Linnéuniversitetet
Kalmar Växjö

Examensarbete

Diskkrypteringsprestanda i GNU/Linux

Författare: Sebastian Ahlstedt &
David Granath Karlsson
Handledare: Marcus Wilhelmsson
Examinator: Jacob Lindehoff
Termin: VT13
Kurskod: 1DV41E
Nivå: B

Abstract

This thesis compares the impact on disk performance in a GNU/Linux environment with three encryption algorithms: AES, Serpent and Twofish in three different implementations: DM-crypt, Loop-AES and Truecrypt. For all three algorithms a key length of 256 bits is used.

The thesis shows that the least performance impact during data encryption, and thus file writing, is reached by using AES or Twofish encryption implemented in DM-crypt or TrueCrypt. The thesis shows that some data operations with a sufficiently low processor utilization barely affects disk performance at all if encrypted using the optimal implementation and algorithm.

It is also discovered that the performance impact during data decryption, or file reading, can be minimized by using the most efficient implementation and algorithm. The best results are met with the AES or Twofish cipher, regardless of implementation.

An important conclusion that is drawn is that it is hard to determine a superior encryption solution for all purposes. However, by reviewing and examining the collected data from all aspects of disk performance the AES implementation in TrueCrypt is, with small marginals, determined to be the most optimal.

Key Words: Encryption, AES, Rijndael, Serpent, Twofish, DM-crypt, Loop-AES, Truecrypt, hard drive, performance, benchmark.

Abstrakt

Undersökningen jämför hur krypteringsalgoritmerna AES, Serpent och Twofish implementerade i DM-crypt, Loop-AES och TrueCrypt påverkar diskprestandan i en GNU/Linux-miljö. Samtliga krypteringsalgoritmer tillämpas med en nyckellängd på 256 bitar.

Undersökningen visar att högst skrivhastighet och således krypteringshastighet uppnås med algoritmerna AES och Twofish i DM-crypt eller Truecrypt. Krypteringens påverkan på diskprestanda är beroende av vilka typer av operationer som utförs och hur hög processorsysselsättningsgraden är. Vid simplare I/O-operationer har den mest optimala krypteringslösningen knappt någon påverkan på diskprestandan överhuvudtaget.

Undersökningen visar också att påverkan av prestanda vid dekryptering är minst vid tillämpning av AES eller Twofish oavsett implementering.

En viktig slutsats som dras är att det inte finns en överlägsen krypteringslösning för samtliga användningsområden. Genom att sammanställa och granska insamlad data gällande samtliga aspekter beträffande diskprestanda fastställs TrueCrypts implementation av AES till att vara den mest optimala, trots små marginaler.

Nyckelord: Kryptering, AES, Rijndael, Serpent, Twofish, DM-crypt, Loop-AES, Truecrypt, hårddisk, prestandamätning.

Tack

Tack till vår handledare Marcus Wilhelmsson som varit en betydande resurs vid genomförandet och författandet av undersökningen.

Innehåll

1 Introduktion

I det här kapitlet förklaras bakgrunden till problemet som undersökningen ämnar lösa. För undersökningen defineras också en frågeställning, vilken delvis baserats på tidigare forskning, och en avgränsning. Begrepp och termer som används i undersökningen förklaras i kapitlet begrepp och terminologi.

1.1 Bakgrund

Det existerar ett flertal krypteringsalgoritmer vars säkerhet är fastställd och bekräftad [4]. Den stora kvarstående utmaningen är att avgöra hur mycket införandet av datakryptering påverkar prestandan i ett datorsystem. I den här undersökningen jämförs tre krypteringsalgoritmer i tre olika implementationer under GNU/Linux för att avgöra om, och i vilken utsträckning, diskprestanda påverkas vid kryptering.

1.2 Tidigare forskning

Utbudet av tidigare publicerad forskning inom området är högst begränsad. I artikeldatabasen *IEEE Xplore* finns publiceringar tillgängliga där undersökningar och granskningar av enstaka algoritmer har genomförts. I några få artiklar jämförs prestanda för olika algoritmer men i samtliga fall skiljer sig utförande och användningsområde från det som är relevant för den här undersökningen.

I publiceringen *AES Finalists Implementation for GPU and Multi-core CPU based OpenCL* jämförs prestanda för krypteringsschiffrena AES, Twofish och Serpent. Tillämpningsmetod, syfte och sammanhang skiljer sig till stor del från det som behandlas i den här undersökningen. I deras undersökning mäts prestanda som genomströmningskapacitet och OpenCL-implementationer i s.k XTS-läge tillämpas i syfte att kunna genomföra prestandatester mot grafikprocessorer. Två tester är genomförda mot flerkärniga huvudprocessorer och de resultaten ger tydliga indikationer på vad som kan förväntas av den här undersökningen [1].

Övrig tidigare forskning inom området har bedrivits av Robin Olsson, Linnéuniversitetet, år 2012. Olsson presenterar sin forskning i form av en B-uppsats med titel *Performance differences in encryption software versus storage devices*[1]. I sin undersökning granskar Olsson prestandan hos tre olika krypteringsapplikationer, vilka alla tillämpar de symmetriska krypteringsalgoritmerna AES, Twofish och Serpent. I Olssons studie undersöks, till skillnad från den här undersökningen, krypteringsapplikationer i en Windowsmiljö samt att prestandajämförelser utförs mellan traditionell mekanisk disk och SSD-disk [43].

I kapitel 6.2, "Proposal for future research" uppmuntrar Olsson till och ger förslag på vidare forskning inom ämnet. Ett av de förslag som Olsson presenterar innebär liknande tester som han själv genomförde, med skillnaden att andra krypteringsapplikationer undersöks i en Linuxmiljö. De resultat som Olsson presenterar i sin undersökning ger, precis som *AES Finalists Implementation for GPU and Multi-core CPU based OpenCL*, indikationer på vilka resultat som kan förväntas [43].

[1] http://urn.kb.se/resolve?urn=urn:nbn:se:lnu:diva-20315

1.3 Problemformulering och syfte

Med kryptering införs ytterligare operationer, vilka utförs mellan operativsystem och lagringsmedie. Att en viss kostnad i form av prestandaförsämring uppstår vid införandet av datakryptering, såvida inte dedikerad hårdvara används, är ett ofrånkomligt faktum. Diskprestanda definieras i undersökningen som läs- och skrivhastighet samt söktider vid såväl sekventiella som slumpartade operationer.

Frågeställningen är av intresse både ur ett vetenskapligt och praktiskt perspektiv. Ur en vetenskaplig betraktningsvinkel förväntas resultatet från undersökningen väcka nya frågeställningar, vilka förhoppningsvis uppmuntrar till vidare forskning. Ur ett praktiskt perspektiv innebär den besvarade frågeställningen att val av krypteringstillämpning går att motivera med hjälp av empirisk data. Genom att kunna välja den minst prestandakrävande kombinationen av implementation och algoritm sänks också krav på hårdvara, vilket i slutändan innebär förbättrade förutsättningar vid inköp av hårdvarukomponenter i förhållande till pris.

Frågan undersökningen ämnar besvara är: hur kan prestandaförlusten hos ett lagringsmedie minimeras genom valet av krypterings-implementation och algoritm?

1.4 Avgränsning

Undersökningen tar endast hänsyn till diskprestanda, vilket innebär att säkerheten beträffande valda implementationer och algoritmer ej granskas. Vidare är undersökningen begränsad till krypteringsimplementationerna DM-Crypt, Loop-AES och TrueCrypt samt krypteringschiffrena AES-256, Twofish-256 och Serpent-256. Implementationerna undersöks i deras standardutförande och inga försök till prestandaoptimering genomförs då det anses vara upp till utvecklaren av programvaran. Samtliga moment i genomförandet utförs mot samma hårdvara.

Kontentan av den förundersökning som föregick undersökningen visar att inga kända och dokumenterade för- eller nackdelar berör aktuell hårdvara och krypteringsimplementationer samt algoritmer. Trots genomförd förundersökning existerar en potentiell risk för odokumenterade bristfälligheter mellan aktuell hårdvara samt tillämpade algoritmer. Således blir reservationer vad beträffar eventuella hårdvarurelaterade för- och nackdelar nödvändiga.

Undersökningen utförs endast mot mekanisk disk och hur krypteringen påverkar prestandan hos andra lagringsmedium undersöks inte.

1.5 Begrepp och terminologi

AES: Advanced Encryption Standard - Benämningen på krypteringsalgoritmen Rijndael efter att den 2001 kronats vinnare i tävlingen som hölls av NIST.

GPL: GNU General Public License - En mjukvarulicens ursprungligen utvecklad av Richard Stallman. Licensen finns i olika versioner men innebär kortfattat friheten att använda, granska, distribuera samt modifiera mjukvaran.

NIST: National Institute of Standards and Technology - En del av USAs *Department of Commerce*.

FIPS: Federal Information Processing Standards - Standarder som på begäran av USAs regering framtagits av NIST. Ett FIPS-dokument utvecklas av NIST vid statliga krav på säkerhet och interoperabilitet när ingen acceptabel industristandard existerar [2].

Feistelnätverk: Ett feistelnätverk är en generell metod för att transformera en funktion till en permutation. En permutation kan beskrivas som produkten av flera transformationer. Den funktion som utgör grunden i varje block i ett feistelnätverk har funktionen att skapa den nyckelberoende mappningen mellan en sträng indata till en sträng utdata [3].

1.6 Målgrupp

Undersökningens resultat har såväl vetenskapliga som praktiska tillämpningsområden. Vetenskapligt riktar sig undersökningen till forskare och studenter inom datavetenskap, vilka uppmuntras till vidare forskning inom området. Praktiskt förväntas resultaten kunna utgöra en resurs vid införandet av privata och kommersiella krypteringstillämpningar.

Läsaren bör ha grundläggande förståelse för problemområdet samt syftet med kryptering.

2 Teori

I det här kapitlet presenteras den tekniska och teoretiska bakgrund vilken utgör en grundförutsättning för läsarens fullständiga förståelse av genomförandet och de resonemang som förs. Vidare förses läsaren med teori och bakgrundsinformation vad beträffar tillämpade krypteringsalgoritmer, krypteringsimplementationer samt testverktyg.

2.1 Krypteringsalgoritmer

Fastställd säkerhet är ett krav samt den mest grundläggande egenskap för de krypteringsalgoritmer som tillämpas i undersökningen. Det som i den här undersökningen anses vara avgörande för ett krypteringsschiffers trovärdighet ur ett säkerhetsperspektiv är: om schiffret har varit tillgängligt för granskning, hur länge schiffret har varit tillgängligt för granskning, antalet kvalificerade kryptoanalyser schiffret har genomgått, kvaliteten på dokumentation av genomförda granskningar samt källan varifrån dokumentationen härstammar.

Rijndael, Twofish och Serpent utgjorde tre av fem finalister i AES-tävlingen som arrangerades av NIST [4]. Efter krav från USAs regering formulerades ett FIPS-dokument, vilket 1997 resulterade i en officiell förfrågan efter förslag på algoritmer vilka skulle uppfylla följande grundläggande krav: Algoritmen skulle vara ett blockschiffer baserad på symmetrisk nyckel-kryptografi, varje block skulle utgöras av minst 128 bitar och nyckellängder utgörande av 128, 192 och 256 bitar.

År 1998 accepterades femton kandiderande algoritmer som deltagare till tävlingen och NIST efterfrågade analysering av de antagna kandidaterna från kryptografiska forskarsamfundet. Analyserna innefattade djupgående undersökningar och forskning vad beträffar effektiviteten samt säkerheten hos de accepterade kandidaterna [5].

Baserat på resultatet av den preliminära forskning som presenterades 1999, vid den andra AES-kandidatkonferensen, tog NIST beslutet att välja MARS, RC6, Rijndael, Serpent och Twofish som de slutgiltiga finalisterna. I oktober år 2000 presenterades vinnaren av tävlingen, Rijndael, som därefter kom att benämnas Advanced Encryption Standard [6].

För ytterliggare information om den genomgångna processen samt kriterier som de fem finalisterna bedömdes utifrån finns rapporten från tävlingens andra runda att tillgå i NISTs dokumentarkiv[2]. I rapporten beskriver NIST bland annat att samtliga av de fem finalisterna verkar, baserat på genomförda analyser, besitta adekvat säkerhet för att kunna vinna AES-tävlingen [7].

2.1.1 AES (Rijndael)

Rijndael, numera AES, utvecklades av kryptologerna Joan Daemen och Vincent Rijmen inför valet av NISTs officiella avancerade krypteringsstandard. AES är ett 128-bitars blockschiffer, vilket kan tillämpas med hjälp av 128, 192, eller 256 bitar långa krypteringsnycklar [8]. Det karaktäristiska och variabla för varje variant av AES är nyckellängden, vilken också används för att referera till olika AES-implementationer enligt: AES-128, AES-192 samt AES-256 [8]. Schiffret bygger på en unik substitutionslinjär transformation som görs i tre lager [9]. Antalet rundor, i vilken

[2] http://csrc.nist.gov/archive/aes/round2/r2report.pdf

transformationen genomförs beror på nyckellängden och vid tillämpning av AES-256 gäller 14 rundor.

2.1.2 Twofish

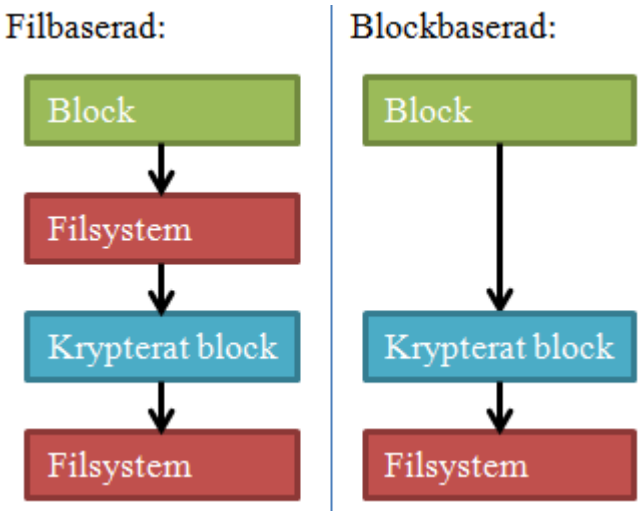
Twofish utvecklas och publicerades av Bruce Schneier, John Kelsey, Niels Ferguson, Doug Whiting, David Wagner och Chris Hall. Twofishalgoritmen är ett 128-bitars blockschiffer, med stöd för nyckellängder på 128, 196 och 256 bitar [10]. Algoritmen bygger på ett Feistelnätverk i ett utförande om 16 rundor.

2.1.3 Serpent

Serpent utvecklades av Ross Anderson, Eli Biham samt Lars Knudsen och är licensierad under GPL [11] . Serpent är ett schiffer som bygger på ett 32-rundors substitution- och permutations-nätverk (SP-nätverk) vilket utgörs av fyra ord om 32 bitar. Varje serpent-block utgörs av 128 bitar och algoritmen har stöd för nyckellängder om 128, 196 och 256 bitar [12].

2.2 Krypteringsimplementationer

DM-crypt, Loop-AES och TrueCrypt har stöd för kryptering av både blockenheter och filer. Den här undersökningen behandlar endast kryptering av blockenheter för att undvika det ytterligare abstraktionslager en filbaserad blockemulering medför [13]. Förhållandet mellan de olika lösningarna presenteras i figur 1.



Figur 1: Filbaserade respektive blockbaserade krypteringsimplementationer.

Implementationerna som behandlas i undersökningen arbetar i realtid vilket innebär att data som skrivs till det krypterade blocket krypteras automatiskt och transparent för den skrivande applikationen. När data efterfrågas från det krypterade blocket dekrypteras denna automatiskt och läses ifrån RAM-minnet [13][14][15].

2.2.1 Loop-AES

Loop-AES bygger delvis på Linuxkärnans loop-modul som används för att hantera virtuella blockenheter skapade från godtycklig data såsom fysiska blockenheter eller filer. Detta ger möjlighet till att till exempel låta ett filsystem existera i en vanlig fil [16]. I Linuxkärnan finns också funktionen cryptoloop som utökar loop-modulen med krypteringsstöd. Med cryptoloop kan data transparent krypteras och dekrypteras i bakgrunden när den skrivs eller läses mot den virtuella blockenheten.

Linuxkärnans cryptoloop och därmed dess loop-modul besitter dock kryptologiska svagheter vilket påpekades av kryptologen Markku-Juhani O. Saarinen i hans whitepaper *Encrypted Watermarks and Linux Laptop Security* där han bevisade uppenbara brister i implementationen [17].

Kryptologen Jari Ruusu, skapare och huvudansvarig utvecklare av Loop-AES, fann liknande svagheter i andra krypteringsimplementationer, bland annat de dåvarande versionerna av TrueCrypt och DM-crypt. Svagheterna bevisade han via källkod publicerad i offentliggjord elektronisk kommunikation [18].

Syftet med Loop-AES är att ersätta kryptologiskt underlägsna alternativ, såsom cryptoloop, med en säkrare implementation. För att uppnå detta ersätts Linuxkärnans nativa loop-modul med Loop-AES egna modul. Ovanstående kryptologiska svagheter övervinns genom att datan delas upp och krypteras med olika nycklar. Användaren behöver dock bara memorera en nyckel som sedan används för att dekryptera en fil innehållande resterande nycklar. Om nyckelfilen skapas på korrekt vis i enlighet med Loop-AES medföljande dokumentation görs den resistant mot krypto-attacker såsom bruteforce och kan utan säkerhetsrisker lagras i klartext [13].

Med Loop-AES medföljer inga särskilda användarrymdsverktyg och administration sker med program normalt härrörande från GNU-paketet util-linux såsom mount och losetup. För att ge de programmen stöd för Loop-AES flernyckelsystem krävs det dock att util-linux modifieras med hjälp av en patch som levereras tillsammans med källkoden för Loop-AES [13].

Modulen har stöd för krypteringsalgoritmerna AES, Blowfish, Twofish samt Serpent [13].

2.2.2 TrueCrypt

TrueCrypt är en krypteringsimplementation med stöd för plattformarna Windows, Linux och Mac OS X. Truecypt kan skapa virtuella krypterade blockenheter utifrån filer eller fysiska blockenheter [19]. På Linuxplattformen är TrueCrypt anpassningsbart och kan nyttja funktioner i Linuxkärnan såsom device-mapper om de finns tillgängliga och annars användarrymdsbibliotek såsom FUSE [20].

För att minimera prestandaförlusten använder TrueCrypt bland annat parallellisering för att nyttja multipla processorkärnor. Datan som ska krypteras eller dekrypteras delas upp i mindre bitar och operationerna sprids ut över samtliga tillgängliga kärnor. Med parallellisering menar TrueCrypt Foundation att krypterings- och dekrypteringshastigheten är direkt proportionerlig mot antalet kärnor [21].

TrueCrypt har stöd för AES, Twofish och Serpent samt kombinationer av dessa utförda i seriella operationer av krypteringsalgoritmerna [22].

2.2.3 DM-crypt

DM-crypt är en nativ funktion i Linuxkärnan som bygger på device-mapper-infrastrukturen. Device-mapper verkar som ett virtualiseringslager mellan kärnan och fysiska blockenheter och kan bortsett från kryptering användas för upprättandet av olika förhållanden mellan blockenheter såsom RAID eller logisk volymhantering [23].

För administrering av device-mappers funktioner krävs verktyg i användarrymden - Cryptmount är ett exempel på ett dylikt verktyg. Syftet med Cryptmount är bland annat att göra det enkelt för oprivilegierade användare att montera och avmontera krypterade filsystem vid behov [24].

Då DM-crypt direkt bygger på krypteringsgränssnittet i Linuxkärnan stöds samtliga underliggande krypteringsimplementationer vilket bland annat medför nativt stöd för AES, Twofish och Serpent [15].

2.3 Testverktyg

I det här kapitlet förklaras och presenteras de testverktyg som tillämpas vid genomförandet av undersökningen. För samtliga prestandatestmjukvaror följer beskrivning om vilka typer av operationer som utförs mot disk eller filsystem. Kapitlet ska ge läsaren en klar uppfattning om hur tillämpade testverktyg tillgodoser behovet av att kunna testa de egenskaper som i kapitel 1.3 definieras som diskprestanda.

2.3.1 Bonnie++ 1.96

Bonnie++, utvecklat av Russel Coker och licensierad under GPL 2.0, är ett program för prestandamätningar baserat på Tim Brays *Bonnie benchmark*. De tolv tester som mjukvaran innefattar är alla I/O-relaterade och egenskaper för lagrinsmedia och tillhörande filsystem prestandatestas utifrån sekventiell läs- och skrivhastighet samt slumpade sökningar. De sex integrerade testerna från *Bonnie benchmark* innefattar filsystemsaktiviteter, vilka av University of Waterloo, genom observering, har bekräftats utgöra flaskhalsar i applikationer med hög I/O-frekvens. Testerna utförs mot en fil av känd storlek och resultatet returneras i antal bearbetade kilobytes per sekund samt processorns sysselsättningsgrad i procent [25].

De andra sex testerna innefattar filoperationer baserade på POSIX-funktionerna *stat*, *creat* och *unlink* [25]. *Stat* används för att erhålla information om status på önskad fil [26], *unlink* är en funktion som används för att avlägsna länkar till önskad fil [27] och *creat* används för att skapa ny eller skriva över en befintlig fil [28]. Filoperationerna kombineras i syfte att simulera ett verkligt scenario där överföringar sker tvåvägs mellan användarrymd och fysisk diskenhet [25].

2.3.2 dd (coreutils 8.13)

dd är en del av kärnuppsättningen, *coreutils*, av paket i GNUs bibliotek av förvaltade paket och verktyg. Verktyget används primärt för kopiering och konvertering av filer, med möjligheter att ändra I/O-blockstorleken. dd stödjer läsning från både fil och *standard in* samt skrivning till både fil och *standard out* [29].

2.3.3 Phoronix-test-suite 3.6.1

Phoronix-test-suite erbjuder en uppsättning av tester, vilka tillhandahålls av openbenchmarking.org. Tester som inkluderas i Phoronix-test-suite grupperas i olika testsamlingar i syfte att förenkla kategoribaserade testsessioner relevanta för den hårdvara som önskas testas. Samtliga tillämpade tester utgör moduler i Phoronix disk-svit, vilket är en samling av tester vars syfte är att tillhandahålla tester som utför verklighetsbaserade disk- och filsystems-operationer [30]. För att uppnå tillförlitliga resultat exekverar Phoronix varje test flera gånger tills standardavvikelsen når en acceptabel nivå [31].

Nedan följer en beskrivning av det urval av tester ur Phoronix-sviten som tillämpas i undersökningen och således utgör relevans för en slutgiltig bedömning. Varje rubrik innehåller testets namn följt av versionen på testprofilen i Phoronix-test-suite.

2.3.3.1 Gzip Compression 1.1.0

Testet mäter tiden det tar att komprimera en 2 GB stor fil bestående av nollor. Datan läses från mediet som utvärderas och utdatan kastas. Vid utförande av test nyttjar Gzip Compression den nativa versionen av Gzip i operativsystemet. Testet utvärderar således processorprestandan samt läshastigheten hos lagringsmediet [32].

2.3.3.2 Compile Bench 1.0.0

Compile Bench är ett testverktyg framtaget av Chris Mason vid Oracle. Verktyget avser att mäta tiden ett antal olika filsystemsoperationer tar att utföra i en katalogstruktur som utsätts för en simulerad realistisk degradering. Testet innefattar bland annat uppackning, patchning, kompilering och borttagning av filer [33]. Testerna görs i slumpartad ordning och Phoronix ansvarar för att minimera standardavvikelsen genom att exekvera compile bench flera gånger [34].

2.3.3.3 IOzone 1.8.0

IOzone är en programvara som utgörs en samling av testverktyg vars syfte är att prestandatesta I/O-enheter i form av filsystemsoperationer. Rättigheterna för mjukvaran ägs av William D. Norcott men det är fritt för allmänheten att använda och distribuera programmet så länge upphovsrättsinformationen inte manipuleras [35]. Iozone erbjuder möjligheten att utföra tester som innefattar filhanteringspresanda för standard- och specialoperationer. Standardoperationer finns tillgängliga i både sekventiellt och slumpmässigt utförande. Följande operationer stöds: read, write, re-read, re-write, read backwards, read strided, fread, fwrite, random read/write, pread/pwrite variants, aio_read, aio_write och mmap [36].

Läs- och skrivoperationer utförs genom både obuffrade samt, med hjälp av biblioteksfunktioner som fwrite() och fread(), buffrade läsningar och skrivningar. Vid skrivoperation skapas en helt ny fil och vid läsoperation läses en befintlig fil. Både läs- och skrivoperationer finns i ett alternativa utförande som innebär återskrivning samt omläsning. Vid återskrivning genomförs skrivning av en redan existerande fil. I praktiken innebär det att metadata ej blir nödvändig att skrivas om och således är återskrivning mindre resurskrävande än en standardskrivning. Vid omläsning läses en fil som nyligen blivit läst, vilket innebär lägre resursförbrukning då cache-funktioner nyttjas [36].

Läs- och skrivoperationer finns även tillgängliga i ett slumpmässigt utförande där läsning och skrivning av och till en fil genomförs mot slumpmässiga punkter i en fil. Systemcache, antal diskar och söktider utgör de mest vitala och grundläggande faktorerna vid slumpmässiga tester [36].

2.3.3.4 Dbench 1.0.0

Dbench är ett testverktyg framtaget av Andrew Tridgell och Ronnie Sahlberg, vilka är delaktiga i Samba-projektet. Dbench-verktyget använder så kallade loadfiles för att utföra specifika I/O-operationer. Exempel på dylika operationer är öppning och stängning av filer eller sökningar. Operationerna sker vid specifika tidpunkter vilket innebär att identiska operationer minutiöst kan jämföras på olika datorsystem. Loadfiles kan utformas för olika ändamål till exempel att emulera en databasserver eller

godtycklig filserver [37]. Phoronix använder Dbenchs medföljande loadfile, client.txt, som emulerar en lokal filserver. Phoronix testar med 1, 6, 12, 48, 128 och 256 simultant anslutna emulerade klienter [38].

3 Metod

För att uppnå tillförlitliga resultat genomfördes en kvantitativ och deduktiv studie. För att kunna säkerställa relevansen och trovärdigheten beträffande insamlad data inhämtades först nödvändig och relevant kunskap inom ämnet. Studien bestod av ett flertal olika tester vars resultat insamlades som empirisk data. Resultaten ligger till grund för de analyser och slutsatser som dras i senare kapitel.

3.1 Genomförande

För att bevara en konsekvent experimentmiljö utfördes samtliga tester under liknande förhållanden. Testerna utfördes på samma server och potentiella störningskällor minimerades genom terminering av irrelevanta tjänster och processer innan de utfördes. De processer som var igång under testen återfinns i bilaga A. Samtliga mätningar utfördes åtskilliga gånger i syfte att minimera risken för anomaliteter och i slutändan inkorrekt data samt felaktig bedömning.

3.1.1 Förundersökning

I syfte att fastställa ett lämpligt urval av algoritmer och implementationer, samt de verktyg, vilka dessa ska granskas med, initierades undersökningen i form av en mindre förundersökning i tre delar.

Målet med den första delen av förundersökningen var att fastställa vilka algoritmer som lämpas för att inkluderas i den här typen av undersökning. Huvudsakliga informationskällor som låg till grund vid valet av krypteringsalgoritmer utgjordes av resurser i NISTs dokumentarkiv. NIST betraktas som tillförlitlig källa då det är en del av USAs *Department of Commerce* [39]. NIST grundandes år 1901, vilket gör det till ett av USAs äldsta vetenskapslaboratorier [39]. Utöver dokumentation tillhandahållen av NIST utgjorde Robin Olssons *Performance differences in encryption software versus storage devices* en betydelsefull tillgång vid val av algoritmer.

Utifrån NISTs dokumentation angående tävlingen i sig, längden på tävlingen, engagemanget kring tävlingen samt den stora mängd kryptoanalyser som finalisterna har genomgått betraktas AES, Twofish och Serpent i den här undersökningen som säkra schiffer.

Syftet med den andra delen av förundersökningen var att välja ut ett antal olika krypteringsprogramvaror, där samtliga besitter egenskapen att stöd för implementering av valda algoritmer finns tillgängligt. Krypteringsimplementationernas stöd för aktuell plattform, GNU/Linux, utgjorde också ett grundläggande krav och en förutsättning. På grund av bristen av officiella eller vetenskapliga undersökningar angående effektiviteten för tillgängliga implementationer på marknaden genomfördes en informell frekvens- och kompabilitetsanalys. Analysen innebar att undersöka förekomsten av olika krypteringsimplementationer i fem stora Linuxdistributioner. Vid genomförandet av analysen utgjorde distributionernas användarforum samt dokumentation den största tillgången. De distributioner som analysen tar hänsyn till är: Ubuntu, Archlinux, Debian, Fedora och CentOS, vilka, enligt distrowatch, alla under det senaste året har ett snitt på mer än 800 träffar per dag [40].

Tredje delen av förundersökningen genomfördes i syfte att identifiera, kvalitetssäkra samt eliminera tillgängliga verktyg vad beträffar test av diskprestanda under Linuxplattformen. Identifiering av testverktyg innebar endast inhämtning av

information om välkända metoder och programvaror som kan tillämpas för aktuella prestandatester. Att kvalitetssäkra innebar att bastester, det vill säga mot okrypterad disk, genomfördes i syfte att utvärdera kompatibilitet samt resultatets trovärdighet. Kvalitetssäkringen inkluderade också inhämtning av information som eventuellt kan påvisa eller antyda att någon av valda algoritmer drar uppenbar nackdel av prestandatestet i fråga. Eliminering av testverktyg innebar att prestandatestprogramvaror, vilka visade sig otillfredsställande eller tveksamma av någon anledning, betraktades som ej tillämpningsbara och således exkluderades från undersökningen. Vid identifiering av testverktyg användes tillgänglig dokumentation som en del av kvalitetssäkringen och vid uppenbara brister genomfördes eliminering i tidigare skede, vilket innebar att dessa verktyg inte kvalificerade sig för bastester.

3.1.2 Testmiljö

Samtliga tester utfördes på en Dell PowerEdge Server 1850 med en SCSI-ansluten hårddisk av märke och modell Seagate ST373207LC. I tabell 1 presenteras fullständiga hårdvaruspecifikationer för Dell PowerEdge 1850:

Processor	Intel Xeon 3.4 GHz @ 3.39GHz (dual core)
Moderkort	Dell 0d8266
Diskkontroller	LSI Logic / Symbios Logic 53c1030 PCI-X Fusion-MPT Dual Ultra320 SCSI
Chipset	Intel E7520 + ICH5 / ICH5R
Minne	4 x 512 MB DDR2 @ 400 MHz
Disk	73 GB Seagate ST373207LC
Grafikkort	AMD Radeon 7000/VE 128 MB
Nätverkskort	Intel 82541GI Gigabit

Tabell 1: Hårdvaruspecifikationer för Dell PowerEdge 1850.

3.1.3 Testmjukvara

I syfte att kunna säkerställa dragna slutsatsers korrekthet definieras prestanda som en kombination av följande egenskaper och kriterier: läshastighet, skrivhastighet och söktid. Såväl sekventiell som slumpartad läsning och skrivning undersöks för att emulera realistiska scenarion.

De program som används är dd från coreutils-8.13, bonnie++-1.96 samt diskrelaterade delar av testbiblioteket i Phoronix-test-suite-3.6.1. Samtliga tester utförs, explicit eller implicit, flera gånger i syfte att uppnå tillförlitliga resultat. Antalet gånger varje test utförs baseras på standardavvikelsen för testen. Testerna utförs sekventiellt för varje krypteringsimplementation och algoritm på samma partion för att utesluta eventuella externa faktorer. Mellan varje test återställs partionen i syfte att återskapa identiska förutsättningar för samtliga test.

3.1.4 Basundersökning

Basundersökningen, vilken är en del av förundersökningen, innebär att bastester genomförs, dvs tester utförs mot en okrypterad partition, med syftet att skapa referenspunkter för vidare tester i en miljö med tillämpad kryptering. Resultatet av bastesterna ligger till grund för bedömning av trovärdigheten av insamlad data. Förutom att säkerställa trovärdighet och rimlighet för respektive test utgör bastestresultaten en grundläggande tillgång när beslut om vilka tester som betraktas som väsentliga ska tas.

Vid utvärdering av bastester undersöks samtliga tester i syfte att säkerställa att ingen av valda krypteringsimplementationer besitter egenskaper som utgör en risk för avvikelser eller felaktiga mätvärden. Efter slutförd utvärdering av bastester förväntas en tydlig bild ha erhållits över vilka tester som är aktuella eller inaktuella för vidare användning.

3.1.5 Motivering till val av testverktyg

Det första testet som utförs för varje krypteringsimplementation och algoritm är sekventiell skrivning och läsning med dd. Anledningen till att testet exekveras är att det är intuitivt och dess resultat lättförståeligt. Testet exekveras fem gånger och en standardavvikelse på max tre procent anses vara acceptabel. Då testet snabbt levererar ett resultat lägger det grunden inför vad som kan förväntas av efterföljande tester.

Samtliga tester ur Bonnie++-sviten exekveras fem gånger vilket resulterar i information om läs/skriv-hastighet, antal sökningar och metadataoperationer per sekund samt processorbelastningen för varje operation.

Ur sviten Phoronix-test-suite utförs följande tester:

- Gzip Compression
- Compile Bench
- IOzone
- DBench

Flera av testerna undersöker ett antal olika realistiska scenarion vilket innebär att verklighetsbaserad utvärdering av krypteringsimplementationerna blir möjlig.

3.1.6 Installation och användning av mjukvara

Kapitlet ämnar att detaljerat beskriva hur undersökningen förbereddes genom installation av testservern, krypteringsimplementationerna samt testmjukvaran. Kapitlet ämnar ge en grund för att möjliggöra konsistenta upprepningar av undersökningen och med ekvivalent hårdvara uppnå likartade resultat.

3.1.6.1 Ubuntu server 12.04

64-bitars-versionen av operativsystemet installerades via installationsmediet vilket hämtades från Ubuntus officiella webbplats[3]. Under installationen accepterades standardinställningarna bortsett från att under diskpartitioneringsdelen upprättades följande struktur:

Operativsystem: 10 GB
Testpartition: 30 GB

Partitioneringen innebar att det inte skapades någon swap-partition vilket var ett medvetet val då det hade kunnat påverka testresultaten. En användare med godtyckligt användarnamn skapades under installationen vilken är irrelevant för testresultaten då samtliga tester utfördes av root-användaren.

3.1.6.2 Testmjukvara

Bonnie++-1.96 installerades från Ubuntus officiella programbibliotek med instruktionen *apt-get install bonnie++*. Vid utförandet av mätningarna exekverades programmet med följande parametrar:

[3] http://www.ubuntu.com/download/server

- -d /mnt: Utförs i katalogen /mnt.
- -u root: Som användare root.
- -s 4g: Skapar filer på 4 GB, dokumentationen för Bonnie++ rekommenderar att filstorleken minst motsvarar den dubbla mängden tillgängligt RAM-minne [3].
- -n 256: Skapar 256 * 1024 filer under filskapningstestet, Om ett test utförs för snabbt kan inte bonnie++ kalkylera fram ett trovärdigt resultat. Ett stort antal filer medför att testet tar längre tid och resultatet blir således mer tillförlitligt.
- -x 5: Utförs fem gånger, resultatet från förstudien visar att detta ger en godtagbar standardavvikelse.

Phoronix-test-suite-3.6.1 installerades med instruktionen *apt-get install phoronix-test-suite*. För att konfigurera testsviten exekverades det först utan parametrar vilket ledde till att standardkonfiguration för programmet genererades i */root/.phoronix-test-suite/*. I filen */root/.phoronoix-test-suite/user-config.xml* modifierades parametern *enviromentDirectory* till /mnt/installed-tests vilket innebär att samtliga test i sviten utförs mot katalogen */mnt*. För att utföra testen exekverades *phoronix-test-suite* följt av önskat test. Samtliga av de inkluderade testprogrammen i Phoronix-sviten exekverades med standardparametrar.

dd är en del av Ubuntus coreutils-paket, *coreutils-8.13*, och krävde således ingen explicit installation. Vid mätning av skrivhastighet exekverades dd med följande parametrar:

- bs=1M: Operationen utförs mot block av storleken 1 MB.
- if=/dev/zero: /dev/zero används som indatafil.
- of=/mnt/test: /mnt/test används som utdatafil.
- count=4096: Operation utförs på 4096 block, vilket med ovanstående blockstorlek betyder 4 GB totalt.

Vid mätning av läshastighet exekverades dd med identiska parametrar med undantag för följande modifikationer:

- if=/mnt/test: /mnt/test används som indatafil.
- of=/dev/null: /dev/null används som utdatafil.

3.1.6.3 TrueCrypt 7.1a

TrueCrypt installerades från den exkeverbara installationsfilen truecrypt-7.1a-linux-console-x64 som hämtades från TrueCrypts officiella webbplats[4]. För att skapa krypterade block med programmet exekverades det med parametern *--create* varpå resten av instruktionerna infördes via en interaktiv meny. För undersökningarna accepterades standardinställningarna i TrueCrypt bortsett från att aktuell krypteringsalgoritm; AES, Serpent eller Twofish valdes samt filsystemet ext4. För att montera krypterade block exekverades TrueCrypt med sökvägen till blockenheten, */dev/sda2*, och monteringspunkten, */mnt*, som parametrar.

3.1.6.4 Cryptmount 4.2.1 för DM-crypt

För att administrera DM-crypt installerades användarrymdsverktyget Cryptmount med instruktionen *apt-get install cryptmount* vilket installerade paketet *cryptmount-4.2.1-1*. För att skapa krypterade block redigeras filen */etc/cryptmount/cmtab*. För AES256 utformades den enligt följande:

[4] http://www.truecrypt.org/downloads

test {
dev=/mnt/test
fstype=none
fsoptions=defaults
cipher=aes
keyformat=builtin
keyfile=/etc/cryptmount/test.key
}

För övriga krypteringsalgoritmer behöver endast direktivet *cipher* redigeras. En nyckelfil genererades sedan med instruktionen *cryptmount --generate-key 32 test* där *test* motsvarar blockets namn i */etc/cryptmount/cmtab* och *32* anger nyckellängden i bytes. För att förbereda blocket för montering exekverades Cryptmount med parametern *--prepare test*, vilket skapar en virtuell krypterad uppmappad enhet under */dev/mapper/test*. Den virtuella enheten är därefter tillgänglig för skapande av filsystem samt montering.

3.1.6.5 Loop-AES 3.6h

Installationen av Loop-AES innebar modifieringar av Linuxkärnan och vissa användarrymdsverktyg vilket hade kunnat påverka övriga krypteringsimplementatoner. I syfte att undvika påverkning av övriga testresultat utfördes installationen och utvärderingen av Loop-AES efter att övriga undersökningar slutförts. Användarrymdsverktygen installerades med instruktionen *apt-get install loop-aes-utils* vilket installerade paketet *loop-aes-utils-2.16.2-2*.

För att installera kärnmodulerna hämtades nödvändiga kompileringsverktyg samt Linuxkärnans omodifierade källkod med instruktionen *apt-get install dpkg-dev linux-source*. För att minimera antalet modifieringar behölls konfigurationen från Ubuntus aktuella kernel genom att kopiera konfigurationsfilen */boot/config-3.5.0-23-generic* till filnamet *.config* i root-katalogen för källkoden. Kompilering av Linuxkärnan genomfördes med hjälp av instruktionen *make*.

Källkoden[5] till Loop-AES hämtades från Sourceforge, vilket är programvarans officiella distributionskanal. Kompileringen av kärnmodulerna genomfördes med följande instruktioner: *make LINUX_SOURCE=~/linux-source-3.5.0 EXTRA_CIPHERS=y*. Anledningen till förekomsten av parametern *EXTRA_CIPHERS=y* är att moduler för Serpent och Twofish inte byggs som standard.

För att förbereda ett krypterat block användes verktyget losetup vilket härrör från paketet loop-aes-utils. För till exempel krypteringsalgoritmen AES256 exekverades programmet med instruktionerna *-e AES256 /dev/loop0 /dev/sda2* där */dev/loop0* är en godtycklig ledig loop-enhet och */dev/sda2* är den faktiska filen eller blockenheten som det krypterade blocket ska existera inom. För att kunna exekvera losetup krävs det att följande moduler laddas för respektive algoritm:

- loop för AES
- loop_serpent för Serpent
- loop_twofish för Twofish

[5] http://loop-aes.sourceforge.net/

4 Resultat

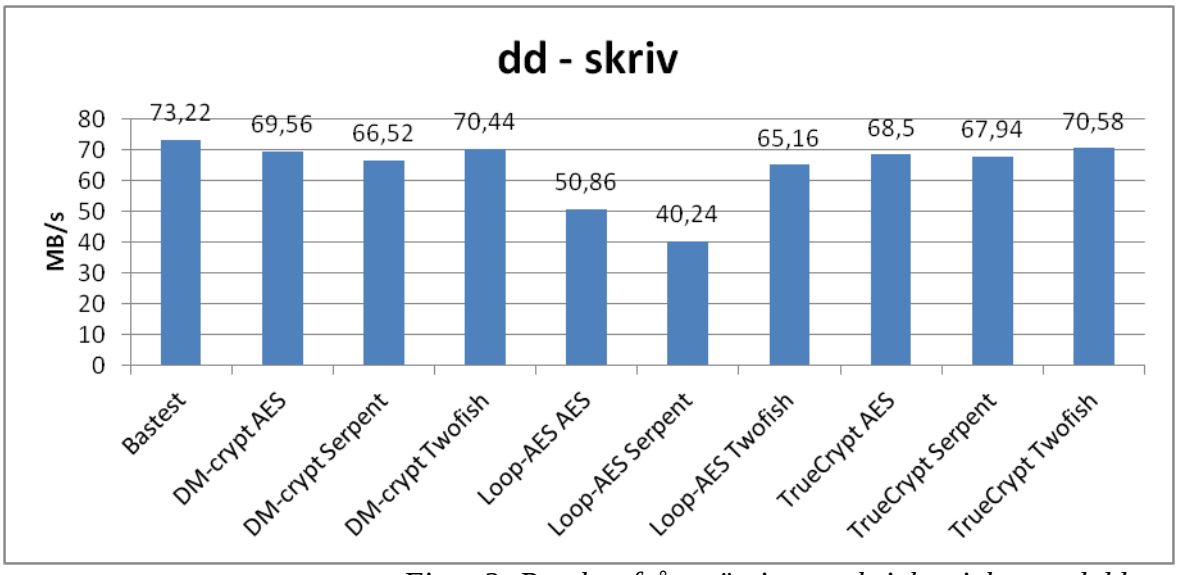
I det här kapitlet presenteras resultatet från de mätningar och tester som genomfördes i undersökningen.

4.1 dd

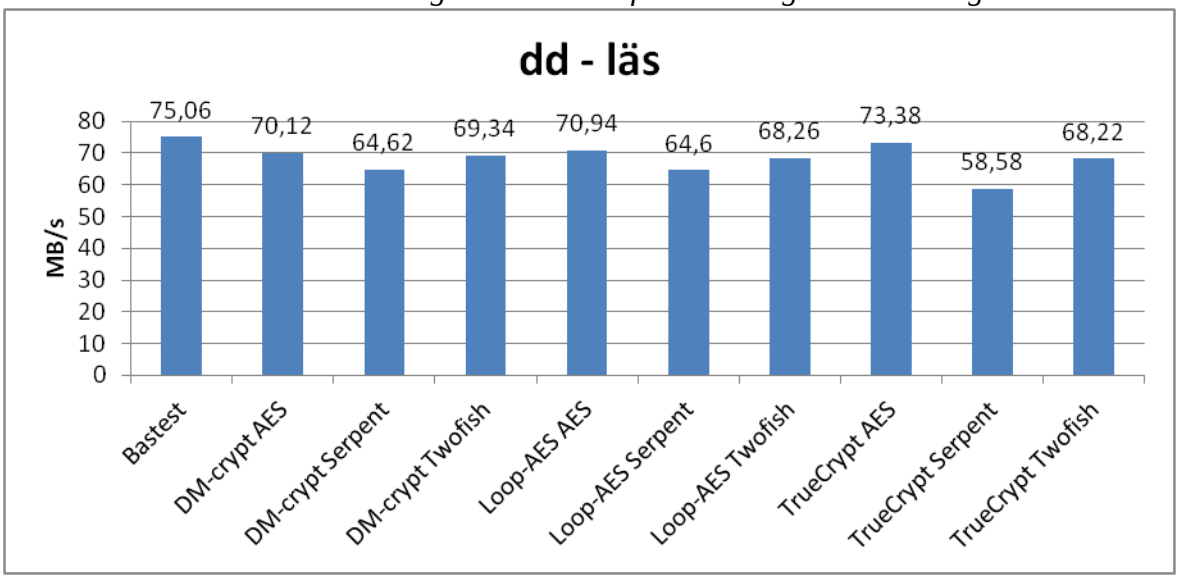
Figur 2 och 3 beskriver resultaten från testen av skriv- samt läshastigheten som uppmättes med dd. Den vertikala axeln beskriver hastigheten för skriv- respektive läshastigheten där ett högre värde är bättre.

För skrivhastighet presterar TrueCrypt och DM-crypt med Twofish eller AES bäst och de når cirka 96 procent av skrivhastigheten från bastestet. Med Loop-AES redovisas sämst resultat för skrivning oavsett algoritm.

Gällande läshastigheten har TrueCrypt med AES högst uppmätt resultat vilket motsvarar cirka 98 procent av resultatet från bastestet. Bland algoritmerna når AES och Twofish markant högre resultat än Serpent.



Figur 2: Resultat från mätning av skrivhastighet med dd.

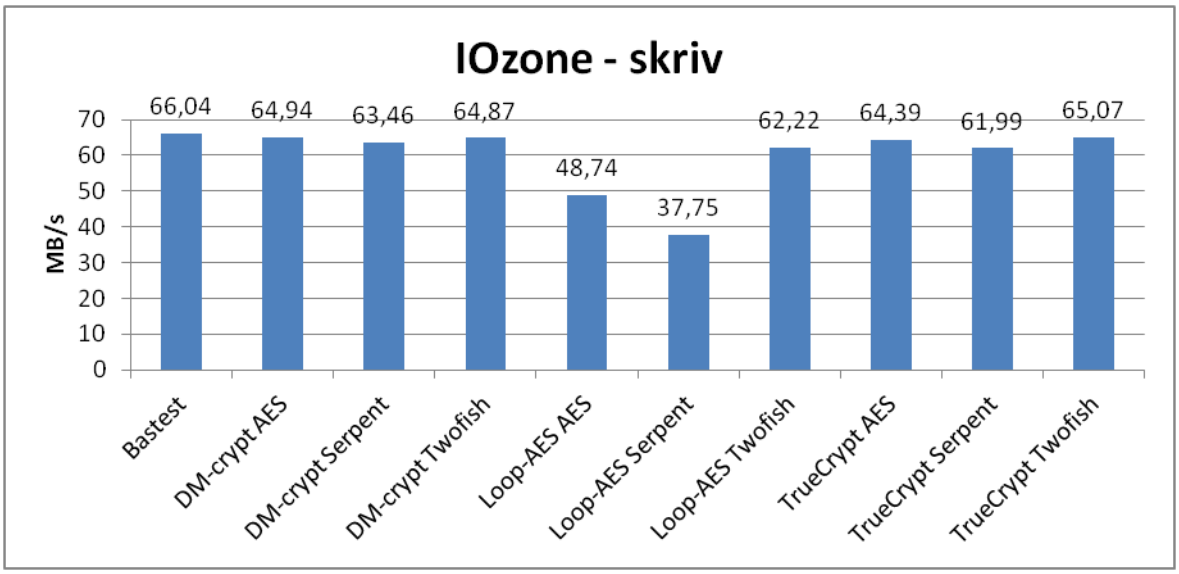


Figur 3: Resultat från mätning av läshastighet med dd.

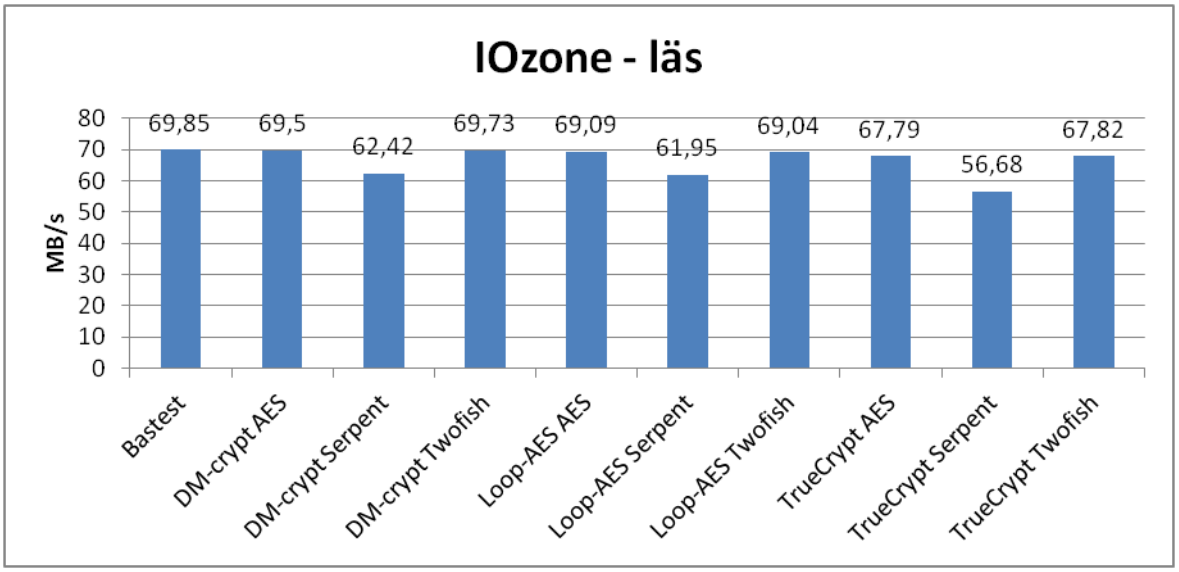
4.2 IOzone

I figur 4 och 5 presenteras resultat från mätningarna som utfördes med IOzone via Phoronix-test-suite. Värdena som presenteras är ett genomsnitt för de olika mätningarna som ingår i IOzone. Den vertikala axeln beskriver skriv- respektive läs-hastigheten där ett högre värde är bättre.

I mätningen av skrivhastighet presterar DM-crypt och TrueCrypt med AES eller Twofish bäst och deras resultat motsvarar cirka 97 procent av värdet från bastestet. Gällande läshastighet är det också jämnt mellan AES och Twofish och oavsett implementantion motsvarar resultatet cirka 98 procent av resultatet från bastestet.



Figur 4: Sammanfattning från mätning av skrivhastighet med IOzone.

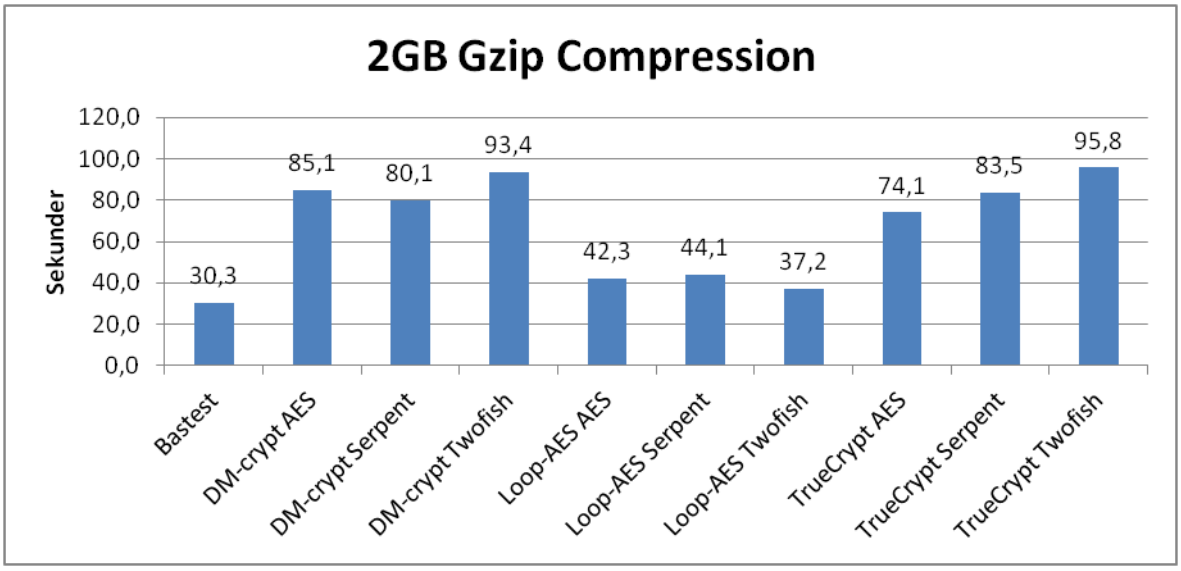


Figur 5: Sammanfattning från mätning av läshastighet med IOzone.

4.3 Gzip Compression

I figur 6 presenteras resultaten från mätningen som utfördes med testet 2GB Gzip Compression via Phoronix-test-suite. Den vertikala axeln beskriver tidsåtgång i sekunder där ett lägre värde är bättre.

Loop-AES presterar klart bättre än övriga implementationer i mätningen. Bland algoritmerna finns det ingen klar vinnare då de presterar olika bra inom implementationerna. Sammanfattningsvis presterar Loop-AES med Twofish bäst och genomför operationen på 81 procent av tiden det tog under bastestet.



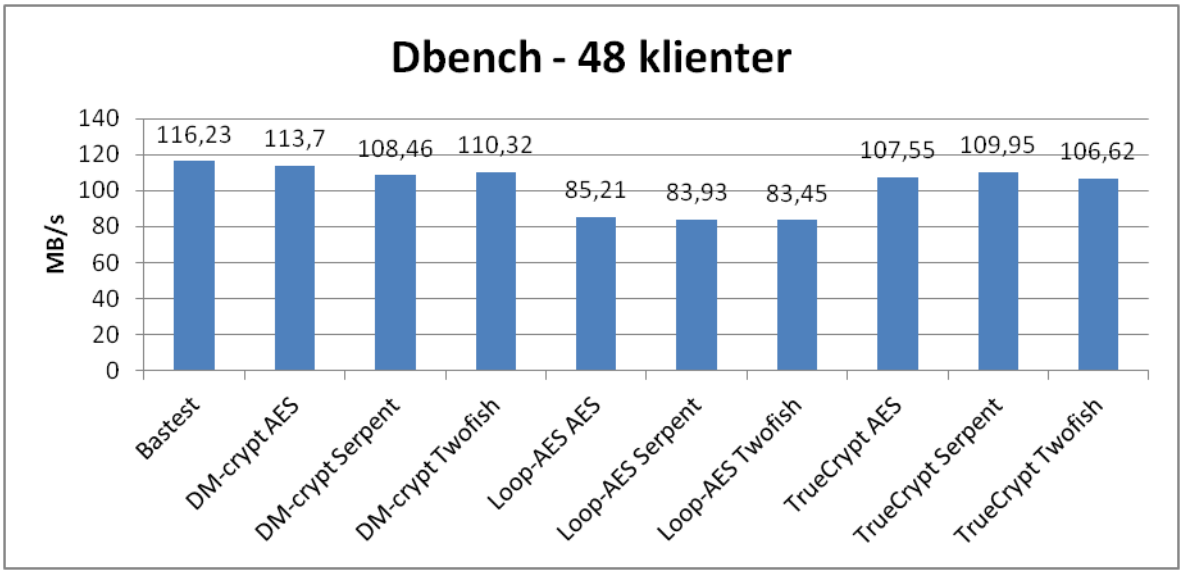
Figur 6: Tiden det tar att kryptera en 2GB stor fil med Gzip.

4.4 Dbench

I figur 7 presenteras resultatet från mätningen med Dbench. Den vertikala axeln beskriver genomströmningskapaciteten i MB/s för en virtuell filserver med 48 simultana klienter där ett högre värde är bättre.

DM-crypt och TrueCrypt presterar klart bättre än Loop-AES i Dbench. Mellan algoritmerna finns ingen klar segrare då de presterar olika bra inom implementationerna. Sammanfattningsvis uppnår DM-crypt med AES bäst resultat vilket motsvarar cirka 98 procent av resultatet från bastestet.

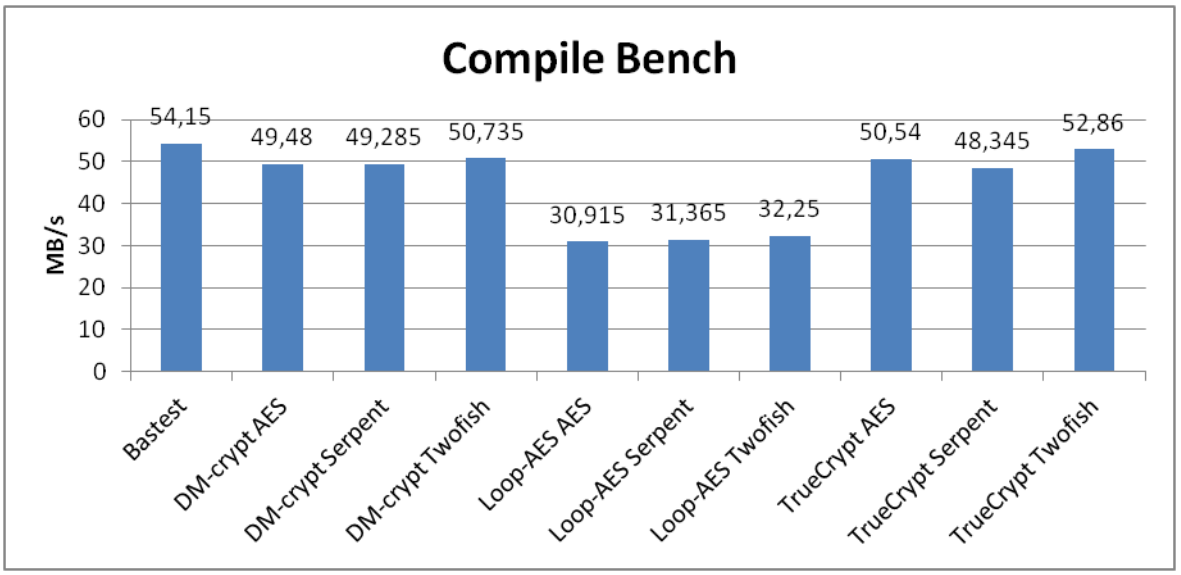
Mätvärdena för 1, 6, 12 och 128 klienter uppvisade liknande förhållanden mellan implementationerna och algoritmerna. Resultatet för dem återfinns i bilaga 2.



Figur 7: Resultat från Dbench.

4.5 Compile Bench

I figur 8 presenteras en sammanfattning av resultaten från *Initial Create* och *Compile* från Compile Bench. På den vertikala axeln presenteras medelvärdet av genomströmningskapaciteten för testen där ett högre värde är bättre. TrueCrypt presterar bäst och i samtliga implementationer ger algoritmen Twofish bäst resultat. Den uppmätta prestandan med TrueCrypt och Twofish motsvarar cirka 98 procent av resultatet för bastestet.



Figur 8: Sammanfattning av resultaten från Compile Bench.

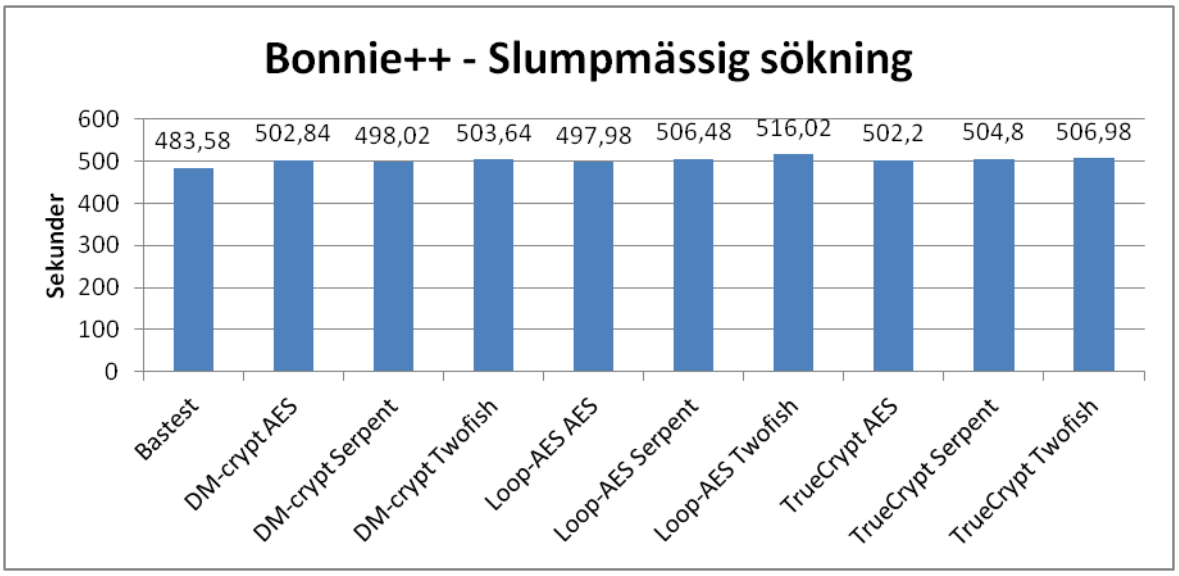
4.6 Bonnie++

I figur 9 beskrivs tiden det tog att slumpmässigt genomsöka en 4 GB stor fil. Den vertikala axeln beskriver tidsåtgången i sekunder där ett lägre värde är bättre. Loop-AES och DM-crypt med AES respektive Serpent presterar bäst och motsvarar cirka 97 procent av prestandan vid bastestet.

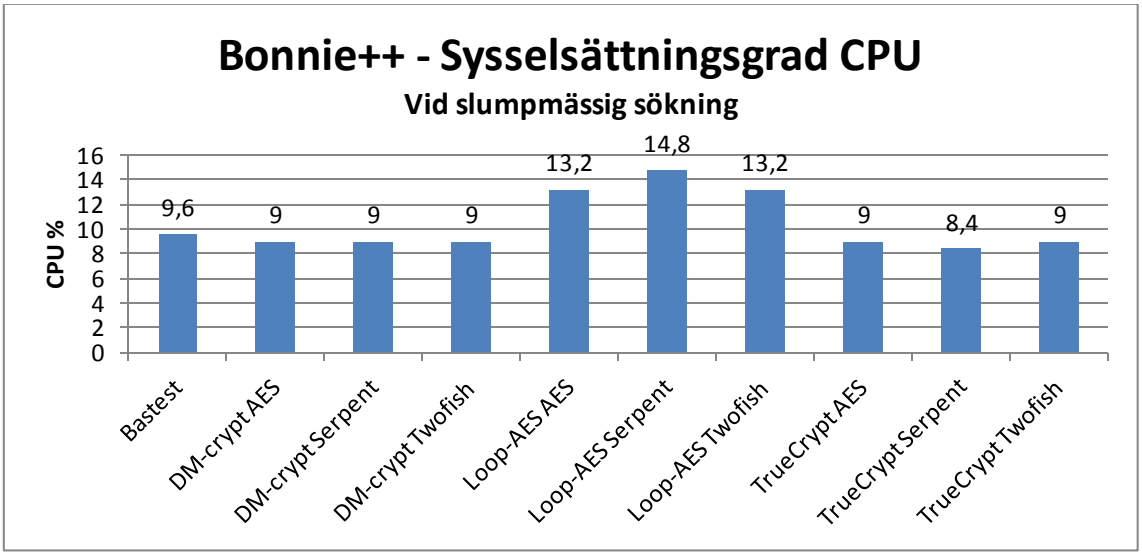
Figur 10 visar processorns sysselsättningsgrad vid slumpmässig sökning. Den vertikala axeln beskriver processorns belastning i procent. Samtliga krypteringsalgoritmer implementerade med hjälp av Loop-AES innebär högre belastning för processorn än vid kryptering med hjälp av TrueCrypt eller DM-crypt.

Figur 11 visar mätresultaten vid sekventiell läsning i block av en fil på storlek om 4 GB. Den vertikala axeln beskriver läshastigheten, där ett högre värde är bättre. TrueCrypts implementation av AES presterar bäst med en prestandaförsämring på drygt två procent sämre än bastestet. Marginalerna mellan TrueCrypt och Loop-AES implementation av AES, som är den sjätte bästa, är små och skillnaden är knappt sex procent.

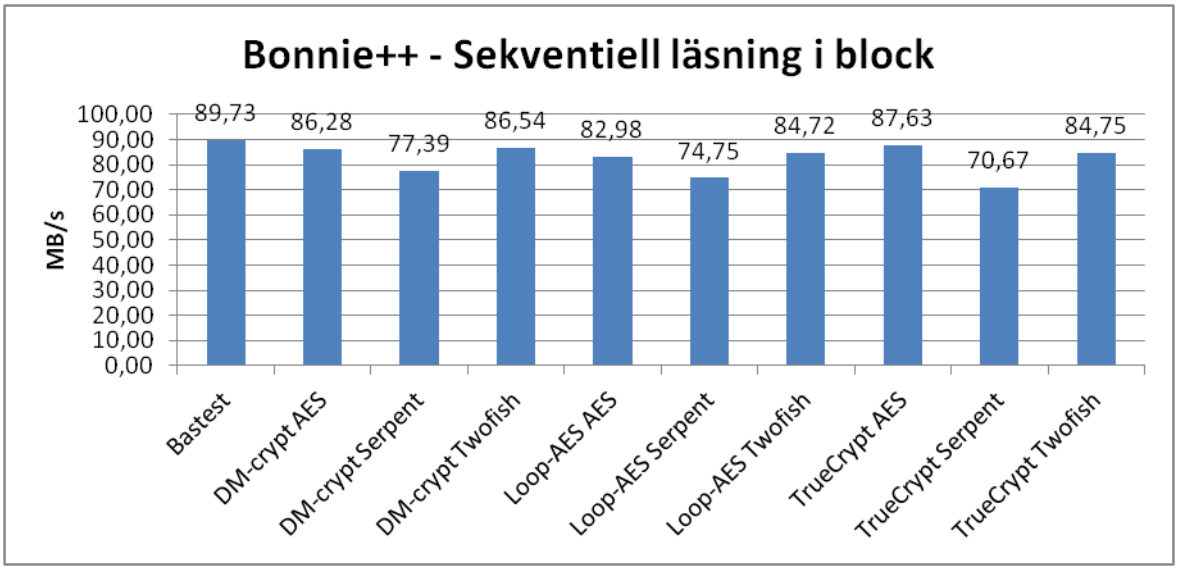
I figur 12 presenteras mätresultaten vid sekventiell skrivning i block till en 4 GB stor fil. Den vertikala axeln beskriver skrivhastigheten, där ett högre värde är bättre. Marginalerna mellan bastestet och den femte bäst presterande krypteringslösningen, Serpent implementerat i DM-crypt, är väldigt små och utgörs av en prestandaförsämring på endast tre procent.



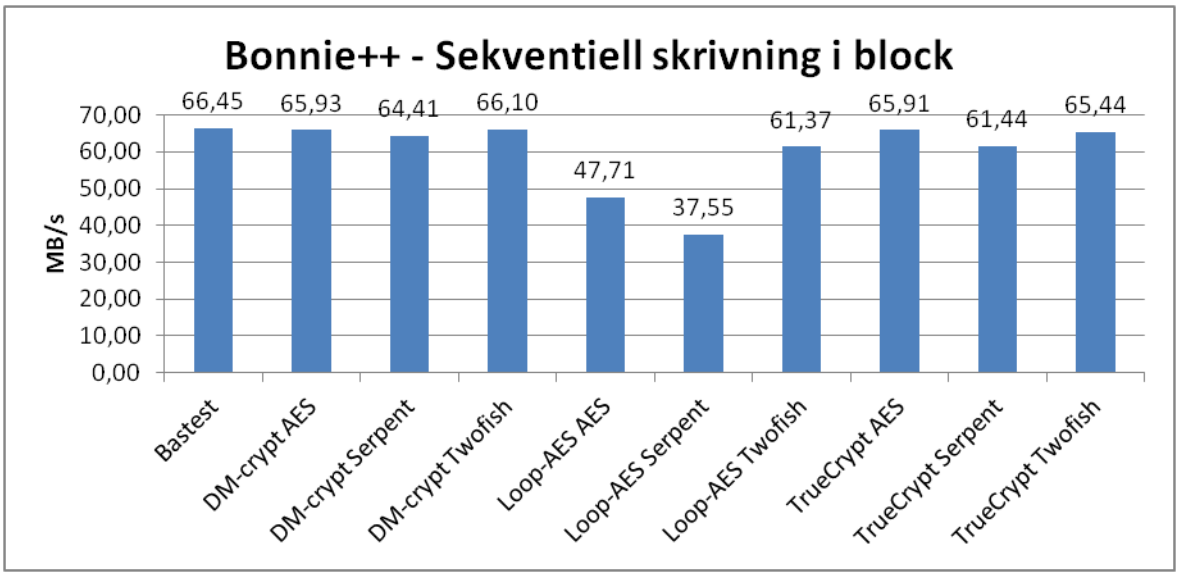
Figur 9: Slumpmässiga sökningar i Bonnie++.



Figur 10: Processorsysselsättningsgrad vid slumpmässiga sökningar i Bonnie++.



Figur 11: Sekventiell läsning i block i Bonnie++.



Figur 12: Sekventiell skrivning i block i Bonnie++.

5 Analys

I det här kapitlet jämförs och analyseras de resultat som presenterades i kapitel 4. Genom att jämföra och granska förhållandena mellan olika testresultat skapas optimala förutsättningar för att i kapitel 6 kunna dra korrekta slutsatser.

För många av mätningarna visade sig införandet av diskkryptering ha en liten effekt på diskprestandan jämfört med bastestet. För rena läs- och skriv-operationer, vilket mättes med till exempel dd, var prestandaskillnaden minst. Vid mer avancerade operationer som krävde större sysselsättning av processorn var skillnaden större såsom Gzip Compression.

De stora skillnaderna som noterades i mätningarna var mellan implementationerna DM-crypt och TrueCrypt gentemot Loop-AES. Resultaten för DM-crypt och TrueCrypt, oavsett algoritm, var vid flera mätningar markant annorlunda än de för Loop-AES, vilket framgår vid granskning av resultaten från Gzip Compression, Dbench, Compile Bench och processorsysselsättningsgraden uppmätt i Bonnie++.

5.1 Läs- och skrivresultat

Läshastighen, som är starkt knuten till dekrypteringshastigheten, var högst med AES och Twofish vilket framgår tydligast av läshastighetsmätningarna med dd, IOzone och Bonnie++. Mellan de olika implementationerna såg fördelningen av mätresultaten förhållandevis jämn ut.

Algoritmerna AES och Twofish uppnådde också högst skrivhastighet och således krypteringshastighet vilket är tydligt i skrivhastighetsmätningarna med dd, IOzone och Bonnie++. De testresultaten visar också att Loop-AES presterade betydligt sämre än DM-crypt och TrueCrypt.

För att på ett nyanserat vis kunna komma fram till vilken av krypteringslösningarna som generellt sett presterar bäst krävs det att hänsyn tas till standardavvikelsen. För läs- och skrivtester utförda med hjälp av dd, Bonnie++ och IOzone nås som mest en standardavvikelse på tre procent. En metod för att avgöra vilken krypteringsvariant som presterar bäst är att granska resultaten för respektive testprogram och utifrån en felprocent, motsvarande den maximala standardavvikelsen, sortera ut ett antal krypteringsvarianter som har presterat tillräckligt bra.

Genom att utifrån den bäst presterande krypteringslösningen för varje test räkna ut vilka övriga lösningar som innebär en prestandaförsämring på max tre procent, kan ett snitt dras för både läs- och skrivhastighet i tre olika prestandatester. Fördelningen bland de olika implementationerna och algoritmerna skulle då se ut som tabell 2 beskriver.

Test\Operation	Läs	Skriv
Bonnie++	DM-crypt AES, Twofish TrueCrypt AES	DM-crypt AES, Serpent, Twofish TrueCrypt AES, Twofish
IOzone	DM-crypt AES, Twofish Loop-AES AES, Twofish TrueCrypt AES	DM-crypt AES, Serpent, Twofish TrueCrypt AES, Twofish
Dd	TrueCrypt AES	DM-crypt AES, Twofish TrueCrypt AES, Twofish

Tabell 2: Fördelning av högpresterande krypteringslösningar utan inbördes ordning.

Vad beträffar skrivningar så placerar sig DM-crypts och TrueCrypts implementationer av AES och Twofish inom treprocentsgränsen för bästa mätvärde för respektive test. Gällande läsningar så är det endast TrueCrypts implementation av AES som i alla tre tester placerar sig inom treprocentsgränsen. Resultatet från läshastighetstestet i dd blir helt avgörande då inga andra krypteringslösningar placerar sig inom treprocentsgränsen i förhållanden till TrueCrypts implementation av AES.

Gemensamt för det slutgiltiga urvalet av krypteringslösningar, med hänsyn till både läs- och skrivhastighet, framgår det att TrueCrypts implementation av algoritmen AES är den enda krypteringslösning som lyckas placera sig inom treprocentsgränsen i samtliga fall.

5.2 Söktidsresultat

Mätningen för slumpmässiga sökningar med Bonnie++ visar att prestandaskillnaden mellan testets bästa och sämsta krypteringslösning, Loop-AES med AES respektive Twofish, utgörs endast av drygt fem procent. Prestandaförsämringen i resultaten från bastestet och Loop-AES med AES är mindre än tre procent.

Genom att jämföra sökhastigheten i hur många MB som genomsöks per sekund framhävs de små marginalerna ännu tydligare. Sökhastigheten vid slumpmässig sökning är mot okrypterad disk 8,9 MB/s. Mot krypterad disk, tillämpad med hjälp av Loop-AES implementation av AES, är hastigheten 8,6 MB/s.

Figur 10 som beskriver resultatfördelningen för slumpmässiga sökningar med Bonnie++ visar också att AES-algoritmen, oavsett implementation, presterar bra vid slumpartade sökningar.

5.3 Processorbelastning

Som figur 10 visar var processorsysselsättningsgraden högre för Loop-AES än för övriga implementationer oavsett algoritm. Detta är en möjlig förklaring till de dåliga resultaten för implementantionen i Compile Bench och Dbench men också övriga mätningar där Loop-AES uppmätte relativt dåliga resultat. Den ökade processorbelastningen för krypteringsoperationerna innebar att mindre processorcykler lämnades tillgängliga för I/O-operationer.

6 Slutsats

Som analysen i förgående kapitel visade på är det svårt att utpeka en tydlig vinnare bland implementationerna och algoritmerna. Generellt så visar mätresultaten att marginalerna är mindre vid skrivning än vid läsning. I många fall har förhållandet mellan flera mätresultat inom samma test inneburit en skillnad på mindre än tre procent. De tester som har genomförts med hjälp av dd har som mest en standardavvikelse på två procent, vilket innebär att samtidigt som resultaten med väldigt stor sannolikhet är korrekta så kan inga större slutsatser dras i de fall där mätvärdena inte skiljer sig med mer än två till tre procent.

Implementationerna DM-crypt och TrueCrypt presterade generellt bäst resultat, framförallt med algoritmerna AES och Twofish. Vissa mätningar hade dock avvikande resultat såsom Gzip Compression där Loop-AES presterade klart bättre än övriga implementationer. Det finns ingen uppenbar förklaring till varför och abnormaliteten kräver vidare forskning för att kunna dra en slutsats kring det.

En slutsats som är rimlig att dra är att implementationerna och algoritmerna har särskilda områden där de presterar bättre och är därmed olika bra för olika system. Loop-AES visade sig kräva en större belastning av processorn vilket resulterade i dåliga resultat för mer processorintensiva operationer.

Baserat på analysen som genomfördes i kapitel 5.1, vilket resulterade i tabell 2, kan TrueCrypts implementation av AES betraktas som den bäst presterande krypteringslösningen, med avseende på både läs- och skrivhastighet. Slutsatsen som dras är således att i generella datorsystem bör bäst resultat uppnås genom kryptering med AES implementerat i TrueCrypt.

6.1 Förslag till vidare forskning

Avgränsningarna i undersökningen ledde till att vissa aspekter i området lämnades outforskade. Det här kapitlet ämnar ge förslag på hur forskning i området kan fortlöpa.

6.1.1 Optimering av utvalda implementationer

En avgränsning som gjordes var att implementantionerna användes i sitt standardutförande och inga försök till prestandaoptimering gjordes. För flera av implementationerna som undersöktes finns det förslag på outforskade lösningar för prestandaoptimering. Dokumentationen[6] för Loop-AES innehåller till exempel ett kapitel angående optimering som kan konsulteras. För device-mapper, vilket inkluderar DM-crypt, finns det inofficiella patchar[7] tillgängliga som bland annat ger stöd för trådning och således bättre nyttjande av multipla processorkärnor. Vid genomförandet av den här undersökningen fanns ingen detaljerad utvärdering av prestandaoptimering via dessa metoder tillgänglig.

För vissa Intel-processorer finns det stöd för en uppsättning instruktioner för just kryptering och dekryptering med AES [41]. Instruktionsbiblioteket kallas AES-NI och samtliga krypteringsimplementationer som utvärderades i undersökningen har stöd för det [13][42]. Processorn som användes under utvärderingen hade dock inte stöd för

[6] http://loop-aes.sourceforge.net/loop-AES.README
[7] http://mbroz.fedorapeople.org/dm-crypt/3.6-rc/

AES-NI och utvärdering av hur instruktionsbiblioteket påverkade undersökningens resultat var således omöjlig.

6.1.2 Utvärdering av implementationerna och algoritmerna mot RAM-disk

I syfte att utesluta hårddiskens påverkan för undersökningen gjordes försök med att mäta prestandaförlusten på ett krypterat block monterat från RAM-minnet. Med utrustningen som fanns tillgänglig för undersökningen visade det sig dock vara svårgenomfört då mängden tillgängligt RAM-minne påverkade testresultaten. Med dd var det till exempel omöjligt att minimera standardavvikelsen med tanke på att samtliga algoritmer uppnådde läshastigheter på över 1 GB/s och det endast fanns 2 GB RAM-minne tillgängligt.

För vidare forskning inom området rekommenderas en större mängd primärminne. En undersökning av hur förekomsten av en RAM-disk påverkar cache-funktioner i Linuxkärnan bör också genomföras.

Referenser

[1] Xingliang Wang, Xiaochao Li, Mei Zou och Jun Zhou, "AES Finalists Implementation for GPU and Multi-core CPU based OpenCL", *Anti-Counterfeiting, Security and Identification (ASID), 2011 IEEE International Conference on*, Xiamen, 2011.

[2] National Institute of Standards and Technology (2010) FIPS General Information [elektronisk] http://www.nist.gov/itl/fipsinfo.cfm. Verifierad 2013-05-23

[3] Schneier (1998) The Twofish paper[elektronisk] Tillgänglig: http://www.schneier.com/paper-twofish-paper.pdf, s 4. Verifierad 2013-05-23

[4] National Institute of Standards and Technology (1999) NIST Announces Encryption Standard Finalists [elektronisk] Tillgänglig: http://csrc.nist.gov/archive/aes/round2/AESpressrelease-990809.pdf. Verifierad 2013-05-23

[5] National Institute of Standards and Technology (2000) Report on the Development of the Advanced Encryption Standard [elektronisk] Tillgänglig: http://csrc.nist.gov/archive/aes/round2/r2report.pdf, s 7. Verifierad 2013-05-23

[6] National Institute of Standards and Technology (2000) Commerce Department Announces Winner of Global Information Security Competition [elektronisk] Tillgänglig: http://www.nist.gov/public_affairs/releases/g00-176.cfm. Verifierad 2013-05-23

[7] National Institute of Standards and Technology (2000) Report on the Development of the Advanced Encryption Standard [elektronisk] Tillgänglig: http://csrc.nist.gov/archive/aes/round2/r2report.pdf, s 88. Verifierad 2013-05-23

[8] National Institute of Standards and Technology (2001) Announcing the Advanced Encryption Standard [elektronisk] Tillgänglig: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf, s 5. Verifierad 2013-05-23

[9] Daemen, J & Rijmen, V (2003) AES Proposal: Rijndael [elektronisk] Tillgänglig: http://csrc.nist.gov/archive/aes/rijndael/Rijndael-ammended.pdf. Verifierad 2013-05-23

[10] Schneier (1998) The Twofish paper[elektronisk] Tillgänglig: http://www.schneier.com/paper-twofish-paper.pdf, s 3. Verifierad 2013-05-23

[11] Serpent (okänt) Serpent homepage [elektronisk] Tillgänglig: http://www.cl.cam.ac.uk/~rja14/serpent.html. Verifierad 2013-05-23

[12] Anderson, R & Biham, E & Knudsen, L (okänt) A proposal for the Advanced Encryption Standard [elektronisk] Tillgänglig: http://www.cl.cam.ac.uk/~rja14/Papers/serpent.pdf. s 2. Verifierad 2013-05-23

[13] Jari Ruusu (2013) Loop-AES README [elektronisk] Tillgänglig: http://loop-aes.sourceforge.net/loop-AES.README. Verifierad 2013-05-23

[14] TrueCrypt Foundation (2013) Introduction [elektronisk] Tillgänglig: http://www.truecrypt.org/docs/intro.Verifierad 2013-05-23

[15] Broz, M (2012) dm-crypt: Linux kernel device-mapper crypto target [elektronisk] Tillgänglig: http://code.google.com/p/cryptsetup/wiki/DMCrypt. Verifierad 2013-05-23

[16] Cox, A (2009) LINUX ALLOCATED DEVICES (2.6+ version) [elektronisk] Tillgänglig: https://www.kernel.org/doc/Documentation/devices.txt. Verifierad 2013-05-23

[17] Saarinen, M-J O. "Encrypted Watermarks and Linux Laptop Security", "*Workshop of Information Security Applications"*, Jeju Island, Korea, 2004.

[18] Ruusu, J (2005-11-17) *Re: gonzo cryptography; how would you improve existing cryptosystems?* [elektronisk]. Tillgänglig: http://marc.info/?l=linux-crypto&m=117497737325177&w=2 . Verifierad 2013-05-23

[19] TrueCrypt Foundation (2013) Creating a New TrueCrypt Volume [elektronisk] Tillgänglig: http://www.truecrypt.org/docs/creating-new-volume. Verifierad 2013-05-23

[20] TrueCrypt Foundation (2013) Version History [elektronisk] Tillgänglig: http://www.truecrypt.org/docs/version-history. Verifierad 2013-05-23

[21] TrueCrypt Foundation (2013) Parallellization [elektronisk] Tillgänglig: http://www.truecrypt.org/docs/parallelization. Verifierad 2013-05-23

[22] TrueCrypt Foundation (2013) Encryption Algorithms [elektronisk] Tillgänglig: http://www.truecrypt.org/docs/encryption-algorithms. Verifierad 2013-05-23

[23] Red Hat, Inc (2013) Appendix A. The Device Mapper [elektronisk] Tillgänglig: https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Logical_Volume_Manager_Administration/device_mapper.html. Verifierad: 2013-05-23

[24] Penney, R.W. (2012) cryptmount - a utility for accessing encrypted filesystems [elektronisk] Tillgänglig: http://cryptmount.sourceforge.net. Verifierad: 2013-05-23

[25] Coker (2001) Bonnie++ - readme [elektronisk] Tillgänglig: http://www.coker.com.au/bonnie++/readme.html. Verifierad 2013-05-23

[26] The Open Group (2004) Stat [elektronisk] Tillgänglig: http://pubs.opengroup.org/onlinepubs/009696699/functions/stat.html. Verifierad 2013-05-23

[27] The Open Group (2004) Unlink [elektronisk] Tillgänglig: http://pubs.opengroup.org/onlinepubs/009695399/functions/unlink.html. Verifierad 2013-05-23

[28] The Open Group (2004) Creat [elektronisk] Tillgänglig: http://pubs.opengroup.org/onlinepubs/009695399/functions/creat.html. Verifierad 2013-05-23

[29] GNU (2013) Dd invocation [elektronisk] Tillgänglig: http://www.gnu.org/software/coreutils/manual/html_node/dd-invocation.html. Verifierad 2013-05-23

[30] Openbenchmarking (2010) Disk test suite [elektronisk] Tillgänglig: http://openbenchmarking.org/suite/pts/disk. Verifierad 2013-05-23

[31] Phoronix Test Suite (2010) Phoronix Test Suite Features [elektronisk] Tillgänglig: http://www.phoronix-test-suite.com/?k=features Verifierad 2013-05-23

[32] Phoronix Test Suite (2010) GZIP COMPRESSION 1.1.0 [elektronisk] Tillgänglig: http://openbenchmarking.org/innhold/94f36c1757ae5e0b5cf883edf2da2806797c63e8. Verifierad 2013-05-23

[33] Chris Mason (2008) Compilebench [elektronisk] Tillgänglig: https://oss.oracle.com/~mason/compilebench/. Verifierad 2013-05-23

[34] Phoronix Test Suite (2010) Compile bench 1.0.0 [elektronisk] Tillgänglig: http://openbenchmarking.org/innhold/f219b57eb61e0a28618878ee55ba6b2e5bc31c68. Verifierad 2013-05-23

[35] IOzone (2006) License [elektronisk] Tillgänglig: http://iozone.org/docs/Iozone_License.txt. Verifierad 2013-05-23

[36] IOzone (2006) Documentation [elektronisk] Tillgänglig: http://www.iozone.org/docs/IOzone_msword_98.pdf. Verifierad 2013-05-23

[37] Tridgell, A & Sahlberg, R (Okänt) DBENCH [elektronisk] Tillgänglig: http://dbench.samba.org/. Verifierad 2013-05-23

[38] Phoronix Test Suite (2010) DBENCH 1.0.0 [elektronisk] Tillgänglig: http://openbenchmarking.org/innhold/91394baa1217292d4bbba46ac8c921f3718325cc. Verifierad 2013-05-23

[39] National Institute of Standards and Technology (2012) About NIST [elektronisk] Tillgänglig: http://www.nist.gov/public_affairs/nandyou.cfm. Verifierad 2013-05-23

[40] Distrowatch (2013) Page hit ranking [elektronisk] Tillgänglig: http://distrowatch.com/index.php?dataspan=52. Verifierad 2013-05-23

[41] Rott, J (2012) Intel® Advanced Encryption Standard Instructions (AES-NI) [elektronisk] Tillgänglig: http://software.intel.com/en-us/articles/intel-advanced-encryption-standard-instructions-aes-ni. Verifierad 2013-05-23

[42] Basu, A (2012) Intel® AES-NI Performance Testing over Full Disk Encryption [elektronisk] Tillgänglig: www.intel.com/content/dam/www/public/us/en/documents/white-papers/healthcare-aes-ni-full-disk-encryption-paper.pdf. Verifierad 2013-05-23

[43] Olsson, R (2012) Performance differences in encryption software versus storage devices [elektronisk] Tillgänglig: http://lnu.diva-portal.org/smash/record.jsf?pid=diva2:535985. Verifierad 2013-06-07

Bilagor

Nedan prestenteras de bilagor undersökningen refererar till.

Bilaga A Processer

```
PID TTY      STAT   TIME COMMAND
  1 ?        Ss     0:00 /sbin/init
  2 ?        S      0:00 [kthreadd]
  3 ?        S      0:00 [ksoftirqd/0]
  5 ?        S      0:00 [kworker/u:0]
  6 ?        S      0:00 [migration/0]
  7 ?        S      0:00 [watchdog/0]
  8 ?        S      0:00 [migration/1]
 10 ?        S      0:00 [ksoftirqd/1]
 11 ?        S      0:02 [kworker/0:1]
 12 ?        S      0:00 [watchdog/1]
 13 ?        S<     0:00 [cpuset]
 14 ?        S<     0:00 [khelper]
 15 ?        S      0:00 [kdevtmpfs]
 16 ?        S<     0:00 [netns]
 18 ?        S      0:00 [sync_supers]
 19 ?        S      0:00 [bdi-default]
 20 ?        S<     0:00 [kintegrityd]
 21 ?        S<     0:00 [kblockd]
 22 ?        S<     0:00 [ata_sff]
 23 ?        S      0:00 [khubd]
 24 ?        S<     0:00 [md]
 26 ?        S      0:00 [khungtaskd]
 27 ?        S      0:00 [kswapd0]
 28 ?        SN     0:00 [ksmd]
 29 ?        SN     0:00 [khugepaged]
 30 ?        S      0:00 [fsnotify_mark]
 31 ?        S      0:00 [ecryptfs-kthrea]
 32 ?        S<     0:00 [crypto]
 40 ?        S<     0:00 [kthrotld]
 59 ?        S<     0:00 [devfreq_wq]
181 ?        S<     0:00 [mpt_poll_0]
182 ?        S<     0:00 [mpt/0]
189 ?        S      0:00 [scsi_eh_0]
204 ?        S      0:00 [jbd2/sda1-8]
205 ?        S<     0:00 [ext4-dio-unwrit]
417 ?        S<     0:00 [edac-poller]
467 ?        S<     0:00 [ttm_swap]
477 ?        Ss     0:00 /usr/sbin/sshd -D
613 tty5     Ss+    0:00 /sbin/getty -8 38400 tty5
623 tty2     Ss+    0:00 /sbin/getty -8 38400 tty2
624 tty3     Ss+    0:00 /sbin/getty -8 38400 tty3
626 tty6     Ss+    0:00 /sbin/getty -8 38400 tty6
3200 ?       S      0:00 [flush-8:0]
3262 ?       S      0:00 [kworker/1:2]
3287 ?       S      0:00 [scsi_eh_2]
```

```
3288 ?        S      0:00 [usb-storage]
3297 ?        S      0:00 [kworker/u:2]
3805 ?        S      0:00 [kworker/1:0]
4268 ?        S      0:00 [kworker/0:0]
5416 ?        S      0:00 [jbd2/sda2-8]
5417 ?        S<     0:00 [ext4-dio-unwrit]
6137 tty4     Ss+    0:00 /sbin/getty -8 38400 tty4
6154 tty1     Ss+    0:00 /sbin/getty -8 38400 tty1
6174 ?        S      0:00 upstart-socket-bridge --daemon
```

Bilaga B Dbench

kryptering\klienter	1	12	48	128
Okrypterat 20,58	85,21	116,23	89,28	
DM-crypt AES	20,38	82,72	113,7	77,11
DM-crypt Twofish	20,45	80,31	110,32	76,59
TrueCrypt Serpent	21,42	81,5	109,95	71,44
DM-crypt Serpent	20,31	81,83	108,46	73,59
TrueCrypt AES	19,31	79,22	107,55	75,93
TrueCrypt Twofish	19,3	76,41	106,62	75,18
Loop-AES AES	19,66	71,03	85,21	48,38
Loop-AES Serpent	20,19	69,64	83,93	47,89
Loop-AES Twofish	20,27	71,64	83,45	47,36