VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

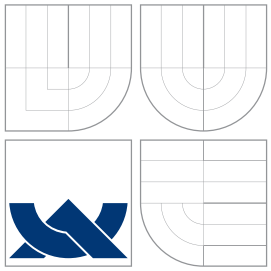
ŠIFROVANÉ SOUBOROVÉ SYSTÉMY

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

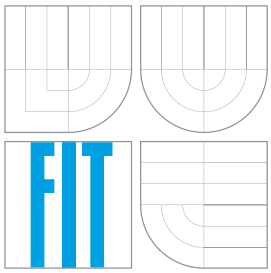
AUTOR PRÁCE
AUTHOR

Bc. MICHAL HLAVINKA

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

ŠIFROVANÉ SOUBOROVÉ SYSTÉMY

ENCRYPTED FILESYSTEMS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MICHAL HLAVINKA

VEDOUCÍ PRÁCE
SUPERVISORS

doc. Dr. Ing. PETR HANÁČEK
PETER VRABEC

BRNO 2008

Abstrakt

Tato práce se věnuje problematice šifrovaných souborových systémů a je zaměřena převážně na řešení pro operační systém Linux. Nejprve uvádí do problematiky zdůvodněním, proč je nutné šifrovat. Dále pak popisuje principy šifrování souborů jako je šifrování celého disku, oddílu, kontejneru a souborů. Uvedeny jsou jejich jednotlivé výhody a nevýhody. Po úvodu jsou zmíněna bezpečnostní rizika, která mohou útočníkovi pomoci získat data nebo je pozměnit. Zpráva dále obsahuje stručný popis nejznámějších šifrovaných souborových systémů. U možností pro OS Linux je uveden způsob jejich použití a porovnání jejich rychlosti. V další části práce je nejprve uvedeno, jakým způsobem funguje šifrování dm-crypt s LUKS. Poté je pro tuto metodu popsán nově vytvořený nástroj od specifikace, přes implementaci až po testování. V závěru je uvedeno shrnutí informací, uvedeno přínos této práce a jakým způsobem by bylo možné v této práci pokračovat.

Klíčová slova

eCryptFS, EncFS, dm-crypt, LUKS, šifrované souborové systémy, QCryptool

Abstract

This thesis is about encrypted filesystems and is aimed mainly for Linux solutions. At first there is explained why is it necessary to encrypt your data at all. Next is written about pros and cons of encrypting methods such as whole disk encryption, partition encryption, container encryption and file encryption. After the introduction there are mentioned most common security issues which can help attacker to decrypt your data or modify them. Furthermore this thesis contains short description about the most often used encrypted filesystems. There is also small how-to for the most important encrypted filesystems available in Linux and compaaration of their speed. Next part of this thesis contains dm-crypt and LUKS description. In the last chapter all information are concluded. There is also mentioned benefit of this work and possibilities, what can be done in future.

Keywords

eCryptFS, EncFS, dm-crypt, LUKS, encrypted filesystems, QCryptool

Citace

Michal Hlavinka: Šifrované souborové systémy, diplomová práce, Brno, FIT VUT v Brně, 2008

Šifrované souborové systémy

Prohlášení

Prohlašuji, že jsem tuto semestrální práci vypracoval samostatně pod vedením pana doc. Dr. Ing. Petra Hanáčka a Petera Vrabce.

.....
Michal Hlavinka
19. května 2008

Obsah

Kapitola 1

Úvod

Každá práce by měla mít nějaký důvod, nějakou motivaci, proč se problémem zabývat. Nemá cenu znovu vynalézat kolo, ani vytvářet něco, co akorát skončí zapomenuto v archivu. Tato práce důvod svého vzniku má. Šifrování souborů dnešní počítače nijak znatelně nezpomaluje, přesto jej málokdo používá. Nejčastějšími důvody jsou názory, že je to příliš složité a není to potřeba. Ani jeden z těchto názorů však není zcela pravdivý.

V současné době je běžné, že software je dražší než hardware. Ještě cennější než nainstalované programy jsou však obchodní informace o vývoji nějakého zařízení či softwaru a nejedna konkurenční firma by za taková data i zaplatila nemalou částku. Krádeže dat se však netýkají jen velkých firem, ale týkají se i jednotlivců. Prvotním cílem zloděje může být jen krádež samotného notebooku, externího disku či flash disku. Zvědavost zlodějovi nedá a podívá se, co tam máte uloženého pěkného. Určitě byste pak nechtěli, aby třeba kolegové v práci našli někde na webu vystavené vaše soukromé pikantnější fotky z minulé dovolené a vy pak museli chodit postranními uličkami, aby si na vás někdo neukazoval.

Nejedná se ani o žádné výjimečné případy. Podle různých zdrojů je ve výsledku ukradeno až deset procent ze všech prodaných notebooků [11].

Podle analýzy provedené firmou IBM bylo v roce 2005 provedeno přes 200 milionů útoků týkajících se oblasti software a dat, z nichž byla část zaměřena na krádež důležitých dat či identit. To mělo útočníkům vydělat vymáháním peněz nebo prodáním ukradených dat. Meziroční nárůst útoků dosahuje zhruba 50 % [3].

Krádež dat se netýká tedy jen několika málo lidí, kteří měli jen smůlu, ale je reálným nebezpečím pro jakoukoliv firmu a čímkoli soukromí. Data je tedy nezbytné chránit.

Tato práce se nezabývá tím, jak zabránit zloději ukrást notebook (rady v tomto směru se dozvíte jinde na internetu [4]), ale tím, jak minimalizovat škody, které krádež způsobí.

Účinným způsobem, jak útočníkovi ve zneužití dat zabránit, je jejich šifrování. Pokud nechcete data šifrovat po jednom souboru v nějakém programu, je nejlepší volbou použití šifrovaného souborového systému.

Právě tématem šifrovaných systémů se bude zabývat tato práce. Často si pod slovem „šifrování“ lidé představí něco ohromně složitého, co se nepodaří zprovoznit ani po několika hodinách usilovné práce. V případě chyby lze o data přijít a výsledek je spíše ke vzteku. Po přečtení této práce ale asi zjistíte, že to tak složité skutečně není, a aby to bylo ještě jednodušší, bude posléze vytvořen i snadno použitelný nástroj, který celou práci ještě více usnadní.

V následující kapitole se dozvíte, co jsou to šifrované systémy, jak fungují, jaké jsou mezi nimi základní rozdíly v jejich konceptu a nastínění dalších možností zabezpečení dat.

Jakým způsobem se při šifrování zpracovávají sektory disku a proč se vůbec nějak zvlášť

zpracovávají je popsáno ve třetí kapitole.

Čtvrtá kapitola se věnuje bezpečnostním problémům u šifrovaných souborových systémů. Popisuje některé typy možných útoků a jaké okolnosti mohou útočníkovi usnadnit (nebo dokonce přímo umožnit) získání či úpravu cenných dat.

V páté kapitole dojde k porovnání různých typů šifrovaných souborových systémů. Stručně o jejich vlastnostech, schopnostech, problémech a vývoji.

Šestá kapitola popisuje jak si lze šifrované souborové systémy vyzkoušet, uvádí tedy způsob instalace vybraných šifrovaných souborových systémů, konkrétně dm-crypt/LUKS, eCryptFS a EncFS.

Různé šifrované souborové systémy jsou realizovány různými způsoby, což má vliv na jejich výkon. Měření rychlosti práce s daty je popsáno v sedmé kapitole.

V osmé kapitole je provedeno shrnutí doposavad zjištěných informací a vybrán šifrovaný souborový systém, kterému je později věnována hlavní pozornost.

K vybranému šifrovanému souborovému systému je v deváté kapitole uvedeno několik dalších informací s podrobnějším popisem, jak funguje.

Pro zjednodušení práce s vybraným šifrovaným souborovým systémem je vytvořen uživatelský nástroj. Jeho návrhu, implementaci a výjimečnostem je věnována desátá kapitola.

V závěrečné kapitole je shrnutí zjištěných informací, stručně popsán přínos vytvořeného nástroje a uvedeny náměty, kterými by bylo možné dále v tomto tématu pokračovat.

Kapitola 2

Popis šifrovaných souborových systémů

Nejprve je důležité zmínit, co je to souborový systém. Souborový systém je systém uspořádání dat na disku tak, aby byla data snadno nalezitelná, dalo se k nim snadno přistupovat a dala se snadno upravovat. Podrobnější vysvětlení lze nalézt například na wikipedii [15]. Kromě samotných dat jsou ukládána ještě další metadata potřebná pro práci souborového systému. Tato metadata (např. datum vytvoření) můžeme také chtít mít utajena, ale v některých šifrovaných souborových systémech šifrována nejsou.

2.1 Ruční šifrování

Nejjednodušším způsobem šifrování souborů je ruční šifrování. Jde o šifrování souborů po jednom pomocí nějakého programu. Před každým použitím souboru je nutné spustit program, zadat klíč a soubor dešifrovat. Program vytvoří kopii souboru v dešifrované podobě nebo soubor přepíše dešifrovanými daty. S takovým souborem pak lze pracovat běžným způsobem. Pokud byl šifrovací program mezitím ukončen, je nutné jej opět spustit, znovu zadat klíč a teprve pak lze soubor zašifrovat.

Hlavní výhodou ručně šifrovaných souborů je jejich snadná přenosnost a ve většině případů se dají využívat i s omezenými oprávněními uživatele. Soubor, ať už jej zkopírujete kamkoliv, zůstává stále zašifrovaný. V naprosté většině ostatních způsobů šifrování bude, při kopírování na jiné médium, soubor uložen v dešifrované podobě.

Nevýhodou však je, že tento způsob uživatele poměrně zdržuje. Dále se může stát, že některá data zapomenete po použití zpátky zašifrovat nebo kvůli plánovanému brzkému použití je „zatím“ nezašifrujete.

Je-li také soubor na disku uložen v dešifrované podobě, není zaručeno, že při přepsání souboru zašifrovanými daty budou vždy tato data přepsána i fyzicky na disku. Data mohou být uložena do jiného bloku a původní blok označen jako prázdný, přesto s původními dešifrovanými daty.

Poslední nevýhodou je nutnost dalším prostředkem šifrovat i odkládací paměť a úložiště dočasných souborů, jinak samotné šifrování nemá příliš velký smysl.

2.2 Šifrované soubory

Souborové systémy mohou obsahovat zašifrované soubory. Nejjednodušeji způsobem, že máte na disku uložen soubor, který jste před tím sami nějakým způsobem zašifrovali. Právě proto, abyste nemuseli toto provádět ručně, existují šifrované souborové systémy, které převezmou tuto práci za vás. Šifrování je pak realizováno transparentně. To znamená, že pokud je zadán klíč a/nebo heslo, jsou soubory přístupné pro běžnou práci běžným způsobem a nikoho nic neobtěžuje a nezdržuje ručním (de)šifrováním.

Systém při práci na šifrovaném souborovém systému po obdržení požadavku o přečtení souboru po přečtení dat z disku data nejprve dešifruje a poté teprve vrátí již dešifrovaná data. Naopak při zapisování souboru jsou data nejprve zašifrována a teprve poté zapsána na disk. Data na disku jsou tedy pouze v zašifrované podobě a při jejich přímém přečtení získáte pouze hromadu „náhodných“ bytů, která vám bude k ničemu. Stejně jako útočník, tak i vy budete ale bez šance na získání dat, pokud heslo zapomenete nebo případně ztratíte šifrovací klíč. Jde tedy o souborový systém, který obsahuje zašifrované soubory (může obsahovat i nezašifrované soubory). Podle typu může být bez zadání hesla nepřístupný celý souborový systém nebo jen zašifrovaná data.

Tento typ šifrovaných souborových systémů obvykle dokáže různé soubory šifrovat různými klíči, takže může mít na stejném oddílu zabezpečená data více uživatelů a každý svým vlastním klíčem. To je určitě bezpečnější, než se spoléhat jen na zabezpečení systému, že jiný uživatel nebude moci číst má data. Nebo případně útočník, který získá klíč jiného uživatele.

Nevýhodou však je, že k šifrování odkládací paměti a úložiště dočasných souborů je nutné použít další nástroj.

2.3 Šifrovaný oddíl

Výše uvedený způsob šifrování je jen jednou z možností. Kromě toho, že mohou být šifrovány jednotlivé soubory, tak může být zašifrován celý oddíl obsahující souborový systém. Pak by bylo možná místo šifrované souborové systémy vhodnější označení zašifrované souborové systémy. Pro větší jednoduchost budou tyto dvě varianty v textu přímo rozlišovány pouze tam, kde to bude nezbytné.

Při tomto přístupu jsou zašifrována i veškerá metadata naprosto automaticky. Nelze tedy zjistit jména souborů, adresářovou strukturu, časy souborů (vytvoření, modifikace, čtení) a ani poměr zabraného místa. Zároveň je možné do šifrovaného oddílu umístit prakticky cokoliv, jedno jestli swap nebo FAT32, NTFS, ext3 atd.. Nevýhodou tohoto přístupu je, že pokud má být oddíl přístupný pro více uživatelů, tak přestože uživatelé mohou mít různé klíče, bude vždy dešifrován obsah celého disku a útočníkovi stačí jakýkoliv z klíčů.

Data jsou pak dostupná stylem buď všechno nebo nic (samozřejmě se uplatní oprávnění pro jednotlivé soubory). Při šifrování oddílu je možné využít další možnost, kterou je vytvoření souboru s obrazem oddílu (tzv. kontejneru), který bude obsahovat obraz celého zašifrovaného oddílu. Po připojení se v systému objeví nový disk, například jako když připojíte flash disk. Výhodou je snadná přenosnost, nevýhodou pak to, že tento kontejner má pevnou velikost a jeho zvětšení bývá problematické nebo přímo nemožné.

Opět je nutné šifrovat i odkládací paměť a úložiště dočasných souborů.

2.4 Šifrování celého disku

Ještě o stupínek výše položenou metodou je možnost šifrování celého disku, kdy jsou pak automaticky zašifrovány všechny oddíly (včetně swapu) a není známa ani jejich velikost. Některé notebooky umožňují provádět šifrování celého disku. Problém je, že tato data obvykle nejsou (nebo jen velmi obtížně) přenosná mimo notebook. Jinak pro tuto variantu platí z hlediska použitelnosti to samé jako pro šifrování oddílů.

2.5 Steganografie

Šifrování souborů však někdy nestačí. Může se stát už jen to, že někdo na vás bude dotírat s tím, že máte na disku zašifrovaná data a co tam schováváte. Vyrukuje třeba i s dost pochybným tvrzením, že kdyby bylo vaše svědomí čisté, nemuseli byste nic skrývat. Někdy se uvádí i (spíše teoretický) příklad, že někdo z vás může chtít získat klíč k zašifrovaným datům pod pohrůžkou násilí.

O soukromí však můžete přijít například i u soudu. K zamyšlení a pozornosti je zajímavý případ [14], kdy ve Velké Británii hrozí presumpce viny v tom směru, že pokud máte na disku zašifrovaná data, musíte k ním vydat dešifrovací klíč, jinak můžete být odsouzeni až na dva roky. Příkladem může být třeba, že máte na disku uložena třeba videa ze svatby zahrnující i vaši svatební noc. Těžko si vsadit, jestli by se z tohoto záznamu nestalo podpultové zboží mezi kolegy z policie, které by se časem dostalo i na internet...

Je tedy vidět, že někdy se může stát, že jen samotné šifrování dat nestačí. Výhodnější může být (zašifrovaná) data raději skrýt tak, že je nikdo nenajde. Tímto se zabývá steganografie [16]. Stručně jde o ukrytí dat tak, že je nelze najít ani prokázat, jestli někde uložena jsou či nikoliv.

Nejjednodušší používanou steganografií je skrývání krátké textové zprávy v obrázku. Vezmete text, který zašifrujete. Poté vezmete fotku z vaší dovolené a na ní změníte nepatrně hodnoty některých pixelů na nejméně významném bitu a zprávu tak bit po bitu uložíte do obrázku. Výsledný obrázek pak vypadá stále normálně, přitom ale obsahuje ukrytou zprávu, kterou bez dešifrovacího klíče nezískáte. Nelze ani zjistit, jestli v obrázku nějaká taková zpráva je, či nikoliv.

Některé šifrované souborové systémy dokáží vypadat jako volné nevyužité místo na disku, kde nebyl dosud vytvořen žádný oddíl. Jedná se víceméně o ty, které jsou zašifrovány komplet včetně metadat a není zapsána žádná informace ani v tabulce rozdělení disku. V tomto případě pak není možné bez správného klíče prokázat, jestli se jedná o náhodná data nebo právě o zašifrovaný oddíl. Jednou z metod je uložení dat na oddíl, který je veden jako swap (systém musí být nastaven, aby tento swap nepoužíval, jinak dojde ke ztrátě dat).

2.6 Shrnutí

Každá z metod má některé výhody a některé nevýhody. Záleží tedy dost na tom, za jakým účelem budou data šifrována. Pokud se jedná o připojené vzdálené úložiště, je nutné použít soubor s obrazem oddílu nebo data alespoň šifrovat pomocí virtuálního šifrovaného souborového systému. Dobře použitelným je v tomto případě například EncFS, který může být spravován pro osobní potřeby uživatele bez nutnosti mít administrátorské oprávnění. Je však nutné, aby bylo zašifrováno i úložiště dočasných souborů a odkládací prostor, což obvykle povede k použití další šifrovací metody.

Naopak šifrování celého disku nebo jednotlivých oddílů má výhodu v tom, že dokáže šifrovat i zmíněná místa, kde se data mohou při práci objevit, nebo je alespoň možné tato místa (oddíly) šifrovat stejným způsobem. Nejvýhodnější použití této metody je na soukromém počítači nebo notebooku, kdy není třeba, aby bylo schopno data dešifrovat více uživatelů. Tento způsob má však také několik nevýhod. Není například možné šifrovat různá data různými algoritmy nebo klíči. Zálohovací programy musí mít minimálně pro provádění inkrementálních záloh k dispozici dešifrovaná data a ty je tedy nutno dodatečně nějakým způsobem šifrovat. Při posílání souborů přes síť nebo při ukládání na jiné médium jsou data přenášena (resp. ukládána) dešifrovaná. Poslední nevýhodou je, že dochází k šifrování i těch dat, která šifrovat nutné není, což systém zpomaluje.

Nelze říct, který přístup je nejvhodnější a vybrat jen jeden. Každý způsob je vhodnější v jiných situacích a je dobré, že je z čeho si vybírat.

Kapitola 3

Metody šifrování bloků dat

Data jsou na disku ukládána v sektorech, které mají obvykle velikost 512 bajtů. Kvůli dostatečné výkonnosti musí být možné tyto sektory šifrovat a dešifrovat nezávisle na ostatních. Proto je na šifrování kladen požadavek, aby bylo možné dešifrovat jen jeden sektor, bez nutnosti zpracovávat ostatní data.

Šifrovací algoritmy lze rozdělit na proudové šifry a blokové šifry. Proudové šifry zpracovávají data po bitech (případně po bajtech) za sebou. Na velikosti šifrovaných dat nezáleží. Problém ale je, pokud je zapotřebí dešifrovat data, která se nachází někde uprostřed. Tento typ šifrování se pro šifrování souborových systémů nepoužívá. Mezi zástupce proudových šifer patří například Vernamova šifra nebo RC4.

Obvykle se pro šifrování dat na disku používají blokové šifry, které zpracovávají data po blocích (např. 8 nebo 16 bajtů). Sektory disku jsou velké 512 bajtů, což je bez problémů dělitelné velikostí bloku, takže není nutné provádět zarovnávání dat (blokové šifry umí zašifrovat vždy pouze blok dat dané velikosti). Zpracovávaná data sektoru disku se tedy rozdělí do bloků a tyto bloky se šifrují. Jakým způsobem se zpracovávají bloky uvnitř sektoru a jak jsou na sobě případně závislé, udává použitá metoda šifrování bloků dat. Těchto metod bylo postupem času vytvořeno několik. Obyčejně ale platí, že jednodušší metoda je méně bezpečná.

3.1 Electronic Codebook

Metoda ECB (Electronic Codebook) je nejjednodušší metodou. Každý blok dat je šifrován pouze samotným klíčem, který je pro všechny bloky stejný. Kvůli své jednoduchosti (nezpracovává bloky vlastně nijak) není doporučena pro šifrování dat.

Je nutné si uvědomit, že šifrování je ve své podstatě zpracování dat pomocí funkce, která pro zadaná data vrátí jejich zašifrovanou podobu. Tato zašifrovaná podoba dat je ale pro stejná data vždy stejná. Šlo by tedy vytvořit si (alespoň část) převodní tabulky podobně jako je tomu u monoalfabetických substitučních šifer. Je-li zašifrován oddíl, tak na něm umístěný souborový systém má na svém začátku hlavičku, která je z větší části vždy stejná. Známe tedy, jak se určitá data zašifrují. Najdeme-li někde jinde blok, který zašifrovaný vypadá stejně, víme, že byla šifrována stejná data. Toto je ale příliš velký a poměrně snadno zneužitelný bezpečnostní problém, kterého využívá například tzv. watermark attack.

3.2 Cipher-Block Chaining

Bezpečnější metodou je CBC (Cipher-Block Chaining). Při zašifrovávání je každý blok dat nejprve xorován s předchozím zašifrovaným blokem. Při dešifrování jsou data xorována po provedení dešifrování. Každý blok tedy závisí na všech předchůdcích. První blok nemá předchůdce, proto je použit inicializační vektor (blok náhodných dat). Změna jednoho bloku vede k potřebě přešifrovat všechny následující bloky.

Každý úsek dat je šifrován s touto metodou a každý má vlastní inicializační vektor. Původně byl odvozen ze známých dat (např. pořadové číslo). Později byl nahrazen bezpečnějším ESSIV (encrypted salt-sector initialization vector), inicializační vektor pak není útočníkovi znám. Inicializační vektor je získán zašifrováním čísla sektoru hashem hlavního klíče.

Jelikož jsou však bloky na sobě závislé, může útočník za určitých okolností pozměnit data, pokud nevadí, že první blok bude obsahovat nesmyslná data.

Přestože je tato metoda velmi často používána (v kombinaci s ESSIV), byla více doporučovaná metoda LRW.

3.3 Liskov, Rivest a Wagner

Metoda LRW používá postup, kdy se data se před a po zašifrování sčítají (xorují) výsledkem násobení druhého klíče a logického indexu. Sčítání a násobení se provádí v aritmetice nad konečnými tělesy (Galoisovo pole). Tato metoda není náchylná na watermark attack, je považována za vcelku bezpečnou a byla dlouho doporučována.

I u metody LRW se však našla zranitelnost. Útočník může odvodit LRW klíč K2 ze zašifrovaných dat, pokud původní data obsahují K2||0n nebo 0n||K2. Kde || je operátor konkatenace a 0n je blok n nul. To může být problém pro software, který šifruje oddíl operačního systému na kterém v té době běží. Operační systém mohl uložit tento klíč do zašifrovaného odkládacího / uspávacího souboru [9].

Pokud je znám druhý klíč, může být útočník schopen umístit nějaká data, která se správně dešifrují (v data vytvořená útočníkem) [12].

V současné době je proto od IEEE P1619 doporučována metoda XTS.

3.4 XEX a XTS

Jako pokročilejší a bezpečnější metody jsou v současné době doporučovány XEX (Xor Encrypt Xor) a hlavně XTS (XEX-TCB-CTS). Oba fungují na velmi podobném principu, popis lze nalézt ve specifikaci IEEE P1619 [8]. K zašifrování bloku j uvnitř sektoru I. Zašifrovaná data získáme tak, že data před a po zašifrování klíčem K_1 xorují vektorem X:

$$C = E_{K_1}(P \oplus X) \oplus X$$

Tento vektor se získá vynásobením šifry (v aritmetice Galoisova pole) čísla sektoru I klíčem K_2 s primitivním prvkem Galoisova pole (2^{128}) umocněným na číslo bloku:

$$X = E_{K_2}(I) \otimes \alpha^j$$

XTS je doporučován od prosince 2007 jako standard od IEEE 1619.

3.5 Wide-block metody

K výše uvedeným metodám, nazývaným narrow-block metody, vznikly alternativy, které šifrují celý blok najednou. Těmto metodám se říká wide-block metody. Tyto metody jsou zajímavou alternativou, ale jelikož poskytují výrazně nižší výkon a některé (např. EME) jsou pokryty patenty, tak nejsou příliš využívány v praxi.

3.5.1 CMC

Metodu CMC zavedli Halevi a Rogaway. CMC zde znamená CBC-mask-CBC. Celý sektor je zašifrován s metodou CBC, poté je vymaskován xorováním a opět je použita metoda CBC. Jak je z principu vidět, tak hlavní nevýhodou je nutnost pokaždé data zpracovat dvakrát. Tento problém má řešit paralelizovatelná metoda EME.

3.5.2 EME

Jako řešení problému s nutností zpracovávat data dvakrát, přišli autoři CMC, Halevi a Rogaway, s paralelizovatelnou variantou EME neboli ECB-mask-ECB. Metoda funguje podobně jako předchozí CMC. Jednotlivé bloky jsou na počátku vyxorovány a poté posunuty vlevo (každý blok o jinou hodnotu). Výsledkem tedy je možnost paralelního zpracování bloků a zároveň jsou stejné bloky na různých místech zašifrovány různě.

3.6 Watermark attack

Některé šifrované souborové systémy používaly přístup, kdy byl každý blok zašifrován stejným klíčem (což je u metody electronic codebook — ECB). Problém však je, že stejná data, zašifrovaná stejným klíčem, vypadají stejně. Útočník tak může pomocí souboru (návnady) zjistit, máte-li na disku nějaká data, aniž by znal šifrovací klíč [19].

Celý způsob lze vysvětlit na jednoduchém příkladu. Mějme dva známé Karla a Frantu. Karel dluží Frantovi peníze a jelikož je Franta sklerotik, napsal si před Karlem do souboru: „Karel mi do května vrátí 20000,-“. Franta si dává pozor na svá data a tak používá šifrovaný souborový systém. Karel má fyzický přístup k počítači, ale nezná heslo. Pošle tedy Frantovi nějaké vtipné video, které si Franta uloží na disk. Za videem však byla připojena další data. Jednalo se o vzory dat, které se složitým způsobem opakují. Další část obsahovala větu o Karlově dluhu a poslední část větu: „Karel mi do června vrátí 12000,-“.

Když se pak Karel dostal k počítači, na kterém spustil live cd systému, tak prohledal disk a hledal v něm opakující se vzor dat, jaký byl připojený k videu. Jak data vypadají vědět nemusel, protože stačí nalézt vzor, s jakým se zašifrovaná (neznámá) data opakují. Když je našel, tak věděl, že v další části je uložena zašifrovaná Frantova věta. Nechal si ji tedy nalézt na jiném místě. Tuto větu pak přepsal posledním blokem dat, kde byla uložena změněná věta o dluhu.

Až se pak Franta podívá do souboru, bude tam již Karlem změněná věta o nižším dluhu a pozdější splátce.

Tento Karlův podlý čin byl možný pouze díky tomu, že Franta používá šifrování náchylné na watermark attack. Stejná data (věta o dluhu) uložená na různých místech jsou šifrována stejně a proto jsou i zašifrovaná data stejná, takže je lze identifikovat. Aby bylo šifrování proti tomuto útoku odolné, tak by data musela být šifrována na různých místech jinak. Díky tomu by nebylo možné jejich přítomnost určit.

V současné době je již většinou používána jiná metoda šifrování bloků, kdy je klíč upraven pomocí inicializačního vektoru. Mezi nejčastější metody patří LRW (odvozeno od jmen autorů Liskov, Rivest, Wagner) a ESSIV (Encrypted Salt-Sector Initialization Vector) [5]. Je důležité dát si pozor, protože některé šifrované souborové systémy (či jiné šifrovací nástroje) by mohly používat v defaultním nastavení metodu ECB (například z důvodu zpětné kompatibility).

Kapitola 4

Bezpečnostní rizika

Pokud jsou data uložena v zašifrované podobě, neznamená to, že máme vyhráno a nehrozí, že útočník data získá. Ochrana dat šifrováním je důležitá, nestačí však sama o sobě. Musí být splněno i několik dalších podmínek.

4.1 Nedůvěryhodný počítač

Nejste-li si jistí bezpečností počítače, neměli byste data na počítači dešifrovat (u dat šifrovaných jednotlivě), ani připojovat šifrované oddíly (například z USB disku). Na počítači může být logovací software, který zaznamená vaše hesla, případně rovnou zkopíruje veškerý obsah. Po zadání dešifrovacího klíče jsou data dešifrována transparentně a pro všechny programy, které k nim budou chtít přistupovat. Takže nejen prohlížeč dokumentů, ale i útočníkův skrytý „zálohovací“ nástroj.

Podobně u unixových systémů může správce programy, pomocí kterých dešifrujete, nahradit za jiné, které kromě dešifrování zároveň zaznamenají heslo a klíč, který jste k dešifrování použili. Případně může jednoduše proces běžící na pozadí data po připojení zkopírovat. Jelikož programy připojující a dešifrující oddíl vyžadují zvýšená oprávnění, mívají nastaveno, aby se spouštěly s právy superuživatele. Takovéto programy si však nemůžete přinést svoje vlastní (důvěryhodné), protože bez oprávnění nebudou schopny fungovat.

Měli byste tedy mít k počítači, na kterém data dešifrujete, důvěru.

4.2 Dešifrované pozůstatky

Pokud data dešifrujete a používáte, jsou uložena v operační paměti. Při práci mohou být data programem pro práci uložena i do adresáře pro dočasná pracovní data (v Linuxu adresář /tmp). V případě, že systém bude mít méně paměti, mohou být odložena do swapu na disk. Pokud provedete hibernaci (uspání počítače na disk), bude celý obsah operační paměti také uložen na disk. Ve všech těchto případech se na disk dostanou použitá dešifrovaná data, případně i rovnou dešifrovací klíč. Aby bylo bezpečí zachováno, je nutné šifrovat nejen data, která používáte, ale i místa, kde se data mohou objevit. Konkrétně tedy swap a adresář pro dočasné soubory.

4.3 Žurnálovací souborové systémy

Využíváte-li výhod žurnálovacího souborového systému, pro větší zabezpečení proti nekonzistenci dat na disku, číhá zde další úskalí. Uvažuje se pouze varianta, že šifrované soubory nebo obraz šifrovaného disku je uložen na žurnálovacím souborovém systému. Použití tohoto systému souborů uvnitř zašifrovaného oddílu nebo rovnou celého disku následujícími problémy netrpí.

V záznamu žurnálu se mohou objevit informace o přístupech k šifrovaným datům, což samo o sobě může být dost nebezpečné. Větší problém ale skýtá možnost, že se na disku objeví hlavička šifrovaných dat obsahující zaheslovaný klíč k dešifrování. Šifrované souborové systémy většinou obsahují zašifrovaný vlastní klíč, kterým jsou šifrována ostatní data. V případě, že chcete změnit svůj klíč a/nebo heslo, je těmito novými údaji zašifrován pouze klíč pro šifrování zbylých dat. Zbylá data zůstávají zašifrována tímto klíčem. Problém ale je, že pokud útočník získá původní hlavičku s klíčem a k tomu váš klíč (a heslo), tak přestože vy si tyto údaje změníte, je schopen se starým klíčem data dešifrovat. Tento starý klíč však může při změně zůstat uložen na disku a nebýt přepsán [17].

4.4 „Šetrné“ ukládání dat

Některá úložná zařízení, zejména flash disky, obsahují mechanismus pro prodloužení jejich životnosti. Jelikož mají paměti flash omezenou životnost v počtu zápisů, nejsou data zapsána vždy na stejný blok, jako je tomu u klasických pevných disků, ale bloky se střídají. Tím se dosáhne toho, že není neustále přepisována malá skupina bloků, které by s životností brzy skončily, zatímco ostatní bloky by byly skoro jako nové. V praxi to znamená, že pokud přepíšete nějaký soubor, není blok dat přepsán, ale data jsou zapsána do jiného bloku a původní blok je označen jako prázdný. Přesto však obsahuje původní data a může se jednat právě třeba o starší hlavičku od šifrovaného souborového systému, ke které bylo prozrazeno heslo a vy jste ho tedy změnili. Útočník však může s pomocí této staré hlavičky a znalosti starého hesla data dešifrovat.

4.5 Paměťová remanence

Počátkem tohoto roku přišla Princetonská univerzita se zjištěním, že data v operační paměti zůstávají i nějakou dobu po vypnutí napájení [1]. Obvykle se věřilo tomu, že s vypnutím počítače jsou data z operační paměti ztracena. Provedením experimentů se zjistilo, že data se neztratí hned, ale mohou v paměti vydržet několik desítek vteřin. Poté se data začnou postupně ztrácet. Pokud však paměť ještě zchladíte například tekutým dusíkem, vydrží data v paměti okolo deseti minut. Paměť je možné v této době vyndat, umístit do jiného počítače a přečíst její obsah. Potom již stačí v paměti identifikovat klíč a útočník může data dešifrovat. Pokud je počítač v režimu spánku nebo podobném stavu, tak je paměť stále napájena a tudíž stále obsahuje klíč. To dává útočníkovi ještě více času. Jediný způsob, jak se proti tomuto bránit, je počítač skutečně vypnout a ne jej převádět do úsporného režimu. Po vypnutí je lepší ještě chvíli počítač hlídat, aby v kritickém okamžiku nedošlo k jeho krádeži. Při běžném ukončení by šifrovací nástroje měly paměť, která obsahovala důležitá data, přepsat a zničit tak obsah. Zda toto konkrétní šifrovací nástroje provádějí se obvykle bohužel neuvádí.

4.6 Útok hrubou silou

Při vytváření šifrovaného souborového systému je možné vybrat si některý z přístupů zabezpečení. Pomocí hesla, pomocí klíče a nebo pomocí zaheslovaného klíče. V případě použití pouze hesla je důležité dbát na kvalitu hesla proti útokům hrubou silou, protože jinak samotné šifrování dat skoro ztrácí smysl. Kromě hesla je také důležité, aby nebyly zvoleny příliš slabé parametry pro šifrování — zejména příliš malý klíč nebo nedostatečně bezpečné metody šifrování bloků.

4.7 Šifrovací algoritmy

Většina šifrovacího softwaru obsahuje dostatečný výběr šifrovacích algoritmů. Mezi nejčastější patří tyto:

- DES – Data Encryption Standard (neboli DES) je jedním z nejstarších používaných algoritmů pro šifrování dat. Zejména kvůli malé velikosti klíče umožňující útok hrubou silou již není příliš používaný ani doporučovaný. DES používá 64 bitů velké bloky. Klíč je velký 64 bitů, použito pro šifrování je však pouze 56 bitů.

- 3DES – Triple DES je v podstatě třikrát použitý DES pro zesílení bezpečnosti. Algoritmus je však mnohem pomalejší než ostatní dnes používané varianty. Zároveň kvůli útoku typu „meet in the middle“ není přes použití 3 klíčů (tj. celkem 168 bitů) síla algoritmu tak velká, uvádí se, že odpovídá pouze přibližně 112 bitům.

- Blowfish – Algoritmus Blowfish je považován za bezpečný a poměrně rychlý algoritmus, nezatížený patenty. Agoritmus používá klíč velikosti 32-448 bitů s krokem velikosti 8 bitů. Blowfish používá bloky velké 64 bitů. Jedná se ve své podstatě o 16-ti kolovou Feistlovu šifru, používá S-boxy závislé na klíči. Autor tohoto algoritmu však doporučuje, aby byl raději místo něj používán algoritmus Twofish.

- Twofish – Algoritmus je částečně považován za nástupce alg. Blowfish, je bezpečný a poměrně rychlý, také jako předchůdce je nezatížen patenty. Algoritmus používá bloky velikosti 128 bitů. Velikost klíče je buď 128, 192 nebo 256 bitů.

- Serpent – Tento algoritmus byl v pořadí druhý při rozhodování o zařazení mezi standard. Algoritmus používá bloky velikosti 128 bitů. Velikost klíče je buď 128, 192 nebo 256 bitů. Jedná se o 32 kolovou permutačně-substituční síť pracující s 32-bitovými slovy. Tento algoritmus je také dobře paralelizovatelný.

- AES – Tento algoritmus je brán jako standard pro šifrování dat, je dostatečně bezpečný a v současné době je jako volba číslo jedna implementován v naprosté většině šifrovacího softwaru. Algoritmus používá bloky velikosti 128 bitů. Velikost klíče je buď 128, 192 nebo 256 bitů.

Z hlediska bezpečnosti je doporučován algoritmus AES. Použití algoritmů Blowfish a Twofish je také dostatečně bezpečnou volbou. Algoritmus DES ale je vhodný spíše jen z hlediska zpětné kompatibility.

4.8 Porovnání

Veškerý šifrovací software není stejně kvalitní a stejně bezpečný. Některé programy jsou bezpečnější a některé zas novější (bezpečnější) algoritmy (zatím) nepoužívají. Všechny uvedené programy používají (umí používat) šifrovací algoritmus AES. Přehled nejdůležitějších vlastností je v následující tabulce:

name	source	pre-boot	root	swap	tajný IV	LRW	XTS
dm-crypt	ano	ano	ano	ano	ano	ano	ne
EncFS	ano	ne	ne	ne	ne	ne	ne
eCryptFS	ano	ne	ne	ne	ano	ne	ne
TrueCrypt	ano	ano	ano	ano	ne	zpětně	ano
loop-aes	ano	ano	ano	ano	ano	ne	ne
EFS	ne	ne	ne	Vista a novější	ano	ne	ne
SD4L	ano	ne	ne	ne	ano	ano	ne
PGP Disk	RO	ano	ano	ano	ano	EME	ne
cryptoloop	ano	ano	ano	ano	ne	ne	ne
BitLocker Drive Encryption	ne	ano	ano	ano	ano	ne	ne

Význam jednotlivých položek:

- source – dostupnost zdrojových kódů
  - ano
  - RO – zdrojové kódy jsou pouze pro čtení, nesmí být modifikovány
  - část – ze zdrojových kódů nelze zkompilovat program, k dispozici je obvykle jen tzv. kritická část. Díky tomu však také nejde zaručit, že dodaný program v binární podobě skutečně je vytvořen z konkrétních zdrojových kódů.
  - ne
- pre-boot – zda je možné provést autentizaci před zavedením systému a zabránit tak přístupu kompletně.
- root – zda je možné šifrovat i oddíl se systémovými soubory.
- swap – možnost šifrování oddílu swap či odkládacího prostoru (pagefile.sys).
- tajný IV – zda je při použití metody šifrování bloků CBC (nebo podobné) použit inicializační vektor, který není možné odvodit (např. je šifrován).
- LRW – zda program podporuje šifrování bloků metodou LRW či jinou obdobně bezpečnou např. při použití wide-block encryption (EME, CMC).
- XTS – podpora metody XTS nebo XEX.

4.9 Shrnutí

Pro šifrování je vhodné, aby software podporoval algoritmus AES či Twofish s 256 bitovým klíčem a metodu šifrování bloků alespoň CBC s tajným inicializačním vektorem (např. ESSIV) nebo lépe XTS či LRW.

Zároveň je důležité pomocí tohoto či jiného nástroje šifrovat odkládací prostor a úložiště dočasných souborů.

Nejdůležitějším prvkem je však použití dostatečně silného hesla či kombinace např. s USB tokenem.

Kapitola 5

Typy šifrovaných souborových systémů

Šifrovaných souborových systémů existuje několik variant. V této kapitole zmíníme několik nejpoužívanějších nebo alespoň dříve známých. Stručné porovnání šifrovaných souborových systémů lze nalézt například na wikipedii [2]. U šifrovaných systémů se hodnotí hlavně, jsou-li použitelné pro konkrétní plánovaný účel použití (například jestli umí vytvářet skryté oddíly nebo kontejnery). Zároveň je vhodné, aby používaly nějakou z bezpečnějších metod šifrování bloků, která není náchylná na watermark attack.

Ve většině případů jsou šifrované souborové systémy zhruba stejně bezpečné jako ostatní ve stejné kategorii (šifrování souborů, oddílů,...). Obyčejně se používá šifra AES-256 a způsob šifrování bloků LRW. Případné objevené nedostatky bývají u aktivně udržovaných nebo vyvíjených nástrojů brzy odstraňovány.

5.1 NTFS – EFS

EFS neboli Encrypting File System je šifrovaný souborový systém dostupný v operačních Microsoft Windows (2000 a novější) při použití souborového systému NTFS. Šifrování probíhá pro uživatele transparentním způsobem. Systém souborů NTFS umožňuje kromě šifrování i kompresi, ale soubory nemohou být přímo souborovým systémem šifrovány a komprimovány zároveň.

Data jsou zašifrována pomocí klíče, který je zašifrován veřejným klíčem uživatele a uložen v příslušných metadatech. K dešifrování se použije privátní klíč odemčený heslem uživatele.

Každý adresář má nastaven atribut, zda je šifrovaný. Pokud ano, jsou všechna data v něm (soubory i podadresáře) defaultně zašifrována. Při kopírování nebo přesouvání na jiný souborový systém jsou uložena v nezašifrované podobě. Stejně tak jsou data uložena nezašifrovaně, pokud jsou zkopírována do adresáře, který není nastaven jako šifrovaný. Pokud jsou data kopírována na vzdálené úložiště pomocí protokolu SMB/CIFS, jsou v dešifrované podobě přenášena po síti a tak i uložena. Je nutné si dávat pozor na použití zálohovacích programů, protože některé zálohovací programy mohou data ukládat v nezašifrované podobě. Je proto důležité používat zálohovací programy, které jsou schopné používat tzv. „raw“ API a soubory tak budou zkopírovány v zašifrované podobě včetně veškerých metadat.

Od Windows Vista může být uživatelův soukromý klíč uložen na smart kartě. Ve Windows 2000 je lokální administrátor zároveň jako výchozí Data Recovery Agent (DRA), takže je schopen dešifrovat všechny soubory místních uživatelů. Od Windows XP a novějších není po-

užíván žádný výchozí DRA. Odkládací soubor pagefile.sys je v systému šifrován od Windows Vista.

Systém souborů EFS není plně podporován systémy Windows Vista Starter, Windows Vista Home Basic ani Windows Vista Home Premium.

5.2 cryptoloop

Cryptoloop je šifrovací virtuální zařízení pro Linuxové systémy pod licencí GNU GPL. Může šifrovat diskový oddíl nebo i obraz disku uložený v souboru. Používá speciální zařízení (smyčku) přes kterou jsou data jedním směrem zašifrována, druhým dešifrována. Kernel obsahuje CryptoAPI od verze 2.6, pro starší verze je možné použít patch.

Cryptoloop byl označen jako zastaralý, což znamená, že není již nadále aktivně vyvíjen. Za následníka byl zvolen dm-crypt, který je dostupný od kernelu verze 2.6.4.

5.3 dm-crypt

Jako nástupce pro cryptoloop slouží dm-crypt, je primárně určen pro Linuxové systémy s kernelem od verze 2.6 a je pod licencí GNU GPL. Data zašifrovaná použitím dm-cryptu lze používat i v MS Windows pomocí FreeOTFE.

Jelikož pracuje s šifrováním bloků transparentně, podporuje šifrování celého disku a oddílů, stejně tak jako souborů. S použitím initrd lze v Linuxu používat pre-boot autentizaci. Od kernelu verze 2.6.10 není náchylný na watermark attack a od verze 2.6.20 podporuje šifrování bloků metodou LRW (defaultně je kvůli zpětné kompatibilitě používán náchylný mód CBC).

Mapování zařízení dm-cryptem je v prostoru kernelu a slouží spíše na nižší úrovni k šifrování a dešifrování dat. K jeho využití pro šifrované souborové systémy je potřeba použít ještě nějakou nadstavbu. Nejpoužívanější jsou cryptsetup a cryptsetup-LUKS. Kvůli přidávání snadno identifikovatelné hlavičky nelze bohužel vytvářet skryté šifrované oddíly.

5.3.1 cryptsetup

Cryptsetup je interface pro příkazovou řádku, nepřidává k šifrovaným datům žádná metadata. Parametry musí být zadávány při každém připojování šifrovaného zařízení, zároveň může každý oddíl mít pouze jeden klíč. Symetrický šifrovací klíč je přímo odvozen ze zadané fráze(hesla), proto je vhodnější používat jako frázi třeba soubor.

5.3.2 LUKS

Dalším rozšířením je LUKS (Linux Unified Key Setup), který je založen na původním cryptsetup, ponechává si plnou kompatibilní a přidává další funkce pro práci s diskovým formátem LUKS. Tento formát umožňuje správu klíčů, zesilování hesel (hash+sůl) a zapamatovává si konfiguraci šifrování i po restartu systému.

LUKS funguje tak, že na začátek oddílu přidává svoji hlavičku, která obsahuje informace o šifrování. Její součástí je „magické číslo“ pro identifikaci, že se jedná o tuto hlavičku. Dále mimo jiného obsahuje číslo verze, jméno šifrovacího algoritmu, jméno metody šifrování bloků, jméno hash algoritmu, uuid a sadu dešifrovacích klíčů (pro každé heslo resp. uživatele jiný).

Dešifrovací klíč je hlavní klíč pro dešifrování samotných dat. Tento klíč je uložen v zašifrované podobě. Při změně hesla se tedy akorát uloží hlavní klíč přešifrovaný novým heslem.

Jedná se asi o nejrozšířenější a nejčastěji používanou metodu šifrování v Linuxu.

5.4 AES-loop

AES-loop je alternativou ke cryptoloop, je více propracovanější a podle autora by měl být skoro až dvakrát tak rychlý díky použití v assembleru vysoce optimalizované implementace šifry AES. Jeho možnosti použití jsou zhruba na stejné úrovni jako dm-crpyt/LUKS. Je šířen pod licencí GNU GPL a měl by fungovat na Linuxu s jádrem 2.0 a novějším.

Používá hlavní náhodně generovaný klíč, který šifruje data. Tento klíč je zašifrován heslem, proto je velice důležité používat dostatečně silné heslo. Pro obranu proti watermark attacku AES-loop také umožňuje použití více klíčů zároveň, kdy jsou sektory šifrovány 64 samostatnými klíči. Nepodporuje režim šifrování bloků LRW, používá metodu CBC s tajným init. vektorem, který je odvozen od šifrovacího klíče a čísla sektoru. Při změně dat stejného sektoru je tedy použit vždy stejný IV.

5.5 TrueCrypt

TrueCrypt je nástroj pro šifrované souborové systémy, který je použitelný v MS Windows i v Linuxu. Je volně šiřitelný a má dostupné zdrojové kódy. Nešifruje jednotlivé soubory, ale celý souborový systém. Ten může být uložen jako obraz nebo přímo na některém oddílu pevného disku.

Využívá se i skrytí dat, takže není možné prokázat, jestli jsou na disku (nebo v souboru) zašifrovaná data. Oddíl totiž neobsahuje žádná identifikovatelná data, takže není možné prokázat, jestli se jedná o náhodná data v nevyužitém místě disku, nebo jestli je to zašifrovaný oddíl.

Na začátku každého oddílu je uložena hlavička, která mimo jiné údaje obsahuje sůl, náhodný primární a sekundární klíč pro šifrování datových sektorů (je využíváno šifrování bloků metodou LRW).

Další informace o TrueCrypt lze nalézt na jeho domovských stránkách [18].

5.6 PGP Whole Disk Encryption

PGP Whole Disk Encryption je komerční nástroj od firmy PGP, zdrojové kódy jsou dostupné k nahlédnutí. Šifrován je celý disk a používá se pre-boot autentizace. Zašifrováno je tedy vše : bootovací disk Windows, adresář Windows, Program Files a Documents and Settings. Toto například TrueCrypt neumí. Nelze však vytvářet skryté oddíly.

5.7 eCryptFS

eCryptFS je šifrovaný souborový systém, který je podobný EncFS. Sám sebe označuje jako „An Enterprise-class Cryptographic Filesystem for Linux“ [6].

Sestává se ze dvou částí, jednak z jaderného modulu a z části v uživatelském prostoru. Na stránce věnující se eCryptFS je uvedeno, že vývojový stav je ve stádiu Beta. Podporován je v operačním systému Linux a je licencován pod GNU GPL.

Ke každému souboru jsou v hlavičce ukládána šifrovací metadata, takže zašifrované soubory lze kopírovat na jiné počítače a tam s pomocí platného klíče dešifrovat.

eCryptFS funguje a byl testován na souborových systémech ext3 a jfs. Na souborovém systému xfs mohou vznikat problémy, protože modul jádra pro xfs může spotřebovat většinu z limitovaného místa programového zásobníku. Na vzdálených souborových systémech s protokolem CIFS je dostupná částečná funkčnost, na zlepšení podpory se pracuje. S použitím NFS však funkční není.

5.8 EncFS

Šifrovaný souborový systém EncFS je implementován v uživatelském prostoru pomocí FUSE (Filesystem in USErspace). Jedná se o open source software licencovaný pod GNU GPL.

Funguje na principu připojení šifrovaného adresáře na cílové místo, kdy jsou v cílovém adresáři dešifrované soubory a při jejich změně se změní soubory v šifrovaném adresáři. Dešifrovaná data nejsou fyzicky na disku, při přístupu k nim jsou data čtena ze šifrovaných souborů a dešifrována (a naopak při zápisu jsou data zašifrována a uložena do zmiňovaných souborů).

Po připojení nového šifrovaného adresáře je v něm vytvořen soubor .encfsX s daty pro šifrování, kde X je číslo dle verze programu encfs. Pokud dojde k poškození souboru nebo jeho smazání, jsou data nenávratně ztracena, proto je dobré si jej raději zazálohovat.

Nevýhodou vůči podobnému eCryptFS umístěném v prostoru jádra je zdržování při přepínání režimu uživatelský prostor — prostor jádra.

EncFS je postaven nad jiným souborovým systémem a má několik výhod. Hlavní výhodou je dynamická velikost, protože není nutné předem určit pevnou velikost šifrovaného oddílu. Může být zálohován s využitím běžného software a může být umístěn na vrchu jiného souborového systému např. na CD, NFS nebo CIFS.

Má však také několik nevýhod oproti šifrování celého oddílu. Viditelné zůstává počet šifrovaných souborů, oprávnění jednotlivých souborů, jejich velikost a přibližná délka jména souboru. Samozřejmě také není šifrován oddíl swap.

5.9 ScramDisk 4 Linux

ScramDisk 4 Linux vychází z původního šifrovaného souborového systému ScramDisk. Je šířen pod licencí GNU GPL a funguje v Linuxu s jádrem 2.4 a 2.6. Šifruje celé oddíly, od verze 1.0 dokáže pracovat i s šifrovanými oddíly vytvořenými pomocí TrueCrypt. Dokáže vytvářet skryté oddíly.

5.10 Pointsec

Pointsec šifruje celý pevný disk a poté modifikuje MBR disku, využívá tedy pre-boot autentizaci podobně jako PGP Whole Disk Encryption. Jedná se o komerční nástroj určený pro operační systémy MS Windows od verze Windows 95 a nově je dokonce podporován i Linux. Software v MBR předává autentizační údaje pro přihlášení dále operačnímu systému, takže není nutné zadávat heslo vícekrát. K dispozici je také funkce úschovy klíče, která umožňuje, aby byla zašifrovaná data dešifrována důvěryhodnou autoritou.

Kapitola 6

Způsob instalace

Šifrovaných systémů souborů existuje několik. Práce je zaměřena na použití šifrovaných souborových systémů v Linuxu, takže se dále budeme věnovat šifrovaným souborovým systémům EncFS, eCryptFS a dm-crypt/LUKS. Každý má jiné vlastnosti a jinak se používá. Nejlepší způsob, jak si udělat přehled, je si tyto šifrované souborové systémy vyzkoušet. V této kapitole je tedy uveden a stručně popsán způsob, jak je lze nainstalovat.

Způsob instalace bude uveden pro Linuxovou distribuci Fedora verze 8. V ostatních Linuxových systémech by měla být instalace obdobná s ohledem na případné úpravy v různých distribucích.

6.1 dm-crypt/LUKS

Vytvoření souborového systému šifrovaného dm-crypt pomocí cryptsetup-LUKS je záležitostí několika málo jednoduchých kroků. Pro uvedené použití je nutné mít dostupné programy losetup (balík util-linux-ng) v případě použití souboru s obrazem disku, cryptsetup s rozšířením LUKS (balík cryptsetup-luks).

1. Připravení zařízení
   (a) Použití souboru jako obrazu s oddílem
      i. Vytvoření souboru pro obrazu oddílu
      Soubor vytvoříme pomocí utility dd a zaplníme ho náhodnými daty:

      ```
      dd if=/dev/urandom of=/cesta/k/souboru bs=1k count=500k
      ```

      Velikost souboru se uvádí do parametru count a pro uvedenou velikost kopírovaného bloku je 1000x větší než daná hodnota. Proto pro soubor velký 500 MB uvedeme 500k, pro 100 GB by to bylo 100M. Hodnota count udává kolik bloků velikosti bs bude zkopírováno (viz. manuálové stránky dd). Jelikož je velikost bloku 1 kB, tak count=požadovaná_velikost/1kB.
      Je důležité také nezapomínat, že při vytvoření souboru s obrazem disku velkým 1 GB nebude na tento disk možné uložit celkem 1 GB, ale o něco méně, protože kousek zabere hlavička LUKS a část zaberou data samotného souborového systému.
      ii. Připojení souboru jako loop zařízení. Nejprve zjistíme volné loop zařízení pomocí příkazu `losetup -f` a poté k tomuto zařízení připojíme náš soubor.

Pokud bylo volné zařízení /dev/loop2 tak k němu soubor připojíme následujícím příkazem:

```
losetup /dev/loop2 /cesta/k/souboru
```

Dále uvažované zařízení bude v tomto případě /dev/loopX podle volného loop zařízení. Dle příkladu /dev/loop2

(b) Použití přímo oddílu na disku nebo flash disku
Oddíl i v tomto případě nejprve zaplníme náhodnými daty:

```
dd if=/dev/urandom of=/dev/zarizeni
```

2. Vytvoření hlavičky LUKS na zařízení
Na zařízení vytvoříme hlavičku (zařízení inicializujeme pro šifrování) následujícím příkazem:

```
cryptsetup --verbose --cipher "aes-xts-benbi" --key-size 256
      --verify-passphrase luksFormat /dev/zarizeni
```

Parametry předané programu po řadě zleva jsou: podrobnější výpis informací, volba algoritmu šifrování a metody šifrování bloků (v tomto případě šifrou AES a šifrování bloků dle XTS), velikost klíče, zadání hesla 2x pro ověření, akce vytvoření hlavičky a cílové zařízení. Podrobnější informace lze nalézt na manuálových stránkách programu cryptsetup.

3. Namapování šifrovaného zařízení
Potom, co máme připravené zařízení pro ukládání zašifrovaných dat, vytvoříme šifrované zařízení. Mezi těmito dvěma zařízeními bude pracovat šifrovací vrstva, která bude data jedním směrem šifrovat a druhým dešifrovat.

```
cryptsetup --verbose luksOpen /dev/zarizeni sifrovaneZarizeni
```

Parametry zleva jsou: podrobnější výpis informací, akce otevření zařízení, zdrojové zařízení (kam se ukládají šifrovaná data), výsledné namapované zařízení, které data transparentně šifruje (objeví se jako /dev/mapper/sifrovaneZarizeni).

4. Vytvoření souborového systému
Na zařízení vytvoříme libovolný souborový systém, například ext3:

```
mkfs.ext3 -m 0 /dev/mapper/sifrovaneZarizeni
```

Kromě nutného parametru zařízení, na kterém bude souborový systém vytvořen, stojí za zmínku i parametr -m 0. Tento udává kolik procent místa na disku bude rezervováno pro uživatele root. Toto místo je vyhrazeno pro snadné provedení nutných oprav např. v konfiguračních souborech, kdy je normálními uživateli již zabráno veškeré místo na disku. Obzvláště, pokud je soubor s obrazem oddílu používán pro přenos soukromých dat jedním uživatelem, bývá rezervování místa zbytečné a nežádoucí. Defaultní hodnota parametru je 5.

5. Připojení šifrovaného zařízení
Po naformátování zařízení s konkrétním souborovým systémem je již připraveno k použití a lze jej připojit jako běžný disk:

```
mount /dev/mapper/sifrovaneZarizeni /cilove/misto/pripojeni
```

6. Odpojení šifrovaného zařízení
   Po skončení práce s šifrovaným oddílem jej odpojíme klasickým způsobem pomocí příkazu umount:

   ```
   umount /dev/mapper/sifrovaneZarizeni
   ```

7. Odebrání šifrovaného zařízení
   Jako poslední krok odebereme šifrované zařízení, aby nebylo možné jej opět připojit:

   ```
   cryptsetup luksClose sifrovaneZarizeni
   ```

   Pokud jsme použili soubor s obrazem oddílu, uvolníme také zařízení smyčky:

   ```
   losetup -d /dev/loopX
   ```

   V našem konkrétním případě bychom místo /dev/loopX použili /dev/loop2

Při opětovném použití šifrovaného oddílu vynecháme inicializaci náhodnými daty příkazem `dd` a kroky číslo 2 (vytvoření hlavičky LUKS) a 4 (vytvoření souborového systému).

6.2 eCryptFS

Pro použití šifrovaného souborového systému eCryptFS použijeme několik následujících kroků:

1. Zavedení modulu ecryptfs příkazem `modprobe ecryptfs`
2. Připojení šifrovaného adresáře

   Zdrojem i cílem je v tomto případě adresář. Doporučuje se, aby zdrojový i cílový adresář byly totožné (nemusí být), čímž dojde k překrytí zdrojového adresáře a aplikace tedy nebudou moci přistupovat přímo k šifrovaným souborům.

   ```
   mount -t ecryptfs /zdrojovy/adresar /cilovy/adresar
   ```

3. Odpojení adresáře
   Po skončení práce adresář opět odpojíme klasickým způsobem:

   ```
   umount /cilovy/adresar
   ```

Místo hesla lze využít i klíče uloženého na jiném médiu. Klíč vytvoříme pomocí programu `ecryptfs-manager` zadáním voleb: vytvoření nového klíče, modulem openssl, zadání cílového místa klíče a zadáním hesla pro klíč. Šifrovaný oddíl poté připojíme příkazem:

```
mount -t ecryptfs -o key=openssl:keyfile=/cesta/ke/klici/mujklic.pem
        /zdrojovy/adresar /cilovy/adresar
```

Před připojením je nutné spustit démona `ecryptfsd`.

Tento šifrovaný souborový systém nebylo možné zatím důkladněji otestovat, protože byl v aktuální verzi jádra zcela nefunkční — docházelo k vyvolání segfault v modulu při pokusu otevřít soubor, patrně kvůli vyčerpání prostoru zásobníku. Bug-report byl odeslán vývojářům. Zprovoznění eCryptFS se nakonec podařilo, ale až v době těsně před odevzdáním této diplomové práce s jádrem 2.6.25.3. Bohužel díky tomu již nemohl být tento šifrovaný souborový systém řádně prozkoumán.

6.3 EncFS

Použití EncFS je asi nejjednodušší ze všech uvedených možností.

1. Připojení šifrovaného adresáře
   Zdrojem i cílem je v tomto případě opět adresář. Zdrojový a cílový adresář se nemohou překrývat. Z hlediska oprávnění uživatele pro práci s tímto šifrovaným souborovým systémem je dostatečné, pokud je uživatel ve skupině `fuse`.

   ```
   encfs /zdrojovy/adresar /cilovy/adresar
   ```

   Při prvním spuštění se program zeptá na parametry šifrování. Zadávání parametrů není nezbytným krokem, je možné zvolit „přednastavené paranoidní parametry“. Použití přednastavených parametrů ale způsobí nefunkčnost hardlinků u šifrovaných souborů, takže některé programy (například `mutt` a `procmail`) přestanou fungovat. Na tento problém je však uživatel viditelně upozorněn.

   Při ručním zadávání (volba expertního režimu) může uživatel mimo jiné vybrat, jaký má být použit šifrovací algoritmus, velikost klíče a velikost bloku. Dále je možné také zvolit jestli jména souborů mají být šifrována a pokud ano, jestli se má zachovávat délka názvu nebo má být délka názvu zarovnávána do malých bloků.

   V obou případech je uživatel ještě dvakrát dotázán na heslo.

2. Odpojení adresáře
   Po skončení práce adresář opět odpojíme způsobem pro FUSE:

   ```
   fusermount -u /cilovy/adresar
   ```

Kapitola 7

Porovnání rychlosti

Prozatím vybrané šifrované souborové systémy mají různé vlastnosti z hlediska jejich použití. Liší se však i ve způsobu, jak jsou vytvořeny a jak fungují. To se samozřejmě projevuje i v rychlosti těchto souborových systémů i když jsou použity stejné parametry šifrování.

Pro porovnání rychlosti bylo provedeno testování na 64-bitovém notebooku Asus A6T, který má procesorem AMD Turion TL-52, 1 GB RAM a pevným diskem Seagate Momentus s 5400 otáčkami. Použitá byl Linuxový systém v distribuci Fedora 9 s jádrem 2.6.25.2. Základem byl souborový systém ext3. Systém byl po čerstvé instalaci s více než 70 % volného místa na disku, proto se předpokládá minimální fragmentace zapisovaných dat. Vliv na průběh testu byl zanedbán.

Testování probíhalo vždy po restartu systému do init úrovně 2 a po vyčkání půl minuty na dokončení všech operací systému souvisejících s jeho startem. Po kompletním startu byl vytvořen šifrovaný souborový systém a byla provedena synchronizace disků vyprázdněním všech vyrovnávacích pamětí. Měřena byla doba od začátku kopírování 10 000 souborů do ukončení synchronizování disku. Soubory byly velké 4 kB a byly kopírovány v náhodném pořadí stejném pro všechny měřené souborové systémy. Druhá část měření se týkala zápisu 800 MB velkého souboru. Po těchto operacích byl systém restartován, připojil se šifrovaný souborový systém a měřily se doby čtení pro 10 000 malých souborů a jeden 800 MB soubor.

Testy se prováděly pro šifrovaný souborový systém EncFS, eCryptFS a dm-crypt/LUKS s různými parametry šifrování. Test s eCryptFS se však nepodařilo realizovat, protože tento nejnovější a hodně vyvíjený souborový systém vždy vyvolal chybu v jádře. S největší pravděpodobností byla chyba způsobena vyčerpáním velikosti zásobníku v jádře.

V následující tabulce jsou výsledky měření. Jedná se o zprůměrované hodnoty dvaceti měření. Pokud jsou někde hodnoty času CPU větší než hodnoty reálného času, je to dáno tím, že testy byly prováděny na dvouprocesorovém počítači a některé části testu běžely paralelně.

	10 000 x 4 kB		1 x 800 MB	
šifrovaný syst. soub.	reálný čas	čas s CPU	reálný čas	čas s CPU
nešifrovaný loop (zápis)	2:40,975	0:58,476	0:37,039	0:39,827
nešifrovaný loop (čtení)	2:40,845	0:58,362	0:36,678	0:39,459
EncFS (zápis)	2:43,316	0:56,757	0:36,960	0:39,805
EncFS (čtení)	2:43,485	0:57,044	0:37,005	0:39,885
EcryptFS (zápis)	—	—	—	—
EcryptFS (čtení)	—	—	—	—
dm-crypt/LUKS (zápis) CBC-ESSIV:SHA256	2:41,012	0:58,907	0:36,820	0:39,522
dm-crypt/LUKS (čtení) CBC-ESSIV:SHA256	2:41,542	0:58,944	0:36,940	0:39,681
dm-crypt/LUKS (zápis) LRW-BENBI	2:41,402	0:59,058	0:36,955	0:39,598
dm-crypt/LUKS (čtení) LRW-BENBI	2:41,272	0:59,162	0:37,058	0:39,800
dm-crypt/LUKS (zápis) LRW-plain	2:41,252	0:59,097	0:37,122	0:39,918
dm-crypt/LUKS (čtení) LRW-plain	2:41,390	0:58,900	0:36,961	0:39,595
dm-crypt/LUKS (zápis) XTS-BENBI	2:40,809	0:58,831	0:36,849	0:39,488
dm–crypt/LUKS (čtení) XTS-BENBI	2:40,363	0:58,510	0:36,693	0:39,274
dm-crypt/LUKS (zápis) XTS-plain	2:40,787	0:58,790	0:36,952	0:39,667
dm–crypt/LUKS (čtení) XTS-plain	2:40,926	0:58,799	0:36,815	0:39,518

Šifrovací algoritmy a blokové metody již zmíněny byly. Pro dm-crypt/LUKS ještě zbývá vysvětlit způsoby získání inicializačního vektoru:

- plain — little-endian 32-bitové číslo sektoru doplněné nulami dle potřeby.
- ESSIV:*alg* — "encrypted sector|salt initial vector" — zašifrovaný sektor+sůl. Sůl je zde získána aplikací algoritmu *alg* na klíč.
- BENBI — big-endian 64-bitové číslo bloku (začínající od jedné).
- NULL — nulový inicializační vektor, slouží pouze pro zpětnou kompatibilitu a neměl by se využívat.

Před zhodnocením výsledků je nutné zmínit, že šifrování s dm-crypt bylo použito ne s oddílem disku, ale pouze s jeho obrazem. V jeho čase je tak započítána ještě navíc režie systému souborů, ve kterém byl obraz oddílu umístěn, a zařízení smyčky. Přesto vše je jeho rychlost obdobná jako rychlost EncFS. Při umístění přímo na oddílu disku by byl výsledek ještě lepší.

Dále je zajímavé, že bezpečnější metoda šifrování bloků XTS je z testovaných metod nejrychlejší.

Proč je čas některých operací pro šifrování přes zařízení smyčky nižší, než jen samotný souborový systém umístěný na zařízení smyčky, je však nejasné.

Kapitola 8

Výběr dále zpracovávané technologie

Při prozkoumávání možností uvedených šifrovaných souborových systémů je znát, že dm-crypt je již nějakou dobu na světě, jeho použitelnost je dobrá a možnosti velké. eCryptFS vypadá jako zajímavý projekt, který ale nebyl v době testování použitelný, kvůli chybě v jeho modulu. EncFS poskytuje velkou výhodu v bezproblémovém použití i na vzdálených úložištích a orientací na uživatelský prostor pro použití uživatelem bez práv superuživatele.

Pro použití v praxi je nutné šifrovat i oddíl swap a místa, kde se mohou dešifrovaná data objevit. To je jednoznačně práce pro dm-crypt. Uživatel tedy může šifrovat jednotlivé soubory a k tomu používat dm-crypt pro swap nebo rovnou použít dm-crypt/LUKS a šifrovat vše. Data zašifrovaná pomocí dm-crypt je možné používat i v prostředí MS Windows pomocí programu FreeOTFE. Z těchto důvodů a dále z důvodu větší odladěnosti byl pro následující práci vybrán právě dm-crypt/LUKS. Pro tuto technologii bude v další části práce vytvořen nástroj pro správu.

Užitečnost zbylých dvou šifrovaných systémů je i přesto velká a dokáže bez problémů zaplnit místo vytvořené poptávkou pro konkrétní účely.

Kapitola 9

Jak funguje dm-crypt s LUKS

Před vytvořením nového nástroje pro princip šifrování s dm-crypt/LUKS je vhodné se nejprve seznámit jak konkrétní technologie funguje.

V Linuxu bylo nejprve používáno šifrování přes zařízení zpětné smyčky, kde se jen datům do cesty přidalo šifrování. Podobným způsobem vzniklo i šifrování pomocí dm-crypt. Nejprve byl vytvořen modul device-mapper, což je ve své podstatě abstraktnější vrstva nad zařízeními, která dokáže z jednoho či z více zařízení namapovat jejich určitou část a vytvořit tak nové zařízení. Později byla opět datům do cesty přidána šifrovací vrstva a vznikl tam dm-crypt.

9.1 dm-crypt bez LUKS

S možností vytváření mapování, vznikl nástroj dmsetup pro vytvoření konkrétního mapování. Šifrování u něj byla také spíše „vedlejší záležitost“, takže jeho obsluha byla poměrně obtížná, např.:

```
blksize=$(blockdev --getsize /dev/loop/0)
key="0123456789abcdef0123456789abcdef"
echo "0 $blksize crypt aes-plain $key 0 /dev/loop/0 0" |
    dmsetup create crypto_mapped_device
```

Jistý pokrok přinesl program cryptsetup, který se zabýval právě jednodušším vytvořením mapovaného šifrovaného zařízení. Zdrojem dat byla ale pouze samotná surová šifrovaná data, takže z nich nebylo možné získat žádné informace o šifrování (což nemusí být někdy na škodu, viz. skryté kontejnery). Při požadavku používat (dešifrovat) tato data jste museli kromě klíče zadat opět všechny parametry jako šifrovací algoritmus, velikost klíče, režim šifrování bloků a další. Zároveň nikde neexistuje nic jako kontrolní součet, takže je nutné mapování s určitými konkrétními parametry vytvořit a pak teprve ověřit, jestli oddíl připojit jde a je-li tedy zadán správný klíč. Pokud na mapované zařízení se špatným klíčem zapíšete data, tak samozřejmě o všechna data přijdete.

Samotné šifrování a věci okolo se nachází přímo v jádře systému, takže se mění i s jeho vývojem. Jde hlavně o podporu šifrovacích algoritmů a režimů šifrování bloků.

9.2 Rozšíření LUKS

Po určité době byl napsán program cryptsetup-luks, který před šifrovaná data přidával hlavičku LUKS [10] (Linux Unified Key Setup). Původním řešeným problémem byla nutnost

zadat vždy všechny parametry pro dešifrování, jinak by nebyla data správně dešifrována. S využitím hlavičky není třeba zadávat již všechny parametry, stačí jen zadat zařízení s daty a klíč a program zjistí potřebné informace z hlavičky. Díky jednoduššímu používání byl program cryptsetup v naprosté většině linuxových distribucí nahrazen za cryptsetup-luks. Program zachovává zpětnou kompatibilitu, takže jej lze použít i stejným způsobem (se stejnými parametry), jako dříve. Díky tomu je tedy možné i data identifikovat jako šifrovaná od původně vypadajícího smetí.

Při šifrování je na zařízení umístěna hlavička, která dle použitých parametrů zabírá obvykle velikost půl až dva megabajty. S touto velikostí je potřeba počítat. Je-li v plánu zašifrovat konkrétní data například pomocí souboru s obrazem oddílu, je třeba použít dostatečně velký soubor. Větší komplikace to však přináší v kombinaci s problémem posunutí dat. U starší varianty šifrování, kde nebyla přítomna hlavička, bylo možné data zašifrovat či dešifrovat přímo (s vědomím rizika ztráty dat v případě problému) pomocí běžného nástroje `dd`. Toto však již možné není, data je nutné posunout a ani se na cílové zařízení nevejdou.

Hlavička LUKS je umístěna na začátku šifrovaného zařízení spolu s klíči následujícím způsobem:

hlavička	data klíčů	šifrovaná data
4 kB	$8 * X$	velikost zařízení $-4\,\texttt{kB} - 8 * X$

Velikost hlavičky je ve skutečnosti výrazně menší, ale veškerá data se ukládají zarovnána do bloků 4 kB. Hodnota X je velikost bloku s materiálem klíče (bude vysvětleno později). Tato velikost při defaultní kombinaci pro klíč 128 bitů odpovídá 64 kB. Velikost bloku je lineárně závislá na velikosti klíče, tedy pro 256 bitový klíč je 128 kB. Hlavička obsahuje následující údaje:

offset	velikost	název	popis
0	6	magic	Identifikační údaj, že se jedná o hlavičku LUKS. Vždy obsahuje „LUKS“ a bajty 0xba a 0xbe
6	2	version	Číslo verze. Aktuální verze mají číslo 1.
8	32	cipherName	Název šifrovacího algoritmu, např. „AES“.
40	32	cipherMode	Název režimu šifrování bloků pomlčka název způsobu získání inicializačního vektoru, např. „lrw-benbi“
72	32	hashSpec	Název použitého hashovacího algoritmu. V současné době je to výhradně SHA1.
104	4	payloadOffset	Offset, kde začínají šifrovaná data.
108	4	keyBytes	Počet bytů hlavního klíče.
112	20	mkDigest	Výsledek kontrolního hashe hlavního klíče.
132	32	mkDigestSalt	Hodnota salt pro hashování hlavního klíče.
164	4	mkDigestIterations	Počet iterací hashování hlavního klíče.
168	40	uuid	Univerzální unikátní identifikátor zařízení.
208	n*48	keyblock	Informace o jednotlivých klíčích.

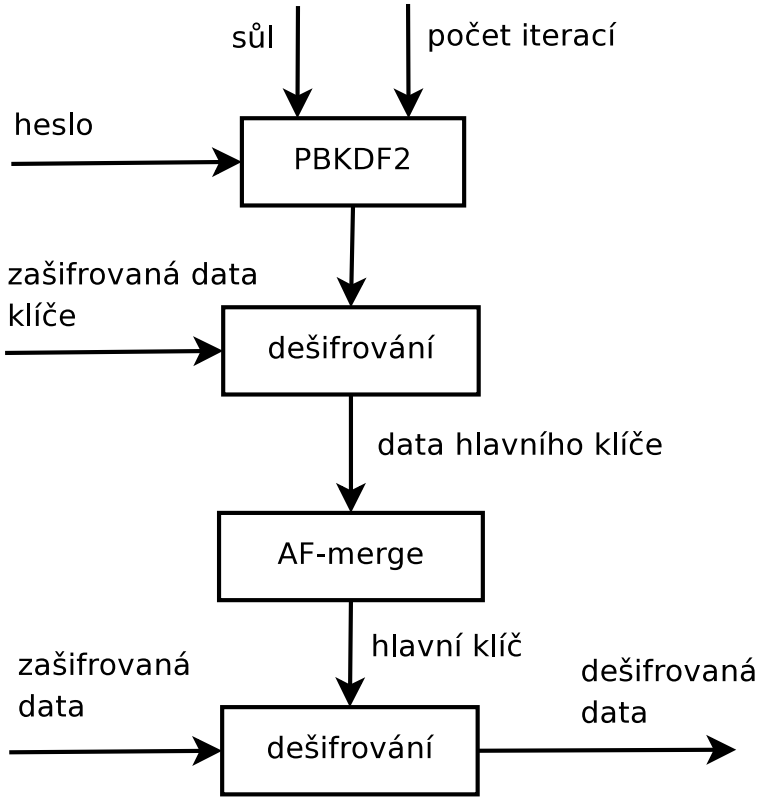
Informace o klíčích (keyblock):

offset	velik.	název	popis
0	4	active	Zda je pozice klíče aktivní (hodnota 0x00AC71F3) nebo neaktivní (hodnota 0x0000DEAD).
4	4	passwordIterations	Počet iterací hashování hesla.
8	32	passwordSalt	Hodnota salt pro hashování hesla.
40	4	keyMaterialOffset	Offset, kde je uložen materiál klíče.
44	4	stripes	Na kolik kousků je rozdělen blok dat hlavního klíče.

Hlavička obsahuje v současnosti 8 pozic pro klíče, takže je možné mít pro šifrovaný oddíl 1–8 klíčů, více možné není. V hlavičce je také uložen několikanásobný hash hlavního klíče a hodnoty salt (také uložena v hlavičce), takže je možné ověřit platnost hesla dříve, než při zkusmém použití dešifrovaných dat. Díky použití UUID se lze na oddíl jinde odkazovat právě pomocí něj, což zvyšuje flexibilitu použití.

9.2.1 Správa klíčů

Jak již bylo řečeno, rozšíření LUKS dává možnost použít až 8 klíčů. Pro každý klíč existuje blok dat, kde je v zašifrované podobě uložen hlavní klíč. Tento blok dat je šifrován klíčem, který se získá ze zadaného hesla (tím se rozumí samotné heslo ale i klíč z USB tokenu) a hodnoty salt uložené v hlavičce funkcí PBKDF2-HMAC-SHA1. Jako generátor náhodných čísel je použito systémové zařízení `/dev/urandom`.



Dešifrovaný blok hlavního klíče je ještě zpracován pomocí funkce AF-split (případně AF-merge), originálně nazvané „anti-forensic information splitter“. Cílem funkce je, aby změna jediného bitu v bloku dat, kde je uložen hlavní klíč, vedla k získání úplně jiného klíče. Blok dat každého klíče, kde je uložen hlavní klíč, vypadá následovně:

úsek 1	úsek 2	...	úsek $n-1$	hlavní klíč

Výchozí hodnota pro počet úseků je 1000. Prvních 999 úseků obsahuje náhodně vygenerovaná data a poslední úsek je hlavní klíč, všechny úseky jsou stejně velké.

Funkce AF-split (resp. AF-merge) funguje tak, že každý úsek je vyplněn náhodnými daty. Tato data jsou vyxorována předchozím hashem (nulami u prvního bloku) a zahashována SHA1. Poslední blok je hlavní klíč xorovaný výsledným hashem. Funkce AF-split do takto zpracovaných dat na poslední pozici klíč uloží, zatímco AF-merge klíč z poslední pozice získá.

Při vytváření nového šifrovaného zařízení je na začátek zařízení umístěna hlavička spolu s jedním klíčem. Celý proces vypadá následovně:

1. Vygenerování hlavního klíče
2. Vygenerování soli pro PBKDF2 (mkDigestSalt)
3. Výpočet hashe hlavního klíče pomocí PBKDF2 (mkDigest)
4. Určení počtu iterací PBKDF2 tak, aby výpočet trval na konkrétním stroji jednu vteřinu (keyblock.passwordIterations)
5. Vygenerování soli pro klíč (passwordSalt)
6. Získání odvozeného klíče z hesla pomocí funkce PBKDF2
7. Vytvoření bloku dat klíče pomocí funkce AF-split
8. Zašifrování bloku dat klíče na disk. Zašifrování se provádí se stejnými parametry, jaké jsou určeny v hlavičce, a pomocí klíče odvozeného z hesla.
9. Zapsání hlavičky na zařízení

Otevření šifrovaného zařízení:

1. Získání odvozeného klíče z hesla pomocí funkce PBKDF2
2. Dešifrování bloku dat klíče.
3. Získání hlavního klíče z bloku dat klíče pomocí funkce AF-merge
4. Výpočet hashe hlavního klíče pomocí PBKDF2
5. Porovnání získaného hashe hlavního klíče s hodnotou uloženou v hlavičce (mkDigest)
6. Pokud si hodnoty odpovídají, je s hlavním klíčem vytvořeno dešifrované mapované zařízení. Jinak je celý proces opakován pro všechny použité klíče.

V následujících částech se používá funkce nalezení pozice klíče. Tato funkce je stejná jako otevření zařízení, jen s tím rozdílem, že po ověření platnosti hlavního klíče je vráceno číslo klíče. K vytvoření dešifrovaného mapovaného zařízení nedojde.

Přidání klíče probíhá tak, že se získá hlavní klíč zadáním libovolného již platného klíče a postupuje se obdobným způsobem, jako při vytváření klíče u nového šifrovaného zařízení, konkrétně od kroku 4.

Při odebírání klíče se buď rovnou odebere klíč zadaný číslem klíče nebo klíč, který odpovídá zadanému heslu. Po odebrání klíče jsou jeho data několikrát (aktuálně 39-krát) přepsána střídavě náhodnými a přednastavenými daty.

Aktuálně nástroj cryptsetup nepodporuje přímo změnu klíče. Klíč je nutné odebrat a přidat nový klíč. Nevýhodou je, že pokud je obsazeno všech 8 pozic klíčů a uživatel zná jen jeden klíč, není schopen takto změnit svůj klíč, protože pro přidání je nutné nejprve uvolnit pozici jeho odebráním, a tak již pak není splněna podmínka přidání klíče, že uživatel zná alespoň jeden platný klíč. Pokud jsou také obsazeny všechny pozice klíče, tak při ověřování se ověřují všechny pozice. Je-li zadáno chybné heslo nebo je-li zadáno heslo pro poslední klíč, trvá celá operace 8 vteřin.

Jako heslo je programu předhozeno pole bajtů. Program nerozlišuje mezi zadáním krátkého hesla a klíče v souboru. Cryptsetup nepodporuje přímo používání zašifrovaného klíče jako hesla, to je nutné provést pomocí jiného programu, jehož výstup se přesměruje na vstup cryptsetupu.

PBKDF2

Při odvozování klíče je zde používána funkce PBKDF2. Tato funkce je doporučena ve standardu PKCS (Public Key Cryptography Standards) číslo 5. Přesnou specifikaci funkce PBKDF2 (Password-Based Key Derivation Function 2) lze nalézt například v RFC 2898 nebo v PKCS #5 [13].

Heslo se jako klíč nedoporučuje používat přímo, protože se nejedná o příliš náhodnou sekvenci bitů. Proto se nejprve z hesla vyrobí klíč a to netriviálním způsobem, který významně zpomaluje útok hrubou silou.

Klíč je složen z několika bloků za sebou oříznutých na požadovanou velikost klíče. Nejprve se pseudonáhodný algoritmus (HMAC) inicializuje se zadaným heslem. Každý blok klíče je pak získán tak, že se vypočítá hash ze soli+čísla bloku, kde číslo bloku je 32-bitové big-endian číslo. Výsledná hodnota (blok klíče) se získává iterativně opakovaně xorováním s hashem dosavadního hashe. Toto se děje iterativně tolikrát, dokud operace trvala dostatečně dlouho.

Funkce PBKDF2 využívá algoritmus HMAC [7]. Tento algoritmus funguje následovně:

1. K úseku hesla jsou doplněny nuly, pokud je úsek menší, než je délka hashe. Pokud je heslo delší, je nahrazeno hashem hesla.
2. Provede se xorování úseku hesla s hodnotou 0x36
3. K výsledku se připojí zpráva a vše se zahashuje
4. Provede se opět xorování hesla doplněného nulami, tentokrát s hodnotou 0x5c.
5. K výsledku se připojí předchozí hash a vše se zahashuje.
6. Výsledek hashování je výsledkem algoritmu HMAC.

Tento algoritmus má zároveň jako jeden z parametrů typ použitého hashovacího algoritmu. V případě cryptsetupu se jedná o SHA1.

Kapitola 10

Tvorba uživatelského nástroje

10.1 Specifikace vlastností

Prakticky jediným nástrojem pro správu šifrování s dm-crypt/LUKS je program cryptsetup. Ten však není úplně uživatelsky přívětivý. Existuje několik různých nadstaveb pro cryptsetup, ale jejich rozsah schopností bývá nevelký. V další části práce proto bude vytvořen uživatelský nástroj pro OS Linux s šifrováním pomocí dm-crypt/LUKS pro správu šifrování. S následujícími požadovanými schopnostmi:

- vytvoření šifrovaného oddílu (nebo disku či kontejneru)
- možnost použití USB tokenu
- změna šifrovacího algoritmu
- přidání, odebrání či změna hesla nebo klíče
- kompletní přešifrování jiným hlavním klíčem

Zejména schopnost přešifrování s jinými parametry na stejném místě a odolně proti poruše je vlastnost, kterou, pokud je mi známo, žádný jiný nástroj zatím neumí. Nástroj bude mít dvě rozhraní a to grafické (s použitím knihovny Qt 4) a textové. Zejména grafické rozhraní bude co nejjednodušší pro usnadnění ovládání uživatelům, kteří v oblasti kryptografie nemají žádné znalosti. Výsledný nástroj bude dostupný v balíku rpm pro distribuci Fedora.

10.2 Způsob implementace

Program bude rozdělen do tří částí:

1. Hlavní část obsahující veškerou funkcionalitu
2. Grafické uživatelské rozhraní
3. Textové uživatelské rozhraní

Kompilace je prováděna pomocí sestavovacího nástroje cmake. Kompletní instrukce pro zkompilování a instalaci se nachází v souboru INSTALL v archivu se zdrojovými kódy. Ke kompilace je nutný nástroj cmake, překladač C++ (např. g++), knihovna Qt 4 a knihovna OpenSSL. Případně je možné využít i balík rpm s binární podobou.

10.2.1 Hlavní část

Program provádí správu šifrování s rozšířením LUKS, je založen na knihovně cryptsetup. Bohužel v současné době knihovna nemá použitelné API a jedná se spíše o oddělenou část programu cryptsetup. Knihovna například nepodporuje přímé zadání hesla — to si chce vždy přečíst ze souboru nebo přímo z konzole. Proto byl tedy nejprve vytvořen patch upravující knihovnu tak, aby ji bylo možné použít. Tento patch by měl být po domluvě s autorem cryptsetupu zahrnut do jeho příští verze. Jelikož je však třeba knihovnu používat již nyní, obsahuje program vlastní kopii této knihovny. Závislost na této upravené verzi by měla být v budoucnu odstraněna.

Základní funkce pro správu šifrování byly zahrnuty do třídy cLuksDevice. Ta obsahuje následující hlavní funkce:

- vytvoření šifrovaného zařízení — zdrojové zařízení předvyplní/přepíše náhodnými daty, vytvoří na zařízení hlavičku LUKS, připojí jej a na šifrovaném zařízení vytvoří souborový systém.
- přidání klíče — klíčem se rozumí heslo, klíčem nebo heslem chráněný klíč. U klíče chráněného heslem je využito knihovny OpenSSL pro šifrování klíče algoritmem AES.
- odebrání klíče
- otevření šifrovaného zařízení
- uzavření šifrovaného zařízení

Z této základní třídy je odvozena třída cReencryptLuksDevice, která obsahuje všechny metody potřebné pro přešifrování.

Další třídy jsou:

- cAutodeleteString — pole charů inicializované buď polem nebo proměnnou QString, které se s automatickou destrukcí dealokuje a přepíše původní data nulami.
- cBFIO — třída pro I/O operace na blokovém zařízení se zarovnáním čtení a zápisu pro přímý přístup.
- cErrorHandler — předávání a správa chybových zpráv.
- cSwapSpace — třída pro správu odkládacího místa pro přešifrování.

Většina schopností je poměrně jednoduchých, takže nestojí příliš za zmínku. Při nastavování parametrů šifrování jsou již nabízeny jen kombinace parametrů, které lze použít. Původní nástroj cryptsetup totiž má nedostatek, kterým je vrácení ne příliš srozumitelného chybového hlášení při použití nefungující kombinace parametrů.

Nejpodstatnější funkce programu je samozřejmě schopnost přešifrování. Základem je otevření zařízení dvakrát. Jednou se starými parametry a jednou s novými parametry. Data se nejprve přečtou ze starého umístění a poté zapíšou na stejné místo v novém umístění. Při přečtení jsou data dešifrována a při zápisu zašifrována dle nových parametrů. Pokud se při tom zapíše pouze na místo, které je právě přečteno, jsou-li obě místa stejně velká a je-li mezi nimi nulové posunutí, nedojde nikdy k přepsání dat, která jsme ještě nečetli. Problém je jen v případě, kdy dojde k poruše, například k výpadku napájení. Přešifrování dat je k poruše celkem náchylné, protože celá operace trvá velice dlouho, a nikdo nechce kvůli tomu přijít

o svá data. Při poruše nejde bez hlubší analýzy a dávky štěstí určit, kde program skončil, a ani jestli jím naposled zapsaná data jsou v pořádku. Proto se k práci používá odkládací místo a log zaznamenávající, jaká je právě prováděná činnost. Z těchto informací je poté možné nalézt místo, kde program skončil, a co má dělat dál. Místo pro log a odkládací prostor se získá v prostoru hlavičky. Při přešifrování je vždy vytvořen nový hlavní klíč a tak jsou ostatní hesla než to, které se právě použilo, neplatná. Jejich místo se tedy využije k uložení informačního bloku (informace a klíč k otevření zařízení původním šifrováním), bloky s informacemi o pozici logu a odkládacího prostoru a samotný log a odkládací prostor.

Místo pro odkládací prostor je však omezené a tak je použito vícekrát. Ještě vícekrát je ale použit log — klidně i více jak několik deset tisíckrát. Aby se zabránilo nadměrnému opotřebování stejného místa, dochází po několika přepisech k posunutí logu a odkládacího prostoru dopředu do místa, které již bylo přešifrováno. Data z nových pozic jsou uložena do původních pozic. Musíme ale udržovat ještě informaci o jejich pozici, která by se stále musela přepsat příliš hodněkrát. Proto je adresa (pozice) zřetězena stylem pozice pozice pozice pozice logu a odkládacího prostoru. Počet zřetězení je vypočítán podle velikosti přešifrovávaného zařízení a tím tedy podle celkového počtu kroků.

Po přešifrování jsou vlivem stěhování odkládacího prostoru a logu některé bloky posunuty, proto se provede jejich posunutí na původní pozici.

Vše se vždy děje tak, že je možné kdykoliv provést zjištění, co se již provedlo a co se má provést dále. Pokud dojde k výpadku či poruše dříve, než jsou nastaveny a zazálohovány všechny parametry, tak po poruše se vše vrátí do původního stavu. Pokud již parametry známy jsou, tak se po poruše operace bez problému dokončí.

Aby při přešifrování nebylo možné provádět cryptsetupem na zařízení žádné operace, je na začátku operace přepsána hodnota magic v hlavičce. Cryptsetup pak pracovat se zařízením odmítne. Po skončení je hodnota magic vrácena.

Z důvodu bezpečnosti program po spuštění zamyká svoje stránky paměti, které říkají systému, že je nesmí odkládat do swapu. Zároveň vždy po skončení práce s proměnnou, která by mohla obsahovat důležitá data, je její hodnota přepsána.

10.2.2 Grafické rozhraní

Grafické rozhraní bylo zvoleno jako co nejvíce klikací. Nevyžaduje po uživateli žádné větší znalosti. Všechny operace lze provést snadno v několika krocích.

Jednotlivé části jsou od sebe odděleny, takže se po uživateli nevyžaduje příliš informací najednou. Při vybírání parametrů šifrování se vždy nabízené možnosti mění tak, aby se vybíralo jen z té kombinace, která je skutečně funkční. Nelze tedy například vybrat pro algoritmus AES velikost klíče 64 bitů.

Kvůli použití knihovny Qt pro grafickou část nelze bohužel zaručit, aby všechna místa v operační paměti, kde bylo uloženo zadávané heslo, byla před ukončením přepsána. Použití souboru jako klíče se tento problém netýká, protože je zadáváno jen jeho umístění. Samotný klíč je zpracován již mimo knihovnu Qt a všechna místa v operační paměti jsou pod kontrolou.


QCrypt
QCryptool 0.1
Create encrypted device
Select encryption parameters
algorithm: aes
block mode: xts
init.vect.: benbi
key size: 256 bits
back
next
cancel


QCrypt
QCryptool 0.1
Easy to use disk encryption manager
status
Key slot 0 ACTIVE
Key slot 1 empty
Key slot 2 ACTIVE
Key slot 3 ACTIVE
Key slot 4 empty
Key slot 5 empty
Key slot 6 empty
Key slot 7 empty
Add key
Remove key
Change key
cancel

10.2.3 Textové rozhraní

Textové rozhraní je jednodušší variantou oproti grafickému, hlavně z důvodu zachování co největší přehlednosti. I tak však uživateli nabízí dostatečný komfort. Například při připojování souboru s obrazem oddílu jen stačí zadat cestu k obrazu a program poté provede i jeho připojení jako loop zařízení, které je nezbytným krokem k získání blokového zařízení, jenž je teprve možné šifrovat.

Obdobně při uzavírání šifrovaného zařízení stačí zadat cokoliv z jména namapovaného zařízení nebo šifrovaného blokového zařízení nebo souboru s obrazdem oddílu a program si všechny ostatní informace získá sám.

Všechny parametry jsou získávány a ověřovány vždy před zahájením jakýchkoliv operací, takže při chybném zadání posledního parametru se s šifrovaným zařízením stále ještě nic nestalo. Zároveň je již pak běh plně automatický, takže veškeré časově náročné operace probíhají na konci a uživatel se tak nemusí vracet a v polovině provedených kroků zadávat další parametry. Při všech operacích, kde je to možné, je zobrazen ukazatel postupu s odhadem zbývajícího času.

Ukázka výchozího menu textového rozhraní:

```
QCryptool 0.1
Encrypted device management

actions:
        [1] Open encrypted device
        [2] Close encrypted device
        [3] Create encrypted device
        [4] Add key
        [5] Remove key
        [6] Change key
        [7] Luks device info
        [8] Change encryption

        [q] Quit

command :
```

10.3 Testování

Přešifrování dat trvá obvykle velice dlouho. Během té doby může dojít k libovolnému problému, který by normálně vedl ke ztrátě dat. Aby k tomuto nedocházelo, byl tento program navržen tak, aby dokázal výpadku v libovolném okamžiku odolat.

Při vývoji se program průběžně měnil a tak bylo nutné stále kontrolovat, jestli je program i nadále schopný se zotavit z výpadku. Vznikla proto sada testů. Každý test funguje tak, že na začátku je připraveno (vytvořeno) šifrované souborové zařízení ze souboru s obrazem oddílu. Jelikož test by zatěžoval disk nadměrným způsobem (a byl by i diskem zpomalován), je soubor s obrazem oddílu umístěn pouze v operační paměti (umístěn v adresáři /dev/shm).

Pro zajištění spolehlivosti proti chybě při vytváření šifrovaného zařízení programem jsou všechny přípravné a kontrolní operace prováděny nástrojem cryptsetup.

V každém místě přešifrování, kde dochází k zápisu, je kód schopný uměle vyvolat zhroucení. Kód testuje globální proměnnou, která určuje testovací pozici, jenž se má aktivovat.

V případě, kdy je to vhodné, je testováno zhroucení při prvním průchodu místem a zároveň i při několikátém průchodu místem.

Po poruše je spuštěna obnova, která musí šifrované zařízení vrátit do stavu, kdy je zašifrováno původním klíčem a parametry, nebo musí operaci přešifrování dokončit.

Po dokončení přešifrování je ověřeno, že je možné zařízení připojit a je zkontrolována konzistence dat. Kontrola dat se provádí vyplněním původního zařízení danými hodnotami a ověřením, že po dokončení operace jsou všechna data v pořádku a na správném místě. U testů, které neprošly, jsou vypsány debugovací informace umožňující provedení nápravy.

Při kompilaci programu dojde i k vytvoření kontrolního nástroje, takže je možné si spolehlivost předem ověřit. Testovacích pozic je celkem 54. Projití všech testů trvá zhruba jeden a půl hodiny, podle výkonu počítače. Ve finální verzi program samozřejmě úspěšně projde všemi testy.

10.4 Způsob použití

Program byl navržen tak, aby byl maximálně intuitivní. Všechny parametry lze zadat buď interaktivně nebo na příkazové řádce, takže je možné jej využít i jinými programy či skripty.

Místo uvádění, jak se program používá, je lepší říci, jak by měl být program použit. Hlavním záměrem je umožnit výměnu hlavního klíče za jiný. Současné nástroje to snadno neumožňují, proto zatím není zvykem hlavní klíč měnit (nebo třeba změnit parametry šifrování za bezpečnější při nalezení slabin u těch aktuálně používaných).

Při změně hesla je sice blok dat se zašifrovaným hlavním klíčem několikrát přepsán, ale jak bylo zmíněno v kapitole o bezpečnostních rizicích, starší klíč se může někde objevit a být použit. Právě proto by při výměně prozrazeného hesla mělo dojít k přešifrování dat jiným hlavním klíčem. I kdyby byla data změněného klíče skutečně přepsána, může se stát, že si útočník udělá kopii původní hlavičky a poté vyčkává. Až dojde k prozrazení hesla a jeho výměně, tak se znalostí starého hesla a s využitím staré hlavičky může data bez problému dešifrovat.

Kapitola 11

Závěr

Závěrem lze říct, že přestože je práce soustředěna obzvláště na šifrované souborové systémy s použitím dm-crypt/LUKS, tak se nejedná o jedinou existující možnost. Například při šifrování dat v místě, kde musí mít k počítači přístup více uživatelů může být výhodnější použít „soukromé“ šifrování pomocí EncFS.

Šifrování dat je jednoznačným přínosem pro zajištění soukromí a ochranu dat před jejich zneužitím.

Jako u většiny činností je nutné dodržovat bezpečnostní zásady, což zde platí dvojnásob. Jde zejména o uvědomování si bezpečnostních rizik uvedených ve čtvrté kapitole. Nejvíce při použití individuálně šifrovaných souborů bývá častým nedostatkem opomenutí šifrování úložiště dočasných souborů a odkládacího oddílu.

Hlavním přínosem této práce je vytvoření nástroje umožňujícího snadno vytvářet a spravovat šifrované oddíly, a zejména nová funkce pro přešifrování dat. Díky této funkci je možné přešifrovat data s novými parametry šifrování nebo třeba jen přešifrovat data novým hlavním klíčem. Změnit parametry šifrování umožňuje například použít jinou metodu šifrování bloků, pokud tu současnou již nelze považovat za dostatečně bezpečnou. Přešifrování dat novým hlavním klíčem by se ale mělo stát důležitější a nemělo by být tolik opomíjeno a zanedbáváno. Změna hesla při prozrazení toho současného je sice důležitá, ale sama o sobě nemusí být dostatečná. Na disku se může někde neúmyslně nacházet starší hlavička nebo si jen trpělivý útočník mohl udělat zálohu hlavičky a vyčkávat. Proto je při výměně klíče více než vhodné změnit i hlavní klíč. A právě toto nebylo možné do teď provádět rozumným způsobem. Buď bylo nutné data zašifrovat na jiné místo a mít tak k dispozici třeba i další velký disk, nebo bylo možné provést dešifrování na místě s vědomím rizika ztráty dat při nějaké závadě.

Další vývoj projektu by mohl zahrnovat vytvoření oddělitelné hlavičky od šifrovaného zařízení. V případě, že uživatel má klíč umístěn například na USB disku, nepřineslo by to žádné zesložitění procesu připojení šifrovaného oddílu. Zároveň by bylo možné mít v takovéto hlavičce jen jediný klíč, takže připojení by bylo výrazně rychlejší. V současné situaci jsou ověřovány všechny použití klíče, což může dobu obzvláště při překlepu znatelně prodloužit. Dalšími výhodami by byla nemožnost prokázat, že data jsou skutečně šifrovaným oddílem a ne jen pozůstatek nějakého smetí. Umožněno by bylo také snadné vytváření a rušení šifrovaných oddílů, protože data by se před i po zašifrování vešla na stejné místo.

Vytvořený program qcryptool je umístěn na https://sourceforge.net/projects/qcryptool/ , kde bude dále vyvíjen.

Literatura

[1] Cold Boot Attacks on Encryption Keys. [online], rev. 22nd February 2008, [cit. 2008-04-29].
URL http://citp.princeton.edu/memory/

[2] Comparison of disk encryption software. [online], rev. 20th December 2007, [cit. 2007-12-20].
URL http://en.wikipedia.org/Comparison_of_disk_encryption_software

[3] CryptoSafe — bezpečné on-line šifrování: proč šifrovat. [online], rev. 3. března 2006, [cit. 2007-12-20].
URL http://www.cryptosafe.cz/index.php?q=proc

[4] Devět způsobů, jak zvýšit zabezpečení notebooku při práci na cestách. [online], rev. 3. března 2006, [cit. 2007-12-20].
URL http://www.microsoft.com/cze/atwork/workspace/laptopsecurity.mspx

[5] Disk encryption theory. [online], rev. 8th December 2007, [cit. 2007-12-20].
URL http://en.wikipedia.org/wiki/Disk_encryption_theory

[6] eCryptFS homepage. [online], rev. 15th September 2007, [cit. 2007-12-20].
URL http://ecryptfs.sourceforge.net/

[7] HMAC: Keyed-Hashing for Message Authentication. [online], February 1997, [cit. 2008-04-29].
URL http://www.ietf.org/rfc/rfc2104.txt

[8] IEEE P1619/D16 Standard for Cryptographic Protection of Data on Block-Oriented Storage Devices. [online], rev. May 2007, [cit. 2008-04-29].
URL http://grouper.ieee.org/groups/1619/email/pdf00086.pdf

[9] IEEE P1619 — Wikipedia. [online], rev. 21st December 2007, [cit. 2008-01-26].
URL http://en.wikipedia.org/wiki/IEEE_P1619

[10] LUKS on disk format. [online], rev. 18th February 2006, [cit. 2008-04-29].
URL http://luks.endorphin.org/LUKS-on-disk-format.pdf

[11] One in ten laptops stolen. [online], rev. 3rd March 2006, [cit. 2007-12-20].
URL http://www.cbronline.com/article_feature.asp?guid=C256F57D-6C54-4CA5-8181-142141ACC0AC

[12] P1619: how seriuous is leak of K2? [online], rev. 2nd June 2006, [cit. 2008-01-26].
URL http://grouper.ieee.org/groups/1619/email/msg00962.html

[13] PKCS #5 v2.1: Password-Based Cryptography Standard. [online], rev. 5th October 2006, [cit. 2008-04-29].
URL ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-5v2/pkcs5v2_1.pdf

[14] Animal rights activist hit with RIPA key decrypt demand. [online], rev. 14th November 2007, [cit. 2007-12-20].
URL http://www.theregister.co.uk/2007/11/14/ripa_encryption_key_notice/

[15] Souborový systém. [online], rev. 3. března 2006, [cit. 2007-12-20].
URL http://cs.wikipedia.org/wiki/Souborovy_system

[16] Steganografie. [online], rev. 25. října 2007, [cit. 2007-12-20].
URL http://cs.wikipedia.org/wiki/Steganografie

[17] TrueCrypt — Journaling File Systems. [online], rev. 25. října 2007, [cit. 2007-12-20].
URL http://www.truecrypt.org/docs/?s=journaling-file-systems

[18] TrueCrypt homepage. [online], rev. 10th May 2007, [cit. 2007-12-20].
URL http://www.truecrypt.org/

[19] Watermarking attack. [online], rev. 14th March 2007, [cit. 2007-12-20].
URL http://en.wikipedia.org/wiki/Watermarking_attack