TALLINNA TEHNIKAÜLIKOOL
INSENERITEADUSKOND
Virumaa kolledž

Kollisiooni leidmine parooliga kaitstud arhiivi murdmisel

Finding a collision when breaking a password-protected archive

Telemaatika ja arukad süsteemid ÕPPEKAVA LÕPUTÖÖ

Üliõpilane: Ksenia Kostyuk
Üliõpilaskood: 193083EDTR
Juhendaja: Larissa Joonas, lektor

Kohtla-Järve, 2023

AUTORIDEKLARATSIOON

Olen koostanud lõputöö iseseisvalt.

Lõputöö alusel ei ole varem kutse- või teaduskraadi või inseneridiplomit taotletud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

".... " ..... 20..... .

Autor: .....

/ allkiri /

Töö vastab rakenduskõrgharidusõppe lõputööle/magistritööle esitatud nõuetele "... "
..... 20..... .

Juhendaja: .....

/ allkiri /

Kaitsmisele lubatud ".... " ..... 20..... .

Kaitsmiskomisjoni esimees .....

/ nimi ja allkiri /

LIHTLITSENTS LÕPUTÖÖ ÜLDSUSELE KÄTTESAADAVAKS TEGEMISEKS JA REPRODUTSEERIMISEKS

Mina Ksenia Kostyuk (sünnikuupäev: 01.12.2000)

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose Kollisiooni leidmine parooliga kaitstud arhiivi murdmisel, mille juhendaja on Larissa Joonas,

1.1. reprodutseerimiseks säilitamise ja elektroonilise avaldamise eesmärgil, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. Olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. Kinnitan, et lihtlitsentsi andmisega ei rikuta kolmandate isikute intellektuaalomandi ega isikuandmete kaitse seadusest ja teistest õigusaktidest tulenevaid õigusi.

TalTech Inseneriteaduskond Virumaa kolledž

LÕPUTÖÖ ÜLESANNE

Üliõpilane: Ksenia Kostyuk, t193083

Õppekava, peaeriala: EDTR17/18 ja Telemaatika ja arukad süsteemid
Juhendaja(d): lektor, Larissa Joonas, larissa.joonas@taltech.ee

Lõputöö teema:

(eesti keeles) Kollisiooni leidmine parooliga kaitstud arhiivi murdmisel

(inglise keeles) Finding a collision when breaking a password-protected archive

Lõputöö põhieesmärgid:

1. Krüptograafiliste funktsioonide kollisiooni leidmine ja tulemuste analüüs

Lõputöö etapid ja ajakava:

Nr	Ülesande kirjeldus	Tähtaeg
1.	Krüptograafiliste funktsioonide ja standardite uurimine	03.10.2022
2.	Paroolide murdmise meetodite ja vahendite uurimine	17.11.2022
3.	Brute Force rünnaku paroolide murdmise rakendamine	10.03.2023
4.	PBKDF2-funktsiooni kollisiooni leidmine	21.04.2023
5.	Tulemuste analüüs	10.05.2023

Töö keel: eesti keel **Lõputöö esitamise tähtaeg:** "…. "…….. 20…. a

Üliõpilane: Ksenia Kostyuk "…. "…….. 20…. a

/allkiri/

Juhendaja: Larissa Joonas "…. "…….. 20…. a

/allkiri/

Programmijuht: Žanna Gratšjova "…. "…….. 20…. a

/allkiri/

SISUKORD

EESSÕNA 7
LÜHENDITE JA TÄHISTE LOETELU 8
SISSEJUHATUS 9
1. PAROOLIDE TÄHTSUS JA NENDE MURDMINE 10
1.1. Paroolide murdmise tüübid 10
1.1.1. Brute force'i rünnaku meetod 10
1.1.2. Sõnastikurünnaku meetod 10
1.1.3. Vikerkaaretabeli rünnaku meetod 11
1.1.4. Rünnakumeetod "sünnipäeva paradoks" 11
1.1.5. Rünnakumeetod "Kohtumine keskel" 11
1.1.6. Rünnakumeetod, mis kasutab andmelekkeid kõrvalkanalites 11
1.2. Artikkel kollisiooni kohta zip arhiivides 12
2. ARHIIVI TURVALISUSE ANALÜÜS PAROOLIGA 13
2.1. Arhiivid 13
2.1.1. Arhiivide kasutamine 13
2.1.2. Arhiivitüübid 13
2.2. Krüptograafia 13
2.2.1. Krüptograafia kasutamine 13
2.2.2. Krüptograafia peamised funktsioonid 14
2.3. PBKDF2 funktsioon 14
2.4. PBKDF2-s kasutatavad funktsioonid 16
2.5. Krüptograafilised räsifunktsioonid 16
2.5.1. Omadused, mis peaksid olema krüptograafilisel räsifunktsioonil 16
2.5.2. Laviiniefekti, kollisiooni ning esimese ja teise astme kollisiooni mõisted 16
2.6. Algoritm SHA-1 17
2.6.1. Krüpteerimisprotsess SHA-1 räsialgoritmiga 18
2.7. Algoritm AES-256 20
2.7.1. Krüpteerimisprotsess AES räsialgoritmiga 21
3. BRUTE FORCE PAROOLIDE MURDMINE 24
3.1. Paroolid 24
3.1.1. Paroolide pikkus 24
3.1.2. Tähemärgid paroolis 24

3.1.3. Parooli häkkamise aeg sõltuvalt sümbolite arvust aastal 2022 25
3.2. Paroolide murdmise tööriistad 26
3.3. Hashcat 26
3.3.1. Hashcat'i paigaldamine 26
3.3.2. Programmiga tutvumine 27
3.3.3. Sõnastikurünnak 28
3.3.4. Brute-force rünnak 28
3.4. Raskused 29
3.5. Parooli murdmine kollisiooni leidmise meetodil 29
3.5.1. Kood Python'i keeles 29
3.6. Saadud tulemuste analüüs 30
3.6.1. Kui ohtlik on selliste paroolide paari olemasolu? 31
3.6.2. PBKDF2 funktsiooni rakendamine ja selle tähtsus 31
3.6.3. Kas selliseid paarid (parool ja räsikood) on võimalik kasutada teistes programmides? 31
3.6.4. Nõuanded arhiivi täiendavaks kaitsmiseks 33
KOKKUVÕTE 34
SUMMARY 35
LISA 1. HASH-FUNKTSIOONIDEGA SEOTUD MÕISTED 40
LISA 2. SALASÕNA VALIMISE JA MURDMISE PROGRAMMID 41
LISA 3. HASHCAT HELP'I JUHENDIST SAADUD TEAVE 43

EESSÕNA

Lõputöö käsitleb krüpteeritud zip-arhiividele kollisiooni leidmist, mis põhineb neis kasutatavas PBKDF2-funktsioonis.

Tahaksin avaldada tänu õppejõule Larissa Joonasele abi eest tema algatusel sõnastatud huvitava teema leidmisel lõputöö jaoks. Samuti soovin teda tänada motivatsiooni hoidmise, tagasiside, nõuannete ja juhendamise eest kogu lõputöö kirjutamise protsessi jooksul.
Tahaksin tänada ka õppejõud Ingrid Preesi ja kaasüliõpilasi, kes on mind toetanud, motiveerinud ja aidanud lõputöö jätkamisel.
Võtmesõnad: krüptograafia, krüpteerimisalgoritmid, hash-funktsioon, kollisioon, diplomitöö.

LÜHENDITE JA TÄHISTE LOETELU

AES-256	progressiivne krüpteerimisstandard (ingl k *Advanced Encryption Standard*)
AND	NING loogiline operatsioon (ingl k The AND logic operation)
ASCII	infovahetuse Ameerika standardkood (ingl k *American Standard Code for Information Interchange*)
hash	räsi väärtus (ingl k Hash value)
hex	kuueteistkümnendsüsteem (ingl k Hexadecimal system)
k	Kuu
n	Nädal
NIST	Riiklik Standardi- ja Tehnikainstituut (ingl k *National Institute of Standards and Technology*)
NOT	inversioon loogiline operatsioon (ingl k The NOT logic operation)
NSA	Riikliku Julgeoleku Amet (ingl k *National Security Agency*)
OR	VÕI loogiline operatsioon (ingl k The OR logic operation)
p	Päev
PBKDF2	paroolipõhine võtmetuletusfunktsioon 2, paindlikum ja turvalisem kui PBKDF1 (ingl k *Password-Based Key Derivation Function2*)
SHA-1	turvaline räsialgoritm (ingl k *Secure Hash Algorithm 1*)
t	tund
tuh	tuhat
XOR	välistav või loogiline operatsioon (ingl k The XOR logic operation)

SISSEJUHATUS

21. sajandil, kui igapäevaelu igale täiskasvanule, teismelisele või lapsele puutub kokku IT-tehnoloogiatega, on nende kasutamisel turvalisus väga oluline. Sellele tuleb pöörata erilist tähelepanu, kuna teeme oste nii kauplustes kui ka veebipoodides, meie andmeid töötlevad pangad ja mobiilioperaatorid. Kui krüptograafiat poleks olemas, võiksid meie isiklikud andmed sattuda teiste inimeste kätte, kellele need pole mõeldud, ning ka kurjategijatele, mis tooks kaasa suuri probleeme. Probleemid võivad puudutada nii üksikisikut kui ka tervet ettevõtet. Andmete kaotus võib lihtsalt rikkuda inimese mainet, kui isiklik info satub võrku, ning rikkuda terve tootmise tööd ja võib-olla ka pankrotistada selle. Seetõttu peab iga inimene mõtlema turvalisusele nii kodus, koolis kui ka töökohal.

Ma olin väga huvitatud krüptograafia teemast küberkaitse ja küberturvalise loengutes ning mõistsin, et soovin ühenda oma lõputöö krüptograafiaga seotud teemaga. Augustis 2022 ilmus artikkel [1], kus öeldi, et kahe parooliga pääseb ligi krüpteeritud zip-arhiividele. See uudis üllatas mind, sest see rikub teabe turvalist salvestamist arhiivis, millel on parool. Hakkas mind huvitama, kui tõenäoline on seda korrata ja proovida arhiivi selle meetodiga häkkida. Positiivse tulemuse korral huvitas mind, kui palju aega selleks kulub. Kuna konkreetset lahendust ei olnud, vaid ainult teiste kasutajate oletused, otsustasin proovida rakendada pakutud lahendust.

Minu töö eesmärk on:

- Krüptograafiliste funktsioonide kollisiooni leidmine ja tulemuste analüüs

On püstitatud 6 ülesannet:

- krüptograafiliste funktsioonide ja standardite uurimine;
- paroolide murdmise meetodite ja vahendite uurimine;
- Brute Force paroolide murdmise rakendamine;
- PBKDF2-funktsiooni kollisiooni leidmine;
- tulemuste analüüs;
- lõputöö vormistamine.

1. PAROOLIDE TÄHTSUS JA NENDE MURDMINE

Tänapäeval kasutatavate traadita mobiiltehnoloogiate arv suureneb pidevalt ning sellega koos kasvab ka turvarikkumiste oht. Seetõttu on olulisem kui kunagi varem kaitsta oma konfidentsiaalset teavet nende inimeste eest, kelle käes see võib meile, meie lähedastele või meie tööle kahju teha. Praktiliselt igal veebilehel on vajalik registreeruda ja sisse logida oma kasutajanime ja parooliga. See võib olla töö- või isiklik e-post, kus saabuvad näiteks korteriarved, vanema konto koolis, veebipood, internetipank ning palju muud. Iga päev kasutatakse suurt hulka veebisaite, mis võib tekitada soovi kasutada sama parooli erinevatel veebisaitidel või muuta ainult üks või kaks sümbolit. Kahjuks võib see viia isikliku teabe kompromiteerimiseni ja selle kasutamiseni inimese vastu. [2]

1.1. Paroolide murdmise tüübid

On palju võimalusi paroolide murdmiseks. Autor tutvus erinevate paroolide murdmise meetoditega ja kirjeldas allpool oma arvates kõige huvitavamaid.

1.1.1. Brute force'i rünnaku meetod

Toore jõu ehk Brute-force meetod eeldab objektide (näiteks räsisõnumite) kõikide võimalike variantide järjestikust proovimist otsitava väärtuseni (näiteks sõnum, mis vastab määratud räsi väärtusele). [3]
Brute-force rünnakuid saab kasutada kõikide võimalike rünnakueesmärkide saavutamiseks krüpteerimisalgoritmide vastu [3]:

- esimese ja teise astme kollisiooni otsimine;
- salajase võtme tuvastamine.

1.1.2. Sõnastikurünnaku meetod

Sõnastikurünnak ehk (dictionary attack) on meetod mingi informatsiooni avastamiseks võimalike variantide proovimist (sõnavara rünnakud on suunatud kasutajate paroolidele, mis on salvestatud nende räsiväärtustena). Rünnaku nimi tuleneb sellest, et suur osa proovitavatest paroolidest koosneb mõne keele sõnadest. Enamik kasutajaid kasutab kergesti meeldejäävaid paroole, mille koosseisu kuuluvad sageli erinevad sõnad ja nende variandid. Inimesed asendavad sõnaosa tähed sarnase kirjutusviisiga numbrid või erimärgid, mis teevad nad sõnastiku rünnakutele haavatavaks. Sõnastikurünnakuid liigitatakse sageli üheks meetodiks "toore jõu meetodi" rünnakuvariantidest, kuna siin tehakse samuti läbi parooli või võtme variantide järjestikune proovimine. Võrreldes "toore jõu meetodi" meetodiga teostavad sõnastikurünnakud proovimist oluliselt väiksemate väärtuste seas. [3]

1.1.3. Vikerkaaretabeli rünnaku meetod

Vikerkaaretabel ehk (Rainbow Table Attack) on rünnak, mis ei ole suunatud mitte parooli enda, vaid krüpteerimismeetodi vastu, mis tagab parooli turvalise salvestamise räsi väärtusena. Vikerkaaretabel on tohutu paroolide andmebaas, kus iga avatud tekstiparool vastab tema räsi väärtusele. Küberkurjategijad võivad võrrelda kasutaja parooli räsiväärtust kõigi andmebaasi räsiväärtustega, et teada saada, milline avatud tekstiparool on seotud selle räsiväärtusega. Lisaks võib mitu avatud tekstiparooli anda sama räsi väärtuse, mis muudab vikerkaaretabeli küberkurjategijate jaoks väga tõhusaks tööriistaks, kuna neil pole vaja teada tegelikku parooli, et kontole pääseda. [4]

1.1.4. Rünnakumeetod "sünnipäeva paradoks"

Sünnipäevade paradoksil põhinev rünnak näitab, et kahe inimese sünnipäevade kokkulangemise tõenäosus on palju suurem kui tavaliselt arvatakse. Näiteks 23-liikmelise grupi puhul on sünnipäevade kokkulangemise tõenäosus 50%. Samamoodi on tõenäosus avastada kollisioone sihträsifunktsioonide vahel märgatavalt suurem, kui arvatakse, mis võimaldab ründajal leida kattuvaid fragmente, kasutades väiksemat arvu iteratsioone. [5]

1.1.5. Rünnakumeetod "Kohtumine keskel"

See rünnak on suunatud algse sõnumi otsimisele, mis vastab määratud räsi väärtusele. Rünnak "Kohtumine keskel " on üks rünnaku variante, mis kasutab sünnipäevade paradoksi, kuigi seda ei kasutata kollisioonide leidmiseks. [3]

1.1.6. Rünnakumeetod, mis kasutab andmelekkeid kõrvalkanalites

Kaitstud räsi loomiseks tarkvara või seadmete abiga, võib see põhjustada lekkeid meie töödeldavate andmete ja/või salajase võtme kohta, mida me kasutame tema loomiseks. [3]

Sellised lekked võivad olla näiteks järgmised [3]:

- elektromagnetkiirgus;
- informatsioon andmetöötluse aja- ja võimsuskulude kohta;
- informatsioon räsimisprotsessis esinevate vigade kohta jne.

Seda teavet võib krüptoanalüütik kasutada räsimisalgoritmi ründamisel, samuti muude rünnakute kontekstis. Selliseid rünnakuid nimetatakse rünnakuteks, mis kasutavad andmelekkeid kõrvalkanalites (side-channel attacks). Tasub siiski öelda, et sagedamini kasutatakse neid rünnakuid sümmeetrilise krüpteerimise algoritmide vastu. [3]

1.2. Artikkel kollisiooni kohta zip arhiivides

Augustis 2022 ilmus twitteris artikkel selle kohta, et krüptitud zip-arhiivile sobivad kaks parooli. Selles artiklis [1] on kasutaja Arseniy Sharoglazovi nime all näidanud, et ühe arhiivi lahtipakkimiseks võib kasutada 2 parooli "Nev1r-G0nna-G2ve-Y8u-Up-N5v1r-G1nna-Let-Y4u-D1wn-N8v4r-G5nna-D0sert-You" ja "pkH8a0AqNbHcdw8GrmSp". Kuid Sharoglazov ei selgitanud selle nähtuse olemust, kuid leidis kasutaja nimega Unblvr, kes esitas oma eelduse: «ZIP kasutab PBKDF2, mis räsib sisendit, kui see on liiga suur. See räsi (töötlemata baitidena) muutub tegelikuks parooliks. Proovige räsida esimene parool kasutades SHA-1 ja dekodeerida kuueteistkümnendsüsteemis ASCII...:)» [6]

Autor tegi selle eelduse ja sai veebilehelt [7] räsi väärtuse reale "Nev1r-G0nna-G2ve-Y8u-Up-N5v1r-G1nna-Let-Y4u-D1wn-N8v4r-G5nna-D0sert-You" ning sai rea "706b4838613041714e62486364773847726d5370". Järgnevalt teisendati see rida teise veebilehega [8] ja üllataval oli vastus tõesti nii "pkH8a0AqNbHcdw8GrmSp" näitel. [9]

Autor uuris oma töös parooliga kaitstud arhiivi murdmise võimalusi ja krüptograafilise lahenduse leidmist selle murdmiseks. Et täielikult mõista, kuidas arhiivides kasutatav funktsioon toimib ja miks see võimaldab kollisiooni, on autor AES-256 ja SHA-1 tegevust detailsemalt tutvunud.

2. ARHIIVI TURVALISUSE ANALÜÜS PAROOLIGA

See peatükk sisaldab teavet arhiivide ja krüptograafia kohta, mis annab esmase ülevaate. Töö käigus kasutas autor zip-arhiivi. See arhiiv kasutab krüpteerimiseks funktsiooni PBKDF2.

2.1. Arhiivid

Arhiivid on failid, mis sisaldavad teisi faile. Tavaliselt on arhiivis failid kokku surutud. Arhiividel on erinevad laiendid (nimede lõpud), nagu ZIP, 7Z, ZIPX, LZH, LHA, GZ või CAB, sõltuvalt sellest, kuidas need on loodud. [10]

2.1.1. Arhiivide kasutamine

Arhiivid lihtsustavad failide rühmitamist ja kiirendavad nende failide allalaadimist ja kopeerimist. Arhiiv on mugavam saata, kui on vaja saata mitu faili, samuti suured failid, mida e-post tavaliselt läbi ei lase. Arhiivide laadimine võtab vähem aega kui failide laadimine eraldi. Paljud internetist allalaaditavad failid levivad samuti arhiividena. [10]

2.1.2. Arhiivitüübid

Saadaval on suur hulk arhiive nagu: 7-Zip Compressed File, Zipped File jt.

7Z fail on kokkusurutud arhiiv, mida loovad erinevad failide kokkusurumise programmid, peamiselt Igor Pavlovi 7-Zip. See on kokkusurutud avatud lähtekoodiga LZMA abil, mis eristub kõrge kokkusurumise astmest ja võib sisaldada AES-256 (256-bitist) krüpteerimist. 7Z failid võivad sisaldada mitut kataloogi või faili, mis on kokku surutud ruumi kokkuhoiu või transportimise jaoks. [11]

Samuti toetab see failitüüp Unicode failinimesid, mitmest osast koosnevaid arhiive, arhiivide päiste kokkusurumist ja faile, mille suurus on kuni 16 miljardit GB. [11]

ZIP file on arhiiv, mis sisaldab ühte või mitut Zip-koopiaga kokku pakitud faili. See hoiab failid üksteisest eraldi, võimaldades faile erinevate meetoditega kokku suruda ja võtta need välja ilma, et oleks vaja kogu arhiivi kokku või lahti pakkida. [12]

2.2. Krüptograafia

Krüptograafia on iidne info krüpteerimise kunst. Egiptuse kirjamees kasutas mittestandardseid hieroglüüfe kirjalikult umbes aastal 1900 e.m.a. Tänapäeval on krüptograafia oluline andmete ja telekommunikatsiooni turvaliseks edastamiseks ebausaldusväärsete võrkude kaudu, sealhulgas Interneti kaudu. [13]

2.2.1. Krüptograafia kasutamine

Tänapäeval kasutatakse krüptograafiat kõikjal, sealhulgas paroolide ja pangatehingute

kaitsmine internetis, inimeste ja seadmete autentimine ja seadmete vahelise seose loomine. Juhul kui kõik krüptograafiafunktsioonid kaoksid, lõpetaks tänapäevane elu, sest meie isiklik teave muutuks avalikkuseks, mis tooks kaasa tõsiseid tagajärgi, kui see satuks ründajate kätte. Krüptograafia võimaldab selliseid ohte ennetada, tagades teabe ja side turvalisuse reeglite abil, mis võimaldavad andmeid vastu võtta ja töödelda ainult neil, kellel on vastav ligipääs. [14]

2.2.2. Krüptograafia peamised funktsioonid

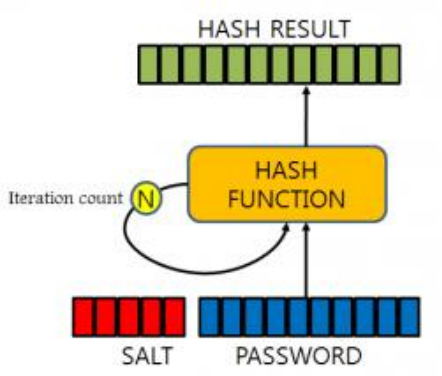
Krüptograafial on viis põhifunktsiooni [15]:

- privaatsus (confidentiality): Tagamine, et keegi ei saa sõnumit lugeda peale eeldatava saaja;
- autentimine (Authentication): identiteedi kinnitamise protsess;
- terviklikkus (Integrity): Tagab, et saadud sõnum ei ole võrreldes originaaliga muudetud;
- mittekeeldumine (Non-repudiation): Mehhanism, mis võimaldab tõestada, et saatja on selle sõnumi tõesti saatnud;
- võtmete vahetamine (Key exchange): krüptovõtmete vahetamise meetod saatja ja saaja vahel.

2.3. PBKDF2 funktsioon

Sellel veebilehel [16] toodud näite kohaselt kasutab funktsioon "PBKDF2" räsifunktsiooni SHA-256. Sellest võib järeldada, et sama funktsioon kasutab ka SHA-1 räsifunktsiooni koos 160-bitise räsiväärtuse suurusega. Autor tugines ka loengutel saadud teadmistele, et olulisemaid andmeid, nagu paroole, krüpteeritakse asümmeetrilise krüpteerimisega, samas kui suuremahulisi, kuid mitte nii olulisi andmeid krüpteeritakse sümmeetrilise krüpteerimisega.

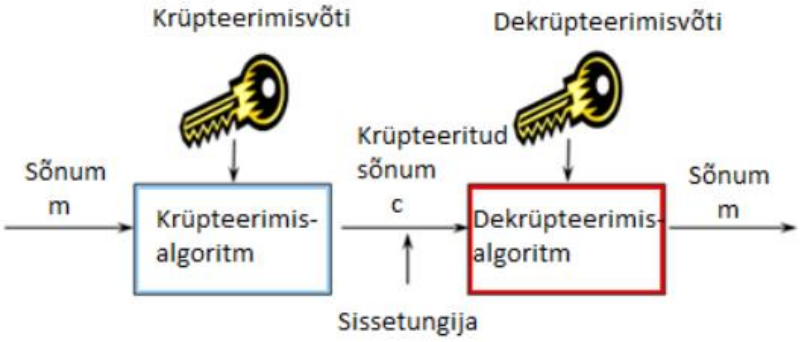
PBKDF2 funktsioon toimib pseudojuhusliku räsifunktsiooni (nt SHA-256, SHA-512 jne) koos soolaga rakendamisel reale ehk paroolile ja selle protsessi kordamisega suur hulk kordi [17]. Selline protsess võib näha joonisel 2.1.



Joonis 2.1 PBKDF2 funktsiooni täitmise protsess

Antud funktsiooni kasutatakse parooli usaldusväärsuse suurendamiseks. Näiteks LastPass kasutab SHA-256 räsi algoritmi, mis tagab suurema kaitsetaseme bruteforce rünnakute vastu. Räsimisprotsess viiakse läbi kliendipoolselt ja hõlmab 5000 kuni 200 000 iteratsiooni, sõltuvalt jõudlusest. Lisaks teeb LastPass serveripoolselt 100 000 iteratsiooni PBKDF2. See lahendus nii kliendi- kui ka serveripoolselt tagab suurema turvalisuse. [17]

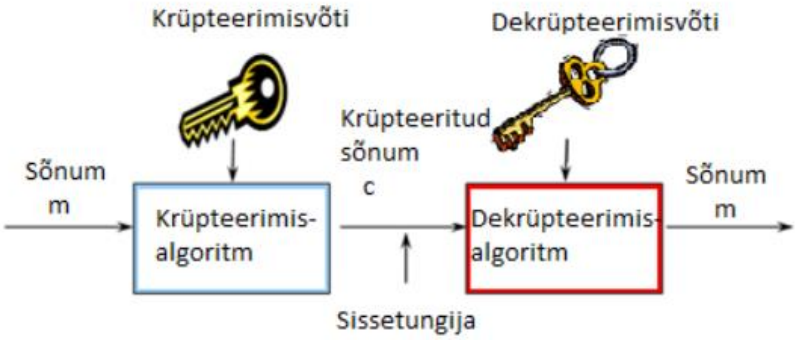
Sümmeetriline krüpteerimine on meetod, mis kasutab andmete krüpteerimiseks ja dekrüpteerimiseks üht võtit, muutes protsessi lihtsamaks ja tunduvalt kiiremaks kui asümmeetrilised analoogid. Lisaks on sümmeetriliste krüpteerimisalgoritmide tööks vaja vähem arvutusvõimsust ning interneti kiirus ei vähene. On mitmeid sümmeetrilisi algoritme, nagu AES, RC4, DES, 3DES, RC5, RC6 ja teised. Arhiivides sisu krüpteerimiseks kasutatakse AES (Advanced Encryption System) algoritmi, mis on perekond plokkšifreid erineva võtme ja plokkide suurusega. [18] Visuaalne näide on esitatud joonisel 2.2.



Joonis 2.2 Sümmeetriline krüpteerimine [19]

Asümmeetriline krüpteerimine kasutab mitut võtit andmete krüpteerimiseks ja dekrüpteerimiseks. See meetod tagab turvalisuse, kuna andmete krüpteerimiseks kasutatakse avalikku võtit, samas kui dekrüpteerimine on võimalik ainult privaatse võtmega, mida tuleb hoida turvaliselt. Lisaks tagab see krüpteerimisviis autentimise,

tagades, et andmeid saab dekrüpteerida ainult need, kes neid peaksid saama. Veebiserverite ja e-posti serverite jaoks kasutatakse ainult ühte võtit, mida tuleb turvaliselt kaitsta. [18] Visuaalne näide on esitatud joonisel 2.3.



Joonis 2.3 Asümmeetriline krüpteerimine [20]

2.4. PBKDF2-s kasutatavad funktsioonid

SHA-1, SHA-256 ja SHA-512 on asümmeetrilised räsifunktsioonid, mida kasutatakse PBKDF2 funktsioonis [17]. AES-256 on sümmeetrilise võtmega krüpteerimine [21].

2.5. Krüptograafilised räsifunktsioonid

Räsifunktsioonid on funktsioonid, mis kuvavad suvalise pikkusega sisendid fikseeritud pikkusega reale - räsikood. Kui need kuvad rahuldavad mõningaid täiendavaid krüptograafilisi tingimusi, saab neid kasutada teabe terviklikkuse kaitsmiseks. [22]

2.5.1. Omadused, mis peaksid olema krüptograafilisel räsifunktsioonil

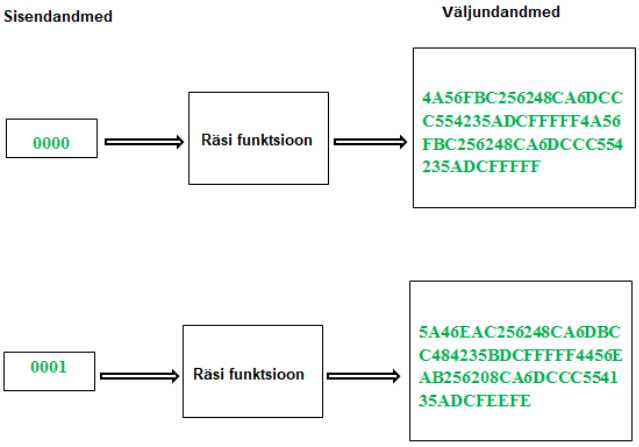
Ideaalse räsifunktsiooni jaoks on täidetud järgmised tingimused [23]:

- räsifunktsioon on determistlik, st sama sõnumi tulemuseks on sama räsiväärtus;
- räsiväärtus arvutatakse kiiresti iga sõnumi jaoks;
- on võimatu leida sõnumit, mis annab antud räsiväärtuse;
- on võimatu leida kahte erinevat sama räsiväärtusega sõnumit (ei tohiks olla kollisiooni);
- väike muutus sõnumis muudab räsi nii palju, et uued ja vanad väärtused on täiesti erinevad ja ei sõltu üksteisest(laviiniefekt).

2.5.2. Laviiniefekti, kollisiooni ning esimese ja teise astme kollisiooni mõisted

Laviiniefekt on krüpteerimisalgoritmi soovitud omadus, kus isegi väike võtme või teksti muutus põhjustab krüpteeritud teksti märgatavat muutumist. [24] Täpsemalt saab

lugeda Lisa 1.



Joonis 2.4 Laviiniefekti näide [24]

Kollisioon on olukord, kus kahel erineval väärtusel on sama räsi. Räsifunktsioon ei saa vältida kollisioonide tekkimist lõputu arvu võimalike sisendite ja lõpliku väljundite arvu tõttu. [23] Täpsemalt saab lugeda Lisa 1.

Esimese astme kollisioon (Pre-image resistance) on räsifunktsiooni turvaomadus, mille puhul kurjategija ei suuda kergesti leida sõnumit, mis viib määratud räsini. [23] Täpsemalt saab lugeda Lisa 1.

Teise astme kollisioon (Second pre-image resistance) on räsifunktsiooni turvaomadus , mis tähendab, et antud räsi h ja sõnumi m1 puhul ei leia kurjategija teist sõnumit m2, millel on sama räsi h. Oluline on, et kollisioonikindel räsifunktsioon on kaitstud ka rünnakute eest teise algkuju leidmiseks. Hea räsifunktsioon peaks käituma juhuslikult, jäädes samas deterministlikuks ja efektiivselt arvestatavaks. [23] Täpsemalt saab lugeda Lisa 1.

2.6. Algoritm SHA-1

SHA-1 on üks neljast turvalise räsialgoritmi (SHA) perekonna algoritmidest. Enamik neist on välja töötatud Riikliku Julgeoleku Ameti poolt. (NSA) ja selle on avaldanud Riiklik Standardi- ja Tehnoloogiainstituut (NIST). [25]

SHA-0 on 160-bitise räsiväärtuse suurusega ja on selle algoritmi esimene variant. Tema räsiväärtused koosnevad 40 numbrist. [25]

SHA-1 on selle krüptograafilise räsifunktsiooni teine iteratsioon. Sellel on ka 160-bitise räsiväärtuse suurusega ja selle eesmärk on suurendada turvalisust. [25]

MD4 algoritm ja hilisemad SHA algoritmid kasutavad 32-bitiseid muutujaid koos biti kaupa boolaarsete funktsioonidega, nagu loogilised operaatorid AND, OR ja XOR,

sisendandmete teisendamiseks nende vastavaks räsiväärtuseks. [26]

2.6.1. Krüpteerimisprotsess SHA-1 räsialgoritmiga

Esialgu luuakse 5 muutujat, mis on funktsiooni sisemised muutujad vahepealsete arvutuste salvestamiseks joonisel 2.5. [26]

```
H0 - 01100111010001010010001100000001
H1 - 11101111110011011010101110001001
H2 - 10011000101110101101110011111110
H3 - 00010000001100100101010001110110
H4 - 11000011110100101110000111110000
```

Joonis 2.5 Loodud muutujad [26]

Järgmine samm on valida sõna räsi jaoks. Näiteks sõna " CRYPTO". [26]

Seejärel konverteeritakse sõna ASCII-ks - "Ameerika teabevahetuse standardkoodiks". [26]

Igale tähele määratakse oma number. [26]

CRYPTO — 67-82-89-80-84-79 [26]

ASCII-koodi teisendamine binaarseks koodiks joonisel 2.6. [26]

```
CRYPTO – 01000011-01010010-01011001-01010000-01010100-01001111
```

Joonis 2.6 CRYPTO iga tähe binaarkood [26]

Tähemärgid pannakse kokku ja lisatakse 1 lõpus joonisel 2.7. [26]

```
CRYPTO – 0100001101010010010110010101000001010100010011111
```

Joonis 2.7 CRYPTO binaarkoodis [26]

Kuna SHA-1-s on ploki suurus 512 bitti, lisatakse sõnumile nullid, et ta oleks mooduliga 512 võrdne 448-ga. Seega peab 48-bitine sõnum, millele on lisatud üks, lõppu lisama 399 nulli joonisel 2.8. [26]

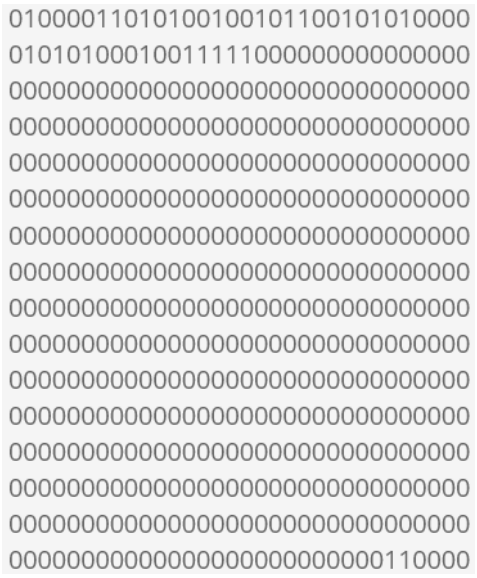
```
01000011010100100101100101010000010101000100111110000000000000000000000000-
00000000000000000000000000000000000000000000000000000000000000000000000000-
00000000000000000000000000000000000000000000000000000000000000000000000000-
00000000000000000000000000000000000000000000000000000000000000000000000000-
00000000000000000000000000000000000000000000000000000000000000000000000000-
00000000000000000000000000000000000000000000000000000000000000000000000000-
0000000000
```

Joonis 2.8 CRYPTO binaarkoodis [26]

Sõnumi pikkus on 48 tähemärki, mis binaarses vormis väljendatuna on 110000. Seega

lisatakse järgmine osa eelmise sammu lõppu. [26]

Seejärel jagatakse sõnum kuueteistkümneks lõiguks, mille pikkus on 32 märki või bit joonisel 2.9. [26]



Joonis 2.9 CRYPTO binaarkoodis [26]

Järgmine samm on 16 sõna teisendamine 80 sõnaks, kasutades tsüklilist funktsiooni. Esimesel tsükli korduse läbimisel valitakse neli sõna, mis on read 1, 3, 9 ja 14 eelmisest sammust. Järgmisel tsükli kordusel kasutatakse sõnu 2, 4, 10 ja 15 jne.

XOR-teisendus 1, 3, 9 ja 14 ridade jaoks joonisel 2.10. [26]

Joonis 2.10 Esimese tsükli tulemus [26]

Teostamine arvude tsüklilist pööramist vasakule - st liigutage vasakpoolseim arv paremale joonisel 2.11. [26]

Joonis 2.11 Tulemus esimese numbri teisaldamine rea lõppu [26]

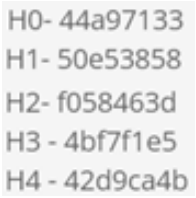
See protsess korratakse kuni saadakse 80 sõna. [26]

Järgmine samm on käivitada funktsioonide komplekt sõnadele kindlas järjestuses, mis töötavad välja viiest muutujast, mis määrati 1. sammus. Funktsioonid ühendavad AND-, OR- ja NOT-operaatoreid koos vasakpoolsete nihega. [26]

Lõpptulemusena jääb järele viis muutujat joonisel 2.12 [26]

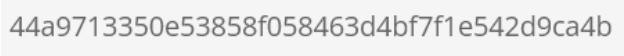
Joonis 2.12 Viie muutuja väärtus jääb alles lõpptulemusena [26]

Teisenda H muutujad kuueteistkümnendas süsteemisse (hex) joonisel 2.13 [26]



Joonis 2.13 H muutujad heksa(hex) [26]

Ühendage muutujad, et saada räsiväärtus (hash) joonisel 2.14 [26]



Joonis 2.14 Lõplik tulemus ehk räsi väärtus [26]

2.7. Algoritm AES-256

Lisaks sellele, et AES on sümmeetrilise võtmega räsialgoritm, peetakse seda ka plokkšifraks. Seda nimetatakse ka Rijndaeliks ja selle on välja töötanud belgia krüptograafid Joan Damen ja Vincent Rijmen. Plokkšifrid jagavad lihtteksti osadeks, mida nimetatakse plokkideks. AES-krüpteerimisel on kolm võtmepikkust. [21]

Igal võtmepikkusel on erinev arv võimalikke klahvikombinatsioone [21]:

128-bitise võtme pikkus: 3.4 x 1038

192-bitise võtme pikkus: 6.2 x 1057

256-bitise võtme pikkus: 1.1 x 1077

Standardne AES kasutab 128-bitist plokipikkust. Kõik lähtetekstis olevad andmed on jagatakse 4x4 massiiviks, milles on 16 baiti. Igas baidis on 8 bitti. Saades 16 baidi ja 8 biti korrutise, on meil igas plokis 128 bitti. Kuigi võtme pikkus võib varieeruda, ei mõjuta see krüpteeritud teksti suurust. See jääb alati 128 bitti. [21]

Tabel 2.1 Bitimassiivi näide [27]

8 bit	8 bit	8 bit	8 bit

8 bit	8 bit	8 bit	8 bit
8 bit	8 bit	8 bit	8 bit
8 bit	8 bit	8 bit	8 bit

2.7.1. Krüpteerimisprotsess AES räsialgoritmiga

Väärib märkimist, et teisendusringide arv (Nr) sõltub ploki pikkusest ja võtme pikkusest ning on esitatud tabelis 2.2. [13]

Tabel 2.2 Sõltuvus ploki suurusest ja võtme pikkusest teisendusringide arv [13]

Teisendusringide arv Nr		Ploki pikkus		
		128 bit Nb=4	192 bit Nb=6	256 bit Nb=8
Võtme pikkus	128 bit Nk=4	10	12	14
	192 bit Nk=6	12	12	14
	256 bit Nk=8	14	14	14

Krüpteerimise Protsess AES räsi algoritmi koosneb järgmistest etappidest:

KeyExpansion-round - võtmed tuletatakse krüpteerimisvõttest, kasutades Rijndaeli võtmetabelit. [28]

Initial Round [28]

- AddRoundKey— Iga bait praeguses olekus segatakse XOR-operatsiooni abil ringi võtme vastava baidiga. [28]

Rounds [28]

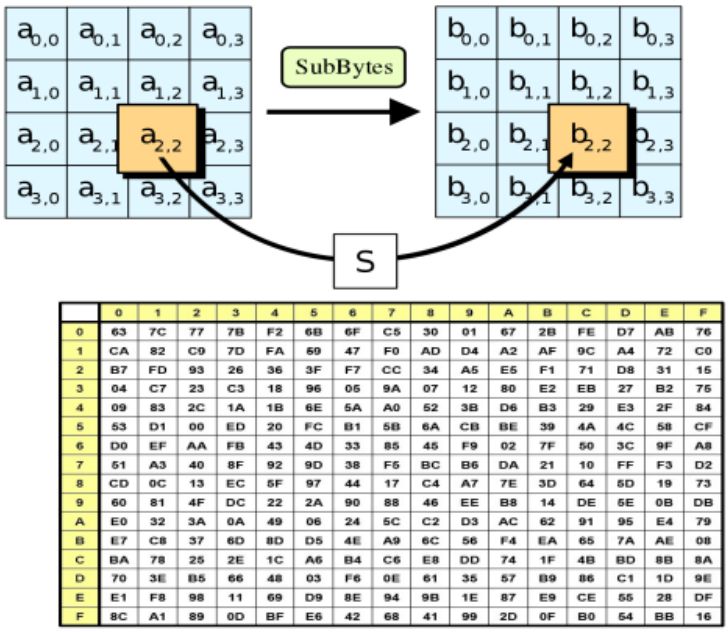
- SubBytes— mittelineaarne asendamine, kus baidid teisaldatakse ümber vastavalt ümberpaigutamistabelile.
- ShiftRows— teisaldamistsükkel, milles iga olekurea segatakse tsükliliselt teatud arvu sammude kaupa.
- MixColumns— Segamisoperatsioon, mis töötleb olekumassiivi veerge, lisades iga veeru 4 bitti.
- AddRoundKey

Final Round (ilma MixColumns) [28]:

- SubBytes
- ShiftRows
- AddRoundKey

Lisateave iga vooru kotha.

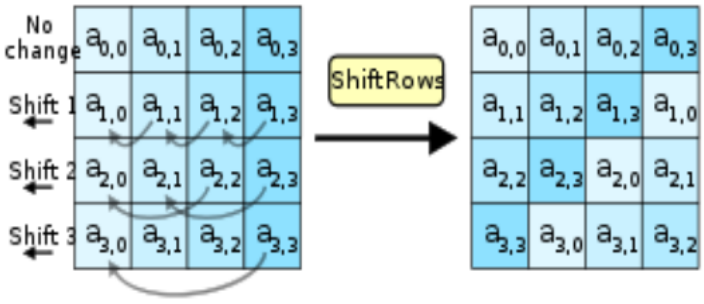
SubBytes - Baitide asendamine S-box tabeli abil. Iga bait esitatakse kuueteistkümnendas süsteemis numbrina b = (x, y), kus x määrab b-i 4 vanemat bitti ja y - 4 nooremat bitti. S-box tabeli suurusega 16x16 sisaldab väärtusi asendamiseks algse baitiga: väärtust b' tabeli rea x ja veeru y ristumiskohas kasutatakse asendamiseks algsele bait'ile b. [27] SubBytes etapp joonisel 2.15



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

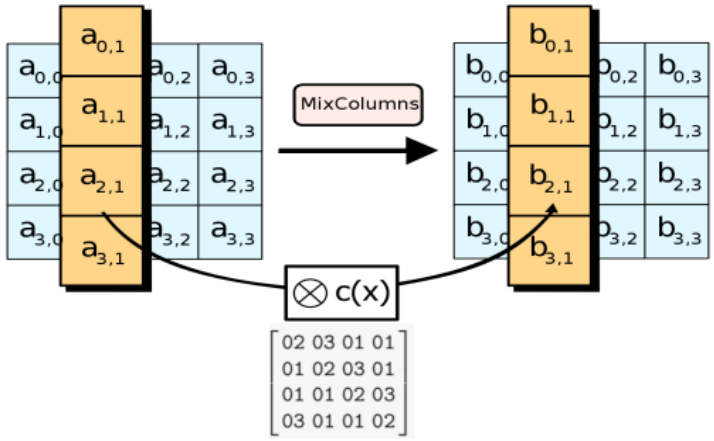
Joonis 2.15 SubBytes etapp [27]

ShiftRows — Tsükliline nihe olekureal. Nullrea jätab muutmata, esimene nihutatakse vasakule 1 bait, teine 2 bait ja kolmas vastavalt 3 bait. [27] ShiftRows etapp joonisel 2.16



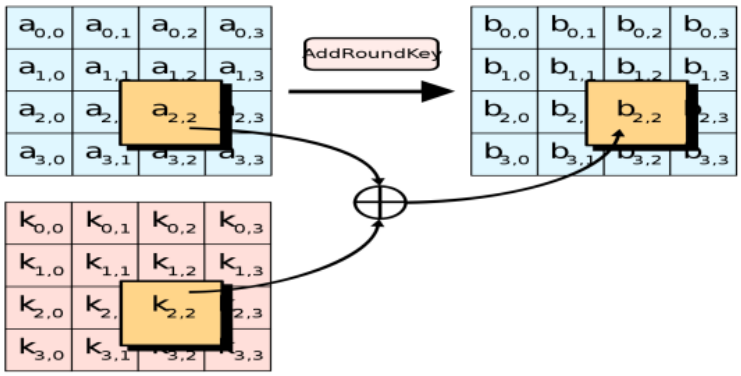
Joonis 2.16 ShiftRows etapp [27]

MixColumns — Iga veeru "state" korrutamine fikseeritud maatriksiga. Nii liidetakse ridade ja veergude korrutajad kokku. [27] MixColumns etapp joonisel 2.17



Joonis 2.17 MixColumns etapp [27]

AddRoundKey — Kui ümmargune võti lisatakse olekusse, võetakse iga element (bait) ümmargusest võtmest ja iga vastav element olekust ja liidetakse bititi kokku. Kui bitid langevad kokku, on tulemuseks 0, kui ei langevad kokku, on tulemuseks 1. Tulemuseks on uus olek, mida kasutatakse järgmises voorus. [27] AddRoundKey etapp joonisel 2.18



Joonis 2.18 AddRoundKey etapp [27]

3. BRUTE FORCE PAROOLIDE MURDMINE

Toore jõu ehk Brute-force rünnak tähendab mingite objektide kõikide võimalike variantide otsimist, kuni otsitava väärtuseni jõudmiseni. Ühe sümboli muutmine annab juba uue variandi. [3]

3.1. Paroolid

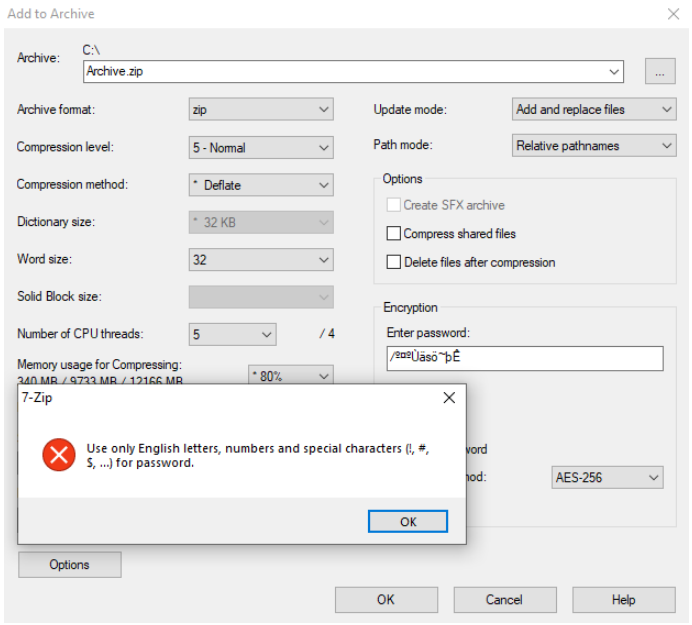
Parooli murdmiseks kuluv aeg sõltub parooli pikkusest ja selles kasutatud märkidest, näiteks suurtest ja väikestest tähtedest, numbritest ja erimärkidest.

3.1.1. Paroolide pikkus

Viidi läbi eksperiment, et teada saada, kui pikk parool võib olla kasutatud arhiivi loomisel. Autor kasutas numbreid 1 kuni 500 ja tireega nende vahel. Arhiiv loodi ja pakkus edukalt lahti. Selleks kulus umbes 10 sekundit. Võrdluseks, 10 tähemärgi pikkune parool vajab sama ülesande jaoks alla sekundi. Seega võib parool olla kuni 1000 tähemärki pikk ja see ei takista arhiivi loomist, kuid pole selleks vajadust.

3.1.2. Tähemärgid paroolis

Selleks, et kontrollida, milliseid võimalikke sümbolite sisendeid saab kasutada ZIP-arhiivi parooli loomiseks, proovis autor sisestada väärtusi erinevatest kodeeringutest. Pärast seda eksperimenti sai autor hoiatuse kasutada parooli loomisel ainult ingliskeelseid tähti, numbreid ja erimärke. Vaata joonis 3.1.



Joonis 3.1 Kasutatavate tähemärkide hoiatus

3.1.3. Parooli häkkamise aeg sõltuvalt sümbolite arvust aastal 2022

IT-ettevõtte Hive Systems eksperdid koostasid statistika tänapäevaste häkkerite töö kiiruse kohta. Selgus, et 8-kohalised paroolid ei ole parim vahend häkkimise vastu. Eksperdid tegid ka soovitusi paroolide loomisel. Vaata tabel 3.1. [29][30]

Tabel 3.1 Parooli murdmise aeg sõltuvalt sümbolitest aastal 2022 [29]

2022					
Tähemärkide arv	**Ainult numbrid**	**Väiketähed**	**Üla- ja väiketähed**	**Numbrid, üla- ja väiketähed**	**Numbrid, üla- ja väiketähed, sümbolid**
4	Koheselt	Koheselt	Koheselt	Koheselt	Koheselt
5	Koheselt	Koheselt	Koheselt	Koheselt	Koheselt
6	Koheselt	Koheselt	Koheselt	Koheselt	Koheselt
7	Koheselt	Koheselt	2 sek	7 sek	31 sek
8	Koheselt	Koheselt	2 min	7 min	39 min
9	Koheselt	10 sek	1 t	7 t	2 p
10	Koheselt	4 min	3 p	3 ndl	5 k
11	Koheselt	2 t	5 k	3 a	34 a
12	2 sek	2 p	24 a	200 a	3 tuh a
13	19 sek	2 k	1 tuh a	12 tuh a	202 tuh a
14	3 min	4 a	64 tuh a	750 tuh a	16 mlj a
15	32 min	100 a	3 mlj a	46 mlj aa	1 mld a

Parooli loomisel on spetsialistide sõnul oluline mitte ainult selle pikkus, vaid ka keerukus. Näiteks kaasaegsete otsingumeetodite kasutamisel leitakse 11-numbrilise kombinatsiooni praktiliselt kohe ja samast arvust väikeste tähtedega - paari tunni jooksul. Kuid suurtähtede lisamisel kasvab bruteforce'i rünnaku aeg viie kuu võrra ja koos numbritega kuni kolme aastani. Avaldatud tabeli põhjal tundub optimaalseima variandina olulise teabe kaitseks 12-tähemärgilist parooli, kus on erinevate registrite numbrid ja tähed. Sellise kombinatsiooni purustamine kaasaegsete arvutite abil võtab aega umbes 200 aastat. [29]

Tuleb märkida, et need numbrid on esitatud arvutite arvutusvõime kasvu arvestamata. Näiteks kaks aastat tagasi kulutasid samad ülesanded häkkeritele rohkem aega. Näiteks 11-tähemärgilise tähtede ja väikeste tähtedega komplekti valimine 2020. aastal oleks võtnud aega kolm aastat viie kuu asemel. [31] Vaata tabel 3.2.

Tabel 3.2 Parooli murdmise aeg sõltuvalt sümbolitest aastal 2020 [31]

2020					
Tähemärkide arv	**Ainult numbrid**	**Väiketähed**	**Üla- ja väiketähed**	**Numbrid, üla- ja väiketähed**	**Numbrid, üla- ja väiketähed, sümbolid**
4	Koheselt	Koheselt	Koheselt	Koheselt	Koheselt
5	Koheselt	Koheselt	Koheselt	Koheselt	Koheselt

6	Koheselt	Koheselt	Koheselt	1 sek	5 sek
7	Koheselt	Koheselt	25 sek	1 min	6 min
8	Koheselt	5 sek	22 min	1 t	8 t
9	Koheselt	2 min	19 t	3 p	3 ndl
10	Koheselt	58 min	1 k	7 k	5 a
11	2 sek	1 p	5 a	41 a	400 a
12	25 sek	3 n	300 a	2 tuh a	34 tuh a
13	4 min	1 a	16 tuh a	100 tuh a	2 mlj a
14	41 min	51 a	800 tuh a	9 mlj a	200 mlj a
15	6 t	1 tuh a	43 mlj a	600 mlj a	15 mld a

3.2. Paroolide murdmise tööriistad

On olemas palju tööriistu paroolide läbimurdmiseks, mis töötavad nii internetiühendusega kui ka ilma selleta. Mõne tööriista puhul on väga oluline omada teatud mahuga kettaruumi.

Autor loetles leitud tööriistad paroolide valimiseks ja murdmiseks:

- Hashcat;
- John the Ripper;
- RainbowCrack;
- Cain & Abel;
- Crackstation.

Teave programmi kirjeldusega saate leida Lisa 2.

Autor valis hashcat, kuna see ei vaja suurt kettaruumi, töötab ilma internetita, mis on väga mugav igasuguste Interneti-probleemide puhul, on tasuta, omab avatud lähtekoodi ja sobib soolatud räsude jaoks. Üksikasjaliku tabeliga paroolide murdmise vahendite kohta saate tutvuda Lisa 2.

3.3. Hashcat

Arhiivi murdmiseks on vaja läbi proovida suur hulk paroole, eeldusel, et algne parool ja selle räsksumma vastavad ASCII sümbolitele. Kaitstud arhiivi parooli sisestamisel saame sisestada ainult ASCII sümbolid. Kõige mugavam vahend selliseks läbimurdmiseks on paroolide brute-force meetodiga mõeldud programmid.

Autor on seadnud järgmised ülesanded, et saavutada eesmärk:

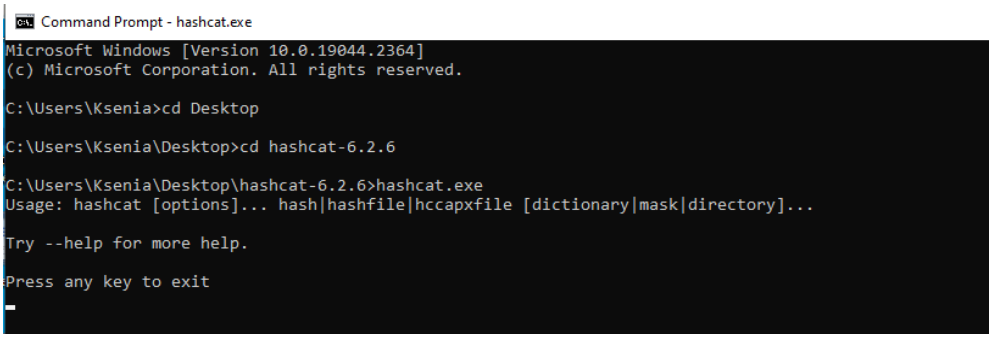
- Hashcat'i paigaldamine;
- sobiva parooli otsimine vastavalt tingimustele;
- saadud tulemuste analüüs.

3.3.1. Hashcat'i paigaldamine

Autor paigaldas programmi, kasutades GitHubi repositooriumi. [32] Kõik failid olid autori töölaual ning ka hashcat kaustas.

3.3.2. Programmiga tutvumine

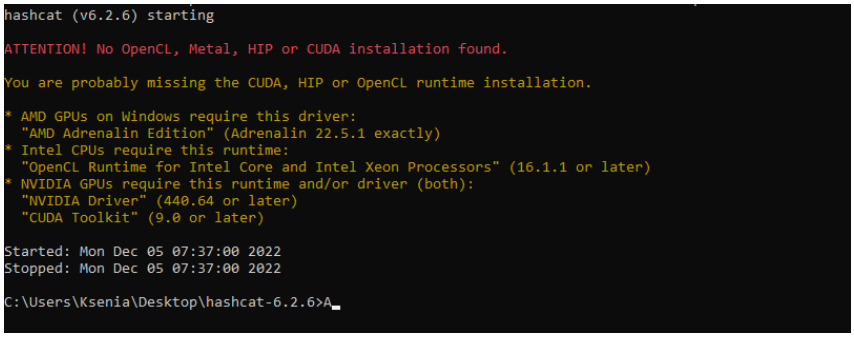
Programm käivitatakse käsurealt. Vaata joonis 3.2.



Joonis 3.2 Hashcat'i käivitatud programm

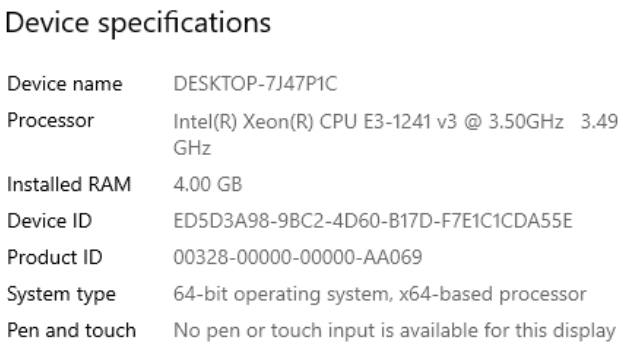
Programmi tutvustamiseks kasutati järgmist rida: hashcat –help. Üksikasjad kõigi valikute, räsi-režiimide, ründemeetodite ja kõigi sisseehitatud tähemärkide kohta leiate Lisa 3-st.

Pärast programmiga tutvumist tekkisid raskused faili enda käivitamisega. Vaata joonis 3.3.



Joonis 3.3 Hoiatus täiendava keskkonna vajalikkuse kohta

Avades spetsifikatsiooni arvuti, veendus autor, et protsessor on Intel ja nagu hoiatuses kirjutatud, installis ta "OpenCL Runtime for Intel Core and Intel Xeon Processors" (16.1.1 or later). Vaata joonis 3.4.



Joonis 3.4 Arvuti spetsifikatsioonid

Autor täitis vormi aadressil [33], kinnitas e-posti aadressi ja laadis alla nõutava käitamise keskkonna. Pärast seda hoiatus enam ei ilmunud.

3.3.3. Sõnastikurünnak

Autor otsustas proovida murda 7zipi parooli, kuna ta kasutas seda oma kogemuse järgi sagedamini. [34] Autor laadis üles sõnastiku, mis sisaldab 1 000 000 kõige levinumat parooli. Faili nimega TEST1 loodi autor poolt, mis sisaldab seda teksti: "Come on! You can do it!".

Selle faili jaoks loodi arhiiv nimega TEST1.zip parooliga 123456.

Kasutades seda veebisaiti [35], saadi arhiivi räsi ja salvestati tekstifailis nimega TEST1Hash.txt."

Seejärel käivitades hashcat, valis autor:

- sõnastikurünnaku režiimi (-a 0);
- räsi-režiim 7z arhiivide jaoks (-m 11600);
- tee räsi-failile /Users/Ksenia/Desktop/TEST1Hash.txt;
- tee sõnastikufailile /Users/Ksenia/Desktop/1mm.txt Vaata joonis 3.5.

```
C:\Users\Ksenia\Desktop\hashcat-6.2.6>hashcat -a 0 -m 11600 /Users/Ksenia/Desktop/TEST1Hash.txt /Users/Ksenia/Desktop/1mm.txt
hashcat (v6.2.6) starting
```

Joonis 3.5 Sõnastikurünnaku näide

Halli värvi joonega on tähistatud räsiväärtus, mida kasutati parooli murdmiseks. Punase joonega on tähistatud parool, mis leiti hashcati abil. Vähem kui kahe sekundiga leidis programm kuuekohalise parooli. Vaata joonis 3.6

```
Started: Fri May 12 05:31:54 2023
Stopped: Fri May 12 05:31:56 2023

C:\Users\Ksenia\Desktop\hashcat-6.2.6>hashcat -a 0 -m 11600 /Users/Ksenia/Desktop/TEST1Hash.txt /Users/Ksenia/Desktop/1mm.txt --show
$7z$2$10$0$$16$c4859c4733ecc0c2870e06faa0474123$3982997338$32$27$bd0e9d1d9051d184ffa06338ff3985c476d2101a07f60fc048190782267056d7$23$00:123456
```

Joonis 3.6 Sõnastikurünnaku tulemus

3.3.4. Brute-force rünnak

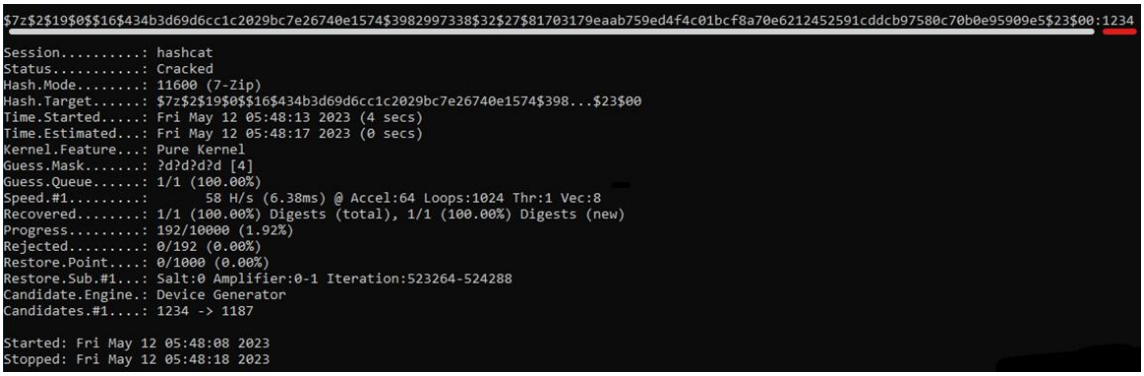
Autor lõi uue arhiivi TEST2, faili oma räsiga ja käivitas selle reaga hashcat -a 3 -m 11600 /Users/Ksenia/Desktop/TEST2Hash.txt ?d?d?d?d. Erinevus antud juhul seisneb selles, et autor otsib ainult numbreid ja seetõttu kirjutatakse d(digit-arv) ja ? (tundmatu märk). Vaata joonis 3.7.

```
C:\Users\Ksenia\Desktop\hashcat-6.2.6>hashcat -a 3 -m 11600 /Users/Ksenia/Desktop/TEST2Hash.txt ?d?d?d?d
hashcat (v6.2.6) starting
```

Joonis 3.7 Brute-force rünnaku näide

Halli värvi joonega on tähistatud räsiväärtus, mida kasutati parooli murdmiseks. Punase

joonega on tähistatud parool, mis leiti hashcati abil. Programm leidis kümne sekundiga neljakohalise parooli. Vaata joonis 3.8.



Joonis 3.8 Brute-force rünnaku tulemus

Kui räsi-režiimi arhiivi lahtimuukimiseks ei määrata, valib programm selle automaatselt ja ülesande täitmiseks kulub umbes 15 sekundit. Vaata Lisa 3.

3.4. Raskused

Nagu eespool mainitud, esimene autori probleem seisnes vajaliku keskkonna puudumisel programmi käivitamisel. Autoril polnud võimalust töö teostamiseks kasutada teist arvutit, seega ta ühendas end "FortiClient" ja Kolledži "kaugtöölaua ühenduse" abil. Programmi töötades ülikooli arvutil, lõpetas programm töö mitu korda enne tulemuse leidmist. Autori arvates võivad põhjused olla järgmised:

- keegi lülitas arvuti välja;
- uuendused laaditi alla, mis nõudsid arvuti taaskäivitamist;
- programm käivitati mitte administraatori õigustega.

3.5. Parooli murdmine kollisiooni leidmise meetodil

Nagu artiklis mainiti, on võimalik kollisiooni leida sellepärast, et PBKDF2 räsib parooli, mille pikkus on üle 64 tähemärgi. Seega otsitakse paari paroole, mis võivad avada sama arhiivi. [36] Autor otsustas välja selgitada, kui raske on sellise paari leidmine. Paari leidmiseks oli vaja luua selline parool, mis sisaldas ainult ASCII-sümboleid, mida saab sisestada paroolivälja pärast räsi väärtuse arvutamist.

3.5.1. Kood Python'i keeles

Autor püüdis Python'i keeles rakendada programmi, mis vaataks läbi algse parooli tähemärgid, leiaks räsi ja kontrolliks seda ASCII tähemärkide vastu. Vaata joonis 3.9.

main.py

```python
from cryptography.hazmat.primitives import hashes #Krüptograafia madalal tasemel liideseid pakkuvast moodulist imporditakse klassid, mis töötlevad räsifunktsioone.
import binascii#Binääride andmete teisendamise moodul tekstivormingusse ja vastupidi
from cryptography.hazmat.backends import default_backend#Imporditakse vaikimisi backend, moodulist, mis pakub backend'id krüpteerimise opereerimiseks.
cons='UTF-8'#Kodeeringu tüüb
string="Nev1r-G0nna-G2ve-Y8u-Up-N5v1r-G1nna-Let-Y4u-D1wn-N8v4r-G5nna-D0sert-Yo"#Räsimiseks string
while len(string)<80:#Tsükkel, mis töötab seni kuni stringi pikkus on alla 80
  try:#Konstruktsioon, mida kasutatakse erindite töötlemiseks
    string=string+"u"#Tekstiline string, millele lisatakse täht "u"
    data=string.encode(cons)#String kodeeritakse baitideks ja salvestatakse muutujasse nimega 'data'
    print(string)#Algse väärtuse kuvatakse
    print(data)#Kuvatakse kooditud UTF-8 string
    digest = hashes.Hash(hashes.SHA1(),backend=default_backend())#Loodakse 'digest' objekt, kasutades SHA1 ja vaikimisi backendi
    digest.update(data)#Baitide jada lisatakse 'digest'-i
    result=digest.finalize()#Arvutused lõpetatakse ja saadakse räsi summa
    hex=binascii.b2a_hex(result).decode(cons)#Tulemus kuueteistkümnendsüsteemis esituses
    b64=binascii.b2a_base64(result).decode(cons)#Tulemus Base64 kodeeringus
    String_ASCII = result.decode(cons)#Tulemus ASCII kodeeringus
    print ("Hex: ",hex, "Base64: ",b64, "ASCII: ",String_ASCII,"\n")#Kõikide väärtuste kuvatamine
  except Exception as e:
     print(e)  # Kui tekib viga, siis kuvatakse see ekraanile
```

Joonis 3.9 Kood Python'i keeles

Pythoni keeles koodi väljund, algne rida, rida baitidena UTF-8 kodeeringus, rida kuusteistkümnendsüsteemis esitusena, Base64 kodeeringus ja ASCII kodeeringus. Vaata joonis 3.10.

```
Nev1r-G0nna-G2ve-Y8u-Up-N5v1r-G1nna-Let-Y4u-D1wn-N8v4r-G5nna-D0sert-You
b'Nev1r-G0nna-G2ve-Y8u-Up-N5v1r-G1nna-Let-Y4u-D1wn-N8v4r-G5nna-D0sert-You'
Hex:  706b4838613041714e624863647738477726d5370 Base64:  cGtIOGEwQXFOYkhjZHc4R3JtU3A=
 ASCII:  pkH8a0AqNbHcdw8GrmSp

Nev1r-G0nna-G2ve-Y8u-Up-N5v1r-G1nna-Let-Y4u-D1wn-N8v4r-G5nna-D0sert-Youu
b'Nev1r-G0nna-G2ve-Y8u-Up-N5v1r-G1nna-Let-Y4u-D1wn-N8v4r-G5nna-D0sert-Youu'
'utf-8' codec can't decode byte 0xdc in position 0: invalid continuation byte
Nev1r-G0nna-G2ve-Y8u-Up-N5v1r-G1nna-Let-Y4u-D1wn-N8v4r-G5nna-D0sert-Youuu
b'Nev1r-G0nna-G2ve-Y8u-Up-N5v1r-G1nna-Let-Y4u-D1wn-N8v4r-G5nna-D0sert-Youuu'
'utf-8' codec can't decode byte 0xb1 in position 1: invalid start byte
Nev1r-G0nna-G2ve-Y8u-Up-N5v1r-G1nna-Let-Y4u-D1wn-N8v4r-G5nna-D0sert-Youuuu
b'Nev1r-G0nna-G2ve-Y8u-Up-N5v1r-G1nna-Let-Y4u-D1wn-N8v4r-G5nna-D0sert-Youuuu'
'utf-8' codec can't decode byte 0xbb in position 0: invalid start byte
Nev1r-G0nna-G2ve-Y8u-Up-N5v1r-G1nna-Let-Y4u-D1wn-N8v4r-G5nna-D0sert-Youuuuu
b'Nev1r-G0nna-G2ve-Y8u-Up-N5v1r-G1nna-Let-Y4u-D1wn-N8v4r-G5nna-D0sert-Youuuuu'
'utf-8' codec can't decode byte 0xc7 in position 1: invalid continuation byte
```

Joonis 3.10 Python'i keeles koodi väljund

Kui muuta kodeerimistüüp 'latin1'iks, siis kodeerimisega probleemi pole, kuid sisaldatud on ka teised sümbolid, mis ei vasta ASCII-le. Näiteks: «l2Mi;^ªò¼ôg_tï» või «ë,-*NùtæÔü-¯¢D»

3.6. Saadud tulemuste analüüs

PBKDF2 funktsioon, mida kasutatakse arhiivi kaitseks, on piisavalt vastupidav Brute

Force meetodile. Autor kulutas mitu kuud, et leida parool, mis sisaldaks suuri ja väikseid tähti, numbreid ja sümboleid, kasutades Hashcati, aga see ei ole piisav. Kirjutatud Pythoni kood muutis algset rida, kasutas SHA-1 algoritmi ja saavutas räsiväärtuse. Kuid ASCII sümbolitega räsiväärtuse leidmine ei õnnestunud.

3.6.1. Kui ohtlik on selliste paroolide paari olemasolu?

Selliste paroolide olemasolu ei peeta ohtlikuks, kuna kasutajad ei loo paroole, mis on pikemad kui 64 tähemärki, et funktsioon saaks neid teha räsi väärtust. Isegi kui leidub inimene, kes loob sellise pika parooli, on väga väike tõenäosus, et tema räsi väärtus sisaldab kõiki ASCII sümbolid.

3.6.2. PBKDF2 funktsiooni rakendamine ja selle tähtsus

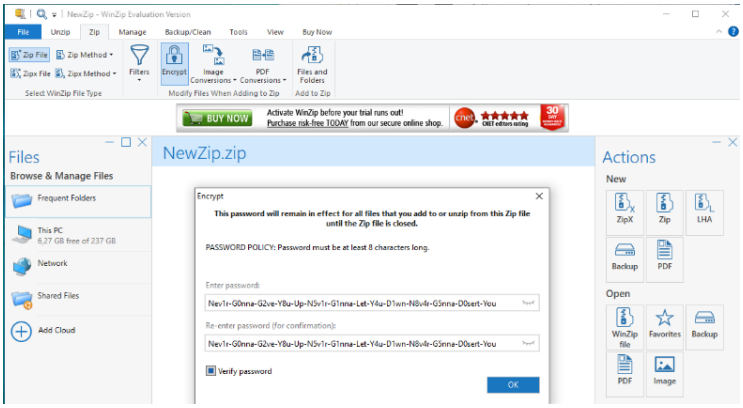
PBKDF2 funktsiooni kasutatakse erinevates programmides, nagu [2]:

- AES krüpteerimisskeem programmides nagu WinZip;
- turvalisuse tagamiseks WiFi kaudu;
- Firefox Sync;
- Cisco IOS;
- kasutajate paroolide ja autentimiskoodide kaitsmiseks;
- Apple'i mobiilsetes operatsioonisüsteemides iOS.

PBKDF2 funktsiooni kasutatakse laialdaselt paroolihaldurites nagu Dashlane, Keeper, RoboForm, RememBear, 1Password, Sticky Password ja teised. [37]

3.6.3. Kas selliseid paarid (parool ja räsikood) on võimalik kasutada teistes programmides?

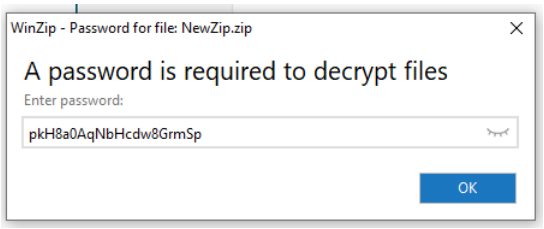
Autoril tekkis huvi, kas oleks võimalik teha sama trikki ka WinZip'iga. Pärast programmi installimist ja katsetusperioodi aktiveerimist loodi faili arhiiv. Vaata joonis 3.11.



Joonis 3.11 Arhiveerimine parooliga WinZip programmiga

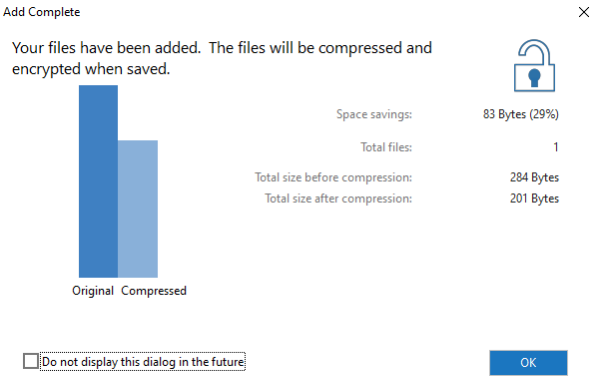
Autor seadis arhiivi parooliks Twitteri postituses olnud parooli ning proovis seejärel sama

Twitteri postituses olnud räsi-väärtust kasutades arhiivi avada. Vaata joonis 3.12.



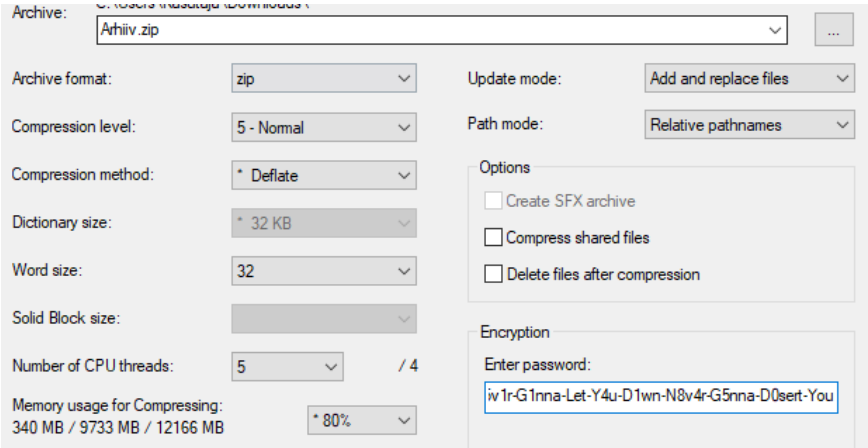
Joonis 3.12 Arhiivi dekrüpteerimiseks sisestatakse sisse hash-väärtus

Arhiiv avanes. Vaata joonis 3.13.



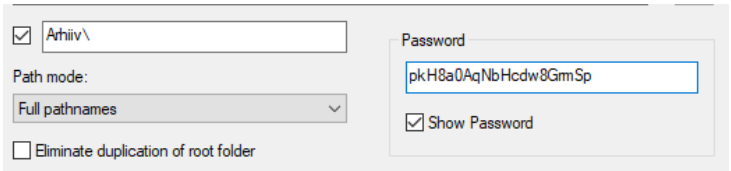
Joonis 3.13 Arhiiv avati edukalt

Teen sama trikki 7zip programmides zip arhiiviga, lõin pika parooliga arhiivi artiklist. Vaata joonis 3.14.



Joonis 3.14 Arhiivi loomine pika parooliga abil

Seejärel proovisin seda avada lühikese räsiväärtuse abil. Vaata joonis 3.15.



Joonis 3.15 Arhiivi avamine lühikese parooliga

Arhiiv avanes ilma probleemita. Vaata joonis 3.16.

Joonis 3.16 Ava arhiiv

3.6.4. Nõuanded arhiivi täiendavaks kaitsmiseks

Arhiivi täiendavaks kaitsmiseks soovitab autor muuta mõningaid parameetreid:

- suurendage iteratsioonide arvu 10 000-ni;
- soola kasutamine;
- kasutage paroole, mille pikkus on vähemalt 12 tähemärki, sealhulgas väiketähti, suurtähti, numbreid ja erimärke;
- kontrollige paroolide keerukust ja neid aeg-ajalt muuta;
- kasutage PBKDF2-funktsiooni kasutavate programmide ja raamatukogude uusimaid versioone, kuna need võivad sisaldada haavatavuse parandusi.

Nüüd soovitatakse kasutada SHA-2 ja SHA-3 algoritme, kuna SHA-1 on aegunud ja seda on juba õnnestunud murda. Kuna SHA-2 ja SHA-3 pakuvad kõrgemat turvalisust. [38]

KOKKUVÕTE

Ksenia Kostyuk lõputöö: "Kollisiooni leidmine parooliga kaitstud arhiivi murdmisel". Kui ei tagata adekvaatset kaitset krüptograafiaga, võivad isikuandmed olla varastatud ja ebaseaduslikult kasutatud, mis võib kaasa tuua tõsiseid tagajärgi nii individuaalsele kasutajale kui ka ettevõttele tervikuna. Paroolikaitse on kõige populaarsem autentimisviis.

Töös uuris autor parooliga kaitstud arhiivi murdmise võimalusi. Liiga pika parooli puhul kasutab ZIP selleks PBKDF2 funktsiooni. Uuriti nii Brute Force meetodit kui ka kollisiooni otsingu võimalust.

Jaotises "Paroolide tähtsus ja nende murdmine" täpsustas autor, kui oluline on tänapäeval kasutada usaldusväärset parooli ja kirjeldas, millised on ründemeetodid. Autor tutvustas lugejale ka ebatavalist nähtust, kus ühe zip-arhiivi jaoks saab kasutada kaht erinevat parooli, mis seavad kahtluse alla infoturve arhiivides.

Jaotises "Arhiivi turvalisuse analüüs parooliga" käsitletakse arhiivi kaitse analüüsi. Autor kirjeldas PBKDF2-s kasutatavaid räsifunktsioone ja nende omadusi.

Viimases peatükis viis autor läbi mitmeid katseid arhiivi loomisel parooliga. Ta pakkus välja parooli murdmise aja sõltuvust kasutatud sümbolitest. Ta tegi ülevaate parooli murdmise tööriistadest ning kasutas üht neist, et teha mitu rünnakut arhiivile. Autor proovis realiseerida ja otsida kollisioone, lähtudes ühe kommentaatori eeldusest, Pythoni programmeerimiskeele abil.

Autor esitas PBKDF2 funktsiooni kasutamise loendi, millest autor valis ühe arhiivihalduri WinZip ning kontrollis võimalust luua ja avada arhiiv algses artiklis esitatud parooli ja räsi väärtuse abil.

Lõputöö tulemusena väljendas autor oma arvamust kolisioonide olemasolu ja nende ohtlikkuse kohta, samuti õnnestus tal edukalt murda paroole Hashcat programmi abil, viia läbi mitmeid eksperimente 7-zip arhiivihalduriga ning luua ja avada arhiiv parooli ja tema räsi WinZipis. Autor andis soovitusi turvalisema parooli loomiseks, krüptograafiliste funktsioonide kasutamiseks ja paroolidega töötamiseks vajalike parameetrite kasutamiseks.

SUMMARY

Ksenia Kostyuk's thesis titled "Finding a collision when breaking a password-protected archive". Inadequate cryptographic protection can lead to theft and illegal use of personal data, which can have serious consequences for both individual users and the company as a whole. Password protection is the most popular authentication method.

In the thesis, the author explored the possibilities of cracking password-protected archives. For a password that is too long, ZIP uses the PBKDF2 function. Both Brute Force and collision search methods were studied.

In the section "The Importance of Passwords and Their Crackability," the author clarified the importance of using a reliable password in today's world and described the attack methods. The author also introduced the unusual phenomenon where two different passwords can be used for one zip archive, which raises doubts about information security in archives.

In the section "Analysis of Archive Security with Passwords," the analysis of archive protection was discussed. The author described the hash functions used in PBKDF2 and their properties.

In the final chapter, the author conducted several experiments in creating password-protected archives. Autor proposed a dependence of the cracking time on the symbols used in the password. Autor reviewed password cracking tools and used one of them to make several attacks on the archive. The author tried to realize and search for collisions based on a commentator's assumption using the Python programming language.

The author presented a list of PBKDF2 function uses, from which the author selected one archive manager, WinZip, and checked the possibility of creating and opening an archive using the password and hash value provided in the original article.

As a result of the thesis, the author expressed her opinion on the existence and danger of collisions, successfully cracked passwords using the Hashcat program, conducted several experiments with the 7-zip archive manager, and created and opened an archive in WinZip using the password and its hash. The author gave recommendations for creating a more secure password, using cryptographic functions, and using parameters necessary for working with passwords.

KASUTATUD KIRJANDUSE LOETELU

[1] M. K. V. A. K. R. G. Levent Ertaul, Implementation and Performance Analysis of, CSU East Bay, Hayward, 2023.

[2] С. Панасенко, Словарные атаки на хэш-функции, 2009. [Online]. http://www.panasenko.ru/Articles/168/168.html (16.05.2023).

[3] CYCLONIS, Атака паролей Rainbow Table - что это такое и как вы защищаетесь от этого, CYCLONIS, 24.05.2020. [Online]. https://www.cyclonis.com/ru/rainbow-table-password-attack-what-is-it-and-how-do-you-protect-yourself-from-it/ (16.05.2023).

[4] e. b. Kaspersky, Атака «дней рождения» (Birthday attack), [Online]. https://encyclopedia.kaspersky.ru/glossary/birthday-attack/ (16.05.2023).

[5] A. Sharoglazov, Backdoor password in a ZIP!, 22.08.2022. [Online]. https://twitter.com/_mohemiv/status/1561044393880178689 (14.05.2023).

[6] Unblvr, twitter, [Online]. https://twitter.com/Unblvr1/status/1561112433812463616?cxt=HHwWgICz_Zjfl6orAAAA (25.05.2023).

[7] snipp.ru, Текст в SHA1, [Online]. https://snipp.ru/tools/sha1 (25.05.2023).

[8] Bfotool, Hex to Ascii Text Converter, Bfotool, [Online]. https://bfotool.com/hex-to-ascii (16.05.2023).

[9] denis-19, Зашифрованный ZIP-архив может иметь два правильных пароля, 22.08.2022. [Online]. https://habr.com/ru/news/683840/ (16.05.2023).

[10] WINZIP, About Zip Files and Other Archives, [Online]. https://kb.winzip.com/help/AboutZIPsAndOtherArchives_4.htm#:~:text=Zip%20files%20are%20the%20most,on%20how%20they%20were%20created.. (16.05.2023).

[11] fileinfo.com, .7Z File Extension, [Online]. https://fileinfo.com/extension/7z (16.05.2023).

[12] fileinfo.com, .ZIP File Extension, fileinfo.com, [Online]. https://fileinfo.com/extension/zip (16.05.2023).

[13] G. C. Kessler, An Overview of Cryptography, Дейтона-Бич, 1998.

[14] Z. Sardar, symmetron, 14.07.2020. [Online]. https://www.symmetron.ru/articles/zachem-nuzhna-kriptografiya/?sphrase_id=451195) (16.05.2023).

[15] EUCIP, Krüptograafia rakendamine võrguturbes, EUCIP, [Online]. https://eopearhiiv.edu.ee/e-kursused/eucip/haldus/423_krptograafia_rakendamine_vrguturbes.html (16.05.2023).

[16] P. C. f. Developers, PBKDF2, 13.10.2021. [Online]. https://cryptobook.nakov.com/mac-and-key-derivation/pbkdf2 (16.05.2023).

[17] L. Safonov, Стандарт PBKDF2, DefconRU, 16.06.2015. [Online]. https://defcon.ru/cryptography/485/ (16.05.2023).

[18] D. Honcharenko, Шифрование: типы и алгоритмы. Что это, чем отличаются и где используются?, [Online]. https://hostpro.ua/wiki/security/encryption-types-algorithms (16.05.2023).

[19] EUCIP, Sümmeetriline krüptograafia, EUCIP, [Online]. https://eopearhiiv.edu.ee/e-kursused/eucip/haldus/4221_smmeetriline_krptograafia.html (16.05.2023).

[20] EUCIP, Asümmeetriline krüptograafia, [Online]. https://eopearhiiv.edu.ee/e-kursused/eucip/haldus/4222_asmmeetriline_krptograafia.html (24.05.2023).

[21] М. АЛЬГРЕН, ЧТО ТАКОЕ ШИФРОВАНИЕ AES-256 И КАК ОНО РАБОТАЕТ?, [Online]. https://www.websiterating.com/ru/cloud-storage/what-is-aes-256-encryption/ (16.05.2023).

[22] B. Preneel, CRYPTOGRAPHIC HASH FUNCTIONS:, Leuven.

[23] CorVVin, Хеш-функция, что это такое?, 23.12.2020. [Online]. https://habr.com/ru/articles/534596/ (16.05.2023).

[24] swetha_vazhakkat, Avalanche Effect in Cryptography, [Online]. https://www.geeksforgeeks.org/avalanche-effect-in-cryptography/ (16.05.2023).

[25] T. Fisher, SHA-1: What It Is & How It's Used for Data Verification, [Online]. https://www.lifewire.com/what-is-sha-1-2626011 (16.05.2023).

[26] A. Madeira, How Does a Hashing Algorithm Work?, 13.03.2019. [Online]. https://www.cryptocompare.com/coins/guides/how-does-a-hashing-algorithm-work/ (16.05.2023).

[27] k_anokhin, Симметричный алгоритм блочного шифрования Advanced Encryption Standart, 23.12.2020. [Online]. https://habr.com/ru/articles/534620/ (16.05.2023).

[28] StudFiles, Алгоритм шифрования aes, [Online]. https://studfile.net/preview/9387408/page:3/ (16.05.2023).

[29] В. Сидоров, Названо среднее время брутфорса паролей в 2022 году, 21.03.2022. [Online]. https://4pda.to/2022/03/21/397835/nazvano_srednee_vremya_brutforsa_parolej_v_2022_godu/ (16.05.2023).

[30] Е. Быстрова, Hive Systems: 8-значные пароли можно взломать менее чем за час, 04.03.2022. [Online]. https://www.anti-malware.ru/news/2022-03-04-111332/38278 (16.05.2023).

[31] В. Сидоров, Названо среднее время брутфорса паролей в 2020 году, 21.03.2022. [Online]. https://4pda.to/2022/03/21/397835/nazvano_srednee_vremya_brutforsa_parolej_v_2020_godu/ (16.05.023).

[32] jsteube, hashcat, 02.09.2022. [Online]. https://github.com/hashcat/hashcat (16.05.2023).

[33] Intel, Intel Registration Center - User Portal, [Online]. https://lemcenter.intel.com/forms/?productid=3207&pass=yes (25.05.2023).

[34] apasscracker, Free dictionaries, [Online]. https://apasscracker.com/dictionaries/ (17.05.2023).

[35] Hashes, Zip2john, [Online]. https://hashes.com/en/johntheripper/zip2john (16.05.2023).

[36] М. Нефёдова, К зашифрованным архивам ZIP подходят два разных пароля, 24.08.2022. [Online]. https://xakep.ru/2022/08/24/zip-rickroll (16.05.2023).

[37] К. Касунич, 9 лучших менеджеров безопасных паролей 2023, (24.05.2023).

[38] CodeSigningStore, Hash Algorithm Comparison: MD5, SHA-1, SHA-2 & SHA-3, [Online]. https://codesigningstore.com/hash-algorithm-comparison (24.05.2023).

[39] kukuxumushi, Брутфорс хэшей в Active Directory, 19.10.2021. [Online]. https://habr.com/ru/company/deiteriylab/articles/584160/ (16.05.2023).

[40] C. &. Abel, Cain & Abel, [Online]. https://xn----7sbabnb7cmacncmoc3p.xn--p1ai/app/cain_and_abel (16.05.2023).

[41] Й. Сантос, John the Ripper Review: это бесплатный сброс пароля Windows с открытым исходным кодом, [Online]. https://www.topsevenreviews.com/ru/john-the-ripper-review/#part1 (16.05.2023).

[42] iNfor24, RainbowCrack, [Online]. https://infor24.ru/RainbowCrack (16.05.2023).

[43] М. Миллс, Каин и Авель: программа для взлома и взлома паролей, 10.09.2020. [Online]. https://itigic.com/ru/cain-and-abel-program-to-crack-and-hack-passwords/ (16.05.2023).

[44] CrackStation, Free Password Hash Cracker, [Online]. https://crackstation.net/ (16.05.2023).

[45] HackWare.ru, Полное руководство по John the Ripper. Ч.6: брут-форс нестандартных хешей, [Online]. https://hackware.ru/?p=15567#61 (16.05.2023).

[46] АЛЕКСАНДРА, Инструменты для взлома паролей, 31.01.2023. [Online]. https://clickfraud.ru/instrumenty-dlya-vzloma-parolej/ (17.05.2023).

[47] QUASAR, Как пользоваться Hashcat на Windows, [Online]. https://spy-soft.net/hashcat-windows/ (17.05.2023).

[48] S. T. Help, 11 Password Cracker Tools (Password Hacking Software 2023), [Online]. https://www.softwaretestinghelp.com/password-cracker-tools/#1_CrackStation (17.05.2023).

LISA 1. HASH-FUNKTSIOONIDEGA SEOTUD MÕISTED

Laviiniefekt on termin, mis on seotud matemaatiliste funktsioonide spetsiifilise käitumisega, mida kasutatakse krüpteerimisel. Laviiniefekti peetakse igasuguse krüpteerimisalgoritmi soovitud omaduseks. Väike muutus võtmes või avatud tekstis peaks kaasa tooma märkimisväärse muutuse krüpteeritud tekstis. Seda omadust nimetatakse laviiniefektiks. [24]

Kollisioon - kahe erineva sisendi väärtusel on sama väljundväärtus ehk räsiväärtus. Tuleb märkida, et kollisioonid eksisteerivad alati igas räsifunktsioonis, kuna võimalikud sisendid on lõputud, aga väljundite arv on piiratud. Räsifunktsiooni peetakse kollisioonidele vastupidavaks, kui kollisiooni avastamise tõenäosus on nii väike, et selleks on vaja miljoneid aastaid arvutusi. Kuigi räsifunktsioone ilma kokkupõrgeteta ei eksisteeri, on mõned neist piisavalt usaldusväärsed ja loetakse kokkupõrgetele vastupidavaks. [23]

Esimese astme kollisiooni (Pre-image resistance)

Esimese astme kollisiooni omadus on tuntud algkuju takistus. Räsifunktsiooni peetakse ettearvamatuks, kui on väga väike tõenäosus, et ründaja leiab sõnumi, mis genereeris antud räsi. See omadus on andmete kaitse seisukohast oluline, kuna sõnumi hash võib tõestada selle autentsust ilma teabe avaldamise vajaduseta. [23]

Teise astme kollisiooni (Second pre-image resistance)

Seda omadus on tuntud teise algkuju takistus. Lihtsustamiseks võib öelda, et see omadus asub kahe eelmise omaduse vahel. Rünnak teise algkuju leidmiseks toimub siis, kui kurjategija leiab kindla sisendi, mis genereerib sama räsi kui teine sisend, mis talle juba teada on. Teisisõnu, kurjategija teab, et hash(m1) = h ja püüab leida m2 sellise, et hash(m2) = h. Siit selgub, et rünnak teise algkuju leidmiseks hõlmab kollisiooni otsimist. Seetõttu on igasugune kollisiooni vastupidav räsifunktsioon ka teise algkuju rünnakutele vastupidav. [23]

Kõik need omadused tähendavad, et kurjategija ei saa sisendandmeid asendada või muuta ilma nende räsi muutmata. Oluline on märkida, et räsifunktsioon peaks käituma nii juhusliku funktsioonina kui võimalik, olles samas määratletud ja efektiivselt arvestatav. [23]

LISA 2. SALASÕNA VALIMISE JA MURDMISE PROGRAMMID

Hashcat see on kõige populaarsem ja jõudluselt parim tööriist, mis toetab paljusid erinevaid räsialgoritme. [39]

John the Ripper on murdmisprogrammide komplekt on üks vanimatest programmide hulgas, mis võimaldab failide teisendamist vastavateks räsiväärtusteks ja hilisemaseks valikute proovimiseks. [39]

RainbowCrack — Antud rakendus kasutab teistsugust lähenemist räsiväärtuste murdmiseks - mitte paroolide räsimise arvutamist ja võrdlemist,vaid otsib räsiväärtuse paari sisaldavaid ette genereeritud tabelites. See erineb teistest tööriistadest väga kõrge murdmiskiiruse poolest (kuni ~5000 korda kiirem), kuid nõuab iga räsialgoritmi jaoks eelnevalt genereeritud tabelite salvestamist, mis võtab kümneid terabaiti kettaruumi. [39]

Cain & Abel see on tööriist Microsofti operatsioonisüsteemide paroolide taastamiseks. See võimaldab hõlpsalt taastada erinevaid tüüpe paroole, kasutades võrgu pealtkuulamist, sõnastikurünnakuid, Brute-Force'i ja krüptoanalüüsi, VoIP-vestluste salvestamist, krüpteeritud paroolide dekrüpteerimist, traadita võrkude võtmete taastamist, paroolikastide avamist, kogutud paroolide avamist ja marsruudi analüüsi. [40]

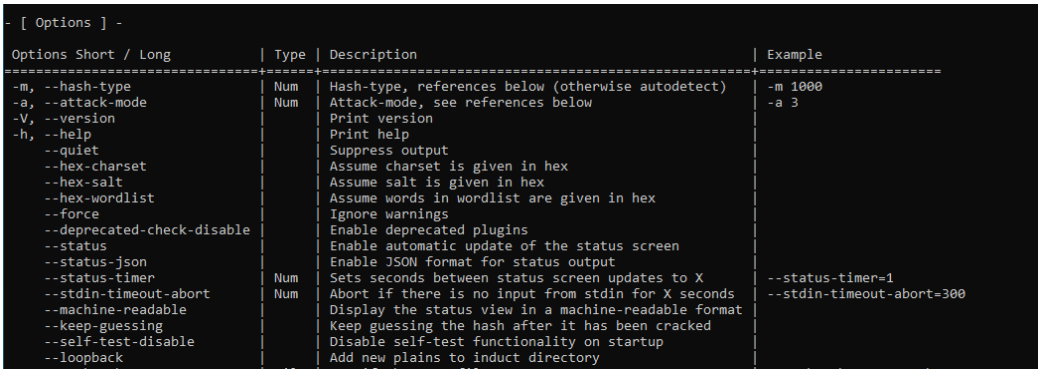
crackstation see on teenus, mis võimaldab otsida räsiväärtusi sõnastikus, mis sisaldab 1,6 miljardit parooli. [39]

Tabel Lisa 2.1 Paroolide murdmiseks vahendit [41][42][43][44][45][46][47][48]

Kriteeriumid	Hashcat	John the Ripper	Rainbow Crack	Cain & Abel	Crackstation
Internetiühendu se vajadus	Ei	Ei	Ei	Ei	Jah
Suur kettamaht	Ei	Ei	Jah	-	-

Tasuta	Jah	Jah/Ei	Jah	Jah	Jah
Ava lähtekood	Jah	Jah	-	Ei	Ei
Erinevate räsi tüüpide arv	350+	-	-	-	-
Uute räsialgoritmide lisamise võimalus	Ei	Jah	Jah	-	-
Sobib soolatud räsile	Jah	Jah	-	-	Ei
Toetatud platvormid	Windows, macOS, Linux	macOS, Linux, BeOS, OpenVMS, Windows.	Windows, Linux.	Windows	-

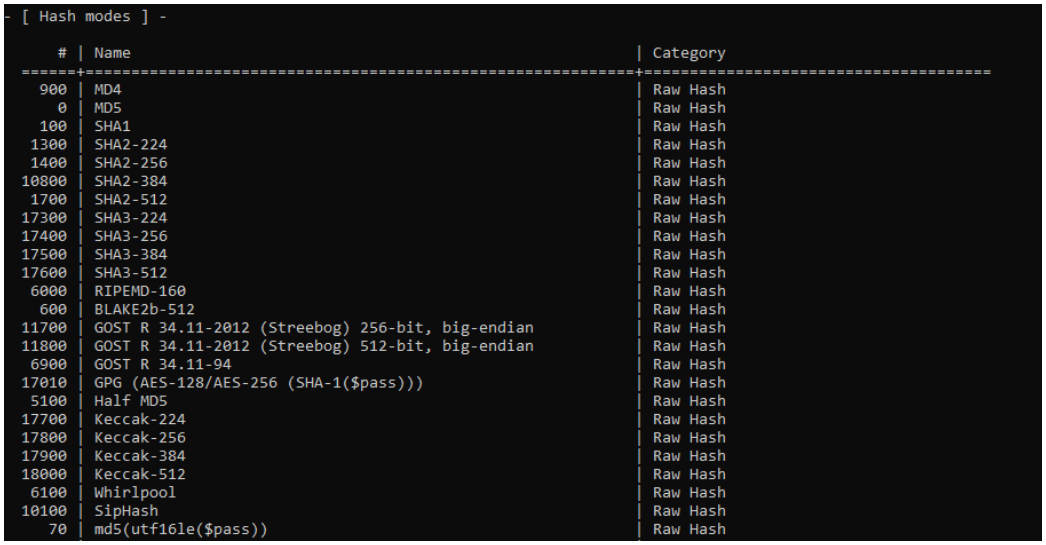
LISA 3. HASHCAT HELP'I JUHENDIST SAADUD TEAVE



```
- [ Options ] -

 Options Short / Long          | Type | Description                                               | Example
===============================+======+===========================================================+=======================
 -m, --hash-type               | Num  | Hash-type, references below (otherwise autodetect)        | -m 1000
 -a, --attack-mode             | Num  | Attack-mode, see references below                         | -a 3
 -V, --version                 |      | Print version                                             |
 -h, --help                    |      | Print help                                                |
     --quiet                   |      | Suppress output                                           |
     --hex-charset             |      | Assume charset is given in hex                            |
     --hex-salt                |      | Assume salt is given in hex                               |
     --hex-wordlist            |      | Assume words in wordlist are given in hex                 |
     --force                   |      | Ignore warnings                                           |
     --deprecated-check-disable|      | Enable deprecated plugins                                 |
     --status                  |      | Enable automatic update of the status screen              |
     --status-json             |      | Enable JSON format for status output                      |
     --status-timer            | Num  | Sets seconds between status screen updates to X           | --status-timer=1
     --stdin-timeout-abort     | Num  | Abort if there is no input from stdin for X seconds       | --stdin-timeout-abort=300
     --machine-readable        |      | Display the status view in a machine-readable format      |
     --keep-guessing           |      | Keep guessing the hash after it has been cracked          |
     --self-test-disable       |      | Disable self-test functionality on startup                |
     --loopback                |      | Add new plains to induct directory                        |
```

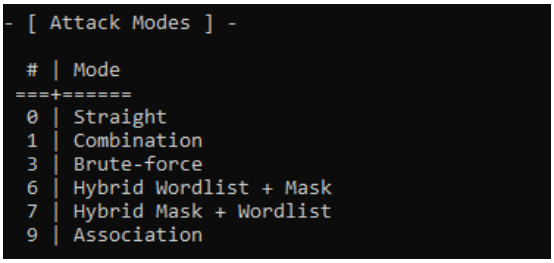
Joonis Lisa 3.1 Seaded Hashcat'is



```
- [ Hash modes ] -

      # | Name                                                      | Category
  ======+===========================================================+======================================
    900 | MD4                                                       | Raw Hash
      0 | MD5                                                       | Raw Hash
    100 | SHA1                                                      | Raw Hash
   1300 | SHA2-224                                                  | Raw Hash
   1400 | SHA2-256                                                  | Raw Hash
  10800 | SHA2-384                                                  | Raw Hash
   1700 | SHA2-512                                                  | Raw Hash
  17300 | SHA3-224                                                  | Raw Hash
  17400 | SHA3-256                                                  | Raw Hash
  17500 | SHA3-384                                                  | Raw Hash
  17600 | SHA3-512                                                  | Raw Hash
   6000 | RIPEMD-160                                                | Raw Hash
    600 | BLAKE2b-512                                               | Raw Hash
  11700 | GOST R 34.11-2012 (Streebog) 256-bit, big-endian          | Raw Hash
  11800 | GOST R 34.11-2012 (Streebog) 512-bit, big-endian          | Raw Hash
   6900 | GOST R 34.11-94                                           | Raw Hash
  17010 | GPG (AES-128/AES-256 (SHA-1($pass)))                      | Raw Hash
   5100 | Half MD5                                                  | Raw Hash
  17700 | Keccak-224                                                | Raw Hash
  17800 | Keccak-256                                                | Raw Hash
  17900 | Keccak-384                                                | Raw Hash
  18000 | Keccak-512                                                | Raw Hash
   6100 | Whirlpool                                                 | Raw Hash
  10100 | SipHash                                                   | Raw Hash
     70 | md5(utf16le($pass))                                       | Raw Hash
```

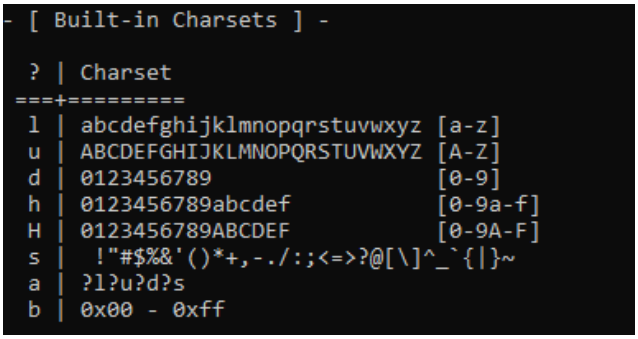
Joonis Lisa 3.2 Räsi-režiimid Hashcat'is



```
- [ Attack Modes ] -

  # | Mode
 ===+======
  0 | Straight
  1 | Combination
  3 | Brute-force
  6 | Hybrid Wordlist + Mask
  7 | Hybrid Mask + Wordlist
  9 | Association
```

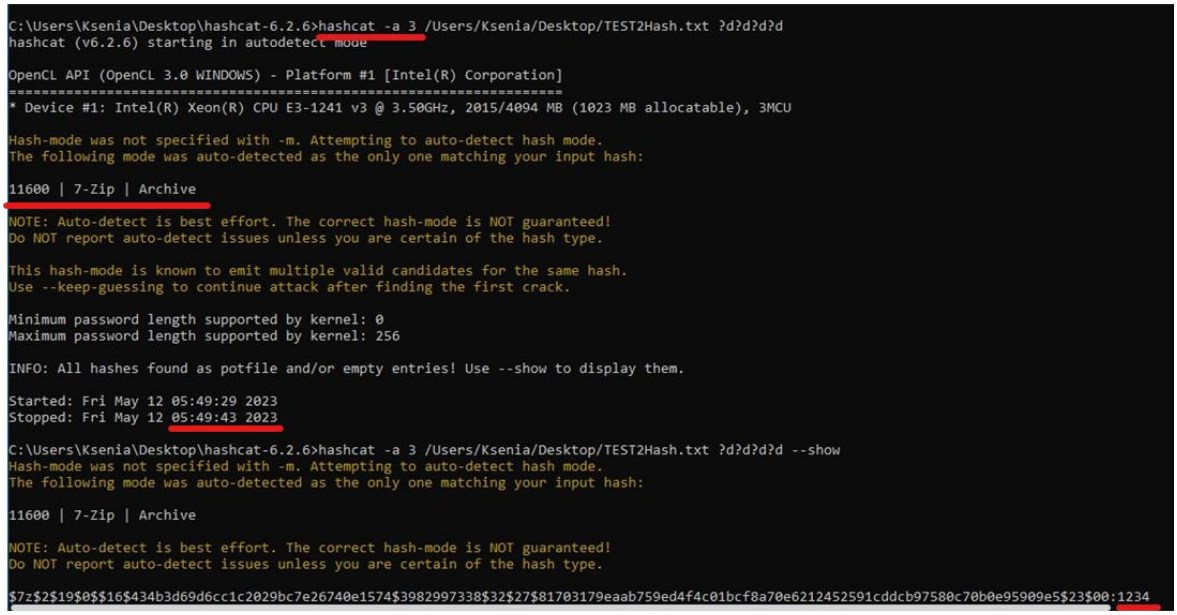
Joonis Lisa 3.3 Ründemeetodeid Hashcat'is



```
- [ Built-in Charsets ] -

  ? | Charset
 ===+=========
  l | abcdefghijklmnopqrstuvwxyz [a-z]
  u | ABCDEFGHIJKLMNOPQRSTUVWXYZ [A-Z]
  d | 0123456789                 [0-9]
  h | 0123456789abcdef           [0-9a-f]
  H | 0123456789ABCDEF           [0-9A-F]
  s |  !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
  a | ?l?u?d?s
  b | 0x00 - 0xff
```

Joonis Lisa 3.4 Sissesisaldatud tähemärkid Hashcat'is



Joonis Lisa 3.5 Parooli otsimine räsi-režiimi määramata