Examensarbete, Högskolan i Halmstad, 10 Juni, 2011

Detektering av krypterade filer

Linus Barkman

Teknologie kandidatexamen i datateknik, 180 ECTS
Inriktning IT-forensik och informationssäkerhet, 15 ECTS

Sektionen för Informationsvetenskap, Data- och Elektroteknik (IDE)
Högskolan i Halmstad

Detektering av krypterade filer

Linus Barkman

Högskolan i Halmstad

Teknologie kandidatexamen i datateknik med inriktning IT-forensik och informationssäkerhet, 15 ECTS

Handledare: Ph.D. Eric Järpe
Examinator: M.Sc. C.E. Urban Bilstrup
Externa handledare: IT-forensiker Mattias Martinsson
Kriminalinspektör Bo Malmsten
Polismyndigheten i Hallands län

10 Juni, 2011

Sektionen för informationsvetenskap, data- och elektroteknik (IDE)
Högskolan i Halmstad

Förord

Jag vill tacka alla som har hjälpt mig med mitt arbete genom att bistå med konstruktiv kritik, idéer, engagemang och tid.

Tack till min handledare Eric Järpe som har visat stort engangemang, lagt ner mycket tid och kommit fram till många bra idéer som jag har använt i arbetet.

Vidare vill jag tacka Mattias Martinsson och Bo Malmsten från Polismyndigheten i Hallands län som urspungligen kom med idén till examensarbetet och har hjälpt till med att svara på frågor som har uppstått.

Tack till Mattias Weckstén som har hjälpt till med programmeringskunskaper.

Abstract

In contemporary encryption the vast amount of text subject to cracking has brought about the demand for methods distinguish files more likely to be encrypted. The encryption software Truecrypt can encrypt files that are not possible to identify with a file signature. To solve the detection problem, an algorithm sensitive to the absence of structure in the very code of files was developed. The program was written in the programming language EnScript which is built into the forensic software suite EnCase. The essential part of the algorithm therefore deployes the statistic of a chi-square test for deviance from a uniform distribution to distinguish files with contents that appear to be random. The program managed to detect encrypted files that were created with Truecrypt. Test results indicate that the newly developed program is nearly double as fast and has at least the same accuracy in the detection as other programs. The software is licensed under open source standard GNU GPL. The procedure developed will drastically facilitate for computer forensic experts to detect if any existing encrypted file is located on the hard drive.

Innehåll

Kapitel 1

Introduktion

Hotet från terrorister och organiserad brottslighet är verklighet och på grund av hoten har en del länder ändrat lagarna. I Sverige har den omdebatterade FRA-lagen införts vilket betyder att all data via kabel eller trådlös trafik som passerar Sveriges gränser kan sparas för att vid ett senare skede analyseras [14]. Efter terroristattacker har Storbritannien stiftat en ny lag som kan leda till fängelsestraff om man inte uppger sina lösenord för krypterad lagringsmedia [15]. En 19-årig man vägrade uppge sitt lösenord till sin krypterade hårddisk och fick sitta 4 månader i fängelse [17]. Att begå brott i Internetvärlden kan vara enklare än i verkligheten, eftersom man inte har den fysiska kontakten med brottsoffren. Det kan även vara så att människor inte har samma respekt för lagar som i den fysiska världen. Det kan finnas en naivitet om att man är anonym och kan göra lite som man vill. Däremot är det väldigt enkelt att spåra en person som har skrivit någonting på Internet om den inte dolt spåren efter sig [3].

Steganografi är konsten att skriva hemliga meddelande och sedan gömma de, t ex i något dokument eller en bild. Steganografi går ut på att inte avslöja dess existens, ingen ska ens kunna veta att det finns hemliga meddelanden i en bild [4]. Inom modern kryptering handlar det inte om att gömma innehållet utan om att transformera texten till oläsbarhet (slumpmässig data) för den som inte har den hemliga nyckeln. Eftersom man enkelt kan använda steganografi och kryptering tillsammans ställer det till en del problem. För en IT-forensiker blir det svårt att identifiera filer eller text som är krypterade inuti en bild.

1.1 Bakgrund

Truecrypt är en välkänd programvara som används för att kryptera användarens data. Det kan vara ett stort problem för IT-forensiker att hitta krypterade filer vid en kriminaltekniskundersökning av en dator. Truecrypt använder sig bland annat utav den omtalade AES (Advanced Encryption

Standard) krypteringen. AES är godkänd krypteringsalgoritm av NSA (National Security Agency) för säkerhetsklassad information (top secret) [13]. För att en IT-forensiker skall kunna göra en kriminalteknisk undersökning av informationen som finns lagrad, krävs det naturligtvis att innehållet inte är krypterat. Polismyndigheten har problem med att komma åt innehållet i en krypterad fil. Beroende på kunskap kan det även vara svårt att identifiera en krypterad fil, eftersom den innehåller inga identifierbara egenskaper som vanliga filer gör. Polismyndigheten handhar sällan fall där personer har krypterat innehållet. När de däremot får ett fall så blir det betydligt svårare, det finns inga klara metoder att arbeta efter. Mattias Martinsson som är IT-forensiker hos Polismyndigheten i Hallands län och spekulerar i att det kommer att bli vanligare med krypterade datamedier.

Ett närmare steg i spekulationen är när Microsoft lanserar sitt operativsystem Windows 7 (Enterprise och Ultimate versioner) förinstallerat med krypteringsprogrammet Bitlocker [12]. Används någon kryptering så blir det genast svårare. Ett alternativ är att försöka lista ut lösenordet, vilket kan vara en minst sagt tidsödande uppgift. Statens Kriminaltekniska Laboratorium (SKL) har skrivit en rapport om hur man bygger ett brute force-kluster som gissar lösenorden [1]. Istället för att använda en dator parallellkopplar man flera datorer som delar på uppgiften, vilket går mycket snabbare. En krypterad fil har inga egenskaper och är på så vis svår att detektera. Min tanke är att försöka detektera krypterade filer med ett program som jag kommer skriva specifikt för ändamålet, vilket kommer underlätta för Polismyndigheten i Hallands län. Programmet kommer att göras i ett programmeringsspråk som heter EnScript. EnScript är utvecklat av Guidance Software och ingår i den forensiska programsviten EnCase [8]. Program som detekterar slumpmässig data finns redan i form av TCHunt [18] som är licensierad under GPLv3 (öppen källkod)[19], däremot finns inte TCHunt som ett EnScript. File Investigator Tools [6] är ett kommersiellt program som detekterar slumpmässig data.

Deniable file system (DFS; förnekelsebara filsystem) är filsystem vars existens inte går att bevisa [5]. Truecrypt tillåter att man gör en krypterad volym där man kan spara filer och anger ett lösenord till detta. Truecrypt tillåter även skapandet av en dold volym i den volym som precis skapades, vilket är vad DFS går ut på. Där behövs ett andra lösenord för att komma åt den sparade informationen. Truecrypt hävdar att det inte går att identifiera den gömda volymen om man inte har ett lösenord. Eftersom informationen på hårddisken är slumpmässig går det inte att identifiera ett mönster [21].

En genomgång av uppsatsen: I detta kapitel introduceras läsaren till bakgrunden om arbetet. Kapitel 2 går igenom modeller och metoder tillsammans med problemformulering och definitioner. I kapitel 3 visas resultaten och de diskuteras senare i kapitel 4. Slutligen redovisas källkoden till programmet i ett appendix.

Kapitel 2

Metoder

Att kunna detektera krypterade filer med en stor träffsäkerhet, dvs. med lågt antal falsk positiva filer, behövs en bra algoritm för att hitta filerna. De krypterade filernas unika egenskaper användes vid detektionen. Det är otroligt viktigt att få ett bra resultat (detektera de krypterade filerna) annars riskerar man att skicka falsk positiva filer för lösenordsknäckning. Att skicka en okrypterad fil (falsk positiv) för lösenordsknäckning innebär att ta upp en köplats i onödan där en krypterad fil skulle kunna knäckas. Att beräkna hur lång tid det i genomsnitt skulle ta att att knäcka ett lösenord använder man följande formel:

$$\text{Antal tecken}^{\text{lösenordets längd}}$$

Antal tecken kan t ex vara bokstäverna a-z (26 bokstäver) och A-Z (26 bokstäver) samt siffrorna 0-9 (10 siffror). Lösenordets längd kan t ex vara 7 tecken.

Ett lösenord på 7 tecken med 62 olika tecken (bokstäver och siffror) ger

$$62^7 = 3\,521\,614\,606\,208$$

olika lösenord. Väljer man att ha ett extra tecken i sitt lösenord så får man 8 tecken i lösenordet. Det ger

$$62^8 = 218\,340\,105\,584\,896$$

olika lösenord. Lösenordets längd är alltså faktorn som kan ge större antal lösenords möjligheter.

Vid SKL (Statens Kriminaltekniska Laboratorium) används ett kluster av datorer som gemensamt försöker knäcka filerna. Det är alltså viktigt att inte skicka filer som inte är krypterade.

2.1 Syfte

Syftet med uppsatsen var att ge Polismyndigheten i Hallands Län ett program som kunde detektera krypterade filer och volymer. Det kan vara svårt att detektera krypterad lagringsmedia manuellt eftersom det finns tusentals filer att gå igenom. Ett program för att automatisera processen och en metod som kan detektera slumpmässig data skulle utformas. Uppsatsen skulle även ge god kunskap inom kryptering och vilka metoder man kan använda för att detektera eller identifiera krypterad lagringsmedia.

2.2 Frågeställning

Hur kan man detektera en krypterad Truecrypt fil med programmet EnCase?

2.3 Encrypted File Locator

Det skrivna programmet Encrypted File Locator letar efter några egenskaper hos de krypterade filerna.

Egenskaper

Programmet letar efter tre egenskaper.

1. Filen får inte innehålla en filsignatur.
2. Filen måste vara minst 19 KB.
3. Filen måste passera ett filter som bygger på χ^2-statistika.

Filter

Första punkten har jag löst genom att göra ett program (Condition) som tar fram alla filer som har en okänd (Unknown) eller en dålig (Bad signature) filsignatur. På så vis kan man enkelt filtrera ut de kända filerna som man vet är vanliga filer och ofarliga. Efter filtret återstår det filer som är okända eller som har dålig signatur. Efteråt exekverar man det andra programmet som letar efter krypterade filer.

Storleksjämförelse

Truecrypt har en filstorleksbegränsning där man inte kan skapa krypterade filer mindre än 19KB. Programmet kontrollerar varje fil och ser om den är 19 KB eller över, annars hoppar programmet över den filen. Det går även att manuellt ändra storlek till t ex 15 MB.

Chi 2-statistika

Beskrivning av Chi 2-statistika finns under metodval.

Inhämtning av data

Programmet går igenom tecken som finns i filen. Funktionen som kontrollerar varje tecken inhämtar ett tecken varje gång och sparar det i en lista (array). När programmet har lyckats inhämta varje tecken från filen är det dags att analysera listan. Vilket använder Chi 2-statistika.

2.4 Definitioner

EnCase

EnCase är en forensisk programsvit som underlättar återskapande av data, sökning efter filer, leta efter information i datorns register och mycket mer.

EnScript

EnScript är programmeringsspråket som används i EnCase. En användare kan skriva egna program för att automatisera processer som skulle ta lång tid om det gjordes manuellt. Programmeringsspråket är en blandning mellan Java och C++.

Container

En Truecrypt container Ãď r en fil i valfri storlek som är krypterad. Användaren väljer en storlek på filen som kan nästan vara obegränsad. Filen kan sedan krypteras med hjälp av Truecrypt. Filen kan ha vilken filändelse som helst, t ex .mp3 eller .doc som skulle kunna vara en mp3 musikfil eller ett textdokument [21].

Volym

En volym syftar på en partition, en hel hårddisk eller ett USB-minne som krypteras likt en fil. Skillnaden är att hela volymen krypteras, det blir inte någon fil. Det är vanligt att kryptera hela USB-minnet som en volym eftersom det är svårare att detektera en stor volym istället för en fil. Detta eftersom hela volymen har slumpmässig data (okrypterad), vilket är svårt att identifiera [22]. Försöker man öppna en krypterad volym med Windows frågar operativsystemet om man vill formatera volymen eftersom det finns slumpmässig data och inget filsystem.

Dold volym

En dold volym är gömd inuti en vanlig volym. På så vis finns det två lösenord, ett första till den vanliga volymen och ett andra till den dolda volymen. Den dolda volymen innehåller också slumpmässig data när den är okrypterad [22].

Operativsystem kryptering

Operativsystem kryptering är en full kryptering av hela hårddisken. Vid uppstart av datorn finns det en ny bootsektor som ber om lösenord. Det är alltså inte en vanlig operativsystem inloggning. När man skrivit in rätt lösenord så startas ditt operativsystemet [20].

Operativsystem kryptering med dold volym

Operativsystem kryptering med dold volym fungerar som operativsystem kryptering, dock finns möjligheten att ha ett gömt operativsystem som loggar in med ett speciellt lösenord. Det fungerar som en dold volym fast man har ett operativsystem installerat istället. Truecrypt hävdar att det är omöjligt att bevisa att en gömd volym existerar [20].

Kryptering

Kryptering betyder att man har en klartext och sedan utför man en så kallad krypteringsalgoritm mot klartexten. Algoritmen utför en serie förutbestämda regler, t ex att byta ut en del bokstäver mot andra bokstäver [16]. När algoritmen är slutförd får man en oläslig text. Det är då en krypterad text.

Krypteringsalgoritm

Krypteringsalgoritmen är den specifika algoritmen som används vid krypteringen. En algoritm kan vara säkrare än vad en annan är. Exempel på algoritmer är AES och RSA.

Dekryptering

Dekryptering är som kryptering fast åt andra hållet. Krypteringsalgoritmen körs baklänges. På så vis får man fram klartexten från det krypterade textmeddelandet [16].

Hashvärde

Ett hashvärde är ett resultat som man får om man använder en hashfunktion mot en text eller en fil. Hashfunktionen är gjord så att man inte skall kunna använda den åt andra hållet, alltså ha ett hashvärde och sedan få fram text. Hashvärdet kan variera i storlek beroende på hashfunktionen [7].

Metadata

Metadata är data om data. Det vill säga information om data. Det vanligaste metadata i Windows sammanhang är MAC (Modified, Accessed, Created) stämplarna. MAC är tidsstämplar när en fil är Modifierad, Åtkommen och Skapad [2].

Filsignatur

En filsignatur är något unikt som varje fil använder sig av för att bli igenkänd. I början av filen skrivs det ut, t ex har .JPEG-bilder signaturen JFIF.

2.5 Presentation av Truecrypt

Truecrypt är en krypteringsprogramvara som är flitigt använd. Den har laddats ner över 17 miljoner gånger och den laddas ner med mer än 10 000 gånger per dag [23]. Anledningen till den stora framgången är att produktsviten är gratis och tjänar sitt syfte mycket väl, dvs att kryptera användarens information.

Truecrypt använder en så kallad *on-the-fly-encryption* och det betyder att informationen man använder dekrypteras och krypteras automatiskt utan att användaren behöver tänka på det, förutom när man skall ange lösenordet. Det betyder också att man inte behöver vänta på att en fil ska hinna dekrypteras eftersom Truecrypt tar en liten bit av filen hela tiden och dekrypterar. På så vis slipper användaren sitta och vänta på att filerna ska dekrypteras, ungefär som när man ser på TV: programmet syns så fort man sätter på TV:n och man slipper vänta i 60 minuter för att programmet ska laddas in. Truecrypt fungerar likadant, användaren slipper vänta på att hela filen ska laddas in. Truecrypt krypterar hela filsystemet, det betyder att filnamn, storlek, metadata osv inte kan ses utan ett korrekt lösenord. Truecrypt sparar aldrig ner krypterat data på hårddisken utan det sparas temporärt i RAM-minnet [24].

Truecrypt är ett enkelt krypteringsprogram för användaren men det finns framförallt avancerade funktioner som är närmast omöjliga att knäcka. Truecrypt använder sig utav flera olika krypteringsalgoritmer för att skydda data. En speciell algoritm som används är Advanced Encryption Standard (AES) med 128, 196 och 256 bitars kryptering. AES är vinnaren av en tävlingen från NIST (National Institute of Standards and Technology). NIST anordnade en tävling där deltagare fick göra en egen krypteringsalgoritm och vara med i tävlingen. NIST efterlyste en krypteringsstandard som skulle användas hos federala myndigheter, AES vann tävling och är de facto standard idag. AES är godkänd av NSA (National Security Agency) för säkerhetsklassad information [13]. Truecrypt har även en funktion som kan addera flera krypteringsalgoritmer ovanpå varandra t ex med Serpent, Twofish och AES

[25].

Truecrypt kan användas på fem olika sätt:

1. Container
2. Volym
3. Dold volym
4. Operativsystem kryptering
5. Operativsystem kryptering med dold volym

Slumptalsgenerator

Truecrypt har en slumptalsgenerator som skall skriva över hårddisken med slumpmässig data så fort man avmonterar en Truecrypt volym. Slumptalsgeneratorn påverkas av följande variabler:

- När användaren rör musen.
- När användaren skriver.
- När klockan på datorn ändras.
- I Windows används en inbyggd funktion för att slumpmässigt hämta data kallad CryptoAPI.
- I Windows används NETAPI32 som tillhandahåller nätverksstatistik som används för slumpmässig data.
- I Linux och Mac OS X så finns det en Random Number Generator (RNG) inbyggd och det finns under /dev/random eller /dev/urandom.

Kombinerar man dessa variabler kan man få en funktion som har flera slumptalsgeneratorer i en. Funktionen i sin tur påverkar hur datorn skall slumpmässigt skriva data till hårddisken.

2.6 Tidigare forskning

Guidance Software

Guidance Software, företaget som utvecklar EnCase har tagit fram ett program (script) som letar efter krypterade filer (slumpmässig data). Tyvärr har man ingen möjlighet att kontrollera källkoden och se vilken metod de använder. Programmet finns att ladda ner på Guidance Software forum, däremot kan bara användare som har betalat licens för deras programvara EnCase få tillgång till forumet.

Tchunt

Tchunt är ett program som letar efter krypterade Truecrypt filer. Programmet är skrivet i C++ och letar efter 4 unika egenskaper som varje fil måste innehålla för att den skall detekteras som krypterad:

1. Filstorleken modulo 512 måste vara noll. Filstorlek $\equiv 0 \pmod{512}$
2. Filstorleken är minst 5MB.
3. Filen måste passera en χ^2-statistika.
4. Filen får inte innehålla en vanlig filsignatur.

2.7 Metodval

Att välja vilken metod som skall användas är en väsentlig del i det skrivna programmet. Metod skall vara snabb, vara relativt enkel att implementera och framförallt ge ett bra resultat. De metoder som var aktuella var följande:

Entropi

Entropi från engelskans Entropy, kan beskrivas som oförutsägbarheten i en datamängd. Entropin kan vara från 0 till 100%. Datamängden i det här fallet är de 256 olika kategorierna. Entropin är noll om teckenföljden är helt förutsägbar och 100% om den är oförutsägbar. En vanlig mening eller ett ord har en låg entropi eftersom det är enkelt att gissa ett ord. Slumpmässig data har därför en hög entropi, vilket skulle kunna användas för detektering av krypterade filer.

Chi 2-statistika

För att detektera slumpmässig data behöver man först analysera de data som finns i varje fil. Det görs genom en frekvensanalys av varje tecken i filen. En frekvensanalys räknar hur många tecken det finns av varje tecken i filen. Man jämför de faktiska frekvenserna mot de förväntade [9]. När det inte finns mer tecken att räkna utför man en analys av de data man får av frekvensanalysen, sen fortsätter man med nästa fil. Eftersom det kan ta lång tid att gå igenom varje tecken på en hårddisk och fil har jag gjort så att användaren kan välja hur många megabyte den vill att programmet ska analysera. Programmet utför ett Chi 2-statistika för att kontrollera om det är slumpmässig data eller inte.

Metoden som valdes var Chi 2-statistika eftersom det är en erkänd statistisk metod.

2.8 Metod att skapa en krypterad fil

För att skapa en krypterad fil i Windows med Truecrypt och dess användargränssnitt valdes följande hårdvara samt inställningar:
Virtualiseringsmjukvara: VMware Player Version 3.1.3 Build 324285
Processor: Intel Core 2 Duo 1.86 GHz
Minne: 2 GB RAM minne
Operativsystem: 32-bitars Windows 7

1. Installation utav VMware Player
2. Installation utav Windows 7 i VMware Player som fick 9 GB hårddisk 1 GB RAM minne och 1 processor kärna.
3. Installation av Truecrypt volym med standard inställningar.

Inställningar för Truecrypt

1. Truecrypt Container (krypterad fil)
2. AES (förinställt) krypteringsalgoritm och RIPEMD-160 (förinställt) hashsumma
3. 1021 MB "Container"
4. Filesystem : FAT Cluster: Default (Ingen "quick format")

2.9 Metod att skapa flera krypterade filer

För att skapa flera krypterade filer på en och samma gång användes följande hårdvara respektive mjukvara:
Virtualiseringsmjukvara: VMware Player Version 3.1.3 Build 324285
Processor: Intel Core 2 Duo 1.86 GHz
Minne: 2 GB RAM minne
Operativsystem: Ubuntu 10.04

Jag använde mig utav ett script som heter make_tc.sh och är ett bash script som är gjort utav företaget 16 Us. Källkoden finns på följande adress https://github.com/16s/TCHunt/blob/master/make_tc.sh.

2.10 Metod att skapa krypterad volym

För installation av operativsystem och programvara användes samma tillvägagångssätt som i metod 2.8.

Inställningar för Truecrypt

1. Encrypt a non-system partition/drive
2. Hidden TrueCrypt volume
3. Normal mode
4. Välj enheten, t ex ett USB-minne
5. AES (förinställt) krypteringsalgoritm och RIPEMD-160 (förinställt) hashsumma
6. 149 GB "Outer Volume"
7. Filesystem : FAT Cluster: Default (Ingen "quick format")
8. Samma inställningar användes för skapandet av den dolda volymen

2.11 Metod vid skapandet av ett fiktivt fall

Polismyndigheten kan tyvärr inte lämna ut riktig data pga. sekretessregler. För att analysera programmen behövs då data. Ett fiktivt fall gjordes med följande metoder.

1. Windows XP installerades med Service Pack 3.
2. Datorn användes genom kopiering, nerladdning av filer, besök på slumpmässiga hemsidor etc.
3. Det skapades ett antal misstänkta krypterade filer samt några helt och hållet slumpmässiga filer (se metod 2.6, 2,7 2,8).
4. Det skapades även en partition som sedan krypterades med Truecrypt och används som en volym med dold volym inuti den.

Kapitel 3

Resultat

3.1 Exempel 1

Filnamn: SCRIPT.BAT
Filstorlek: 524 288 000 bytes

En exempel fil används som har filstorleken 500MB. Ett urval av 1MB (1048576 bytes) där varje bokstav motsvarar 1 byte. Det kommer alltså att observeras 1048576 tecken. Är inte filen 1MB stor beräknar programmet hela filstorleken istället. T ex om en fil är 1038576 bytes vilket är under 1MB ändras programmet till att använda 1038576 tecken. Datorer kan använda olika teckenformateringar. Det går att omvandla den vanliga teckenformateringen ANSI till siffror. Resultatet blir 256 olika kategorier, t ex så motsvarar bokstaven A siffran 65 i ANSI-tabellen. Varje siffra i tabellen motsvarar ett tecken, och även mellanslag finns inräknat som tecken.

För att göra beräkningar behöver man veta de faktiska frekvenserna/ observerade (observed) samt de förväntade frekvenserna (expected). Programmet har ordning på hur många tecken det finns av varje bokstav. Det observerade värdet kan t ex vara 3954st av siffran 0.

Det förväntade värdet får man fram genom att summera ihop alla observerade frekvenser och dividera med antalet grupper, i mitt fall 256 olika tecken i ANSI-tabellen. Ett exempel på det förväntade värdet av 1 MB blir.

$$\frac{1048576}{256} = 4096$$

Hela det statistiska materialet fördelar sig i en frekvenstabell enligt en kurva. Det förväntade värdet blir alltså 4096 och har betydelsen av tyngdpunkten [10].

För att räkna ut Chi 2-statistika använder man denna formel. Det är ett mått på skillnaderna mellan de observerade frekvenserna och de förväntade frekvenserna [11].

$$S = \sum_i \frac{(O_i - E_i)^2}{E_i}$$

Enligt vårt exempel så blir det

$$S = \frac{(3954 - 4096)^2}{4096} + \frac{(4030 - 4096)^2}{4096} + \frac{(4050 - 4096)^2}{4096}$$

Resultatet av varje kategori (256) summeras ihop för att få fram ett S-värde (statistika värde). I exemplet finns det bara tre grupper som summeras, i programmet är det 256 kategorier samt 256 observerade värden. Ett exempel vid summering av tre kategorier är

$$\begin{aligned} S &= 4.99285 + 1.06348 + 0.51660 \\ &= 6.50293 \end{aligned}$$

Summering av 256 kategorier för den ovan nämnda filen blev 252.83 vilket får ett p-värde > 0.05 (5 %) varmed vi inte kan förkasta nollhypotesen om likformig fördelning på någon rimlig signifikansnivå. Det innebär att filen är starkt misstänkt att vara en Truecrypt-fil. Kritiska värden (dvs värden att jämföra statistika med) är listade i följande tabell:

Signifikansnivå	*Värde*
5 %	293.2478
1 %	310.4574
1 ‰	330.5197

3.2 Exempel 2

Filnamn: heidelb.jpg
Filstorlek: 35268 bytes
Räkna ut det förväntade värdet (expected).

$$\frac{35268}{256} = 137.766$$

Det förväntade värdet blir alltså 137. Talet blir mindre än vad det var vid det första exemplet, och det är på grund av att filstorleken är mindre i det här exemplet. Summering av kategorierna sker med följande formel.

$$S = \sum_i \frac{(O_i - E_i)^2}{E_i}$$

S-värdet blir resultatet av de summerade kategorierna, t ex i exemplet nedan med endast tre kategorier:

$$\begin{aligned} S &= \frac{(732-137)^2}{137} + \frac{(209-137)^2}{137} + \frac{(374-137)^2}{137} \\ &= 2584.12 + 37.83 + 409.99 \\ &= 3031,95 \end{aligned}$$

S-värdet av de 256 kategorierna blir 4819.67 vilket innebär att testet i detta fall får p-värdet $< 10^{-6}$. Vi kan alltså förkasta nollhypotesen om likformig fördelning på varje signifikansnivå. Detta innebär att filen definitivt inte bedöms som en misstänkt Truecrypt-fil.

3.3 Exempel 3

Filnamn: Truecrypt.exe
Filstorlek: 1496528 bytes
Räkna ut det förväntade värdet.

$$\frac{1048576}{256} = 4096$$

Det förväntade värdet blir alltså 4096.

$$S = \sum_i \frac{(O_i - E_i)^2}{E_i}$$

Summera ihop alla 256 kategorierna.

$$\begin{aligned} S &= \frac{(3412-4096)^2}{4096} + \frac{(2107-4096)^2}{4096} + \frac{(34337-4096)^2}{4096} \\ &= 114,22 + 965,85 + 223271,01 \\ &= 224351 \end{aligned}$$

Det totala resultatet blev 4 642 482.40, vilket innebär att testet i detta fall får p-värdet $< 10^{-6}$ och vi kan förkasta nollhypotesen om likformig fördelning på varje signifikansnivå. Detta innebär att filen definitivt inte bedöms som en misstänkt Truecrypt fil.

3.4 Jämförelse av resultat för olika program

Jämförelse av filer med filter

Det skrivna filtret (condition) används för att sortera bort kända filtyper.

Inställningar	*Encrypted File Locator*	*Encrypted Data Finder*
Minsta fil (MB)	5	5
Läst data	5	5
Antal filer	2742	2742
Detekterade filer	100%	100%
Falsk-positiva	0	2
Tid (sekunder)	318	622

Jämförelse av filer utan filter

Programmen exekveras på alla filer på hela hårddisken. Progammen använder inte något filter.

Inställningar	*Encrypted File Locator*	*Encrypted Data Finder*
Minsta fil (MB)	5	5
Läst data (MB)	5	5
Antal filer	13866	13866
Detekterade filer	100%	100%
Falsk-positiva	5	5
Tid (sekunder)	427	816

Jämförelse av program vid detektion av krypterade volymer

En jämförelse av program som kan detektera dolda Truecrypt volymer. I programmet EnCase kan man markera filer/partitioner. Encrypted File Locator analyserar den markerade filen/partitionerna, det spelar alltså ingen roll om det är en fil eller en partition.

Inställningar	*Encrypted File Locator*	*Encrypted Data Finder*
Minsta fil (MB)	5	5
Läst data (MB)	5	5
Antal partitioner	1	1
Detekterade filer	100%	0%
Falsk-negativ	0	1
Tid (sekunder)	3	2

Kapitel 4

Diskussion

Det skrivna programmet EFL (Encrypted File Locator) rekommenderas att användas tillsammans med ett filter (condition) som tar bort kända filtyper som den vet inte är krypterade filer, t ex JPEG-bilder. Det forensiska programmet EnCase som utför analyser på data har en funktion som analyserar kända och okända filtyper på bara några sekunder (filsignatur). Efter filsignaturanalysen kan man använda filtret som medföljer EFL, de reducerar antalet misstänkta filer. Filtret skulle även gå att använda tillsammans med EDF (Encrypted Data Finder v1.1) som också använder EnCase. Resultatet visar att EFL är nästan dubbelt så snabbt när filtret används. EFL visar sig också vara nästan dubbelt så snabbt när programmet används utan ett filter. I testet fann EFL alla de krypterade filerna och markerade inga falsk-positiva (false-positives) träffar. EDF fann också samtliga krypterade filer och två filer som var falsk-positiva. Ett falsk-positivt svar betyder i det här fallet att det är två filer som har blivit markerade som krypterade filer fast de inte är så. Att EDF visar falsk-positiva svar är troligtvis på grund av en annan detekteringsalgoritm. Tyvärr går det inte att ta reda på det eftersom källkoden inte finns för granskning.

TChunt har ingen möjlighet att använda filter. Programmet måste söka igenom varje fil för att kontrollera filtypen. Programmet TChunt är uppbyggt på ett mycket snarlikt sätt som EFL. TChunt har dock en ytterligare egenskap, nämligen att den använder en matematisk formel. De allra flesta skapar krypterade filer i Megabyte, och tack vare det kommer alla filer att få resultatet 0 vid uträkning. Uträkningen för filegenskapen följer:

$$\text{Filstorlek} \equiv 0 \pmod{512}$$

Nackdelen med att göra så här är att om man tar bort eller lägger till en byte så kommer resultatet inte att bli 0, vilket skulle vara enkelt att undgå programmet. Exempel på en 6 MB fil. 6 MB blir 6291456 bytes.

$$6\,291\,456 \bmod 512 \quad = \quad 0$$

$$\begin{aligned} 6MB &= 6291456 + 1 \bmod 512 = 1 \\ 6MB &= 6291456 - 1 \bmod 512 = 511 \end{aligned}$$

Detektering av krypterad partition

Det skrivna programmet lyckades detektera en partition som var krypterad med Truecrypt. Jag har inte hittat någon tidigare forskning i ämnet, utan det verkar som de flesta har inriktat sig mot krypterade filer. Resultatet anses väldigt lyckat.

Syftet med undersökningen var att kunna detektera krypterade filer med hjälp av EnCase. Resultatet blev över förväntan.

Den generella slutsatsen är att det skrivna programmet var snabbare och hade bättre eller lika bra träffsäkerhet som de andra programmet.

Fortsatt forskning

Programmet är långt ifrån klart och skulle kunna ses som en arbetsprototyp. För att bygga vidare på programmet skulle det behövas ett grafiskt gränssnitt som kunde underlätta för IT-forensiker. Vidare så skulle funktioner som att läsa flera bytes åt gången och läsa in bytes från olika delar av filerna vara ytterligare förbättringar av programmet. Eftersom programmet kan detektera Truecrypt filer vore nästa steg att försöka gissa lösenordet. Att automatiskt exportera de detekterade filerna till ett beräkningskluster för gissning av lösenordet vore en lösning att få fram innehållet. Eventuellt skulle man kunna göra ett program som söker igenom den misstänktes dator och sammanställer alla ord som finns på hårddisken till en fil. Textfilen används sedan för att försöka få fram lösenordet till den krypterade filen en så kallad ordlistattack. Fungerar inte det får man prova bruteforce metoden som gissar lösenord.

Det finns även ett annat intressant område att forska vidare på, nämligen run-längder. En run-längd är antal tecken som finns i rad eller direkt efter ett likadant tecken. Till exempel

ij24tkmsdf0iannnwkdpaf90h42iptnnngsdfhu8oasd

I texten ovan kan man urskilja att det finns två run-längder som har tre stycken n (nnn) efter sig. Det kan vara att man skulle kunna upptäcka ett visst mönster i slumptalsgeneratorn som skulle kunna indikera att det är en krypterad fil.

Begränsningar med programmet

De begränsningar som finns med programmet är: vid användning av filtret utförs en sortering för att ta bort kända filer. Att ändra programmet Truecrypt till att lägga till en känd filsignatur i varje krypterad fil skulle göra det svårare att detektera filerna eftersom de kommer ha likadan filsignatur som vanliga filer. Det går att hitta filerna om man inte använder filtret, dock kan man få andra falsk positiva svar vilket man vill undvika.

Referenser

[1] J. Bengtsson (2007)
Parallel Password Cracker: A Feasibility Study of Using Linux Clustering Technique in Computer Forensics. S 1-9 Digital Forensics and Incident Analysis, 2007. WDFIA 2007. Second International Workshop on, doi: 10.1109/WDFIA.2007.4299375

[2] H. Carvey (2009)
Windows Forensic Analysis. Syngress; rev 2, Kapitel 5, S 299

[3] E. Casey (2004)
Digital Evidence and Computer Crime. Academic Press; rev 2, Kapitel 18 Sid 477

[4] E. Casey (2001)
Handbook of Computer Crime Investigation. Academic Press, Kapitel 1 Sid 6

[5] A. Czeskis et al. (2003)
Defeating Encrypted and Deniable File Systems: TrueCrypt v5.1a and the Case of the Tattling OS and Applications. Sid 1
Anträffbar: http://www.schneier.com/paper-truecrypt-dfs.pdf

[6] File Investigator Tools (2009)
Anträffbar: http://www.forensicinnovations.com/blog/?p=7

[7] D. Gollmann (2011)
*Computer Security.*Wiley; rev 3, Kapitel 14, Stycke 14.3.2, S 258

[8] Guidance Software, EnCase (2011)
Anträffbar: http://www.guidancesoftware.com/forensic.htm

[9] S. Körner och L. Wahlgren (2009)
Statistiska metoder. Studentlitteratur, S 153

[10] S. Körner och L. Wahlgren (2009)
Statistiska metoder. Studentlitteratur, S 154

[11] S. Körner och L. Wahlgren (2009)
Statistiska metoder. Studentlitteratur, S 155

[12] Microsoft (2010)
Förstärk skyddet för dina filer med hjälp av Bitlocker-diskkryptering.
Anträffbar:
http://windows.microsoft.com/sv-SE/windows7/products/features/bitlocker

[13] National Security Agency (2003)
National Policy on the Use of the Advanced Encryption Standard (AES) to Protect National Security Systems and National Security Information. S 2.
Anträffbar: http://csrc.nist.gov/groups/ST/toolkit/documents/aes/CNSS15FS.pdf

[14] M. Olofsson och M. Odenberg (2007)
Regeringens proposition 2006/07:63, En anpassad försvarsunderrättelseverksamhet
Anträffbar: http://www.regeringen.se/content/1/c6/07/83/67/2ee1ba0a.pdf

[15] Regulation of Investigatory Powers Act 2000 (2003)
Investigation of electronic data protected by encryption. Kapitel III
Anträffbar: http://www.legislation.gov.uk/ukpga/2000/23/part/III

[16] W. Stallings (2007)
Computer Security Principles and Practice. Prentice Hall, Kapitel 2, Stycke 2.1, S 42

[17] Sveriges Radio (2010)
Anträffbar:
http://sverigesradio.se/sida/artikel.aspx?programid=1646&artikel=4081787

[18] TCHunt (2011)
Anträffbar: http://16s.us/TCHunt/

[19] TCHunt (2011)
Anträffbar: https://github.com/16s/TCHunt

[20] Truecrypt (2011)
http://www.truecrypt.org/docs/?s=hidden-operating-system

[21] Truecrypt (2011)
Anträffbar: http://www.truecrypt.org/docs/?s=tutorial

[22] Truecrypt (2011)
Anträffbar: http://www.truecrypt.org/docs/?s=hidden-volume

[23] TRUECRYPT (2011)
Anträffbar: http://www.truecrypt.org/statistics

[24] TRUECRYPT (2011)
Anträffbar: http://www.truecrypt.org/docs/intro

[25] TRUECRYPT (2011)
Anträffbar: http://www.truecrypt.org/docs/?s=cascades

Appendix

Nedan finns programmeringskod som har använts i programmet EnCase och är skrivet i programmeringsspråket EnScript.

4.1 EnScript

```
/**********************************************************************
#    This program is free software: you can redistribute it and/or modify
#   it under the terms of the GNU General Public License as published by
#    the Free Software Foundation, either version 3 of the License, or
#    (at your option) any later version.
#
#    This program is distributed in the hope that it will be useful,
#    but WITHOUT ANY WARRANTY; without even the implied warranty of
#    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#    GNU General Public License for more details.
#
#    You should have received a copy of the GNU General Public License
#    along with this program.  If not, see <http://www.gnu.org/licenses/>.
#
# **********************************************************************
#
#    This program is a result of a 15 ECTS course in the Computer Forensic
#    programme at Halmstad University.
#    If you have any questions, don't hesitate to drop me an e-mail.
#
#    Linus Barkman
#    E-mail: linus.barkman@gmail.com
#    Project website: http://www.0xff.se/Encrypted-File-Locator
#**********************************************************************
*/
class MainClass {
  typedef int[]      IntArray;
  void Main(CaseClass c) {
```

```
    SystemClass::ClearConsole(1);
    //Variables
    int data; //Keeping one byte at a time
    int i;
    double E; //Expected value for Chi 2
    double O; //Observed value for Chi 2
    double M; //Sum of all of the 256 categories (E-O)^2/E + (E-O)^2/E...
    double CalcSampling; //Formula for calculations
    int LoopCounter; //Keeps track of the loop
    double AntalTecken; //Keeps track of number of characters
    int max=331; //Highest value -
    //Everything below 331 indicates random data
    int counter; //A normal counter
    DateClass now; //Creates a dateclass
    now.Now(); //Initiate a variable
    int start = now.GetUnix(); //Gets the time
    Console.WriteLine(now.GetString() +" Script starts");
    //Prints out the date, time and that the scripts starts
      forall (EntryClass entry in c.EntryRoot()){
        int bytes=1024*1024*5; //Characters to sample
// Go through all files bigger then 5242880 bytes
        if (entry.LogicalSize()>5242880 && entry.IsSelected()==1){
          if (entry.LogicalSize()<bytes){
  //A check to see if the file size is smaller then bytes to sample
            bytes = entry.LogicalSize();
//If the file size is smaller the sample size will reduce to file size.
            }
        IntArray tecken(256); //Array with 256 characters
        //Zero out the array
        M=0;E=0;AntalTecken=0;LoopCounter=0;CalcSampling=0;
        EntryFileClass file;
        file = new EntryFileClass(); //Creates a FileClass
         file.Open(entry); //Open the file
          file.SetCodePage(CodePageClass::ANSI);
//Choosing ANSI as the text encoder
          for (i=0;i<bytes;i++){
           data = file.Get(); //Gets one byte at a time ---
//--- Allow the program to get more bytes at a time to speed up the program
           //Console.WriteLine("Read data "+i+ " : " +data);
           tecken[data] = tecken[data]+1; //Add character to the array
          } LoopCounter =i;//Quit for loop (comments if necessary)
            for( i = 0; i < 256; i++){
             O = tecken[i]; //Observed characters in the array
```

```
            E = bytes/256; //Expected value
            CalcSampling = (E-O)*(E-O)/E; //Do some calculations
            //Console.WriteLine("Calculation: "+CalcSampling);
            M=M+CalcSampling;
            AntalTecken=AntalTecken+O;
//Debugging
        //Console.WriteLine("Number: "+i+ " Number of characters: " +O + " M: " + M);
//i = counter for array, O = Observed characters

            }
              if(M<max ){
               counter++;
               now.Now();
               Console.WriteLine(now.GetString() + " "  + counter + ": "
+ entry.FullPath()+" Logical Size: "+ ((entry.LogicalSize()/1024)/1024)+ " MB");
                //Console.WriteLine("Value of M: "+M);
               //Console.WriteLine("For loop Counter: "+LoopCounter);
               //Console.WriteLine("Number of characters: "+AntalTecken);
             }
         //Quit if statement
         }
       } //Quit Foor loop


      //Prints out the execution time
      now.Now();
      if ((now.GetUnix() - start)>60){
      uint minut = (now.GetUnix() - start)/60;
      Console.WriteLine("Finished in: "+ minut + " minutes"
+ " ("+ (now.GetUnix() - start) +") +");
      }else if((now.GetUnix() - start)==1){
      Console.WriteLine("Finished in: "
+ (now.GetUnix() - start) + " second");
      }
      else{
      Console.WriteLine("Finished in: "
+ (now.GetUnix() - start) + " seconds");
      }
      Console.WriteLine("There was "+counter+ " suspected files");
    }//Forall loop quits
  }//Caseclass quits
```

4.2 Condition

```
class MainClass {
  typedef String[] Array1;

  typedef String[] Array2;

  Array1 Variable1;
  Array2 Variable2;

  MainClass():
    Variable1{"! Bad Signature"},
    Variable2{"Unknown"}
  {
  }

  bool Main(EntryClass e) {
    return Variable1.Find(e.Signature()) >= 0
    || Variable2.Find(e.Signature()) >= 0;
  }
}
```