardera.

Efter bcrypt kördes hybridattacker med tillagda specialtecken mot databasen som hashats med osaltad SHA-256. Här kördes attacker där först ett specialtecken adderades till knäckningen, och antalet specialtecken ökade stegvis per attack. Sista attacken av denna

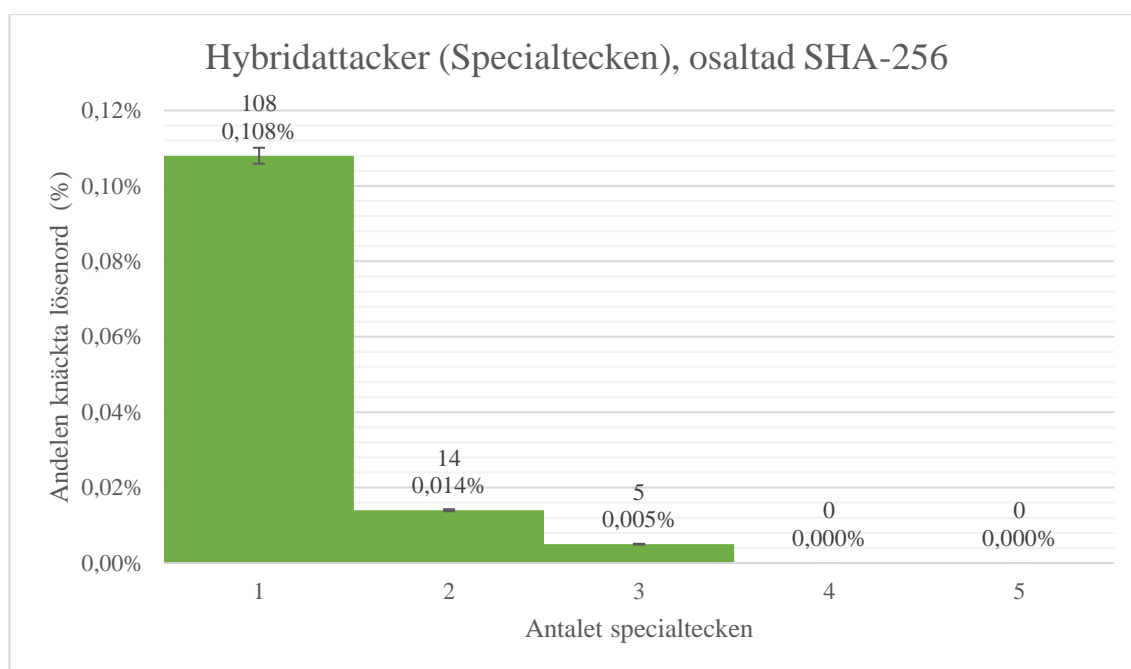
sorts hybridattacker mot osaltad SHA-256 gjordes med fem tillagda specialtecken. I Figur 3 finns alla resultaten från hybridattackerna med specialtecken mot osaltad SHA-256 sammanställda.

I den första av hybridattackerna mot osaltad SHA-256, i vilken programmet lade till ett specialtecken i sökningen, blev resultatet att 108 av 100 000 lösenord knäcktes. *Hashcat* blev klar med denna attack efter en minut och 58 sekunder.

Efterföljande attack genomfördes med att två specialtecken tillades i knäckningen. Där blev resultatet att *Hashcat* hittade 14 av 100 000 lösenord. Resultatet framkom efter att en timme hade förflutit.

För attacken där tre specialtecken adderades i knäckningen blev utfallet att 5 av 100 000 lösenord knäcktes ur databasen. I denna attack kördes *Hashcat* i en timme.

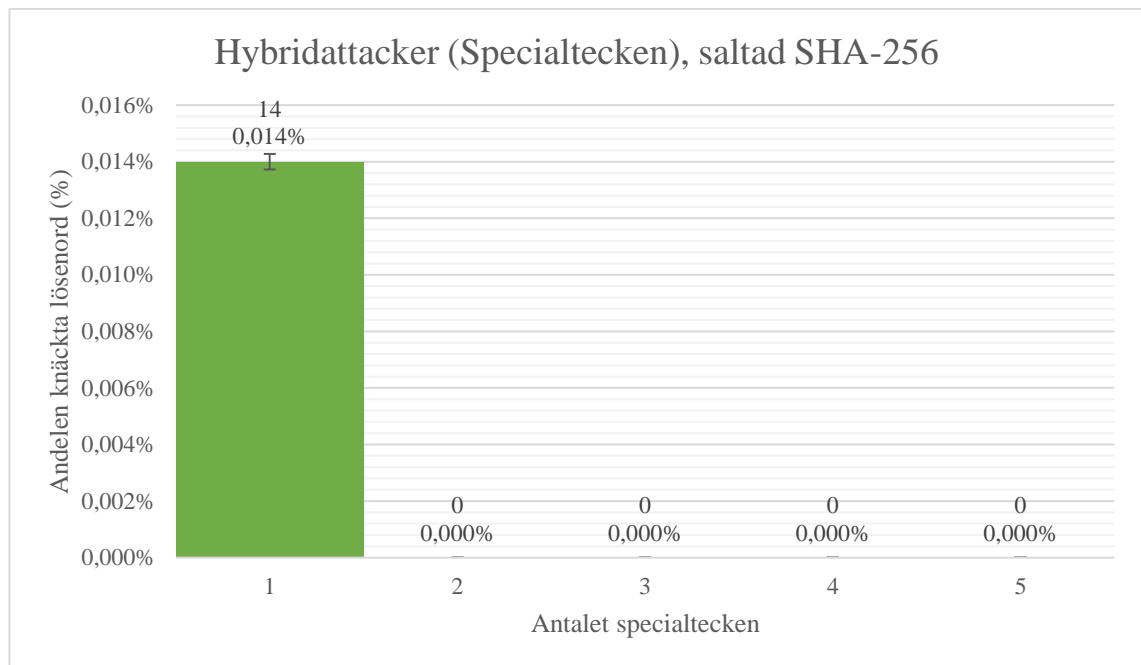
I de två sista hybridattackerna mot osaltad SHA-256, där fyra respektive fem specialtecken adderades till knäckningen, fick *Hashcat* inte fram några av 100 000 lösenord. I båda dessa sista attacker kördes programmet i en timme för respektive attack.



Figur 3: Histogram för antalet och andelen knäckta lösenord för samtliga hybridattacker med specialtecken gjorda med Hashcat mot databasen som hashats med osaltad SHA-256. Osäkerheten visas med felstaplar.

Slutligen gjordes hybridattacker med specialtecken mot databasen som hashats med saltad SHA-256. Precis som attacker mot tidigare nämnda hashingalgoritmer gjordes fem olika hybridattacker där antalet tillagda specialtecken ökade stegvis, upp till att fem specialtecken användes i attacken. En visualisering av samtliga resultat från hybridattackerna med specialtecken mot saltad SHA-256 finns att se i Figur 4.

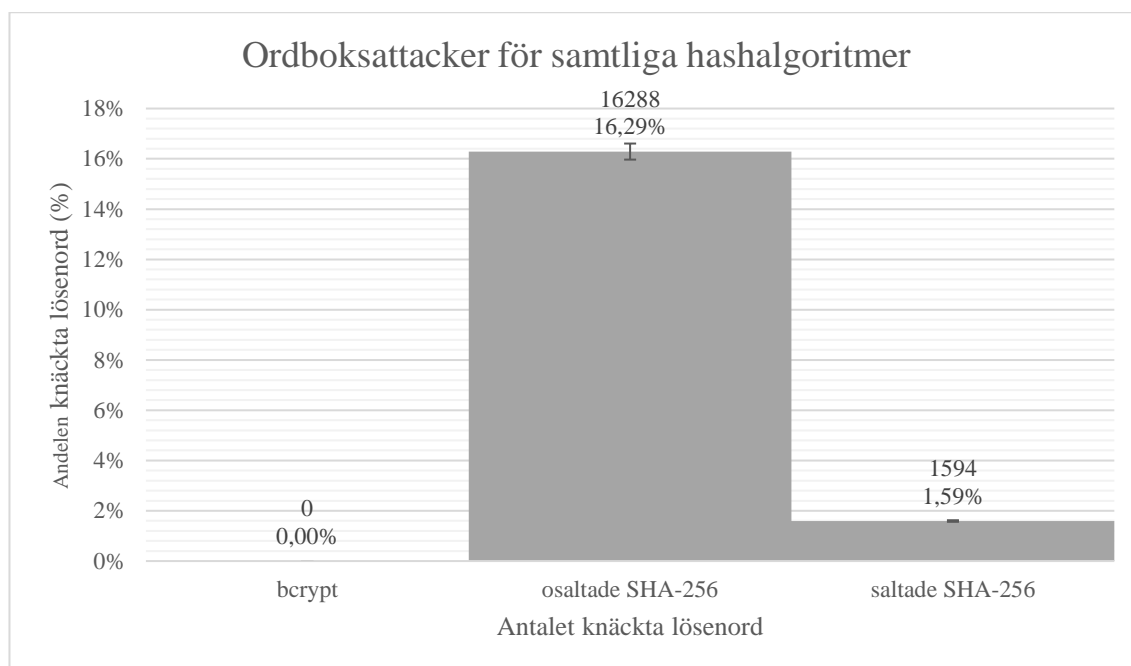
För denna typ av hybridattack mot saltad SHA-256 lyckades endast den första attacken, med ett tillagt specialtecken, att få ut några knäckta lösenord ur databasen. Denna attack fick ut 14 av 100 000 lösenord. Efterföljande attacker fick inte ut några lösenord. Alla attacker lät köras i en timme vardera.



Figur 4: Histogram för antalet och andelen knäckta lösenord för samtliga hybridattacker med specialtecken gjorda med Hashcat mot databasen som hashats med saltad SHA-256. Osäkerheten visas med felstaplar.

3.1.2 Ordboksattacker

Utöver hybridattackerna så genomfördes också ordboksattacker på respektive hashalgoritm. I Figur 5 visas en jämförelse av resultaten genererade för de olika hashalgoritmerna.



Figur 5: Histogram för antalet och andelen knäckta lösenord för samtliga ordboksattacker gjorda med Hashcat på hashalgoritmerna. Osäkerheten visas med felstaplar.

För lösenordsdatabasen som hashades med bcrypt genomfördes en ordboksattack med Hashcat där verktyget uteslutande använde ordlistan RockYou.

Resultatet visade att inga lösenord kunde knäckas inom testets begränsning på en timme. Samtliga 100 000 hashar förblev oknäckta, trots att hela ordlistan användes som input.

För databasen som var hashad med osaltad SHA-256 utfördes en liknande ordboksattack med samma verktyg och ordlista.

Denna gång lyckades Hashcat identifiera och knäcka totalt 16 288 lösenord av 100 000. Körningen avslutades på mycket kort tid, mer exakt åtta sekunder.

Sist genomfördes en ordboksattack mot databasen som hashats med saltad SHA-256. Även här användes RockYou som ordlista i attacken.

Resultatet av denna körning blev att 1 594 lösenord knäcktes av totalt 100 000. Totalt kördes denna attack i en timme.

3.2 John the Ripper

John the Ripper var det andra programmet som kördes i experimentet i denna studie. Programmet kördes mot en lösenordsdatabas som hade krypterats med en av de undersökta hashingalgoritmerna.

Tabell 3. Visar hur många av lösenorden som knäcktes med någon av de genomförda attackmetoderna i *John the Ripper* (ordboksattack, hybrid med siffror eller specialtecken). Dubletter har filterats bort. Tabellen redovisar lösenordens teckenlängd, det totala antalet lösenord för varje längd, hur många som knäcktes samt motsvarande andel i procent.

Teckenlängd	Total mängd lösenord	Knäckta lösenord	Andel knäckta (%)
3	55	51	92,73
4	1 057	995	94,13
5	3 493	2 706	77,47
6	16 773	12 685	75,63
7	16 869	11 947	70,82
8	29 811	14 553	48,82
9	10 087	4 911	48,69
10	7 445	2 689	36,12
11	4 467	1 111	24,87
12	3 259	542	16,63
13	2 438	163	6,69
14	1 440	103	7,15
15–38	2 806	73	2,6

3.2.1 Hybridattacker

Hybridattack var den ena sortens attack som gjordes med *John the Ripper*. Genom experimentet kördes programmet mot alla tre hashingalgoritmer. Hybridattackerna delades in i två kategorier. I den första kategorin blev programmet instruerat att lägga till siffror i knäckningen. Programmet började med en siffra och stegrades sedan successivt antalet till fem. För den andra kategorin lade programmet istället till specialtecken i processen. Samma stegring från ett till fem gjordes för specialtecken.

Hybridattacker – Siffror

Den första sortens hybridattack som genomfördes med *John the Ripper* lade till siffror i knäckningen. Denna sorts hybridattack genomfördes mot alla lösenordsdatabaser som hashats med någon av de tre hashingalgoritmerna. Eftersom programmet successivt stegrade antalet siffror som lades till i knäckningen genomfördes totalt fem hybridattacker

med tillagda siffror för varje hashingalgoritm. Totalt blev det därför femton hybridattacker med siffror gjorda med *John the Ripper*.

För lösenorden som hade hashats med bcrypt genomfördes fem hybridattacker med *John the Ripper*. Liksom i *Hashcat* så lades det till en siffra i första attacken, två i den andra upp till fem siffror hade lagts till. Samtliga av attackerna kördes i en timme.

Vid respektive attack på lösenorden som hashats med bcrypt så lyckades *John the Ripper* knäcka 0 lösenord utav de 100 000 som hade hashats. Samtliga attacker kördes i en timme.

Efter bcrypt var färdigtestat, så genomfördes samtliga hybridattacker med en siffra upp till fem siffror på de lösenord som hashats med osaltad SHA-256. Samtliga av de resultat som framkom för varje hybridattack med *John the Ripper* visas i Figur 6. Alla attacker kördes högst i en timme innan de avbröts.

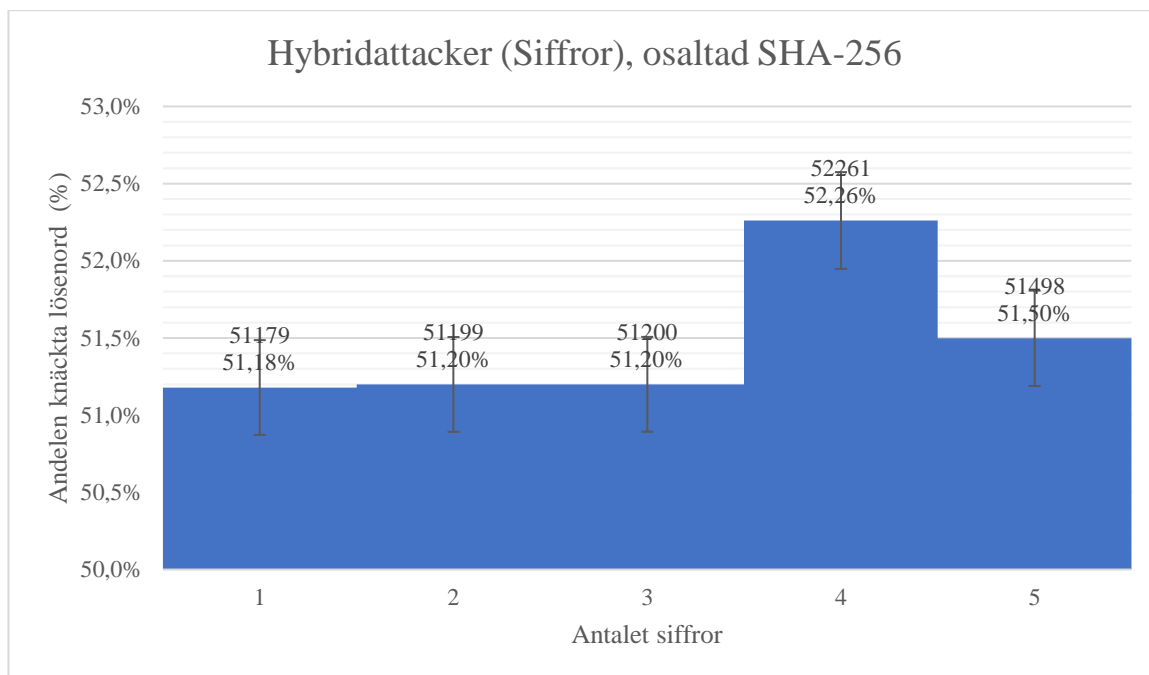
Första hybridattacken genomfördes med en siffra tillagd och då lyckades 51 179 av 100 000 lösenord knäckas. Attacken var färdig efter elva sekunder.

Den andra hybridattacken som genomfördes med två siffror tillagda lyckades 51 199 av 100 000 lösenord knäckas. Attacken var färdig efter en minut och 32 sekunder.

Efter det kördes den tredje hybridattacken som genomfördes med tre siffror tillagda så lyckades 51 200 av 100 000 lösenord knäckas. Attacken var färdig efter 14 minuter och 26 sekunder.

Nästa var den fjärde hybridattacken som genomfördes med fyra siffror tillagda lyckades 52 261 av 100 000 lösenord knäckas. Attacken avbröts efter en timme.

Slutligen, för den femte hybridattacken som genomfördes med fem siffror tillagda lyckades 51 498 av 100 000 lösenord knäckas. Attacken avbröts efter en timme.



Figur 6: Histogram för antalet och andelen knäckta lösenord för samtliga hybridattacker med siffror gjorda med John the Ripper mot databasen som hashats med osaltad SHA-256. Osäkerheten visas med felstaplar.

Slutligen, för lösenorden som hashades med en saltad version av SHA-256 kördes även här fem hybridattacker där en till fem siffror lades till. Likt resultatet för bcrypt så lyckades John the Ripper inte heller knäcka några av hybridattackerna som gjordes på saltad SHA-256.

Hybridattacker – Specialtecken

Likt hybrid attacken som genomfördes med siffror, genomfördes även hybridattacker där specialtecken lades till. Dessa hybridattacker genomfördes mot alla olika löseordsdatabaser. Som hashats med någon av de tre hashingalgoritmer som användes. Eftersom programmet stegrade antalet siffror som lades till för varje knäckning så genomfördes totalt fem hybridattacker med tillagda specialtecken för varje hashingalgoritm. Totalt blev det därför femton hybridattacker med specialtecken som genomfördes med John the Ripper.

Hybridattackerna genomfördes också med ett till fem specialtecken tillagda. Likt resultatet som bcrypt gav vid tidigare hybridattacker med siffror så lyckades John the Ripper knäcka 0 av de 100 000 lösenord oavsett mängden specialtecken som användes. Samtliga av attackerna kördes i en timme.

Efter hybridattackerna på bcrypt var genomförda så gjordes samma hybridattacker med ett till fem specialtecken på lösenorden som var hashade med osaltad SHA-256. Alla attacker kördes i högst en timme innan de avbröts. I Figur 7 finns resultaten från samtliga attacker gjorda mot osaltad SHA-256 visualiserat.

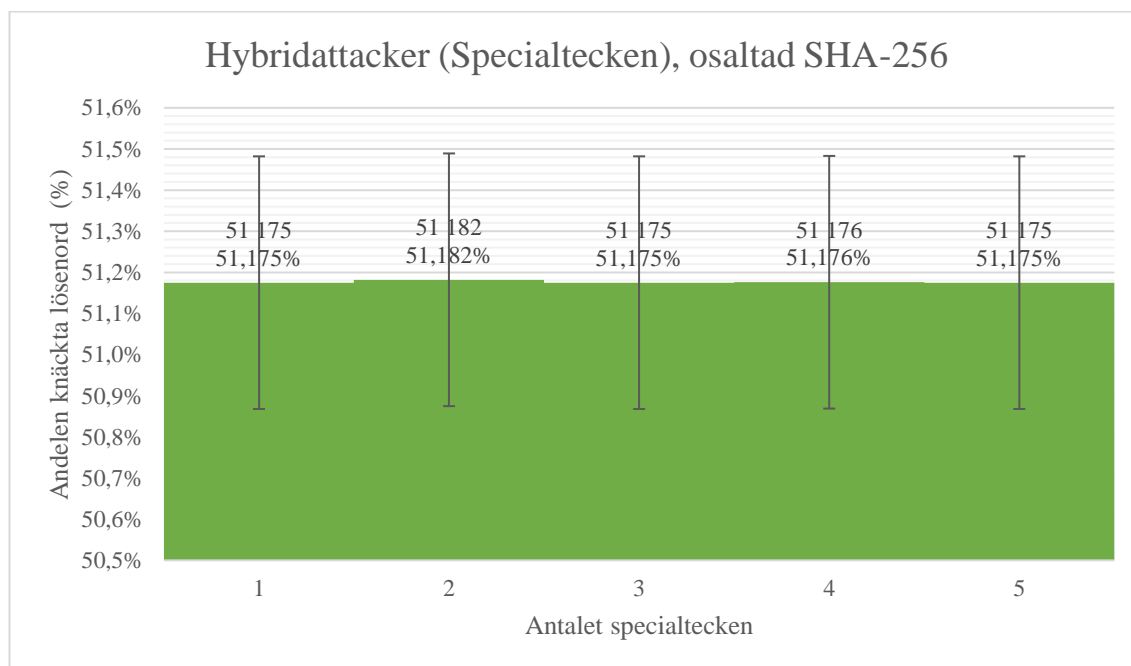
Den första hybridattacken som genomfördes bestod av ett specialtecken tillagt, då lyckades 51 175 av 100 000 lösenord knäckas. Attacken var färdig efter 30 sekunder.

Den andra hybridattacken som genomfördes bestod av två specialtecken tillagda, då lyckades 51 182 av 100 000 lösenord knäckas. Attacken var färdig efter 15 minuter och 56 sekunder.

Den tredje hybridattacken som genomfördes bestod av tre specialtecken tillagda, då lyckades 51 175 av 100 000 lösenord knäckas. Attacken stoppades efter en timme.

Den fjärde hybridattacken som genomfördes bestod av fyra specialtecken tillagda, då lyckades 51 176 av 100 000 lösenord knäckas. Attacken stoppades efter en timme.

Slutligen, den femte hybridattacken som genomfördes bestod av fem specialtecken tillagda, då lyckades 51 175 av 100 000 lösenord knäckas. Attacken stoppades efter en timme.

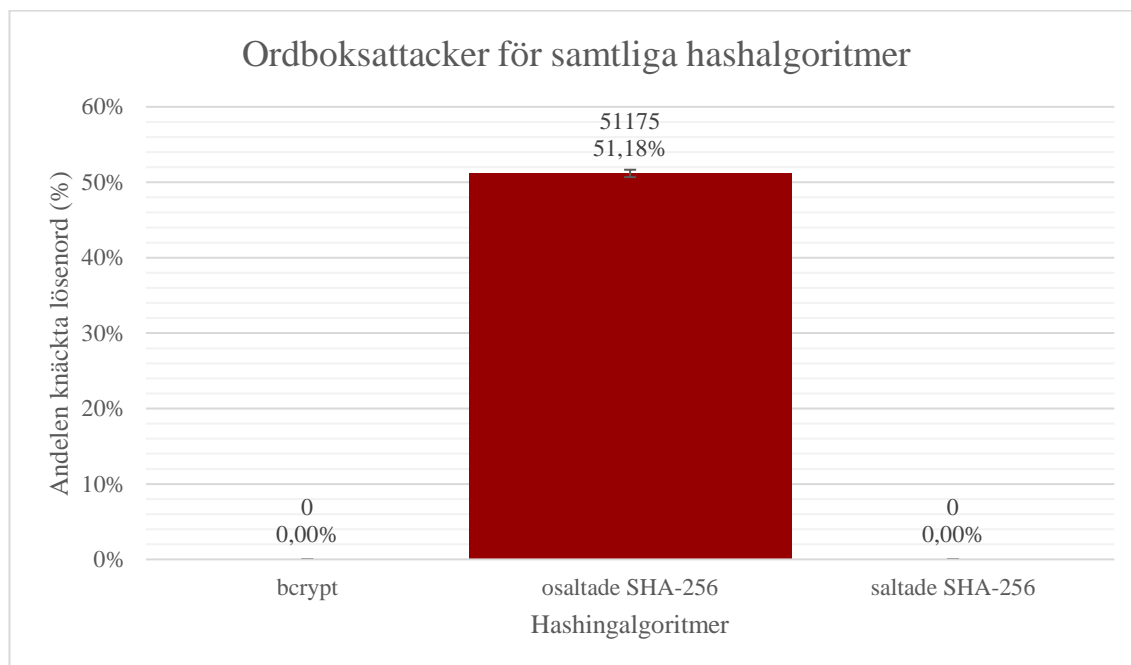


Figur 7: Histogram för antalet och andelen knäckta lösenord för samtliga hybridattacker med specialtecken gjorda med John the Ripper mot databasen som hashats med osaltad SHA-256. Osäkerheten visas med felstaplar.

Avslutningsvis kördes även samtliga hybridattacker med ett till fem specialtecken mot lösenorden som var hashade med saltad SHA-256, resultatet visade att John the Ripper lyckades knäcka 0 av 100 000 lösenord. Samtliga attacker kördes i en timme.

3.2.2 Ordboksattacker

Den andra sortens attack som utfördes mot de hashade lösenordsdatabaserna var ordboksattack. En sammanställning av alla resultat från alla ordboksattacker gjorda med *John the Ripper* finns synligt i Figur 8.



Figur 8: Histogram för antalet och andelen knäckta lösenord för samtliga ordboksattacker gjorda med *John the Ripper* på hashalgoritmerna. Osäkerheten visas med felstaplar.

Det gjordes även en ordboksattack för varje hashfunktion. Både för attacken mot bcrypt och attacken mot saltad SHA-256 så blev resultatet av antalet lösenord som *John the Ripper* lyckades knäcka 0 av 100 000. Båda två attackerna kördes i en timme.

Resultatet för osaltad SHA-256 var däremot att *John the Ripper* lyckades knäcka 51 175 av 100 000 lösenord. Attacken tog en sekund innan den var färdig.

4 Diskussion

I detta kapitel diskuteras de resultat som framkom vid studiens experimentella del. Diskussionen fokuserar på att analysera hur resultaten förhåller sig till tidigare forskning, samt vilka implikationer och begränsningar som kan identifieras utifrån de genomförda experimenten.

4.1 Hur påverkar lösenordets längd dess motståndskraft?

Resultaten från studiens experiment visade tydligt att längre lösenord generellt uppvisade större motståndskraft mot cracking-försök. De hybridattacker som genomfördes med *Hashcat*, där ett till fem siffror eller specialtecken lades till som suffix, visade att flest lösenord knäcktes när ett till tre tecken användes. När masklängden utökades till fyra och fem tecken minskade däremot antalet träffar markant. Denna nedgång förklarades av att sökrymden ökade exponentiellt för varje tillagt tecken, vilket innebar att färre kombinationer hann testas inom den angivna tidsramen, en timme.

Vid motsvarande attacker med *John the Ripper* observerades en motsatt trend, antalet knäckta lösenord ökade vid längre masklängd. Detta framkom i flera testfall och tyder på att verktygets hantering av maskkomponenter, särskilt "?w" (som representerar ett helt ord från ordlistan), möjliggjorde mer effektiva kombinationer av ordlistans innehåll med varierande suffix jämfört med *Hashcat*.

En fördjupad analys av samtliga knäckta lösenord i relation till deras ursprungliga längd visade dock att sambandet mellan längd och motståndskraft inte var helt linjärt. Till exempel hade lösenord med fem och sex tecken den högsta andelen träffar, med 30,58 % respektive 33,54 % knäckta. För sju tecken sjönk däremot träffandelen till 18,18 % (se Tabell 2). Denna variation kan förklaras av angriparens prioritering av vissa längder i ordlistor och maskstrategier, samt av användares benägenhet att skapa mer förutsägbara mönster vid vissa längder. Trots dessa avvikelser visade resultaten sammantaget att längre lösenord generellt var mer motståndskraftiga mot knäckning, särskilt när längden kombinerades med ökad komplexitet.

Vid närmare granskning av attacktypernas effekt visade det sig att lösenordslängden påverkade ordboksattacker och hybridattacker på olika sätt. När det gäller ordboksattacker, är angriparens framgång direkt beroende av att lösenordet återfanns i en fördefinierad ordlista. Dessa ordlistor består ofta av kortare och vanliga lösenord som har förekommit i tidigare dataläckor. Därför är lösenord med kortare längd, i vårt resultat särskilt lösenorden med fem till sex tecken, överrepresenterade bland träffarna. Längre lösenord förekommer mer sällan i ordlistor, vilket minskar sannolikheten att de knäcks vid rena ordboksattacker.

I hybridattacker kombineras ordlistans innehåll med masktillägg, vanligtvis siffror eller specialtecken i slutet av ordet. Även här är längden på lösenordet en avgörande faktor. När

ett suffix på två till tre tecken adderas till ett ord, ökar längden på det prövade lösenordet. Lösenord som i sig redan är längre hamnar då utanför den effektiva sökrymden eftersom attacken snabbt blir för tidskrävande. Det förklarar varför hybridattacker ofta lyckas bäst mot lösenord på upp till sex tecken. När längden överstiger detta, räcker inte tiden till för att täcka alla möjliga kombinationer, särskilt inte när flera specialtecken testas.

Denna skillnad mellan attacktyper innebär att kortare lösenord är särskilt sårbara för båda typerna av attacker, medan längre lösenord, även utan hög komplexitet, erbjuder betydligt bättre motstånd. Längd påverkar alltså inte bara hur många kombinationer som måste testas, utan även vilka attacker som i praktiken blir effektiva inom en rimlig tidsram.

Tidigare forskning visade att lösenordslängd var en central faktor för skydd mot crackingförsök. Kanta m.fl. (2023) lyfte att varje tecken som lades till i ett lösenord ökade antalet möjliga kombinationer som ett verktyg behövde testa, vilket gjorde attacker mer tidskrävande, särskilt i ordboksbaseade och kontextuella angrepp. Denna princip låg till grund för den metod som tillämpades i denna studie.

Även Shi m.fl. (2021) konstaterade att längre lösenord gav ökad motståndskraft i offline-sammanhang, eftersom angriparen inte begränsades av inloggningsförsök utan kunde pröva stora mängder kombinationer. Dessa forskningsresultat bidrog till att förklara varför kortare masker gav fler träffar inom en timme, medan längre masker inte hann täcka hela sökrymden. Litteraturen stödde därmed de skillnader som observerades mellan attacker med varierande masklängd.

Resultaten från denna studie visade att lösenordslängd påverkade cracking-verktygens effektivitet i praktiken. I de attacker som genomfördes med *Hashcat* och *John the Ripper* blev det tydligt att längre lösenord krävde längre tid för att knäckas, även när de inte innehöll avancerade teckenkombinationer. Det innebär att längd i sig hade en skyddande effekt, vilket kunde vara relevant för både användare och systemutvecklare.

Utifrån dessa resultat gick det att fastställa att krav på lösenordslängd kunde vara ett enkelt men verkningsfullt sätt att öka säkerheten. Om användare uppmuntrades att skapa något längre lösenord, kunde motståndskraften mot attacker förbättras utan att lösenorden blev onödigt svåra att komma ihåg. Studien visade därmed att längd var en praktiskt relevant faktor att beakta vid utformning av lösenordspolicyer och rekommendationer.

4.2 Hur påverkar lösenordets komplexitet motståndskraften?

För att undersöka lösenordets komplexitet genomfördes hybridattacker där antingen en till fem siffror eller ett till fem specialtecken lades till som suffix till ord från en ordbok. Resultaten visade att flest lösenord hade knäckts vid låg nivå av komplexitet, särskilt när endast en siffra eller ett specialtecken användes. När fler tecken hade lagts till minskade antalet träffar tydligt. I *Hashcat* hade nästan inga lösenord knäckts vid fyra eller fem tillagda tecken, och ett liknande mönster observerades även i *John the Ripper*.

Minskningen i antalet knäckta lösenord vid ökad komplexitet verkade främst ha berott på att verktygen inte hade hunnit testa tillräckligt många kombinationer inom den angivna tidsgränsen. Det innebär att ett högre antal tecken inte nödvändigtvis hade gjort lösenorden starkare i praktiken, utan snarare svårare att nå inom ramen för en tidsbegränsad attack.

Tidigare forskning visade att komplexitet, såsom variation i siffror, specialtecken och bokstavstyper, kunde förbättra lösenordets motståndskraft. Utöver det visade Veras m.fl. (2021) på att blandade teckenuppsättningar gjorde lösenord mindre förutsägbara, särskilt i attacker som byggde på språkliga mönster och semantik. Samtidigt beskrev Gafni m.fl. (2017) att många användare tendera att skapa lösenord som följer enkla mönster, trots att de formellt sett uppfyllde krav på komplexitet, exempelvis genom att lägga en siffra eller ett specialtecken sist.

Det som framkom i denna studie stämde överens med dessa slutsatser. Lösenord med låg grad av komplexitet lyckades knäckas med högre framgångsgrad, vilket troligen berodde på att de följde vanliga och igenkännbara strukturer. Samtidigt visade resultaten att fler tillagda tecken inte automatiskt innebar ökad säkerhet, verktygen hade inte hunnit pröva hela den större sökrymden. Detta låg i linje med vad Shi m.fl. (2021) hade påpekat, nämligen att praktiska begränsningar i beräkningstid ofta hade större betydelse än själva teckensammansättningen.

Resultaten visade att komplexitet i form av siffror och specialtecken inte automatiskt innebar ett starkare lösenord i praktiken. I de fall där endast ett tecken lades till var träffandelen som högst, vilket antyder att många användare tenderade att skapa lösenord som visserligen uppfyllde formella krav på komplexitet, men samtidigt följde förutsägbara mönster, till exempel ett ord följt av ett utropstecken eller en siffra. Sådana mönster kunde enkelt hanteras av cracking-verktygen, särskilt i hybridattacker med korta suffix.

Däremot minskade andelen knäckta lösenord kraftigt vid högre komplexitet, särskilt när fler specialtecken lades till. Det tydde på att avancerad teckensammansättning, även utan ökad lösenordslängd, kunde fungera som ett effektivt skydd, inte för att lösenorden blev oknäckbara, utan för att verktygen inte hann pröva hela den utökade sökrymden inom den

tilldelade tiden. Detta innebär att komplexitet har en reell skyddseffekt i praktiken, förutsatt att den inte implementeras enligt alltför förutsägbara mönster.

Ur ett användarperspektiv pekar detta på behovet av att utbilda kring inte bara att ett lösenord bör vara komplext, utan också hur komplexiteten utformas. Att lägga till flera varierade tecken i okonventionella positioner kan avsevärt höja motståndskraften utan att göra lösenordet oproportionerligt svårt att minnas. För systemutvecklare innebär resultaten att tekniska krav på komplexitet bör kompletteras med riktlinjer som motverkar triviala mönster, samt medverkande verktyg som hjälper användare att skapa starkare lösenord utan att göra lösenorden svåra att minnas.

4.3 Hur påverkar hashing och saltning lösenordets resistens?

Efter testning av hashingalgoritmerna, både med och utan tillagt salt, går det att fastslå att hashning av lösenord gör det svårare för password cracking-programmen att kunna få fram lösenorden.

Det fanns även en klar skillnad mellan resistensen för olika hashingalgoritmer. Enligt resultaten från studiens experiment gav hashingalgoritmen bcrypt en ökad resistens mot password cracking-programmen jämfört med en hashingalgoritm utan salt, i detta fall den osaltade versionen av SHA-256.

I studien genomfördes experiment där programmen ställdes mot hashade lösenord både med och utan salt. Mer precist var det saltad och osaltad SHA-256 som testades. Resultaten från experimenten, där dessa två sorter av SHA-256 testades, indikerade att det tillagda saltet försvårade knäckningen för password cracking-programmen. Denna skillnad fanns hos både *Hashcat* och *John the Ripper*.

En intressant aspekt är att det fanns klara skillnader mellan hur många lösenord som *Hashcat* och *John the Ripper* lyckades få fram för samma sorts attack.

Ett tydligt exempel på detta är att när en hybridattack genomfördes mot den saltade SHA-256-lösenordsdatabasen, och där programmen instruerades att lägga till en siffra, så fick *John the Ripper* fram 0 av 100 000 lösenord. För samma attack fick *Hashcat* fram 377 av 100 000 lösenord.

Samtidigt fanns det vid flera tillfällen utfall där *John the Ripper* lyckades få ut fler lösenord jämfört med *Hashcat*. Vid hybridattackerna med en siffra, gjorda mot osaltad SHA-256, hittade *John the Ripper* 51 179 av 100 000 lösenord medan *Hashcat* hittade 12 104 av 100 000 lösenord. Generellt hade *John the Ripper* större framgång jämfört med *Hashcat* vid knäckning av osaltad SHA-256.

Nämnda skillnader i resultaten mellan de två programmen visar på begränsningarna hos båda. Baserat på resultaten kan *John the Ripper* mer framgångsrikt få fram lösenord som

hashats med osaltad SHA-256. Däremot verkar *Hashcat* kunna hantera salt bättre än *John the Ripper*, även om båda programmen inte hade någon framgång mot bcrpt. En observation kring denna skillnad är att *Hashcat* har inbyggt stöd för att knäcka saltade SHA-256-hashar. När *Hashcat* instruerades att köra hybridattacker mot saltade SHA-256-hashar fanns det ett förprogrammerat format som stödde denna typ av hash. Motsvarande för *John the Ripper* var ett dynamiskt format. Troligtvis innebär detta att *Hashcat* har bättre stöd för denna typ av hash och därför presterade bättre jämfört med *John the Ripper*.

Hashning av lösenord i en databas är, och har sedan en tid tillbaka, varit en vanligt använd säkerhetsåtgärd för säker lösenordshantering. Samtidigt räcker det inte att endast hasha lösenord för att de ska vara säkra, speciellt inte om den använda hashingalgoritmen inte tillför salt. Vid läckor från databaser med användaruppgifter kan en angripare ta hashvärden av lösenord och jämföra mot listor av hashvärden av vanligt förekommande lösenord (Mellberg Granat & Gustavsson, 2017). Med program som *Hashcat* och *John the Ripper* kan denna jämförelse göras automatiskt utifrån publicerade lösenordslistor. Om endast hashingalgoritmer används bör den valda hashingalgoritmen inkludera automatisk saltning (Kanta m.fl., 2023).

Att tillföra salt till de hashade lösenorden påverkar password cracking-programmens förmåga att gissa rätt då saltning lägger till slumpmässiga tecken i lösenordshasharna. Detta gör att programmen inte kan hasha gissningar och matcha för att hitta rätt. Programmen har inte förmågan att kunna effektivt kringgå saltet i sina gissningar.

Saltning innebär dessutom att en textsträng kan få olika hashvärden även om det är samma textsträng som hashats flera gånger. Beroende på vilket salt som används, och även dess styrka, går det att få ut olika hashvärden vilket ökar säkerheten hos det hashade lösenordet (Kanta m.fl., 2023).

En viktig aspekt i att använda salt är att saltet är unikt för varje lösenord. Används samma salt till flera lösenord sänker det säkerheten av saltet. En angripare kan då enkelt få fram resten av lösenorden som saltats med samma salt, och tillintetgör därmed saltningens effekt (Mellberg Granat & Gustavsson, 2017).

Hashning och saltning är rekommenderad praxis inom säker lösenordshantering. Med koppling till resultaten av experimenten i denna studie står det klart att båda åtgärderna har en direkt påverkan på lösenordssäkerheten. Att endast hasha lösenorden är otillräckligt som en säkerhetsåtgärd. För mer heltäckande säkerhet bör lösenord även saltas. I praktiken bör aktörer därför använda sig av hashingalgoritmer med inbyggd saltning, något som görs med bcrpt.

Vid läckor av databaser med användaruppgifter spelar vilken hashingsmetoden som används en betydande roll. Som experimenten har påvisat gör det väldigt stor skillnad beroende på om det är bcrpt eller SHA-256 som använts för att hasha. Även om SHA-256 har saltats eller ej har stor påverkan på hur väl password cracking-program lyckas få ut rätt lösenord från hashvärden.

Den genomsnittliga användaren har vanligtvis inte kontroll över hur dennes lösenord hashas eller saltas i de databaser där deras användaruppgifter finns förvarade. Det användaren kan göra för att på egen hand påverka sin lösenordssäkerhet ligger i vilket lösenord denne väljer. Längden och komplexiteten hos ett lösenord är faktorer som även spelar in och avgör hashningens effekt. Väljer en användare ett kortare eller vanligt lösenord är det fortfarande osäkert även om det hashas. Anledningen till detta är att en angripare kan kringgå en säker hashingalgoritm genom att låta password cracking-program hasha och matcha gissningar för enkla lösenord. Om lösenordet dessutom är kort behöver inte programmet köra lika många gissningar som om lösenordet vore långt, då fler kombinationer av många tecken skulle behöva itereras. Används då även en säker hashingalgoritm förlängs tiden som det tar för en angripare att få fram lösenordet markant. Ett unikt lösenord som hashas med salt ger det bästa skyddet mot angripare.

4.4 Hur balanseras användarvänlighet och säkerhet vid skapandet av säkra lösenord?

Studiens resultat visade att lösenord kunde göras mer motståndskraftiga mot attacker utan att de blev särskilt svåra att använda eller komma ihåg. I flera fall räckte det med ganska enkla förändringar, som att öka längden något eller undvika typiska mönster, för att minska risken att lösenordet knäcktes. De lösenord som inte följde vanliga strukturer verkade generellt vara svårare att knäcka i de attacker som genomfördes. Även om lösenordens exakta uppbyggnad inte analyserades i detalj, antydde resultaten att verktygen var mest effektiva mot lösenord som liknade ord i ordlistan eller som passade de mönster som användes i hybridattacker.

Ett tydligt exempel var skillnaden mellan hybridattacker med siffror och de med specialtecken. Attacker där siffror lades till i slutet av ord lyckades knäcka betydligt fler lösenord än motsvarande attacker med specialtecken. Det antydde dels att många användare föredrog att använda siffror snarare än specialtecken, dels att verktygen hade lättare att hantera sådana kombinationer, särskilt vid kortare masklängder. Sannolikt berodde det på att fler vanliga lösenord med siffror fanns med i kombinationerna som testades, och att den totala sökrymden för siffror var mindre än för specialtecken, vilket innebar att fler varianter hann prövas inom den givna tidsramen.

Flera tidigare studier belyste utmaningen i att kombinera säkerhet med användarvänlighet vid lösenordsskapande. Gafni m.fl. (2017) visade att användare ofta skapade lösenord som formellt uppfyllde krav på längd och komplexitet, men som ändå följde enkla och förutsägbara mönster, till exempel ett ord följt av en siffra eller ett specialtecken. Dessa mönster upprepades även i denna studie, där just den typen av strukturer visade sig särskilt sårbara i hybridattacker.

Alkhwaja m.fl. (2023) betonade att moderna cracking-verktyg var optimerade för att snabbt identifiera vanliga mönster, vilket ytterligare försvårade balansen mellan användbarhet och skydd. Denna aspekt tydliggjordes i experimenten, där många lösenord knäcktes trots att de uppfyllde tekniska krav på variation. Resultaten låg därmed i linje med litteraturens beskrivning av hur verktygens effektivitet gör det otillräckligt att enbart uppmana användare att öka komplexiteten, utan att också förklara hur komplexitet bör utformas.

De praktiska resultaten visade att säkerheten kunde förbättras utan att lösenorden blev särskilt svåra att komma ihåg. Lösenord som var något längre än genomsnittet, eller som använde specialtecken på ett mindre förutsägbart sätt, klarade sig genomgående bättre i attacker, även om de i övrigt var relativt enkla. Det innebär att god motståndskraft inte krävde att lösenorden var speciellt komplexa, utan snarare att de inte följde mönster som verktygen snabbt kunde identifiera.

I flera fall var skillnaden tydlig mellan lösenord med en siffra i slutet och lösenord som använde specialtecken. De förstnämnda knäcktes i högre utsträckning, vilket tyder på att användares vanor att uppfylla minimikrav på ett standardiserat sätt minskade lösenordets effektivitet. Det visade att policyer som enbart ställer krav på variation kan leda till mönsterbildning snarare än ökad säkerhet. En rekommendation baserat på denna observation är att riktlinjer kring skapandet av lösenord inte ska begränsa användaren till endast ett fåtal specifikationer. För att undvika att skapa mönster som kan utnyttjas av password cracking-program bör användaren, när denne ska skapa ett nytt lösenord, ges en utförlig lista över en stor mängd kriterier för säkra lösenord. Kriterierna bör vara formulerade på ett sätt som ger användaren friheten att själv lägga nivån för både hur långt och komplext lösenordet ska vara, men samtidigt vara säkert.

Samtidigt visade resultaten att användarvänlighet inte behövde stå i direkt konflikt med säkerhet. En viss ökning i längd och variation, utan att lösenorden blev orimligt svåra att minnas, kunde ge påtagliga förbättringar i motståndskraften. Det tydde på att tydlig vägledning i hur komplexitet bör utformas, i kombination med tekniska skydd som saltning, skapade bättre förutsättningar för att uppnå både säkerhet och användarvänlighet.

4.5 Hur långt och komplext behöver ett lösenord vara, och vilka faktorer påverkar dess säkerhet?

För att skapa resistent lösenord som kan motstå välanvända password cracking-program, som *Hashcat* och *John the Ripper*, visade resultaten att det inte räckte att endast lägga till enstaka tecken i slutet av ett ord. Flera av lösenorden som knäcktes innehöll just en siffra

eller ett specialtecken, vilket tydde på att sådana kombinationer var vanliga och därmed prioriteraprioriterades prioriterades av verktygen. Bland dessa två typer av tecken visade sig dock att lösenord där en siffra lades till, knäcktes i större utsträckning än de lösenorden där ett specialtecken lades till. Det visade på att specialtecken kunde ge något högre skydd, förutsatt att de inte följde etablerade mönster. Det minskar dock också användarvänligheten vid användning av många specialtecken.

Lösenordslängden spelade också en avgörande roll för säkerheten. Vid analys av längdfördelningen i *Hashcat* så visade det att 30,58% respektive 33,54% lösenord på fem och sex tecken knäcktes (se Tabell 1). Efter det sjönk andelen markant, vid sju tecken sjönk andelen knäckta lösenord till 18,18%. För längre lösenord fortsatte andelen som lyckades knäckas sjunka lägre. Vid analysen av lösenorden som lyckades knäckas i *John the Ripper* så lyckades en större andel av lösenorden knäckas. Analysen visade att 77,47% respektive 75,63% av lösenorden med fem och sex tecken knäcktes, här knäcktes också 70,82% av lösenorden som bestod av sju tecken. Först vid åtta tecken sjönk andelen knäckta lösenord markant till 48,82% och fortsatte sedan sjunka ju längre lösenordslängden blev.

Utöver längd och komplexitet så visade studien även att valet av hashfunktion och användningen av salt påverkade lösenordens motståndskraft. Attackerna som genomfördes mot osaltade SHA-256 hashar knäckte betydligt fler lösenord än vid motsvarande attacker mot hashar där varje lösenord kombinerats med ett unikt salt. Saltningen gjorde inte själva lösenordet starkare men det förhindrar att samma hashvärde uppstod för identiska lösenord. Det tvingade verktygen att testa varje potentiella lösenord mot varje salt, vilket ökade den beräkningsmässiga belastningen avsevärt.

Resultaten visade tydligt att detta hade en praktisk effekt. Attackerna som lyckades knäcka över tolv tusen osaltade lösenord, knäckte inte ens fyrahundra saltade lösenord. Det bekräftade att var en effektiv metod för att öka den faktiska tiden som krävdes för att genomföra en lyckad attack, även när lösenordet inte var särskilt långt eller komplext.

Tidigare forskning har lyft fram att lösenordets längd och komplexitet är avgörande faktorer för dess säkerhet. Kanta m.fl. (2023) betonade att varje tillagt tecken i ett lösenord förlänger den tid som krävs för en crackingattack genom att utöka sökrymden exponentiellt. Även Shi m.fl. (2021) visade att längre lösenord ökade motståndskraften i offline-sammanhang, där angriparen inte är begränsad av antal inloggningsförsök. Dessa slutsatser stämde överens med resultaten i denna studie, där andelen knäckta lösenord sjönk tydligt vid ökad längd, särskilt vid åtta tecken och uppåt.

Veras m.fl. (2021) lyfte samtidigt vikten av komplexitet, särskilt blandning av olika teckentyper, för att motverka attacker som utnyttjar språkliga eller strukturella mönster. Denna princip kunde även observeras i experimenten, där lösenord med specialtecken uppvisade större motståndskraft än lösenord där endast siffror lades till. Flera tidigare studier har också understrukit att många användare skapar lösenord som visserligen uppfyller formella krav, men som ändå följer förutsägbara mönster (Gafni m.fl., 2017),

vilket förklarar varför vissa typer av komplexitet inte nödvändigtvis gav ökad säkerhet i praktiken.

Slutligen visade resultaten i denna studie att även tekniska faktorer som hashfunktion och saltning hade stor betydelse, vilket låg i linje med Al-Aboosi m.fl. (2022) och Poston (2023), som båda betonade att säkerheten påverkas av hur lösenord hanteras på systemnivå. Güven m.fl. (2022) stärkte denna helhetsbild genom att visa att svaga och återanvända lösenord ofta förekommer, och att effektiv säkerhet kräver samverkan mellan tekniska skydd och medvetna användarbeteenden.

Litteraturen som helhet bekräftade alltså det som denna studie visade, att ett säkert lösenord inte bara handlar om längd eller komplexitet i isolering, utan om kombinationen av flera faktorer, både mänskliga och tekniska.

Resultaten från studien visade att tekniska skyddsåtgärder, som saltning, hade mycket stor effekt på lösenordens motståndskraft, något som i praktiken inte är synligt för användaren. För individen innebar detta att det inte endast var valet av lösenord som har betydelse, utan hashfunktionerna som systemet använde också spelade en avgörande roll. Även starkare lösenord kunde knäckas om de hashades utan salt, samtidigt som enklare lösenord kunde förbli oknäckta när varje lösenord förstärktes med unika salt.

Därför bör det i praktiken inte bara ställas krav på användarna att skapa säkra lösenord, utan också på systemansvariga att tillämpa moderna skyddsmekanismer som saltning och säkra hashfunktioner. Resultaten visade att denna aspekt av säkerheten hade större praktisk effekt än vad användare ofta har insyn i.

En annan implikation var att små förändringar i lösenordets struktur, exempelvis att lägga till specialtecken istället för siffror kunde ge högre skydd, samtidigt som det påverkar användarvänligheten. Det understryker vikten av tydlig vägledning till användare, inte bara att lösenordet ska vara komplext, utan vad det konkret innebär. Ett exempel på vad detta skulle kunna innebära i praktiken är att rekommendationer om att använda specialtecken på oförutsägbara ställen i lösenordet, istället för att enbart uppfylla krav på variation.

Slutligen så visade resultaten att säkerheten inte nödvändigtvis kräver extrema krav. Lösenord som var något längre än genomsnittet och inte följde vanliga mönster klarade sig betydligt bättre än förväntat. Vilket antyder att det finns utrymme att hitta policyer som balanserar användarvänlighet och säkerhet, utan att det riskerar leda till att användare glömmer eller återanvänder sina inloggningsuppgifter.

5 Slutsatser

Syftet med denna studie var att undersöka hur långt och komplext ett lösenord behöver vara för att motstå moderna password cracking-metoder, samt vilka faktorer som påverkade dess säkerhet. Genom experiment med *Hashcat* och *John the Ripper*, i kombination med en litteraturoversikt, identifierades flera avgörande mönster.

Resultaten visade att längre lösenord i regel var svårare att knäcka, särskilt i hybridattacker där sökrymden ökade kraftigt med varje tillagt tecken. Dock var inte längd ensamt avgörande, lösenord med oförutsägbara mönster och varierad struktur stod i många fall emot crackingförsök bättre än längre, men mer förutsägbara, lösenord.

Hashning och saltning hade stor inverkan på motståndskraften. Osaltad SHA-256 knäcktes lättare, medan saltning avsevärt försvårade attackerna. Bcrypt visade sig vara mest effektiv, då inga attacker lyckades inom tidsramen.

Slutsatsen är att säkra lösenord bör vara minst tolv tecken långa, innehålla variation i teckentyper och undvika vanliga mönster. Samtidigt är tekniska skydd som saltning och robusta hashfunktioner avgörande. Måttligt längre lösenord med viss variation och utan vanliga mönster gav ett skydd som var tillräckligt för att stå emot kraftfulla cracking-verktyg inom rimlig tid. Det pekar på att kombinationen av välinformerade riktlinjer och moderna tekniska skydd är avgörande för att skapa ett hållbart lösenordsskydd, både för individer och systemansvariga.

5.1 Framtida arbeten

Denna studie fokuserade specifikt på hur längd, komplexitet, hashning och saltning påverkar lösenords förmåga att stå emot cracking-verktyg. Områden som skulle kunna fördjupas mer är att undersöka fler cracking-verktyg än *John the Ripper* och *Hashcat*. Exempelvis verktyg som använder sig av AI eller maskininlärning för att knäcka lösenord, som bland annat PASSGAN. Detta hade kunnat ge en bättre bild av hur olika typer av attacker fungerar i praktiken.

Det hade också varit intressant att inkludera fler hashningsalgoritmer i experimentet, såsom Argon2 eller scrypt, som är utformade för att stå emot moderna crackingförsök. I ett tidigt skede testades Argon2 med *John the Ripper*, men algoritmen visade sig vara så resurseffektiv att inga lösenord hann knäckas inom den angivna tidsramen. Det gjorde det svårt att dra meningsfulla jämförelser mot övriga hashfunktioner. Därför föll Argon2 bort i den slutliga analysen. Dessutom saknade ett av programmen som användes i denna studie, närmare bestämt *Hashcat*, stöd för denna hashingalgoritm. I framtida studier, med längre körningar eller kraftfullare maskinvara, vore det dock möjligt att inkludera den typen av algoritmer för att undersöka deras motståndskraft mer grundligt.

En annan möjlighet skulle vara att använda större datamängder eller köra testerna i en längre tid än en timme, vilket var gränsen i denna studie. Det hade kunnat ge mer heltäckande resultat för längre och mer komplexa lösenord. Det är också värt att fundera på hur användare faktiskt skapar och kommer ihåg lösenord, vilket är något som framtida studier skulle kunna kombinera med experiment för att förstå hur vardagliga människor resonerar.

6 Referenser

- Al-Aboosi, A. F., Broner, M., & Al-Aboosi, F. Y. (2022). Bingo: A Semi-Centralized Password Storage System. *Journal of Cybersecurity and Privacy*, 2(3), 444. <https://doi.org/10.3390/jcp2030023>
- Alkhwaja, I., Albugami, M., Alkhwaja, A., Alghamdi, M., Abahussain, H., Alfawaz, F., Almurayh, A., & Min-Allah, N. (2023). Password Cracking with Brute Force Algorithm and Dictionary Attack Using Parallel Programming. *Applied Sciences*, 13(10), 5979-. <https://doi.org/10.3390/app13105979>
- Chraïbi, A. (2004). Galland's "Ali Baba" and Other Arabic Versions. *Marvels & Tales*, 18(2), 159–169.
- Darktrace. (n.d.). *Password Cracking: How It Works & How to Protect Your Accounts*. <https://darktrace.com/cyber-ai-glossary/password-cracking>
- Domarboken 12. (2017). <https://www.bibeln.se/bibel/B2000/JDG.12>
- EU 2016/679. *Europaparlamentets och rådets förordning (EU) 2016/679 av den 27 april 2016 om skydd för fysiska personer med avseende på behandling av personuppgifter och om det fria flödet av sådana uppgifter och om upphävande av direktiv 95/46/EG (allmän dataskyddsförordning) (Text av betydelse för EES)* (2016). Tillgänglig: <http://data.europa.eu/eli/reg/2016/679/oj/swe>
- Eum, S., Kim, H., Song, M., & Seo, H. (2023). Optimized Implementation of Argon2 Utilizing the Graphics Processing Unit. *Applied Sciences*, 13(16), 9295-. <https://doi.org/10.3390/app13169295>
- Gafni, R., Pavel, T., Margolin, R., & Weiss, B. (2017). Strong password? Not with your social network data. *The Online Journal of Applied Knowledge Management*, 5(1), 27–41. [https://doi.org/10.36965/OJAKM.2017.5\(1\)27-41](https://doi.org/10.36965/OJAKM.2017.5(1)27-41)

Google. (n.d.). *Tips om hur du skyddar dig online*.

<https://safety.google/security/security-tips/>

Grigutyte, M. (2023, 16 juni). *What is Bcrypt and how it works? | NordVPN*.

<https://nordvpn.com/sv/blog/what-is-bcrypt/#bcrypt-vs-sha256>

Güven, E. Y., Boyaci, A., & Aydin, M. A. (2022). A Novel Password Policy Focusing on Altering User Password Selection Habits: A Statistical Analysis on Breached Data. *Computers & Security, 113*, 102560.

<https://doi.org/10.1016/j.cose.2021.102560>

Hashcat.net. (n.d.a). *Hashcat*. <https://hashcat.net/wiki/doku.php?id=hashcat>

Hashcat.net. (n.d.b). *Hybrid_attack*. https://hashcat.net/wiki/doku.php?id=hybrid_attack

Hashcat.net. (n.d.c). *Mask_attack*. https://hashcat.net/wiki/doku.php?id=mask_attack

Hornby, T. (2021, september 28). *Secure Salted Password Hashing—How to do it Properly*. <https://crackstation.net/hashing-security.htm#normalhashing>

Internetkunskap. (n.d.). *Har mitt lösenord läckt – och vad kan jag göra?*

<https://internetkunskap.se/har-mitt-losenord-lackt/>

Kali. (2024, 9 september). *What is Kali Linux? | Kali Linux Documentation*.

<https://www.kali.org/docs/introduction/what-is-kali-linux/>

Kanta, A., Coisel, I., & Scanlon, M. (2023). Harder, better, faster, stronger: Optimising the performance of context-based password cracking dictionaries. *Forensic Science International. Digital Investigation (Online)*, *44*, 301507-.

<https://doi.org/10.1016/j.fsidi.2023.301507>

Kanta, A., Coray, S., Coisel, I., & Scanlon, M. (2021). How viable is password cracking in digital forensic investigation? Analyzing the guessability of over 3.9 billion real-world accounts. *Forensic Science International. Digital Investigation (Online)*, *37*, 301186-. <https://doi.org/10.1016/j.fsidi.2021.301186>

- Lennon, B. (2015). Passwords: Philology, Security, Authentication. *Diacritics*, 43(1), 82–104. <https://doi.org/10.1353/dia.2015.0000>
- Lurey, C. (2023, 4 augusti). *Understanding RockYou.txt: A Tool for Security and a Weapon for Hackers*.
<https://www.keepersecurity.com/blog/2023/08/04/understanding-rockyou-txt-a-tool-for-security-and-a-weapon-for-hackers/>
- Mani, A. K., & Adedayo, O. M. (2024). Dynamic Calculation of Password Salts for Improved Resilience towards Password Cracking Algorithms. I *2024 12th International Symposium on Digital Forensics and Security (ISDFS)* (s. 1–6).
<https://doi.org/10.1109/ISDFS60797.2024.10527313>
- Marchetti, K., & Bodily, P. (2022). John the Ripper: An Examination and Analysis of the Popular Hash Cracking Algorithm. I *2022 Intermountain Engineering, Technology and Computing (IETC)* (s. 1–6).
<https://doi.org/10.1109/IETC54973.2022.9796671>
- Mellberg Granat, A., & Gustavsson, J. (2017). *Lösenordsstrategier på internet*. Hämtad från <https://urn.kb.se/resolve?urn=urn:nbn:se:du-25551>
- Openwall. (n.d.). *John the Ripper password cracker*. <https://www.openwall.com/john/>
- Passwords have a long history – how much do you know...?* (n.d.).
<https://newsroom.cisco.com/c/r/newsroom/en/us/a/y2022/m11/security-timeline.html>
- Poston, H. (2025, 27 januari). *Top 10 Password Cracking Tools for User Authentication / Infosec*. <https://www.infosecinstitute.com/resources/hacking/10-popular-password-cracking-tools/>

- Schmitt, A. (2024, 28 oktober). *History of passwords: Where passwords come from/ mail.com blog*. https://www.mail.com/blog/posts/history-of-security-password/146/?utm_referrer=www.google.com
- ScienceDirect. (2016). *Hashing Function—An overview*.
<https://www.sciencedirect.com/topics/computer-science/hashing-function>
- Shi, R., Zhou, Y., Li, Y., & Han, W. (2021). Understanding Offline Password-Cracking Methods: A Large-Scale Empirical Study. *Security and Communication Networks*, 2021, 1–16. <https://doi.org/10.1155/2021/5563884>
- Shouling Ji, Shukun Yang, Xin Hu, Weili Han, Zhigong Li, & Beyah, R. (2017). Zero-Sum Password Cracking Game: A Large-Scale Empirical Study on the Crackability, Correlation, and Security of Passwords. *IEEE Transactions on Dependable and Secure Computing*, 14(5), 550–564.
<https://doi.org/10.1109/TDSC.2015.2481884>
- Šimonėlytė, M. (2023, 26 mars). *7 regler som håller dig säker på nätet*.
<https://nordvpn.com/sv/blog/sakerhet-pa-natet/>
- Tatlı, E. İ. (2015). Cracking More Password Hashes With Patterns. *IEEE Transactions on Information Forensics and Security*, 10(8), 1656–1665.
<https://doi.org/10.1109/TIFS.2015.2422259>
- The Speakeasies of the 1920s*. (n.d.). <https://prohibition.themobmuseum.org/the-history/the-prohibition-underworld/the-speakeasies-of-the-1920s/>
- Veras, R., Collins, C., & Thorpe, J. (2021). A Large-Scale Analysis of the Semantic Password Model and Linguistic Patterns in Passwords. *ACM Transactions on Privacy and Security*, 24(3), 1–21. <https://doi.org/10.1145/3448608>
- Waksman, M. (2013, 11 juni). *"Open Sesame!" – Is Your Password So Easy to Guess?*
<https://jetico.com/blog/open-sesame-your-password-so-easy-guess/>

- Wasson, D. L. (2021, 22 mars). *Officers of the Roman Army*.
<https://www.worldhistory.org/article/1711/officers-of-the-roman-army/>
- Weber, J. E., Guster, D., Safonov, P., & Schmidt, M. B. (2008). Weak Password Security: An Empirical Study. *Information Systems Security, 17*(1), 45-.
- Wilson, B. (2025, 24 februari). *Seclists*. <https://gitlab.com/kalilinux/packages/seclists>
- Yao, M., Xue, Z., Li, H., & Shen, S. (2024). An optimized hardware implementation of SHA-256 round computation. *Computer Journal*.
<https://doi.org/10.1093/comjnl/bxae116>
- Zimmermann, V., & Gerber, N. (2020). The password is dead, long live the password – A laboratory study on user perceptions of authentication schemes. *International Journal of Human-Computer Studies, 133*, 26–44.
<https://doi.org/10.1016/j.ijhcs.2019.08.006>