VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

OBFUSKAČNÍ TECHNIKY RANSOMWARE

RANSOMWARE OBFUCATION TECHNIQUES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. JERGUŠ JACKO**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. MATEJ KAČIC, Ph.D.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2018/2019

Zadání diplomové práce

Student: **Jacko Jerguš, Bc.**
Program: Informační technologie Obor: Bezpečnost informačních technologií
Název: **Obfuskační techniky ransomware**
Ransomware Obfucation Techniques
Kategorie: Bezpečnost

Zadání:

1. Prostudujte techniky obfuskace využívané v malware a analyzujte způsoby detekce ransomwaru v antivirových nástrojích.
2. Na základě analýzy navrhněte nové techniky obfuskace založené na entropii dat, diskutujte složitost jejich implementace.
3. Navrhnuté metody implementujte tak, aby je bylo možné použít pro otestování stávajících antivirových řešení.
4. Metody otestujte na dostupných antivirových řešeních. Metodiku testování konzultujte s vedoucím.
5. Diskutujte možnosti dalšího rozšíření.

Literatura:

- A. Liska, T. Gallo: Ransomware: Defending Against Digital Extortion
- D. Khadraoui: Advances in Enterprise Information Technology Security
- Dle doporučení vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- První dva body zadání.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Kačic Matej, Ing., Ph.D.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 22. května 2019
Datum schválení: 1. listopadu 2018

Abstrakt

Táto práca sa snaží navrhnúť, implementovať a poukázať na nové techniky obfuskácie činnosti ransomvéru s využitím princípov entropie dat, ktoré nespadajú do detekčných možností známych anti-ransomvérových a anti-vírových nástrojov. Navrhované techniky sa zameriavajú na zmenu činnosti ransomvéru vo fáze znehodnotenia (šifrovanie alebo obfuskácia) súborov na napadnutom systéme.

Abstract

This master's thesis seeks to design, implement, and point out new techniques for obfuscation of ransomware activity using the entropy principles of data that do not fall within the detection capabilities of known anti-ransomware and anti-virus tools. The proposed techniques are aimed at changing the ransomware activity in the downgrading phase (encryption or obfuscation) of files on the infected system.

Kľúčové slová

ransomvér, obfuskačné techniky, entropia, white-box kryptografia

Keywords

ransomware, obfuscation techniques, entropy, white-box cryptography

Citácia

JACKO, Jerguš. *Obfuskační techniky ransomware*. Brno, 2019. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Matej Kačic, Ph.D.

Obfuskační techniky ransomware

Prehlásenie

Čestne prehlasujem, že som vypracoval túto diplomovú prácu samostatne pod vedením pána Ing. Mateja Kačica, Ph.D. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Jerguš Jacko
23. mája 2019

Poďakovanie

Veľmi rád by som poďakoval vedúcemu mojej diplomovej práce Ing. Matejovi Kačicovi, Ph.D. za jeho odborné rady a pripomienky. Poďakovanie si taktiež zaslúži moja rodina, priateľka a kamaráti za ich trpezlivosť a pomoc.

Obsah

Kapitola 1

Úvod

Ransomvér je škodlivý softvér alebo malvér, ktorý zašifruje informácie v počítači používateľa alebo dokonca aj celé siete. Prístup k týmto súborom uvoľní až po zaplatení poplatku – výkupného. Po zaplatení sa súbory odomknú a používateľ k nim opäť získa prístup. Občas sa prístup k súborom nezíska ani po zaplatení výkupného a obeť zostane s jedinou možnosťou, a to je čakať na dekryptory vyvíjané anti-vírovými spoločnostami na základe nájdených nedostatkov v architektúre ransomvéru (čo sa však nie vždy podarí). Ransomvér sa rýchlo stal najziskovejším typom malvéru v histórii. Kyberzločinci zarábajú ročne miliardy a hrozba exponenciálne narastá. Existuje niekoľko rôznych spôsobov, ako ransomvér môže infikovať počítač obete. Jednou z najbežnejších metód je v súčasnosti škodlivý spam, alebo malspam, čo je nevyžiadaný e-mail, ktorý sa používa na doručovanie škodlivého softvéru. E-mail môže obsahovať infikované prílohy,ako napríklad súbory PDF, dokumenty programu Word, alebo odkazy na škodlivé webové stránky.

Základná myšlienka ransomvéru bola predstavená vo forme krypto víru v roku 1995. Odvtedy sa však už viac ako desať rokov označuje za podstatnú a často diskutovanú tému. V roku 2017 sa ransomvér stal realitou pri niekoľko medializovaných prípadoch, kde ohrozil dôležité počítačové systémy na celom svete. Typický prípad ransomvéru s globálnym dopadom na spoločnosť je CryptoLocker a WannaCry. Tento typ malvéru zašifruje súbory obetí a vyžaduje od nich platbu pre dešifrovanie. Nakoľko využíva kryptografiu s verejným kľúčom, kľúč na obnovu nemožno nájsť v stopách ransomvéru v systéme obete. Pri dobre vyvinutom ransomvéri platí, že ak je systém obete infikovaný, nie je ho možné obnoviť bez zaplatenia "výkupného". Výskumní pracovníci spoločností, vyvíjajúci antivírové nástroje a odborníci v oblasti bezpečnosti sietí prichádzajú s rôznymi metódami riešenia tejto hrozby. Avšak je zrejmé, že kryptografická obrana je nedosiahnuteľná, pretože znovuzískanie súborov obete je výpočtovo rovnako náročné ako prelomenie princípu systému šifrovania s verejným kľúčom. Riešenia sa spoliehajú na detekciu ransomvéru ešte pred jeho spustením, na detekciu ransomvéru v jednotlivých fázach jeho činnosti, alebo na prípadné nedostatky v implementácií, ktoré pomáhajú škody tejto hrozby odvrátiť a navrátiť pôvodný stav systému.

Táto práca sa snaží navrhnúť, implementovať a poukázať na nové techniky obfuskácie činnosti ransomvéru s využitím princípov entropie dat, ktoré nespadajú do detekčných možností známych anti-ransomvérových a anti-vírových nástrojov. Navrhované techniky sa zameriavajú na zmenu činnosti ransomvéru vo fáze znehodnotenia (šifrovanie alebo obfuskácia) súborov na napadnutom systéme. Práve v tejto fáze sa vyskytuje najviac faktorov o ktoré sa opierajú anti-vírové nástroje, čo využívajú navrhované metódy vo svojej architektúre.

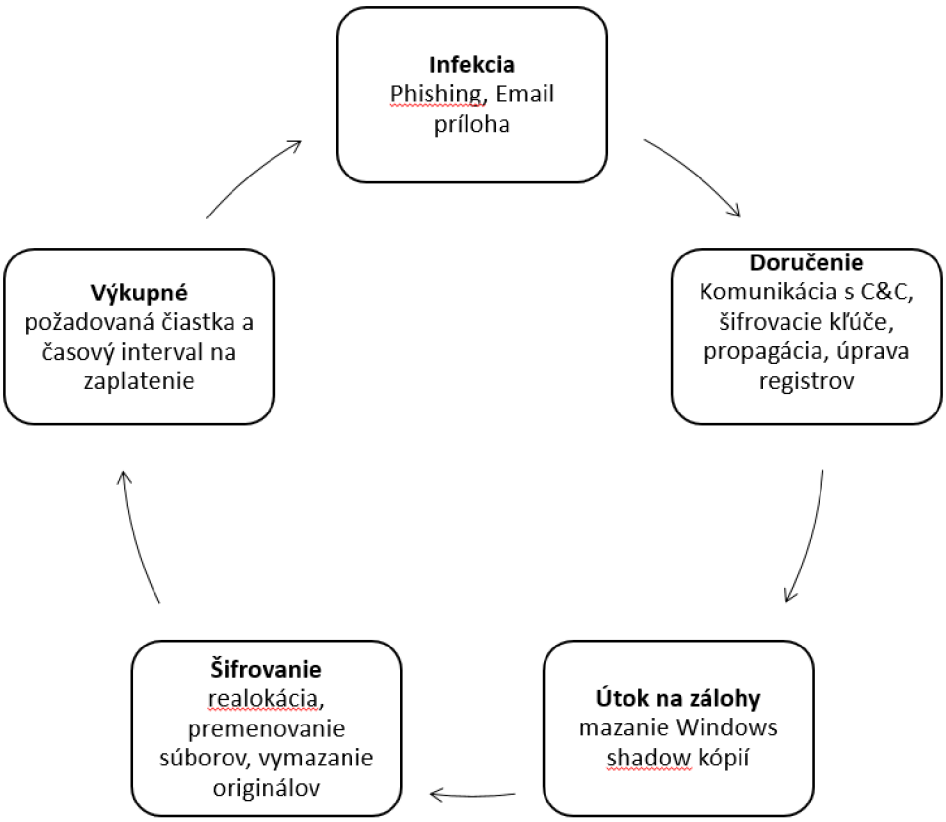
Práca je členená do štyroch kapitol. Pomerne obšírnejšia, prvá kapitola 2 predstavuje čitateľovi malvér typu ransomvér, jeho typy, detekčné možnosti ransomvéru antivírovými nástrojmi na báze statickej a behaviorálnej analýzy a obfuskačné metódy využívané vo fáze šifrovania. V kapitole 3 sú popísané metódy obfuskácie činnosti ransomvéru, ktoré boli navrhnuté na základe znalosti analyzovaných z prvej kapitoly. Tieto metódy využívajú princípy entropie dat, whitebox kryptografie a modernej kryptografie. Nasledovná kapitola 4 obsahuje popis implementácie navrhovaných metód do funkčného celku ransomvéru. Funkčný celok je implementovaný s architektúrou prispôsobenou pre beh na platforme Windows. Kapitola implementácie navrhovaných metód taktiež zahŕňa popis implementácie konvenčného ransomvéru. Záverečná kapitola 5 popisuje metodiku testovania implementovaných metód voči rôznym anti-vírovým a anti-ransomvérovým riešeniam a sumarizuje získané výsledky testovania ako aj efektivitu použitých metód v porovnaní s detekciou konvenčného typu ransomvéru.

Kapitola 2

Analýza a detekcia ransomvéru

Ransomvér je špeciálny typ škodlivého softvéru, ktorý zabraňuje obetiam prístup k vlastným údajom a súborom na vlastných počítačoch alebo iných zariadeniach s cieľom získať výkupné. V posledných rokoch sa stal dominantnou formou cyber-útokov, najmä v dôsledku rastu počtu kryptomien. Vzhľadom na ich anonymitu, útočníci môžu dostávať výkupné bez toho, aby boli identifikovaní. Malwarebytes, spoločnosť, ktorá sa zaoberá bezpečnosťou softvérových systémov, vydala štatistiky, ktoré potvrdzujú popularitu žiadania výkupného. Ransomvér nielenže šifruje súbory na pracovnej stanici, je tiež schopný prechádzať sieť a šifruje súbory uložené na mapovaných aj nemapovaných sieťových diskoch. To môže viesť aj ku katastrofálnej situácii, kde jeden infikovaný počítač (pracovná stanica) môže zastaviť prevádzku oddelenia alebo dokonca celej organizácie. Akonáhle sú súbory šifrované, hackeri prídu s pokynmi ako platiť za odomknutie súborov. Často používaná metóda je zvyšovanie výkupného po určitom čase. Moderné typy tohto škodlivého softvéru vyžadujú výkupné vo forme kryptomien. Vzhľadom na ich anonymitu, útočník môže získať výkupné bez toho, aby bol identifikovaný.

Proces ransomvéru obsahuje päť fáz: infekciu, doručenie, útok na zálohy, šifrovanie a upozornenie používateľa o vykonanom ransomvéri s údajmi na zaplatenie a dešifrovanie (viď. obrázok 2.2). Počas fáze infekcie sa ransomvér dostane do hostiteľského systému. Najčastejšie je vstupný bod do systému povolený práve používateľom, ktorý neúmyselne umožní prístup, napríklad prostredníctvom phishingového útoku alebo kliknutím na podvrhnutý odkaz. Útočníci jednoducho využívajú užívateľskú náchylnosť k útoku kvôli nedostatku vedomostí a nedostatku vzdelania.



Obr. 2.1: Životný cyklus malvéru typu ransomvér.

Počas ďalšej fáze útoku ransomvéru sa iniciuje protokol o infekcii. Ransomvér vytvára spustiteľný súbor a adaptuje sa v hostiteľskom systéme. Ransomvér mení kľúče registra, aby sa zabezpečil, že infekcia zostane dokonca aj po reštarte alebo resetovaní systému. V tomto momente je malvér pripravený a dôjde k šifrovaniu dát podľa definovaného inkubačného obdobia. Ďalej sa ransomvér snaží odstaviť zálohovací systém. Ak je úspešný, tak obmedzí schopnosť obete zotaviť sa a docieli tým pákový efekt na to, aby sa obeť dostala do pozície pre jedinú možnosť vyplatenia výkupného, pretože vzniká škoda na podnikaní spôsobená stratou údajov. Počas fázy šifrovania sa dáta šifrujú, ransomvér vytvára šifrovací kľúč s časom šifrovania, ktorý sa mení na základe faktorov, ako je počet pripojených zariadení, veľkosť súboru a návrh siete. Nakoniec ransomvér zobrazuje upozornenie pre používateľa. Je to oznámenie a oznamuje sumu výkupného a platobné pokyny.

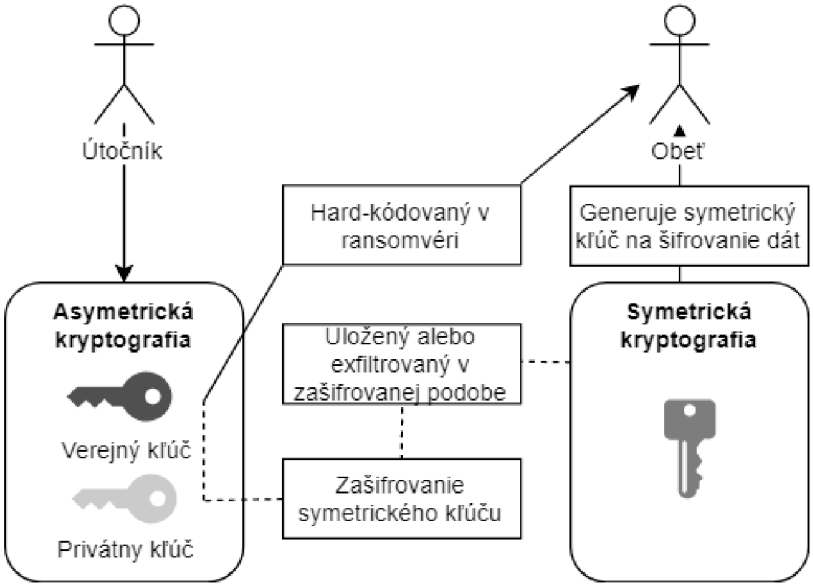
2.1 Typy ransomvéru

Cieľom ransomvéru je žiadanie výkupného prostredníctvom zabránenia prístupu k dátam alebo zablokovaniu funkčnosti systému obete. Kryptografický ransomvér dosiahne tento cieľ šifrovaním súborov pomocou silnej kryptografie, zatiaľ čo drží šifrovacie kľúče pre prinútenie obete zaplatiť výkupné s cieľom opätovného prístupu k svojim súborom. Iný variant, *locker*, dosiahne tento cieľ prostredníctvom prevzatia kontroly systému obete a obmedzuje

funkčnosť. V tomto prípade, používateľské údaje sú nedotknuté, ale infikovaný systém sa stáva nepoužiteľný.

2.1.1 Kryptografický ransomvér (CGR)

Kryptografcký ransomvér (CGR) využíva kryptografické algoritmy pre šifrovanie cieľových súborov. Malvér šifruje dáta v tichosti a po ukončení šifrovacieho procesu je obeť informovaná, že všetky jeho dáta sú zašifrované môžu byť rozšifrované iba v prípade, že zaplatí výkupné. Skoršie verzie CGR ukladali kľúč pre dešifrovanie na počítači obete, čo umožňovalo získanie tohoto kľúča pomocou reverzného inžinierstva. Novšie iterácie malvéru najskôr kontaktujú command and control (C&C) server a následne začnú so šifrovaním, kde šifrovací kľúč je uchovaný na serveri namiesto počítača obete. Skoršie verzie Critroni / CTB Locker malvéru zašifrovali súbory až po kontaktovaní C&C servera, avšak tento nedostatok umožnil bezpečnostným tímom prevenciu na základe auditu komunikácie a zablokovanie komunikácie malvéru pred tým, ako bol malvér schopný svoj útok začať. Ransomvér vývojári opravili tento nedostatok so šifrovaním súborov ešte pred komunikáciou s C&C serverom.



Obr. 2.2: Schéma využitia duálneho šifrovania v CGR (crypto-ransomvéri).

Ochrana symetrického kľúča sa líši, ale väčšinou je verejný kľúč CGR vložený do malvéru alebo je stiahnutý z C&C servera, ktorý sa potom používa na priame šifrovanie symetrického kľúča alebo zohráva úlohu v šifrovacom procese ransomvéru. Po absolvovaní šifrovania je symetrický kľúč často exfiltovaný alebo uložený v napadnutom systéme.

2.1.2 Ne-Kryptografický ransomvér (NCR)

Niektoré typy ransomvér nevyužívajú kryptografiu. V týchto inštanciách malvéru je payload jednoducho aplikácia navrhnutá na obmedzenie interakcie s počítačovým systémom pomocou zablokovania obrazovky alebo modifikácie MBR (Master Boot Record, alebo aj tabuľky s partíciami). Tento typ ransomvér je považovaný za relatívne slabý a škoda môže byť odvrátená bez zaplatenia výkupného.

2.1.3 Ransomvér založený na šifrovacom systéme privátneho kľúča (PKR)

Existujú rodiny ransomvéru, ktoré využívajú šifrovacie systémy ako klasické šifry, DES rodiny alebo moderné šifrovacie systémy založené na využití privátneho kľúča pre šifrovanie cieľových súborov na počítači obete. CryptorBit je príklad, ktorý využíva vlastne navrhnutý klasický šifrovací systém podobný polyalfabetickej substititúcií šifier v prvých 512 bytoch cieľových súborov [37].

2.2 Techniky obfuskácie

Obfuskácia je činnosť vytvárania nezrozumiteľnej softvérovej realizácie pomocou sekvencie transformácií, zatiaľ čo sa uchováva programová sémantika. Vendori pôvodne využívali obfuskáciu na ochranu intelektuálneho majetku vyvíjaného softvéru. Avšak autori malvéru využívajú obfuskáciu na zakrytie škodlivého kódu v binárnych programoch. Obfuskáciou sa malvér môže vyhnúť technikám signaturálnej detekcie, čo jeden z najstarších postupov v boji s malvérom.

Obfuskovaný malvér môže byť rozdelený do štyroch kategórii: *šifrovaný*, *oligomorfný*, *polymorfný* a *metamorfný* malvér. Členovia prvého typu šifrujú segmenty škodlivého kódu v binárnych programoch a dešifrujú ich počas behu. Toto zahŕňa funkciu pre dešifrovanie vložené do tela malvéru pre spracovanie škodlivého kódu. Avšak ani malvérové systémy by takto stále rozpoznali funkcie na dešifrovanie a a identifikovali by škodlivý softvér. Druhý typ, oligomorfný malvér, nesie set zašifrovaných dekryptorov v binárnom dátovom segmente a mení dekryptory v každej generácií. Avšak počet dekryptorov je limitovaný a preto ich napokon všetky anti malvérové softvéry identifikujú. Na druhej strane, polymorfný malvér mutuje jeho spôsoby dešifrovania náhodne a preto sa vyhýba detekcii na základe signatúr. Spôsoby mutácie zahŕňajú vkladanie mŕtveho kódu, register reassignment, menenie poradia sub-rutín programu, substitúciu inštrukcií, transpozíciu kódu a integrácie. Napríklad vkladanie mŕtveho kódu je praktika vkladania častí kódu, ktoré nemajú žiadny efekt na funkciu softvéru.

Všetky vyššie zmienené techniky obfuskácie sú využívané pre obfuskáciu malvéru typu ransomvér, avšak v tejto práci sa zameriame na obfuskáciu ako proces znehodnotenia dát obete ransomvéru.

Kryptografický ransomvér (CGR), ktorý vidíme dnes, používa pokročilé algoritmy na šifrovanie súborov v zariadeniach alebo sieti ktoré fungujú na dvoch základných princípoch: symetrický kľúč a asymetrické šifrovanie. Pre vydieranie má každá metóda výrazné výhody a nevýhody. Niektoré zložitejšie varianty využívajú obidva druhy šifrovania na prekonanie slabých stránok druhého.

2.2.1 Symetrická kryptografia

Malvér, ktorý používa šifrovanie pomocou symetrických kľúčov, často využíva samotné zariadenie na vygenerovanie kľúča, ktorý je nevyhnutný v procese šifrovania. Použitie šifrovania so symetrickým kľúčom zabezpečuje, že sa používa menej systémových prostriedkov, zatiaľ čo škodlivý softvér šifruje súbory. Toto minimalizovanie využitia výkonu pomáha znížiť možnosti detekcie pomocou softvéru na sledovanie procesov a efektívne využíva CPU zdroje infikovaného systému. Použitie krátkeho kľúča vygenerovaného v zariadení môže minimalizovať náklady na zdroje hosťa a maximalizovať objem súborov, ktoré šifruje. Ďalšou výhodou používania šifrovania so symetrickým kľúčom je to, že pre každý infikovaný systém

je vytvorený jedinečný kľúč, a preto výkupcovia ransomvéru môžu určiť, ktoré nasadenia ransomvéru boli úspešné a ktoré neboli. Navyše umožňuje, aby šifrovací proces prebehol online alebo offline (bez C&C). Avšak offline prístup vyžaduje, aby sa počítač vrátil späť online a poslal kľúč k protivníkovi pre jeho využitie na proces výkupného.

Hlavnou nevýhodou symetrického šifrovania je to, že používateľ vytiahne kľúč z aktívnej pamäte a použije ho na dešifrovanie súborov v systéme, keď je offline. To znamená, že pri napadnutí variantom ransomvéru, ktorý používa symetrické šifrovanie kľúčov, je úplne možné, aby si obeť dešifrovala súbory sama.

Obr. 2.3: Šifrovanie dát symetrickou kryptografiou.

2.2.2 Asymetrická kryptografia

Pri tejto metóde útočník využíva v procese šifrovania verejný a súkromný kľúč. Verejný kľúč sa používa na infikovanom systéme na šifrovanie súborov a súkromný kľúč sa používa na dešifrovanie súborov. Tieto kľúčové páry znemožňujú používať forenznú pamäť na dešifrovanie súborov. Namiesto toho sa treba spoliehať na brute-force útoky, slabé miesta v šifrovacích algoritmoch, zaplatenie výkupného alebo pripravenosť na možnosť tohto druhu útok. Pre ransomvér, ktorý využíva šifrovanie pomocou asymetrických kľúčov existujú dva hlavné typy šifrovania: vstavaný (hardkódovaný v malvéri) verejný kľúč a stiahnutý verejný kľúč.

Obr. 2.4: Šifrovanie dát asymetrickou kryptografiou.

V ransomvéri, ktorý využíva vstavaný verejný kľúč, je metodológia pomerne jednoduchá a malvér môže začať šifrovať dáta aj ak je počítač online alebo offline. Nevýhodou tejto techniky je, že musí byť generovaný nový verejný kľúč pre každý útok.

V prípade ransomvéru, ktorý používa stiahnutý verejný kľúč, proces šifrovania nemôže začať pokiaľ počítač nie je späť online a nie je schopný komunikovať so serverom útočníka pre stiahnutie verejného kľúča. Výhodou útočníka je, že útočník môže pri každej infekcii týmto typom ransomvéru využiť rôzne páry kľúčov.

Ďalšou hlavnou výhodou, alebo teda výhodou pre útočníka, metódy asymetrickej šifrovania je to, že vo svojom šifrovacom algoritme používa omnoho väčšie prvočísla, počnúc 2048 bitmi a viac.

Vyššie bolo oddelene rozobraté šifrovanie pomocou asymetrických a symetrických kľúčov, avšak vo väčšine moderných variantov CGR sa obidva typy šifrovania používajú súčasne pre využitie silných stránok z každej metódy. Napríklad ransomvér CryptoDefense[9] používa šifrovanie AES (symetrický kľúč) na manuálne šifrovanie súborov, kde po ukončení šifrovania uloží kľúč lokálne a zašifruje ho pomocou stiahnutého verejného kľúča RSA s počtom 2048 bitov. Následne po zaplatení výkupného je koncovým užívateľom (obetiam) poskytnutý prístup k súkromnému kľúču, ktorý dešifruje lokálne uložený kľúč AES a umožňuje používateľovi dešifrovať svoje súbory.

2.3 Detekcia na základe statickej analýzy

Detekcia malvéru pomocou statickej analýzy znamená analyzovať kód aplikácie pred jeho vykonaním, pre zistenie, či je schopný akýchkoľvek zákerných činností. Ak statická analýza nájde škodlivý kód, spustiteľný súbor bude zastavený v prípade spustenia. Najbežnejší typ statickej analýzy, ktorý sa bežne používa v komerčných vírusových skeneroch, je označovaný ako analýza na základe signatúr. V analýze na základe signatúr sa extrahujú vzory (signatúry) z reťazcov kódu cieľovej aplikácie a porovnávajú sa s repozitárom známych modelov škodlivých kódov. Detekcia založená na signatúrach závisí od obrovského archívu signatúr škodlivého kódu. Toto úložisko musí byť často aktualizované, čo v žiadnom prípade nie je triviálna úloha. Komerčné vírusové skenery majú zvyčajne veľké tímy vedeckých pracovníkov v oblasti počítačovej bezpečnosti, ktoré nepretržite objavujú, vyšetrujú a extrahujú škodlivé signatúry.

2.3.1 Obmedzenia statickej analýzy

Základnou chybou detekcie založenej na signatúrach je jej neschopnosť odhaliť neznámy škodlivý softvér, ktorý ešte nebol spracovaný na signatúru. Škodlivý spustiteľný súbor je odhaliteľný až po tom, čo bol prvý krát nahlásený ako škodlivý a pridaný do úložiska škodlivých signatúr. To má za následok tri nedostatky v účinnosti statickej detekcie:

- **Neúčinné proti obfuskácií kódu** - pre oklamanie detekcie založenej na signatúrach, vývojári škodlivého softvéru používajú techniky obfuskácie kódu. Týmito technikami iteračne modifikujú malvér, kde sa každá verzia signatúry tohoto malvéru zobrazí odlišne. Toto maskovanie kódu neovplyvní zamýšľané škodlivé správanie malvéru, ale ovplyvňuje iba to, ako je staticky vnímaný detekčným systémom založeným na signatúrach.

  Pri ransomware útokoch sa na strane servera zvyčajne používajú techniky obfuskovania kódu pri vytváraní payloadu malvéru. Táto forma je označovaná ako "továreň na škodlivý softvér", ktorá automaticky generuje veľké objemy malvéru s unikátnou hash hodnotou pre každú variantu malvéru z originálneho kusu kódu. Príklad tohoto typu útoku bol videný v ransomware Cerber[8], kde bol server schopný generovať nové, unikátne vyzerajúce vzorky každých 15 sekúnd.

  Druhá forma obfuskácie kódu sa vyskytuje na strane obete, kde malvér rozbalí jedinečný variant jeho spustiteľného kódu pri každom spustení. Tento typ škodlivého softvéru je označovaný ako samo-premenlivý, nakoľko že je schopný replikovať svoje odlišné verzie. Pri zoradení podľa úrovne zložitosť vývoja, samo-premenlivý mal-

vér spadá do troch kategórií: oligomorfný, polymorfný a metamorfný[3]. V súčasnej dobe sa v samo-premenlivý ransomvér moc nerozvíja. Ako dobrý príklad samo-premenlivého malvéru sa možno odkazovať na rodinu Virlock[24], ktorý sa vyhýba statickej detekcie pomocou polymorfizmu.

- **Neefektívne proti vysokému variantu výstupu** - analýza založená na signatúrach je menej účinná v prípade typu škodlivého softvéru, ktorý má rýchle vývojové cykly a variantný výstup. Je to problematické pre detekčné systémy založené na signatúrach, pretože novo vyzerajúci ransomvér je vyvíjaný oveľa rýchlejšie ako sú nové signatúry vytvárané, testované a pridávané do úložiska škodlivých podpisov.

- **Neúčinné voči cieleným útokom** - týka sa to cielených útokov s využitím ransomvéru, kde sa vývojári rozhodnú, že odkryjú nový ransomvér na veľmi konkrétne a dobre vybrané organizácie.

2.4 Detekcia na základe dynamickej (behaviorálnej) analýzy

Skutočnosť, že ransomvér je možné ľahko definovať a zahrnúť do kategórie malvéru, znamená, že je možné vytvoriť dobre definovaný behaviorálny konštrukt, na základe ktorého môžeme predpovedať, že neznámy proces je ransomvér.

V behaviorálnom prístupe obrany proti ransomvéru systémy skúmajú správanie aplikácie a jej interakcie s prostredím, ako napríklad aktivita súborového systému, sieťové pripojenia a modifikácie komponentov operačného systému.

Analýza založená na správaní je vysoko účinná pri detekcií CGR, ktorý vykazuje vysoké charakteristiky správania pri šifrovaní súborov. Tieto charakteristiky sa nemenia z variantu ransomvéru na iný variant a ani z rodiny ransomvéru na inú rodinu. Charakteristiky správania sa dajú rozdeliť do dvoch rôznych kategórií úloh: podozrivá aktivita prípravy ransomvéru a šifrovanie dát.

2.4.1 Statická analýza vs. dynamická analýza

Rozdiel medzi statickou analýzou a behaviorálnou analýzou je možné zovšeobecniť spôsobom: v statickej analýze sú dedukcie behaviorálnych znakov vytvorené z binárneho súboru neznámeho spustiteľného súboru. Toto odvodené správanie sa potom priraďuje jednoduchým algoritmom na úrovne hrozby (to znamená bezpečné alebo škodlivé). Pri behaviorálnej analýze sú behaviorálne charakteristiky spustiteľného súboru známe až počas sledovania v reálnom čase a závery sa robia algoritmom induktívneho rozhodovania na úroveň ohrozenia. Kľúčovým rozdielom medzi statickou detekciou a detekciou založenou na správaní je bod, v ktorom sa uskutočňuje dedukcia - statická analýza odhaľuje behaviorálne vlastnosti z pozorovaného binárneho súboru, dynamické správanie vyvodzuje úroveň hrozby z pozorovaného správania. Pri detekcii založenej na správaní sa všetky spustiteľné súbory považujú za neznáme, kde zaradenie do úrovne ohrozenia závisí od spustiteľného súboru - zvyšuje to schopnosť detekcie zero-day (neznámych) útokov.

2.4.2 Podozrivá aktivita ransomvéru

Ransomvér vykazuje mnohé znaky správania ako iný škodlivý softvér, najmä tie, pri ktorých sa sám inštaluje ešte pred dodaním payloadu. Toto zdieľané správanie sa považuje za

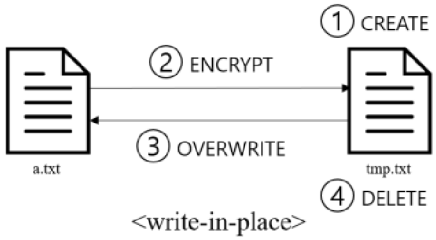
všeobecný návod na úspech, ktorý zväčša dodržujú vývojári malvéru. Môže byť zaradená do šiestich behaviorálnych vlastností:

- **Pretrvávajúce zaťaženie** - pre zaistenie úspešného dokončenia útoku, musí byť zaistené, že útok pretrvá aj počas reštartu systému.
- **Obnova systému** - pre zaistenie, že škodlivá činnosť nebude zmarená obnovou systému, malvér musí zabezpečiť akcie ako napríklad vymazanie Windows shadow kópií.
- **Stealth techniky** - malvér sa pokúša vykonať svoju činnosť tajne, aby sa vyhol tomu, že si ho všimne používateľ alebo bol detekovaný vírusovými skenermi. Bežné techniky zahŕňajú: injekciu do legitímnych procesov vykonávaných z adresára %AppData% a použitím spustiteľných súborov s rovnakým názvom ako bežné spustiteľné súbory Windows.
- **Mapovanie prostredia** - malvér môže ešte pred záškodníckou činnosťou mapovať svoje systémové prostredie. Zvyčajne to robí, aby sa zistil, či je spustený na reálnom počítača alebo v prostredí karantény, ktoré by sa ho mohlo pokúsiť analyzovať. Mapovanie prostredia sa používa aj na určenie nastavenia a politík bezpečnosti, geografickej polohy, používateľského jazyku, architektúry súborového systému a sieťových jednotiek.
- **Sieťová komunikácia** - ransomvér, ktorý vyžaduje pripojenie na C&C, zvyčajne vykonáva sťahovanie súborov súvisiacich s payloadom malvéru a komunikáciu pre ustanovenie šifrovacieho kľúča. Pre zabezpečenie neblokovania komunikácie na konkrétny C&C, vývojári využívajú rôzne taktiky, ktoré zahŕňajú generovanie náhodných názvov domén zaregistrovaných na anonymné top-level domény ako .xyz. .top a ďalšie.
- **Elevácia privilégií** - vykonávanie škodlivých aktivít môže vyžadovať prístupové práva ktoré sú vyššie ako tie, ktoré sú pridelené používateľskému účtu obete.

2.4.3 Správanie ransomvéru pri šifrovaní dát

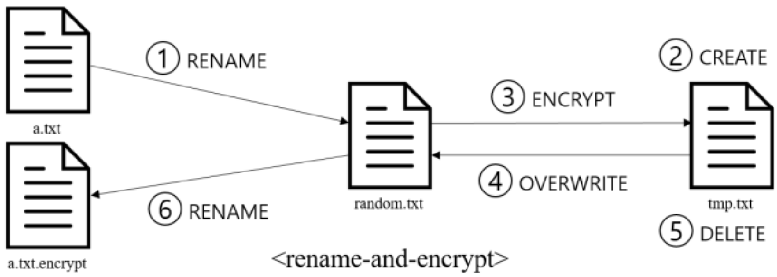
Základ CGR je jeho schopnosť transformovať hromadné množstvo dát z použiteľného stavu do nepoužiteľného stavu. Typicky, transformácia údajov ransomvérom je definovaná z pohľadu operácií nad súbormi, čo vedie k vzniku troch kategórií ransomvéru[23]:

- *Trieda A, Write-in-place,* otvorí pôvodný súbor a priamo prepíše jeho obsah so šifrovanými dátami. Zvyčajne však proces pri väčších súboroch prebieha, že vytvára dočasný súbor a šifruje cieľový súbor, ktorý sa má zašifrovať. Zapisuje šifrovaný cieľový súbor do dočasného súboru a prepíše cieľový súbor so šifrovaným dočasným súborom. Nakoniec dočasný súbor odstráni. Ak je séria niekoľkých čítaní a zápisov operácií so súbormi monitorovaná, ransomvér, ktorý používajú metódu Write-on-place, možno detegovať pomerne ľahko.



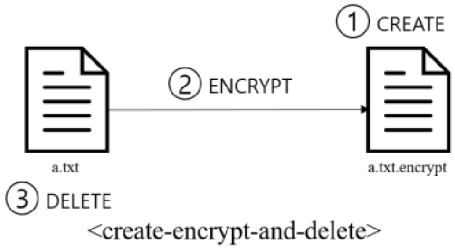
Obr. 2.5: Trieda A spôsobu šifrovania dát obete. Prevzaté z [25]

- *Trieda B, Rename-and-encrypt,* najprv presunie súbor na diskrétne miesto, šifruje súbor ako v triede A a potom presunie súbor späť na pôvodné miesto. Dočasnému súboru zmení názov na iný ako šifrovaný súbor a šifruje pôvodný súbor rovnakým spôsobom ako metóda A. Tento druh ransomvéru taktiež pridá aj ďalšiu príponu súboru za pôvodnú príponu súboru, počas zmeny šifrovaného názvu súboru na pôvodný názov súboru. Pri sledovaní zmien názvov súborov, ransomvér, ktorý používa metódu B, možno zistiť.



Obr. 2.6: Trieda B spôsobu šifrovania dát obete. Prevzaté z [25]

- *Trieda C, Create-encrypt-and-delete,* číta pôvodný súbor, šifruje jeho obsah, zapíše šifrovaný obsah do nového súboru a vymaže pôvodný súbor. Vzhľadom k tomu, že táto metóda je tak podobná metóde presúvania súboru z priečinka do iného priečinka, aj keď je súbor infikovaný ransomvérom je ťažké rozlíšiť, či bol súbor zašifrovaný ransomvérom alebo presunutý používateľom.



Obr. 2.7: Trieda C spôsobu šifrovania dát obete. Prevzaté z [25]

Ak je cieľový súbor, ktorý ransomvér chce zašifrovať, príliš veľký, je veľmi ťažké naraz zašifrovať obsahu súboru v hlavnej pamäti. Preto spôsoby *Write-in-place* a *Create-encrypt-and-delete* vytvoria najprv nový súbor. Potom tieto metódy šifrujú obsah cieľového súboru v malých prírastkoch a zapisujú do nového súboru. V metóde *Rename-and-encrypt*, proces šifrovania premenováva pôvodný súbor, ktorý sa má zašifrovať bez vytvorenia nového súboru, pre vyhnutie monitorovania a ochrany proti ransomvéru.

Podľa vyššie uvedených tried, identifikovanie ransomvéru na základe jeho spôsobov transformácie údajov vyžaduje minimálne schopnosť sledovania operácií so súbormi a schopnosť identifikovať šifrovanie dát. Takéto prístupy k detekcii ransomvéru sa osvedčili v [4], kde sa sledujú operácie so súbormi s použitím I/O požiadavok na súbory a v [23], kde je detekované hromadné šifrovanie súborov pomocou štatistických údajov, ako napríklad zmena entropie.

Behaviorálne založené detekčné metódy založené predovšetkým na detekcii hromadného šifrovania súborov môžu byť účinné, avšak bývajú náročné na zdroje. Šifrovacie opatrenia ako napríklad zmena entropie vyžaduje, aby sa entropia súboru vypočítala pre každú vykonanú operáciu zápisu aplikáciou. Okrem toho tieto operácie potrebujú sledovať operácie so súbormi pre každý súbor zvlášť počas životnosti pozorovaného procesu. Takýto prístup môže značne zhoršiť čítanie a písanie na systémový disk a mať za následok vysoké vyťaženie systému.

Vývojári ransomvéru, ktorí si uvedomujú techniky analýzy transformácie dát, využívajú jednoduché triky na maskovanie prítomnosti hromadného šifrovania súborov, ako napríklad čiastočné šifrovanie súborov (iba sekcie hlavičky) alebo saturácia systému so zápismi s nízkou entropiou, aby sa vytvorila nižšia šifrovacia stopa.

Okrem vykonávania operácií zápisu, presunu a odstránenia má ransomvér tendenciu premenovať zašifrovaný súbor. Toto premenovanie sa často vykoná s cieľom upozorniť používateľa, že došlo k útoku ransomvéru. Príkladom premenovania súboru je pridanie ďalšej prípony k pôvodnej prípone súboru, ako napríklad *nazov.xlsx* na *nazov.xlsx.ransomed*. Nákladovo efektívne riešenie na detekciu ransomvéru je identifikácia operácie premenovania súborov na prípony pridávané známym malvérom. Takáto technika je účinná iba na jednoduchom ransomvéri, ale nie na inteligentnejšie napísanom ransomvéri ako napríklad *CryptXXX*, ktorý náhodne vyberie názov súboru alebo *Spore*, ktorý si zachováva pôvodný názov súboru. Ak sú využité tieto spôsoby, tak pomer false pozitívov pri detekcií ransomvéru je zvyčajne vysoký.

2.4.4 Behaviorálna analýza s využitím strojového učenia

Behaviorálny prístup k detekcii ransomvéru vyžaduje rozhodovací algoritmus, ktorý vyhodnocuje kvantitatívnu behaviorálnu stopu bežiaceho procesu. Tá slúži ako vstup pre jednoduché binárne rozhodnutie - áno, program je bezpečný, alebo nie, je to ransomvér. Oblasť algoritmického rozhodovania je rozsiahla a dáva množstvo prístupov k riešeniu komplexných problémov. Pri detekcii ransomvéru je prístup naučený počítačom najvhodnejší na detekciu. Tento prístup využíva napríklad ani-ransomvér agent *RansomFlare*.

2.4.5 Známe nástroje

V literatúre sa nachádzajú rôzne návrhy, ktoré využívajú behaviorálnu analýzu. Jeden z nich *UNVEIL*[5] vytvára umelé užívateľské prostredie a monitoruje blokovania pracovnej plochy, vzory prístupov k súborom a dátovú entropiu I/O. Ďalšou z nich je *CryptoDrop*[29], ktorý

monitoruje zmeny typov súborov a meria zachovania podobnosti v súboroch pri zmenách pomocou hash funkcií, alebo rozpoznaním ransomvéru za pomoci Shannonovej entropie.

Ďalší nástroj, *ShieldFS*[6] monitoruje okrem predchádzajúcich znakov aj nízko úrovňové činnosti súborového systému a zhromažďuje nasledujúce funkcie: zoznam priečinkov, čítanie, zapisovanie a premenovávanie súborov, typ súboru a entropiu zápisu. Ransomvér odhalí na základe porovnania nameraných charakteristík a charakteristík neškodných aplikácií. Na rozdiel od predchádzajúcich dvoch, *ShieldFS* môže obnoviť aj súbory ktoré boli zašifrované už pre detekciou, hoci táto schopnosť prichádza s významným dopadom na výkon.

2.5 Detekcia na základe kryptografických primitív

V tomto prístupe sú binárne programy analyzované s cieľom identifikácie kryptografických operácií v ich spustiteľných kódoch. Pre tento účel[16] sa sleduje vykonávanie aplikácií a ich vzťah k I / O procesom v toku dát. Na rozpoznanie kryptografických algoritmov sa používajú funkcie pre detekciu výskytu bitových aritmetických inštrukcií a slučiek, vzťahov medzi vstupmi a výstupmi programových rutín a rôzne heuristiky.

Taktiež je pri tomto prístupe využitá statická analýza a graf dátových tokov (DFG), pre identifikáciu kryptografických algoritmov v binárnych programoch. Táto technika najskôr zostaví DFG binárneho programu. Ďalej sa DFG normalizuje pomocou prepisovacích pravidiel s cieľom odstrániť rôzne variácie pre optimalizáciu kompilátorov. Nakoniec sú vyhľadávané v DFG tie podgrafy, ktoré sú izomorfné ku grafovým signatúram kryptografických algoritmov. Zhoda priamo označuje, že zodpovedajúci algoritmus existuje v analyzovanom programe.

Kapitola 3

Návrh techník obfuskácie detekcie ransomvéru

Táto kapitola sa zameriava na pojem entropia dát a jej využitie pre možné prístupy obfuskácie činnosti ransomvéru tak, aby nespadal do detekčných procesov a metód v nástrojoch analyzovaných v kapitole 2. Popísané metódy budú v ďalšej kapitole popísané z hľadiska implementácie a ich efektívnosť otestovaná na konvenčných antivírusových a anti-ransomvérových nástrojoch.

3.1 Entropia dát

Najčastejšie býva entropia spojovaná s menom Claude Elwood Shannon, ktorý tento pojem definoval v roku 1948 a tým kvantifikoval informáciu. Entropia je miera, ktorá popisuje, ako moc je systém organizovaný. Inými slovami, predpokladajme existenciu nejakého systému a nezávislú udalosť, ktorá spôsobí prechod systému do nového stavu.

Entropia v kontexte detekcie ransomvéru je jednoduchý indikátor, ktorý poskytuje informácie o potenciálnej neistote zdroja údajov. Niektoré typy údajov, ako napríklad šifrované alebo komprimované dáta majú prirodzene vysokú entropiou. Intuitívne, útok ransomvérom by mal viesť konštantne k výstupu s vysokou entropiou, nakoľko ransomvér v procese obfuskácie údajov číta súbory obete a zapisuje šifrovaný obsah. Shannonovu entropia poľa pola bajtov možno vypočítať ako súčet:

$$e = \sum_{i=0}^{255} P_{Bi} log_2 \frac{1}{P_{Bi}} \tag{3.1}$$

pričom

$$P_{Bi} = \frac{F_i}{sucet\_bytov} \tag{3.2}$$

, kde F_i je počet inštancií hodnoty bajtu i v poli. Výsledkom je hodnota od 0 do 8, kde 8 predstavuje perfektne rovnomerné rozloženie bajtov v poli.

Šifrované súbory majú tendenciu približovať sa k hodnote 8, pretože každý byte v šifrovanom texte by mal mať jednotnú podobu pravdepodobnosti výskytu. V niektorých prácach je navrhnuté použitie entropie v priestore detekcie škodlivého softvéru, ako [20], [31], [32].

Avšak tieto práce sa zameriavajú na klasifikáciu vzorku škodlivého softvéru a nie na indikáciu transformácie používateľských údajov.

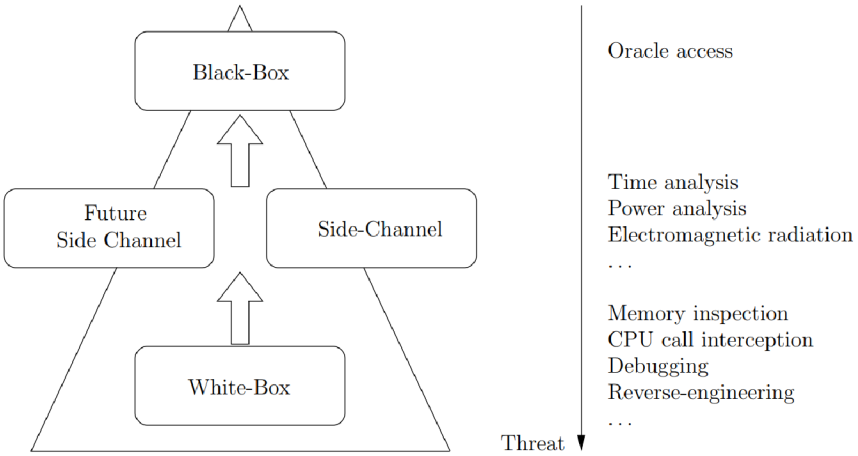
Behaviorálne založené detekčné metódy založené predovšetkým na detekcii hromadného šifrovania súborov môžu byť účinné, avšak bývajú náročné na zdroje. Šifrovacie opatrenia ako napríklad zmena entropie vyžaduje, aby sa entropia súboru vypočítala pre každú vykonanú operáciu zápisu aplikáciou. Okrem toho tieto operácie potrebujú sledovať operácie so súbormi pre každý súbor zvlášť počas životnosti pozorovaného procesu. Takýto prístup môže značne zhoršiť čítanie a písanie na systémový disk a mať za následok vysoké vyťaženie systému.

3.2 White-box kryptografia

V tejto podkapitole je popísané možné využitie white-box kryptografie pre obfuskáciu činnosti ransomvéru spod detekčných schopností anti-ransomvérových nástrojov. Kapitola rozoberá model white-box a konkrétne sa venuje technike vloženia (hard-kódovania) šifrovacieho kľúča priamo do zdrojového programu tak, aby ho bolo obtiažne spätne získať za pomoci reverzného inžinierstva. Implementovanou technikou white-box kryptografie (popísanou v následujúcich kapitolách) získa ransomvér výhodu o jeden faktor, voči analyzovaným konvenčným anti-ransomvérovým nástrojom, ktorým je možné vykonávanie činnost ransomvéru v offline režime bez nutnosti komunikácie s C&C.

Kryptografické šifry sú obyčajne navrhnuté pre využitie v štandardnom kryptografickom modeli, označenom ako model *čiernej skrinky*, kde sú komunikačné koncové body a výpočtové prostredia dôveryhodné. Avšak existujú aplikácie, kde tento model nie je dodržiavaný, ako napríklad beh ransomvéru v prostredí obete a preto je potrebné definovať nový model. Ako model najhoršieho prípadu útoku, v ktorom majú protivníci úplný prístup k implementácii kryptografických primitív a úplnú moc nad ich realizačným prostredím, je definovaný ako *model bielej skrinky*. Pri modeli bielej skrinky má útočník menovite následovné schopnosti[41]:

- **môže skúmať beh aplikácie** - môže pristúpiť k inštrukciám, ktoré sa vykonávajú počas behu programu, môže zaznamenávať priebeh algoritmu a vidí obsah použitej operačnej pamäti,
- **kontroluje prostredie (vykonáva modifikácie za behu)** - môže vykonávať iba určité cykly, podmienky a časti aplikácie (napríklad iba jedno kolo šifrovacieho algoritmu).



Obr. 3.1: Porovnanie modelov white-box a black-box z hľadiska možných útokov. Prevzaté z [40].

Whitebox kryptografia úzko súvisí s obfuskáciou uvedenou v časti 3. Je to aj programová transformácia, ale obfuskácia, ako je definované v literatúre je príliš reštriktívna a neberie do úvahy špecifické bezpečnostné prvky ako napr. reverznosť (invertibilita) šifier. White-box kryptografia sa zameriava na také prípady, kde je treba šifrovať a dešifrovať určitý druh informácií v prostredí white-boxu. Príkladom takýchto prostredí, kde sa white-box kryptografia využíva sú systémy DRM[1]. White-box kryptografiu môžeme definovať ako [42]:

Definícia 1 *Výzva, ktorou sa white-box kryptografia zaoberá, je implementácia kryptografického algoritmu do softvéru takým spôsobom, aby kryptografické aktíva zostali bezpečné aj pri vystaveniu voči možným útokom na white-box model. Implementácie softvéru, ktoré sú odolné voči takýmto útokom v bielej skrinke, sa označujú ako ïmplementácia typu white-box".*

Cieľom white-box kryptografie je vytvoriť program taký, ktorý by mohol byť bezpečne vykonaný v nedôveryhodnom prostredí. Hlavný princíp spočíva vo vytvorení kompilátora, ktorý dokáže pre špecifické kryptografické algoritmy a ich tajné kľúče vytvoriť *white-box kryptografický systém*, ktorý vykonáva kryptografický výpočet s vopred definovaným tajným kľúčom. Ten, kto má kontrolu nad vytvoreným programom, môže vykonať jeho operácie nad ľubovoľnými údajmi a získať očakávaný výstup, ale nemôže vykonať nič iné. Správne navrhnutý a implementovaný white-box kryptografický systém uchováva tajný kľúč rovnako, ako dôveryhodný hardvér a odoláva útokom spomenutým v nasledovných podkapitolách.

[1]Digital Right Management je súbor techník, ktoré umožňujú účastníkom získať prístup k zabezpečenému obsahu pod viacerými prístupovými právami alebo podmienkami. On-demand videá a mobilná televízia sú výraznými príkladmi takýchto služieb. Pri neexistujúcom hardvérovom kryptografickom module realizácia založená na White-box modeli výpočtu dešifrovania obsahu pod individuálnym klientskym kľúčom udržiava kľúč od opätovného využitia a obnovenia útočníkmi (krádež vzhľadom na redistribúciu a zdieľanie kľúčov).

3.2.1 História

Whitebox kryptografia je celkom nová oblasť kryptografie. Štúdia o implementácii white-box modelu pomocou šifier začala prvou implementáciou AES [30] a DES [42] autorom *Chow* v roku 2002.

Spočiatku sa kryptoanalýza algoritmu DES zameriavala na zjednodušený variant. Prvý publikovaný v roku 2002 používa metódu *fault testing* [27], iný publikovaný v roku 2005 využíva štatistickú analýzu [18]. Neskôr bola publikovaná kryptoanalýza plne kódovaného variantu DES v roku 2007 s využitím skrátených diferenciálov.

Podobný prípad platí pre algoritmus AES. Dva roky po zverejnení white-box schémy AES, bola publikovaná úspešná kryptoanalýza v publikácii [30], ktorá umožnila obnoviť vložený symetrický kľúč v menšom počte než 2 na 30tu krokov. Neskôr, v roku 2008, bola publikovaná zovšeobecnená verzia predchádzajúcej kryptoanalýzy, ktorá ovplyvňuje väčšiu skupinu šifier s použitím rovnakej štruktúry ako AES [39].

V prácach sa taktiež objavila whitebox AES schéma pridaním ďalších lineárnych mapovaní a zvýšením veľkosti implementácie v práci [43] ako odpoveď na predchádzajúci útok. Útok na vylepšenú schému pomocou algoritmu lineárnej ekvivalencie bol publikovaný v roku 2012 [46].

Ďalší pokus, ako opraviť slabiny vo white-box AES, priniesol [2] so zavedením náhodných porúch, čo komplikovalo algebraickú kryptoanalýzu, avšak útok na túto implementáciu bol publikovaný vo výskume [45].

Posledná white-box schéma AES bola uverejnená v roku 2011, ktorá bola vylepšená s využitím dvojitých šifier[28]. Dokument tvrdil, že schéma je dostatočne odolná, aby odolala doposiaľ známym útokom na white-box implementáciu. Tento predpoklad bol dokázaný ako nepravdivý v práci [14], ktorá poukázala, že publikovaný útok v predchádzajúcom odstavci funguje rovnakým spôsobom aj na túto implementáciu.

3.2.2 Útoky na white-box s využitím entrópie

Pri implementácií šifrovacieho algoritmu do prostredia white-boxu s hard-kódovaným šifrovacím kľúčom sa vyskytuje možný útok, ktorý pre odhalenie šifrovacieho kľúča využíva princípy entropie dát. Keďže by šifrovací kľúč nemal byť ľahko uhádnuteľný, tak je zvolený náhodne. Náhodnosť sa dá merať entropiou a tento šifrovací kľúč by mal mať vysokú entropiu. Na druhej strane algoritmus na šifrovanie resp. dešifrovanie má nízku entropiu, pretože skompilovaný do programu sa bude skladať len z obmedzeného počtu inštrukcií.

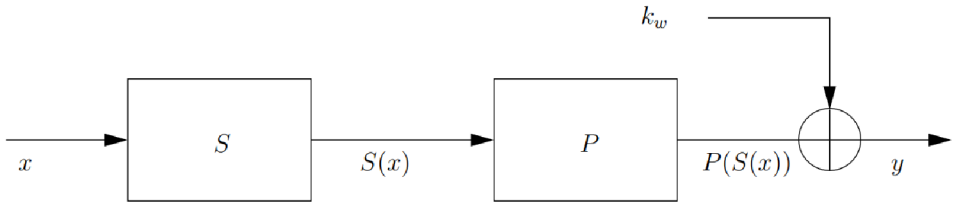
V [33] autori ukázali jednoduchý spôsob na lokalizácie kľúča v skompilovanom programe. Na obrázku 3.2 je znázornená binárna implementácia programu so zabudovaným tajným kľúčom. Zatiaľ, čo časti s nízkou entropiou na ľavej a pravej strane majú nejakú štruktúru, časť v strede, s vysokou entropiou, sa javí ako šum. Preto môžeme relatívne vysokou istotou tvrdiť, že tajný kľúč sa bude nachádzať v časti súboru odpovedajúcej tejto časti obrázku. Na určenie presnej polohy a hodnoty kľúča je potrebný hlbší rozbor, ale tento prístup nám cestu k odhaleniu kľúča výrazne skrátil.

Obr. 3.2: Grafické znázornenie pamäte programu alebo binárnych dát, kde 0 je zobrazená ako čierna a 1 ako biela.

V poslednom desaťročí boli navrhnuté mnohé riešenia, ktoré by mohli zabrániť útokom, ako útok s využitím entropie. Väčšina z nich sú techniky obfuskácie softvéru určené na ochranu dátových štruktúr pred nástrojmi analýzy softvéru. Obfuskácia softvéru označuje súbor techník, ktoré chránia kód pred statickou a dynamickou analýzou. Pre protivníka to sťažuje pochopenie implementácie programu, hoci jeho funkčnosť zostáva nezmenená. Obfuskovaná implementácia programu P je často označovaná ako $O(P)$. Najbežnejším prístupom, ktorý sa vyvinul najmä v deväťdesiatych rokoch, je rozbitie dátových štruktúr a prerušenie abstrakcie premenných v programe. Pre tento prístup bolo prezentovaných niekoľko techník [10], ako napríklad variabilné delenie a transformácia. Myšlienka spočíva v tom, že informácie o kľúči by mohli byť rozsekané na niekoľko častí, z ktorých by každá bola uložená na rôznych miestach (adresách) v binárnom kóde, takže pôvodná hodnota sa nedá ľahko prečítať pomocou nástroja statickej analýzy. Taktiež, časti šifrovacieho kľúča môžu byť využité programom takým spôsobom, že nie sú súčasne uložené v pamäti. Tu však nastáva problém, kde prostredníctvom dynamickej analýzy implementácie (t.j. analýza v čase behu) je možné sledovať polohy pamäte a tým odhaliť pôvodný kľúč [19]. Vylepšené techniky zahŕňajú lineárne transformácie, kde namiesto výpočtu s pôvodnou hodnotou kľúča možno vykonať výpočet s jej transformovanou hodnotou, čo však zahŕňa modifikáciu kódu a preto sú transformácie väčšinou využívané zjednodušene.

3.2.3 Útoky na white-box s využitím techniky key-whitening

Bielenie pomocou kľúča (z anglického key-whitening) je jednou z techník, ako ľahko zvýšiť bezpečnosť blokových šifier. Myšlienkou techniky je skombinovanie kľúča s dátami na vstupe alebo výstupe šifrovacieho a dešifrovacieho algoritmu pomocou operácie XOR a zabrániť tým získanie šifrovacieho kľúče. Medzi blokové šifry, ktoré využívajú túto techniku, patrí okrem iných aj AES alebo TwoFish. V útoku na key-whitening je binárna časť šifrovacieho algoritmu modifikovaná (v kontexte white-boxu) tak, aby sa na výstupe šifrovania objavil šifrovací kľúč.

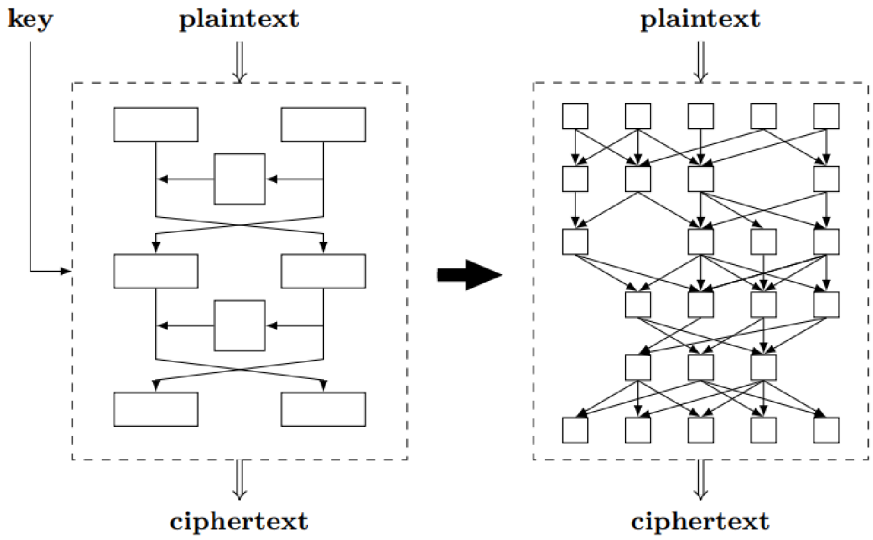


Obr. 3.3: Finálna operácia argoritmov ako AES a TwoFish, ktoré využívajú key-whitening a substitučné boxy (S-Boxy). [40].

Obrázok 3.3 zobrazuje záverečné operácie blokovej šifry, ktorá využíva key-whitening techniku a statické substitučné boxy (ako algoritmy AES a TwoFish). S označuje substitučný box (S-box), ktorý pracuje na vstupe x, P permutáciu, ktorá pracuje na výstupe S. Toto je nasledované finálnym pridaním bieliaceho kľúča s kľúčom pre daný cyklus k_w. Výstup y sa rovná $P(S(x))k_w$. Kvôli princípu Kerckhoffovho prístupu je definícia statických S-boxov všeobecne známa a preto je možné v útoku na bielu skrinku nájsť umiestnenie týchto S-boxov v binárnej podobe pomocou statickej analýzy rôznymy nástrojmi. Keďže S-boxy sú implementované ako vyhľadávacie tabuľky a môžu byť ľahko prepísané v softvérovom binárnom kóde, stačí prepísať vyhľadávaciu tabuľku nulami, kde výsledkom vykonania bude vždy výstup k_w, pretože $P(S(x))$ sa rovná 0 pre daný vstup x [36].

3.2.4 White-box šifrovanie

White-box šifrovanie je koncepcia ochrany citlivých údajov, ktoré sú pevne zakódované v implementácii softvéru [35], [34]. Hlavným zameraním tejto oblasti je vloženie tajných kľúčov do zdrojového kódu takým spôsobom, že ich je ťažké extrahovať z kompilovaných binárnych súborov. Príklad Feistelovej blokovej šifry a jej implementácia s hard-kódovaným kľúčom s využitím metódy white-boxu je znázornená na obrázku 3.4. Hoci white-box kryptografia nie je novým nápadom (prvýkrát bola spomenutá v roku 2002), bezpečná implementácia blokovej šifry AES zatiaľ neexistuje, nakoľko existuje niekoľko prác, kde sa rozoberá útok na extrakciu kľúča a útok na white-box pomocou dekompozície tabuľky. Napriek tomu stále existuje white-box kryptografia v oblasti výskumu [26], [7], [44].



Obr. 3.4: V ľavo je algorimus blokovej šifry založený na Feistelovej sieťovej štruktúre. V pravo, white-box implementácia tohoto šifrovacieho bloku kde je kľúč hard-kódovaný do algoritmu.

Tento prístup je možné použiť pre obfuskáciu kódu ransomvéru s hard-kódovaným šifrovacím kľúčom pre šifrovanie dát obete alebo je ho možné využiť pre samotné šifrovanie binárnych dát obete.

3.2.5 Koncept white-box AES algoritmu

Stratégia navrhnutá *Chowom a spol.*[35] spočíva v transformácii daného bloku šifry do náhodnej, kľúčovo závislej siete vyhľadávacích tabuliek. Zahŕňa tri hlavné kroky:

1. **Čiastočné vyhodnotenie** - vkladanie kľúča do operácií, zvyčajne transformáciou fixných S-boxov S_i do kľúčových vyhľadávacích tabuliek T_i, v prípade v prípade operácie pridávania kľúča pred operáciou S-boxu.

$$T_i(x) := S_i(x \oplus k_i) \tag{3.3}$$

2. **Tabularizácia** - tento proces spočíva v transformácii všetkých zložiek blokovej šifry, vrátane lineárnych transformácií, do vyhľadávacích tabuliek. Avšak neexistuje všeobecný „kompilátor“ alebo algoritmus na transformáciu daného algoritmu do jeho tabuľkového ekvivalentu. Namiesto toho je v literatúre uvedený zoznam techník implementáciách algoritmu DES a AES vo white-boxe. Jedna z týchto techník bude použitá a rozpísaná v sekcií implementácie.

3. **Randomizácia a delinearizácia** - transformácia na vyhľadávacie tabuľky sa využíva kvôli dôvodu, že vyhľadávacie tabuľky môžu implementovať akúkoľvek zadanú funkciu a preto sú ideálnym konštruktom pre skrytie informácií. Hlavná myšlienka spočíva v zameraní sa na reťazce troch po sebe idúcich vyhľadávacích tabuliek v sieti $L_3 o L_2 o L_1$, kde L_2 obsahuje niektoré kľúčové informácie, ktoré musia byť skryté ako napr.:

$$L_2(x) = x \oplus k \tag{3.4}$$

Popis vyhľadávacích tabuliek k dispozícii aj útočníkovi na white-box, kde informácie o kľúči sa môžu extrahovať aj priamo vyhodnotením $L_2(0)$. Tomuto faktu sa dá predísť následovne:

$$\begin{cases} L_1 \mapsto L_1' = b_1 \circ L_1 \\ L_2 \mapsto L_2' = b_2 \circ L_2 \circ b_1^{-1} \\ L_3 \mapsto L_3' = L_3 \circ b_2^{-1} \end{cases} \tag{3.5}$$

kde b_1 a b_2 sú bijekcie rozmerov zodpovedajúcich vstupnému a výstupnému rozmeru operácie L_2. L_i' sú označené enkódované vyhľadávacie tabuľky a enkódovaný reťazec $L_3' o L_2' o L_1'$ má ekvivalentnú funkčnosť ako pôvodný reťazec. Základnou vlastnosťou je, že L_2' neunikne žiadna informácia o kľúči a že white-box útočník je nútený analyzovať viac komponent. V tomto prípade by protivník musel analyzovať L_1' a L_3', aby obnovil informácie o b_1 a b_2.

3.3 Ransomvér s využitím ECB módu šifrovania algoritmom AES

Táto podkapitola popisuje využitie algoritmu AES v ECB móde šifrovanie pre šifrovanie dát obete ransomvéru. Touto metódou sa vyhne ransomvér detekčnej schopnosti anti-ransomvérového nástroja, ktorý sa zameriava na štatistiku veľkosti entropií pôvodných a nových súborov.

V podkapitole je abstraktne popísaný algoritmus AES (ktorou úpravou je docieľený white-box pre daný algoritmus, viď. kapitolu 4.3), jeho činnosť v ECB móde, výhody a nevýhody spojené s použitím ECB módu šifry AES.

3.3.1 Popis algoritmu AES

Advanced Encryption Standard (AES) je substitučno-permutačná sieť , ktorá podporuje kľúče o veľkosti 128, 192 a 256 bitov (označovaných ako AES-128, AES-192 a AES-256). Pracuje s pevnou veľkosťou bloku - 128 bitov.

V závislosti od veľkosti kľúča (128, 192 alebo 256) sa algoritmus AES skladá z 10, 12 alebo 14 kôl(iterácií), ktoré pracujú na poli 4 4 bajtoch (toto pole je nazývané ako "stav"). Kolo (iterácia) je špecifikovaná v štyroch krokoch:

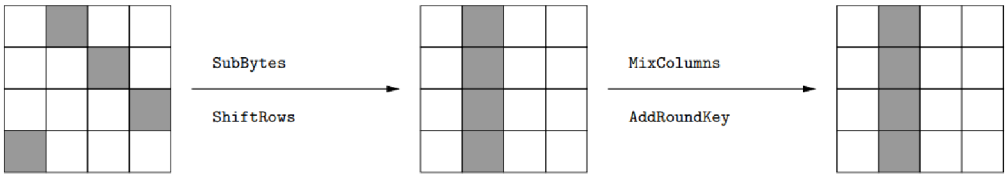
1. **SubBytes** - vyhodnocuje každý bajt daného stavu tým istým S-boxom, 16-krát paralelne. S-box je definovaný rovnicou *S (x) = M (1 / x) + b* na poli GF (2^8)[2], kde M je vhodne zvolené a b je konštanta. Takáto definícia dáva úzke lineárne a diferenciálne hranice.

2. **SiftRows** - cyklický posun každého riadku, ktorý ponecháva horný riadok štyroch bytov nezmenený.

3. **MixColumns** - miešanie štyroch bajtov stĺpca násobením každého stĺpca konštantnou maticou na poli GF (2^8).

4. **AddRoundKey** - binárne pridanie stavu pomocou 128-bitového podkľúča.

[2]Konečné teleso (tiež Galoisove teleso na počesť Évarista Galoisa, zvyčajne značené GF (p_k)) je v matematike, presnejšie v abstraktnej algebre, označenie pre také teleso, ktoré má konečný počet prvkov.[11]



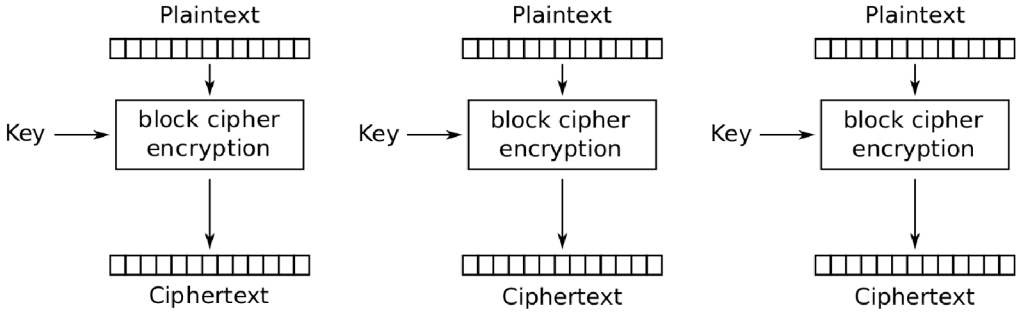
Obr. 3.5: Schématický náhľad na jedno kolo (iteráciu) algoritmu AES.

Pred prvým kolom sa vykoná predbežný krok *AddRoundKey* (s bieliacim kľúčom, pozri kapitolu 3.2.3) a v poslednom kole sa vynechá operácia *MixColumns*. Operácia MixColumns posledného kola nemá význam, nakoľko útočník môže zrušiť túto operáciu použitím známej transformácie.

Kolo (iterácia) algoritmu je navrhnuté tak, že zmena na vstupe do kola môže potenciálne ovplyvniť všetky výstupné bity len po dvoch kolách. Plán kľúčov algoritmu AES-n berie n-bitový kľúč a transformuje ho na NR + 1 podkľúče, kde každý má 128 bitov a kde NR označuje počet kôl. Ďalšie podrobnosti o algoitme AES sú popísané v zdroji [1], z ktorého boli čerpané.

3.3.2 Popis ECB módu šifrovania

Režim "kódovej knihy"(anglicky Electronic Codebook, z čoho pochádza skratka ECB) je najjednoduchším a základným režimom šifrovania. Bloková šifra sa pri ňom aplikuje priamo a nezávisle na jednotlivé bloky, čo znamená, že pri danom kľúči zodpovedá rovnakému bloku otvoreného textu rovnaký blok šifrového textu. Konkrétny blok je v tomto režime *prekladaný* na iný blok podobne, ako sa v rôznych historických substitučných šifrách prekladali jednotlivé znaky alebo slová na iné pomocou tabuľky v kódových knihách. V ECB móde je možné šifrovať a dešifrovať pomocou viacerých vlákien súčasne. Činnosť ECB módu je znázornená na obrázku 3.6.



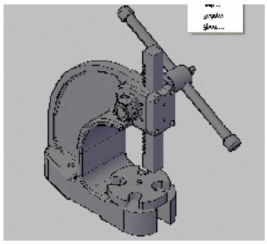
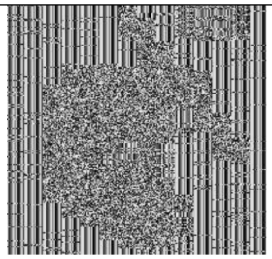
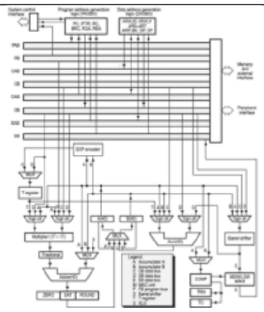
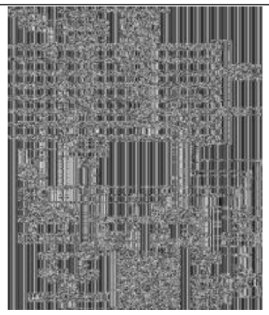
Obr. 3.6: Činnosť ECB módu šifrovania.

Fakt *prekladania* blokov má nežiaduce dôsledky z hľadiska bezpečnosti: ak sa napríklad tento režim použije pre šifrovanie nejakého protokolu s pevne danou štruktúrou, môže útočník po zachytení určitého počtu správ začať rozlišovať ich obsah akonáhle rozpozná opakovanie nejakého kľúčového bloku obsiahnutého v niektorých predchádzajúcich sprá-

vach. Útočník môže aj aktívne sám zostavovať podvrhnuté správy ako nové kombinácie zachytených blokov, čo môže uľahčovať útoky prehraním správy (*replay* útoky).

Všeobecné nevýhody módu ECB umožnujú útočníkovi zistiť, či sú dve správy šifrované v móde ECB identické, či dve správy šifrované v ECB móde zdieľajú spoločnú predponu, či dve správy zašifrované ECB módom zdieľajú iné spoločné podreťazce (pokiaľ sú tieto podreťazce zarovnané na hraniciach blokov) alebo zistiť, či (a kde) správa zašifrovaná ECB módom obsahuje opakované údaje (ako napríklad dlhé úseky medzier alebo nulových bajtov, opakované polia záhlavia alebo náhodne opakované frázy v texte), opakovanie nulových bajtov, opakované polia záhlavia dokumentov alebo náhodne opakované frázy v texte). V iných režimoch šifrovania nemá vplyv na dešifrovateľnost iných blokov. Pre zabezpečenie môže byť použitý len vtedy, ak je zaručená dlhodobá jedinečnosť blokov šifrovaných rovnakým kľúčom.

3.3.3 Entropia šifrovaného súboru v móde ECB

	Obr. 1	Entropia	Obr. 2	Entropia
Nezašifrovaný		2.3005		6.8940
AES ECB		3.2788		7.2513

Tabuľka 3.1: Porovnanie entropie nezašifrovaného obrázku a zašifrovaného obrázku s algoritmom AES v móde ECB. Zdroj informácií obsiahnutých v tabuľke [13].

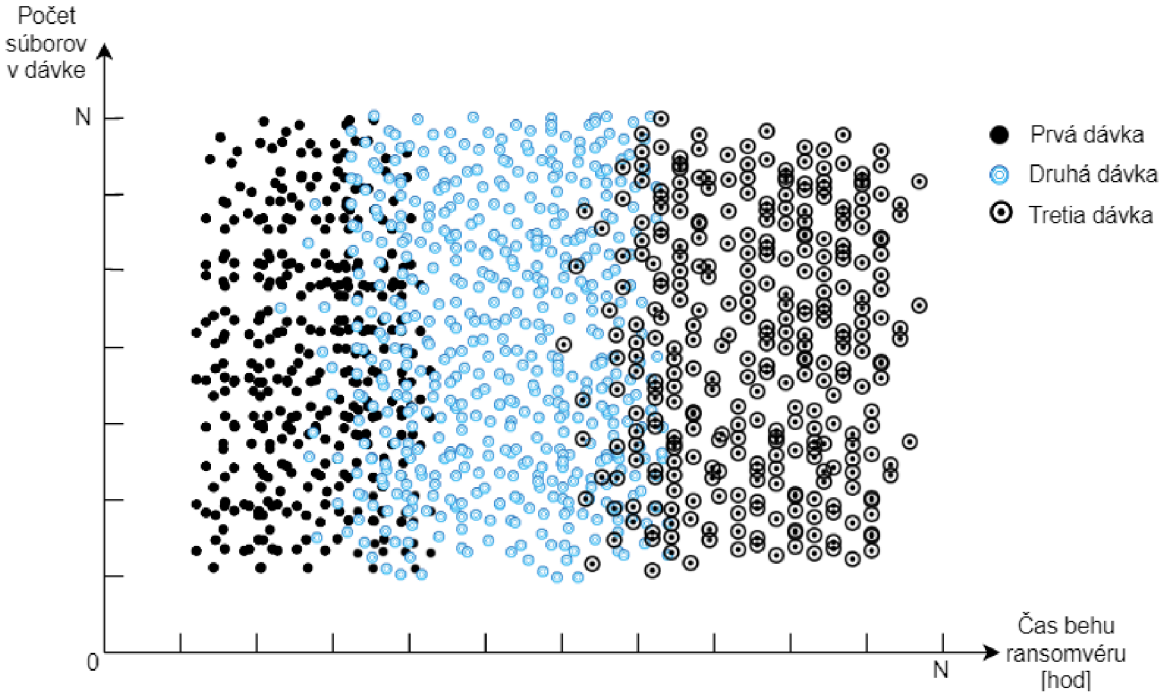
Dáta šifrované ECB módom majú menšiu entropiu (viď. obrázok 3.1) ako dáta šifrované iným módom (ako napríklad CBC, CFB, CTR a mnohé ďalšie), kde sa entropia limitne blíži k hodnote 8, čo je najväčšia možná entropia. Nedostatky spojené s týmto faktom a nedostatky a zraniteľnosti spomenuté v kapitole 3.3.2 nie sú pre činnosť a reverzné inžinierstvo ransomvéru podstatné z hľadiska jeho hlavného cieľa. Tento hlavný cieľ - obfuskovať (šifrovať) dáta obete je schopný ECB mód splniť aj po obnovení opakujúcich sa zašifrovaných blokov, nakoľko sa s najväčšou pravdepodobnosťou vyskytne v zašifrovaných dátach blok, ktorý sa neopakuje, čiže ho nie je možné obnoviť a tým pádom sú dáta nenávratne bez dekryptora (dekryptovacieho kľúča) poškodené.

3.4 Dávkové šifrovanie súborov v náhodnom čase

Táto navrhovaná metóda sa pokúsi obfuskovať jednu konvenčnú činnosť ransomvéru - šifrovanie dát obete, kde šifrovanie prebieha ako kontinuálna činnosť.

Metóda dávkového šifrovania súborov v náhodnom čase spočíva v rozložení šifrovania súborov obete do jednotlivých dávok. Jedna dávka šifruje náhodný počet súborov, kde počet je vybraný náhodne zo špecifikovaného intervalu (rôzne intervaly a ich účinnosť bude popísaná v kapitole testovania) a kde časový rozostup medzi šifrovaním jednotlivých súborov v dávke je taktiež vybraný náhodne zo špecifikovaného intervalu. Časový rozostup jednotlivých dávok bude taktiež vybraný náhodne zo špecifikovaného intervalu.

Táto metóda dosiahne vysokú náhodnosť v časovom rozložení šifrovania počas fáze znehodnotenia dát v ransomvéri. V grafe 3.7, kde jeden bod znázorňuje jednu dávku šifrovania, je možné vidieť rozloženie dávok a počtov súborov v čase behu malvéru. Graf reprezentuje štatistiku behu jednotlivých dávok vo viacerých spusteniach ransomvéru. Jeden beh ransomvéru reprezentuje jeden bod z každej skupiny na grafe, kde skupín môže byť 1 až N v závislosti na vstupnom počte súborov a v počte spracovaných súborov počas behu predchádzajúcej dávky.



Obr. 3.7: Graf rozloženia jednotlivých dávok metódy, zobrazujúci vysokú entropiu naplánovania dávkového šifrovania. Počet dávok je závislý na počte súborov na šifrovanie, kde počet dávok môže byť 1 až N.

Kapitola 4

Implementácia obfuskačných techník

Táto kapitola popisuje implementáciu jednotlivých navrhovaných metód z predchádzajúcej kapitoly do funkčného celku ransomvéru.

Každá metóda je implementovaná ako samostatný program - *ransomvér*, ktorý slúži na šifrovanie (obfuskáciu) dát obete a a program *dekryptor*, ktorý slúži na obnovenie obfuskovaných dát obete do pôvodného stavu. Metódy z kapitol 3.3, 3.4 a konvenčný model ransomvéru využívajú spoločný C&C server pre generovanie a získanie šifrovacích kľúčov na činnosť ransomvéru alebo dekryptoru. Metóda z kapitoly 3.2 nepotrebuje C&C server, nakoľko pre každú inštanciu ransomvéru a dekryptoru využíva unikátny white-box.

Všeobecne pre každú metódu je modul ransomvér klient, ktorý má za úlohu šifrovať (obfuskovať) súbory zo špecifikovanej zložky cieľového systému a zanechať po sebe správu o svojej aktivite s návodom pre zaplatenie "výkupnéhoä získanie dekryptora obfuskovaných súborov.

4.1 Implementácia C&C severa

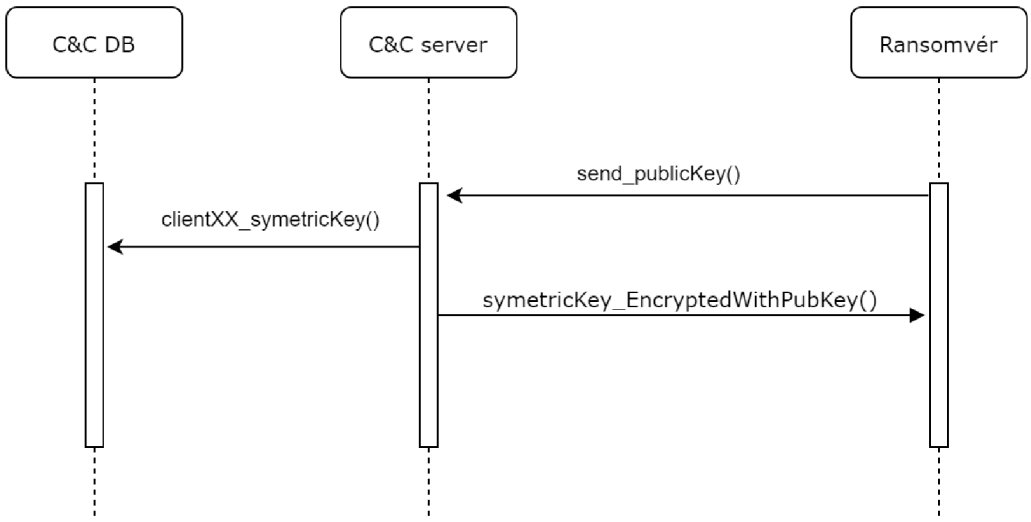
Spoločný C&C sever je implementovaný ako skript v jazyku python (odladený pre verziu 2.7), ktorý počúva na porte 4444 pre spojenia s ransomvérom alebo dekryptorom. Pri novom spojení mu klient zašle verejný kľúč, server vygeneruje náhodné UUID, šifrovací kľúč a inicializačný vektor (zo zdroja */dev/urandom*), tieto údaje zašifruje pomocou obdržaného klientského verejného kľúča a následne zašle klientovi. Tieto vygenerované údaje si server uloží taktiež do svojej internej databázy (implementovanej ako textový súbor vo formáte *JSON*). Tieto údaje budú v neskoršej fáze (po zaplatení výkupného) vyžiadané modulom dekryptor, ktorý zašle UUID stanice, na ktorej bol spustený. Skript využíva štandardný modul *Crypto* pre prácu so šifrovaním pomocou verejného kľúča.


```
{
    "a5219f6be39bfc54e5f0eda38c477c5b966020ec":{
        "uid":"a5219f6be39bfc54e5f0eda38c477c5b966020ec",
        "key":"16c934d72dd98006f8f0f0cf3376a96cb446cbe1e7d96970a74e753f61787502",
        "iv":"c633c53fde026a8d9a2595cbf92e46a0"
    }
}
```


Výpis 4.1: Ukážka záznamu klienta na C&C severi.

4.1.1 Komunikácia ransomvéru

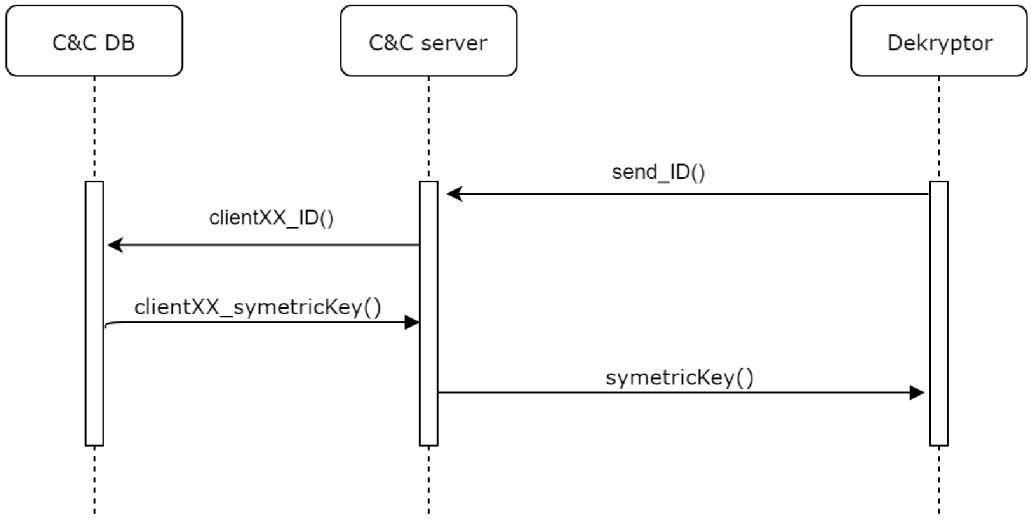
Ransomvér program po úspešnom spojení s C&C severom zasiela na stranu servera vygenerovaný privátny kľúč, ktorým následne server šifruje údaje (inicializačný vektor, šifrovací kľúč a UUID) zasielané napäť ransomvéru (viď. obr.4.1).



Obr. 4.1: Sekvenčný diagram komunikácie ransomvéru s C&C serverom pre vyžiadanie symetrického kľúča na šifrovanie dát obete.

4.1.2 Komunikácia dekryptora

Program dekryptor po svojom spustení na stanici obete zasiela UUID stanice na IP adresu C&C servera, ktorý následne zašle dešifrovací kľúč (rovnaký ako šifrovací keďže sa jedná o symetrickú kryptografiu) napäť dekryptoru (viď. obr.4.2).



Obr. 4.2: Sekvenčný diagram komunikácie dekryptora s C&C serverom.

4.2 Konvenčný typ ransomvéru

Konvenčný typ ransomvéru, slúžiaci ako vzorka pre porovnanie výsledkov testovania navrhnutých metód obfuskácie, je implementovaný ako 3 moduly: samotný ransomvér, C&C server a dekryptor.

Ransomvér po spustení vygeneruje kľúčový pár kryptosystémom RSA2048, verejný kľúč zašle po úspešnom spojení s C&C serverom na server a následne čaká na odpoveď servera. Odpoveď ktorá obsahuje inicializačný vektor, šifrovací kľúč a UUID, program rozšifruje privátnym kľúčom a začne s procesom obfuskácie (šifrovaním) dát.

Program šifruje dáta, ktoré sú uložené na definovanej ceste (cesta je definovaná priamo v programe). Pre svoju činnosť šifrovania dát využíva AES mód šifrovania v móde CBC. Šifrovanie prebieha načítaním obsahu aktuálneho súboru do premennej vo forme stringu (hex hodnoty načítaného binárneho vstupu), vymazaním súboru, šifrovaním načítaných dát a zápisom dát do súboru s rovnakým názvom ako bol názov pôvodného súboru. Po dokončení šifrovania všetkých súborov v špecifikovanej ceste program na plochu obete (*C:\Users\USER\Desktop*) zapíše textový súbor s informáciami ako sú UUID a kontaktom na vyžiadanie dekryptora. Pre prácu s RSA a AES algoritmami bola využitá knižnica Crypto++ [12].

```
initialization;
while notConnected do
    client.Connect();
end
rsa.genKeys();
client.sendPubKey(rsa.pubKey);
client.receiveData();
cc_data = rsa.decryptData(client);
while pathNotTraversed() do
    encrypt_data(path, IV, symetricKey);
end
file.write("Ransomed! Your UUID", "Desktop\ransomed.txt");
return;
```

Algoritmus 1: Princíp implementácie konvenčného ransomvéru.

4.3 White-box AES

Práca sa zameriava na white-box AES metódu popísanú autorom *Chow a spol.*[35] a upravenú podľa práce [41]. Ako už bolo spomenuté v predchádzajúcich kapitolách, všeobecnou stratégiou je zlúčiť niekoľko krokov šifry do siete vyhľadávacích tabuliek, ktoré sú potom obfuskované použitím náhodného alebo definovaného kódovania vstupu a výstupu. Práca implementuje white-box pre AES-128, ktorý počíta s desiatimi kolami algoritmu.

4.3.1 Konštrukcia algoritmu

Najskôr je použitá technika čiastočného vyhodnotenia na zlúčenie doplnku kľúča s hodnotou S-boxu. Funkcia S-boxu, počas kroku *SubBytes* je označená ako S. T-box definujeme následovne:

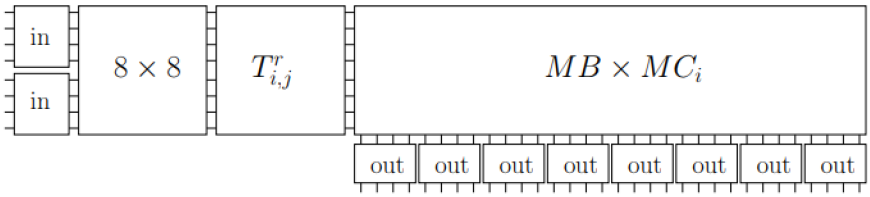
$$T^r_{i,j}(x) := S(x \oplus k^r_{i,j}) \qquad pre \quad 1 \leq r \leq 9$$
$$T^{10}_{i,j}(x) := S(x \oplus k^{10}_{i,j}) \oplus k^{11}_{i,j-i} \tag{4.1}$$

, kde r označuje číslo cyklu (kola algoritmu) a dvojica (i, j) indexové číslo riadka a stĺpca bajtu v stavovom poli. Každý výstup T-boxu plus krok *ShiftRow* prispieva k 4 bajtom stavového poľa po kroku *MixColumns*, čo môže byť opísané pomocou *32x8* submatice MC_i matice MC o veľkosti 32 8 bitovov reprezentujúcej krok *MixColumns*. Celá funkcia môže byť opísaná ako vyhľadávacia tabuľka o veľkosti 32x8 bitov. Stratégia je podobná operácii dekompozície matíc, ktorá zahŕňa vyhľadávacie tabuľky implementujúce dodatočnú operáciu (označované autorom *Chowom a spol.*[35] ako tabuľky typu IV, znázornené na obrázku 4.4.

Táto vyhľadávacia tabuľka musí byť obfuskovaná so 4-bitovým nibble kódovaním.[1]

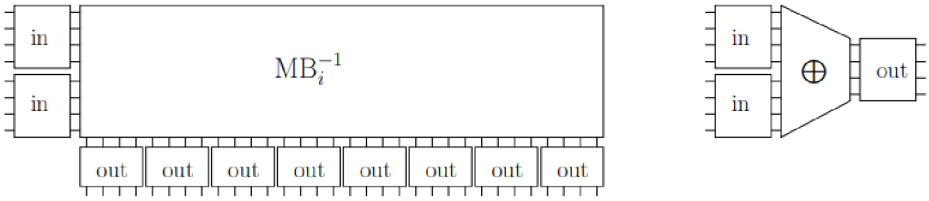
Pre zväčšenie rozptylu sa pred $T^r_{i,j}$ vložia 8x8 bijektné zmiešavače a za časť **MixColumns** zmiešavače o veľkosti 32x32.

Bijekcie sú rušené predchádzajúcim MC (v kole r - 1), ktorý obsahuje inverz (jednoduchým násobením matice) a preto je inverzný krok rozptýlený cez niekoľko vyhľadávacích tabuliek, čo sťažuje jeho odstránenie. Výsledná vyhľadávacia tabuľka je znázornená na obrázku 4.3.



Obr. 4.3: Tabuľka typu II.

Na zrušenie efektu získaným MB (*mixing bijection*) je implementovaná tabuľka typu III, ktorá sa stará o zavedenú inverziu. Výsledná vyhľadávacia tabuľka je zobrazená na obrázku 4.4.



Obr. 4.4: Tabuľka typu III a typu IV.

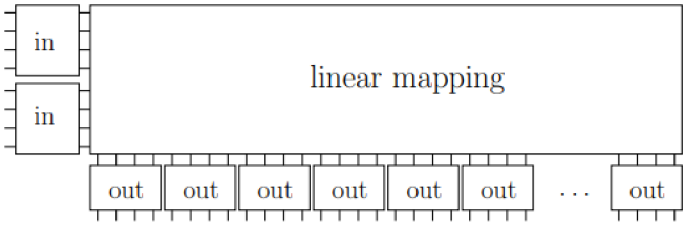
Aby bolo zabezpečené, že zakódované komponenty tabuliek typu II a typu III ponesú maximum informácií a mali maximálnu difúziu, je vybraté MB ako neštandardná matica,

[1] Namiesto 8-bitového kódovania (ktoré by obfuskovalo celú šírku T-boxov) na účely veľkosti implementácie boli zvolené 4-bitové nibble kódovania. To umožňuje použitie dodatočných operácií 8 4 namiesto 16 8, čo by bolo 256-krát väčšie na implementáciu.

kde podmatice 4 4 majú plné ohodnotenie. Účinnú metódu na generovanie takýchto matíc predstavili autori *Xiao a Zhou*[21].

4.3.2 Externé vstupno-výstupné kódovania

Externé vstupné a výstupné kódovania sú implementované v poslednom kroku. Práca [35] navrhla vložiť MB (mixing bijection) tabuľku o veľkosti 128 128 pred prvým výpočtom T-boxu, a 128 128 MB po poslednom výpočte T-boxu. Vstupné kódovanie je zložené z inverznej 8 8 vstupnej MB tabuľky pre $T^1_{\cdot}$, čo môže byť implementované použitím dvoch sád pozostávajúcich zo šestnástich 8-bitových vyhľadávacích tabuliek, znázornených na obrázku 4.5 a sady 960 prídavných tabuliek (typ IV). Namiesto AES_k je implementované N_O o M_G o AES_k o M_F o N_I , kde M_F a M_G sú afinitné vstupné a výstupné kódovania a N_I a N_O sú vstupné a výstupné nelineárne nibble kódovania.



Obr. 4.5: Tabuľka typu I.

Každá *tabuľka kódovania externého vstupu* predstavuje lineárne mapovanie spojené s jedným stĺpcom 128 8 matice 128 128 - je to zloženie M_F a zreťazenie vstupných MB posledného kola algoritmu pre inverzie $T^1_{i,j}$ - obklopeným 4-bitovým na 4-bitové nelineárným kódovaním.

Každá *tabuľka kódovania externého výstupu* predstavuje jeden stĺpec 128 8 matice 128 128 - zloženie výstupnej MB posledného kola, jedného z mapovaní $T^{10}_{i,j}$ a stĺpca matice MG - obklopeným 4-bitovým na 4-bitové nelineárným kódovaním.

Na dokončenie implementácie musia byť výstupy vstupnej a výstupnej kódovacej siete musia byť spojené so sieťou (kódovaných) dodatočných tabuliek. Tieto dekódujú výstupy, XORujú, a re-kódujú s nibble kódovaním zodpovedajúcim tabuľkám typu II a IV.

4.3.3 Metriky a veľkosť implementácie

V tabuľke 4.1 možno vidieť prehľad počtu vyhľadávacích tabuliek ktoré predstavujú implementáci AES white-boxu. Celkovo implementácia, ako bola popísaná aj s externým vstupným a výstupným kódovaním pozostáva z 3008 vyhľadávajúcich tabuliek s implementačnou veľkosťou 770 kilobytov, v porovnaní so štandardnou implementáciou AES, ktorá má 300 operácií (vyhľadávacie tabuľky spolu s XOR operáciami) a implementačná veľkosť 4 kilobytov.

Typ	počet vyhľadávajúcich tabuliek	implementačná veľkosť
Typ I	32	131072 bytov
Typ II	144	147456 bytov
Typ III	144	147456 bytov
Typ IV	2688	344064 bytov
Celkovo	3008	770048 bytov

Tabuľka 4.1: Veľkosť vyhľadávacich tabuliek AES v implementácií white-boxu.

Tabuľka 4.2 ukazuje na počet možných interpretácií vyhľadávacích tabuliek. Výkonnosť implementovaného AES white-boxu má odhadovanú výkonnosť 188 cyklov na byte, čo je takmer 12-krát pomalšie ako šifrovací algoritmus AES[15]. Spomalenie je prevažne spôsobené implementáciou dodatočných operácií do 2688 vyhľadávacích tabuliek[41].

Typ	Počet možností
Typ I	$2^{1483.5}$
Typ II	$2^{556.6}$
Typ III	$2^{486.4}$
Typ IV	$2^{132.8}$

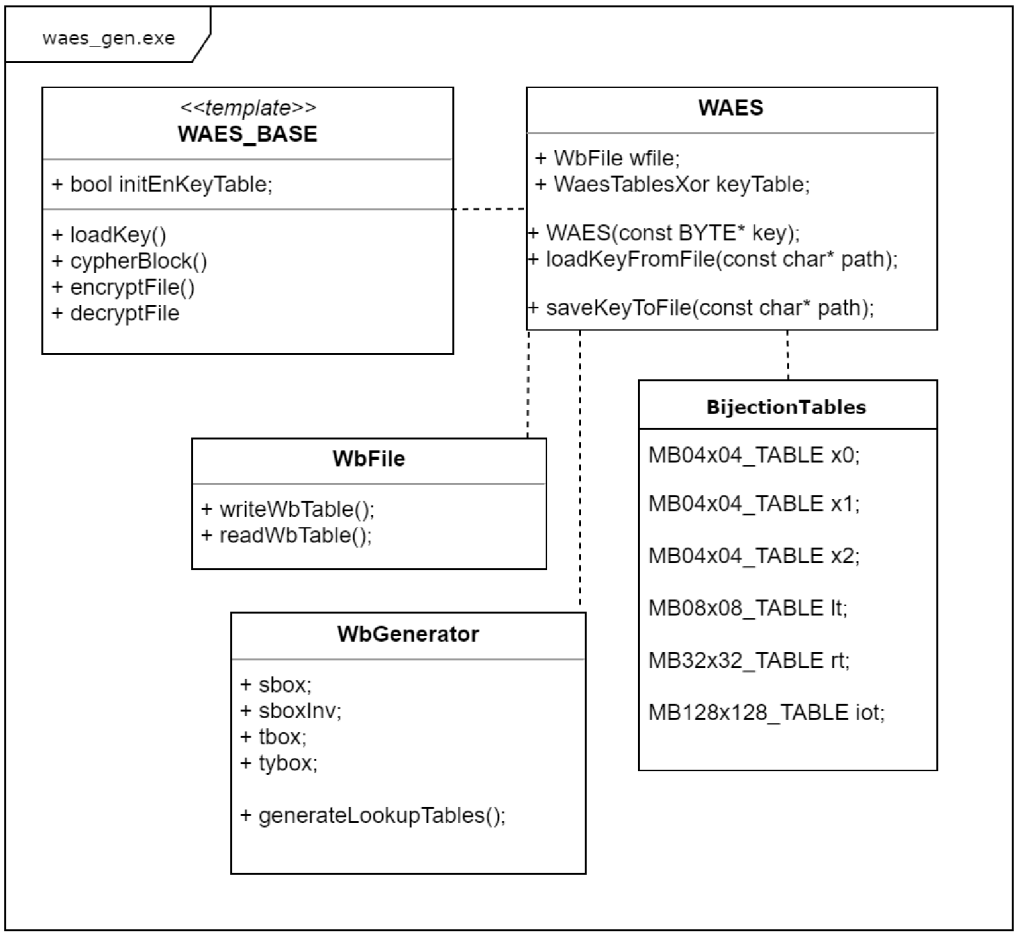
Tabuľka 4.2: Počet možných interpretácií vyhľadávacích tabuliek v implementácií algoritmu white-box AES.

4.3.4 Integrácia white-boxu s ransomvérom

Pre vytvorenie whitebox ransomvéru je implementovaný generátor šifrovacieho a dešifrovacieho white-box AES algoritmu so zadaným kľúčom šifrovania. V prvom kroku je nutné vygenerovať šifrovacie a dešifrovacie tabuľky, ktoré sa ukladajú do binárneho súboru. Tabuľky z binárneho súboru sú v ďalšom kroku uložené ako konštanty v hlavičkovom súbore, ktorý sa použije pri kompilácií white-box ransomvéru.

Pre kompiláciu white-box dekryptora je potrebné vykonať kompiláciu so špecifickou white-box AES tabuľkou. White-box generátor je vo výsledku použitý v rovnakej kostre ako pri implementácií konvenčného ransomvéru s tým rozdieľom, že vo funkcií na prechádzanie súborov v zadanej ceste na obfuskáciu sa využíva namiesto klasickej AES šifrovacej metódy, metóda skompilovaného white-boxu. Zjednodušený diagram implementovaných tried white-box AES generátora je mžné vidieť na obrázku 4.6.

Pri implementácií tried generátora a white-box AES šifrovacích a dešifrovacích metód boli použité funkcie z matematickej knižnice NTL[38], ktorá bola vygenervaná verzií 9.1.1 pre použitie s Windows OS.



Obr. 4.6: Zjednodušený diagram tried a metód white-box generátoru.

4.4 Implementácia metódy dávkového šifrovania

Implementácie tejto metódy sa snaží dosiahnuť vysokú entropiu rozloženia šifrovacieho procesu v čase, čo je dosiahnuté rozdelením šifrovania súborov do jednotlivých dávok ako bolo navrhnuté v kapitole 3.4. Rozdelenie šifrovania na jednotlivé dávky je implementované pomocou makra *nanosleep()* zo štandardnej knižnice *time.h*. Randomizácia intervalov je implementovaná s využitím funkcie *CryptGenRandom()* z windows knižnice *wincrypt.h*.

4.5 Implementácia ECB metódy obfuskácie

Kostra implementácie tejto metódy je rovnaká ako v implementovanom konvenčnom ransomvéri s tým rozdielom, že pre šifrovanie súborov je využitý algoritmus AES v ECB móde šifrovania. AES ECB funkcia šifrovania je využitá z externej knižnice[22] zahrnutej do kompilácie programu.

Kapitola 5

Vyhodnotenie efektivity obfuskačných techník

Táto záverečná kapitola práce popisuje testovanie implementovaných metód a vyhodnotenie ich efektívnosti voči konvenčnému typu ransomvéru. Jadrom kapitoly je vyhodnotenie detekcie implementovaných metód (vo forme ransomvéru) vzhľadom k ich dĺžke behu a počtu zašifrovaných užívateľských súborov.

5.1 Metodika testovania

Implementácia konvenčného ransomvéru sa v následovných kapitolách vyskytuje ako binárny súbor s názvom *ransom.exe*, implementácia ransomvéru s metódou white-box šifrovania ako *waesransom.exe*, metóda s využitím ECB módu šifrovania vo forme ransomvéru ako *ecbransom.exe* a ransomvér s metódou dávkového šifrovania ako *batchransom.exe*.

Testované binárne súbory s implementovanými metódami boli spúštané vo virtuálnom prostredí so 64 bitovým operačným systémom *Windows 10 Enterprise* bežiacom na hypervízori *VMWare Workstation* verzie 12. Prostredie využívalo 4 jadrá procesoru *Intel(R) Core(TM) i7-7820HQ CPU @ 2.90GHz* a 4GB operačnej pamäte.

Pre každý z použitých detekčných nástrojov bola použitá vlastná inštancia prostredia s rovnakou štruktúrou súborov v súborovom systéme. Implementované binárne súbory sa zameriavali na prehľadávanie súborov v dokumentoch aktuálne prihláseného užívateľa v systéme, kde bolo umiestnených celkovo 2500 rôznych súborov pre otestovanie záchytu metód. Štruktúra súborov bola následovná: 725 fotografií, 1232 zdrojových súborov programu a 543 MS Office súborov vo formáte Excel, Word a PowerPoint.

Pre beh C&C serveru bolo použité virtuálne prostredie s operačným systémom *CentOS 7* so sieťovým rozhraním v rovnakej podsieti ako sa nachádzali rozhrania testovacích staníc. Podsieť mala taktiež prístup k internetu pre možné cloudové detekčné procedúry nástrojov.

5.2 Použité detekčné nástroje

Pre otestovanie záchytu implementovaných metód obfuskácie činnosti ransomvéru bolo použitých 6 najpoužívanejších anti-vírových a anti-ransomvérových nástrojov v najnovších verziách vzhľadom k dobe vzniku tejto práce. Každý nástroj bežal v pozadí vlastnej inštancie virtuálneho prostredia. Použité nástroje:

- **Antivírus Kaspersky** vo verzií Small Office Security 6;

- **Nástroj Malwarebytes** vo verzií Premium 3.7.1;
- **Nástroj Hitman** od spoločnosti Sophos vo verzií Pro v3.7.9.;
- **Antivírus Avast** vo verzií Internet Security 19.5;
- **Antivírus Eset** vo verzií NOD32 v12.1.34.0;
- **Antivírus McAfee** vo verzií Endpoint Security v10.6.0.542 so SystemCore verziou 15.8.0.412 a verziou Threat prevention 10.6.0.672 .

5.3 Vyhodnotenie efektívnosti implementovaných metód

Vyhodnotenie efektívnosti implementovaných metód pre obfusáciu činnosti ransomvéru prebiehalo na základe dvoch faktorov: doby behu testovaného programu a počtu úspešne "znehodnotených"(šifrovaných) testových súborov prostredia. Šifrované súbory, pre ktoré v systéme existovala aj obnovená (AV nástrojom) alebo nevymazaná verzia (zablokovanie mazania AV nástrojom) pôvodného súboru sa do výsledkov testovania nerátali.

Nástroj	**Kaspersky Small Office Security 6**	počet zašifrovaných súborov
ransom.exe	3,9s	4
waesransom.exe	4,3s	2
ecbransom.exe	5,6s	6
batchransom.exe	9,2s	3
výsledok	PDM:Trojan.Win32.Generic	

Tabuľka 5.1: Výsledky testovania s AV Kaspersky end-point security.

Prvý použitý nástroj od výrobcu Kaspersky, Small Office Security 6, bol schopný eliminovať činnosť konvenčného ransomvéru v krátkom čase (viď. tabuľku 5.1) od štartu programu z čoho 2 sekundy trvala komunikácia s C&C serverom. Činnosť *waesransom.exe* bol schopný eliminovať v rovnakom čase ako činnosť predchádzajúcej metódy, avšak počet zašifrovaných súborov bol o polovicu menší, čo bolo spôsobené menšou výkonnosťou šifrovacieho algoritmu white-box AES. Metóda s využitím entropie dát zašifrovala v teste najviac súborov, čo bolo spôsobené rýchlym behom ECB módu šifrovania avšak AV Kaspersky nevyhodnotil pomer entropie šifrovaných súborov a pôvodných súborov za nevýznamný a tento beh taktiež vyhodnotil ako škodlivý. Posledná metóda *batchransom.exe* trvala pred odhalením najdlhšie, čo bolo spôsobené pomerne veľkými náhodnými intervalmi šifrovania (málo zašifrovaných súborov) ešte v prvej dávke šifrovania.

Nástroj	**Malwarebytes Premium 3.7.1**	počet zašifrovaných súborov
ransom.exe	16,6s	92
waesransom.exe	2min 37,4s	130
ecbransom.exe	22,7s	123
batchransom.exe	1min 15s	48
výsledok	Malware.Ransom.Agent.Generic	

Tabuľka 5.2: Výsledky testovania s nástrojom Malwarebytes.

Nástroj Malwarebytes sa v testoch (viď. tabuľka 5.2) záchytu choval podobne ako v popise chovania záchytu nástrojom Kaspersky, s rozdielom v metóde použitej v *waesransom.exe*, kde bola metóda pomerne úspešnejšia. Metóda použitá v *batchransom.exe* bola odhalená taktiež ešte v prvej dávke behu šifrovania.

Nástroj	**Hitman Pro v. 3.7.9**	počet zašifrovaných súborov
ransom.exe	4,1s	1
waesransom.exe	4,1s	81
ecbransom.exe	1min, 4,1s	214
batchransom.exe	8,2s	1
výsledok	potenciálny ransomvér	

Tabuľka 5.3: Výsledky testovania s nástrojom Hitman.

Výsledky záchytu nástrojom Hitman (viď. 5.3) boli principiálne rovnaké ako výsledky záchytu v predchádzajúcom nástroji. Rozdiel boľ v implementácií *ecbransom.exe* (kde menšia entropia šifrovaného súboru je menšia ako pri ostatných metódach), ktorej sa podarilo zamaskovať svoju činnosť na najdlhšiu dobu medzi prechádzajúcimi testami.

Nástroj	**Avast Internet Security 19.4**	počet zašifrovaných súborov
ransom.exe	4,3s	0
waesransom.exe	5,2s	0
ecbransom.exe	3,8s	0
batchransom.exe	7,2s	0
výsledok	potenciálny ransomvér	

Tabuľka 5.4: Výsledky testovania s AV nástrojom Avast Internet Security.

Antivírový nástroj od Avastu bol pre prácu najneúspešnejším, kde sa mu podarilo zachytiť všetky implementované metódy v pomerne krátkom čase s nulovou súborovou stratou (viď. tabuľku 5.4).

Nástroj	**Eset NOD32 v12.1.34.0**	počet zašifrovaných súborov
ransom.exe	6min 43s	všetky
waesransom.exe	26min 3s	všetky
ecbransom.exe	6min 52s	všetky
batchransom.exe	6hod 23min	všetky
výsledok	nedetekové	

Tabuľka 5.5: Výsledky testovania s AV Eset NOD32.

Nástroj	**McAfee Endpoint Security v10.6.0.542**	počet zašifrovaných súborov
ransom.exe	6min 58s	všetky
waesransom.exe	25min 4s	všetky
ecbransom.exe	7min 12s	všetky
batchransom.exe	6hod 12min	všetky
výsledok	nedetekové	

Tabuľka 5.6: Výsledky testovania s AV McAfee.

Posledným dvom nástrojom (viď. tabuľku výsledkov 5.5 a 5.6) sa nepodarilo detekovať činnosť konvenčného ransomvéru ani činnosť implementovaných metód.

Kapitola 6

Záver

Táto diplomová práca na základe analýzy spôsobov detekcie ransomvéru, spôsobov šifrovania a znehodnotenia súborov na napadnutom systéme navrhuje nové metódy obfuskácie činnosti ransomvéru, so zameraním na vlastnosti entropie dát, ktoré nespadajú do detekčných možností známych anti-ransomvérových a anti-vírových nástrojov. Navrhnuté metódy boli implementované pre použitie v prostredí Windows a následne vyhodnotené pri ich činnosti vo vykonaných funkčných testoch. Techniky sa zameriavajú na zmenu činnosti ransomvéru vo fáze znehodnotenia (šifrovanie alebo obfuskácia) súborov na napadnutom systéme.

Vo výsledkoch testov práca zaznamenala rozdiely v čase detekcie a rozdiely v počtoch zašifrovaných súborov pre každú implementovanú metódu podľa predpokladov. Rozdieli vo výsledkoch testov medzi jednotlivými antivírovými a anti-ransomvérovými nástrojmi boli spôsobené ich rozdielnymi detekčnými metódami, kde každý z nástrojov využíva vlastné know-how a vlastné vzorce pre vytváranie behaviorálnych znakov činnosti malvéru.

Vo výsledkoch testovania obfuskačných metód prekvapivo vyčnievajú výrobcovia Eset a McAfee, ktorý sa v rebríčkoch hodnotenia spoločnosti Gartner[17] pohybujú na prvých priečkach porovnávaných anti-malvérových riešení. Práca ako celok tvorí analýzu, metódy a nástroje, ktorých princíp a funkčnosť môžu testovať rôzni poprední lídri anti-vírových nástrojov pri zdokonaľovaní svojich behaviorálnych detekčných metód anti-ransomvérových nástrojov.

Literatúra

[1] Joan Daemen and Vincent Rijmen. The Design of Rijndael. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002: .

[2] Julien Bringer, Hervé Chabanne, and Emmanuelle Dottax. White box cryptography: Another attempt. IACR Cryptology ePrint Archive, 2006:468, 2006. URL http://dblp.uni-trier.de/db/journals/iacr/iacr2006.htmlBringerCD06a: .

[3] K. You and I. Yim: *"Malware Obfuscation Techniques: A brief survey"*. Internation conference on Broadband, Wireless Computing Communication and Application, 2016.

[4] A. Kharraz, W. Robertson, D. Balzarotti, L. Bilge and E. Kirda: *"Cutting the Gordian Knot, a Look under the Hood of Ransomware Attacks"*. Lecture Notes on Computer Science, vol. 9148, 2015, 3-24 s.

[5] Amin Kharaz, Sajjad Arshad, Collin Mulliner, William Robertson, and Engin Kirda. 2016. UNVEIL: A Large-Scale, Automated Approach to Detecting Ransomware. In Proc. 25th USENIX Security Symp. (USENIX Security '16): .

[6] Andrea Continella, Alessandro Guagnelli, Giovanni Zingaro, Giulio De Pasquale, Alessandro Barenghi, Stefano Zanero, and Federico Maggi. 2016. ShieldFS: A Self-healing, Ransomwareaware Filesystem. In Proc. 32nd Annu. Conf. Comput. Security Applicat. (ACSAC '16): .

[7] Andrey Bogdanov, Takanori Isobe, and Elmar Tischhauser. 2016. Towards Practical Whitebox Cryptography: Optimizing Efficiency and Space Hardness. In Proc. 22nd Int. Conf. Theory and Application of Cryptology and Inform. Security (ASIACRYPT'16: .

[8] Belcher, P.: "Sofos - Invincea,". Jun 2016, [Online].
URL https://www.invincea.com/2016/06/hash-factory-new-cerber-ransomware-morphs-every-15-seconds/

[9] Ben Herzog, Yaniv Balmas: "Great Crypto Failures". Jun 2016, [Online, accessed 13.1.2019].
URL https://blog.checkpoint.com/wp-content/uploads/2016/10/GreatCryptoFailuresWhitepaper_Draft2.pdf

[10] Christian S. Collberg, Clark D. Thomborson, and Douglas Low. Breaking Abstractions and Unstructuring Data Structures. In Proceedings of the 1998 International Conference on Computer Languages (ICCL 1998), IEEE Computer Society, pages 28–38, 1998: .

[11] Christoforus Juan Benvenuto: Galois Field in Cryptography. 2012, [Online, accessed 20.05.2019].
URL https://sites.math.washington.edu/~morrow/336_12/papers/juan.pdf

[12] Dai, W.: Crypto++ library. 2009, http://www.cryptopp.com (accessed 2019-02-07).

[13] De, S.; Bhaumik, J.: An AES-based Robust Image Encryption Scheme. *International Journal of Computer Applications*, ročník 109, 01 2015: s. 29–34, doi:10.5120/19243-0987.

[14] Dušan KLINEC: White-box attack resistant cryptography. 2013, [Online, accessed 14.1.2019].
URL https://is.muni.cz/auth/th/325219/fi_m/thesis.pdf

[15] ECRYPT Stream Cipher Project. AES-CTR benchmark performance. http://www.ecrypt.eu.org/stream/perf/pentium-m/benchmarks/aes-ctr/aes-128/: .

[16] Felix Gröbert, Carsten Willems, and Thorsten Holz. 2011. Automated Identification of Cryptographic Primitives in Binary Programs. In Proc. 14th Int. Conf. Recent Advances in Intrusion Detection (RAID '11): .

[17] Gartner, Inc.: Reviews for Endpoint Protection Platforms (EPP). 2019, [Online, accessed 20.03.2019].
URL
https://www.gartner.com/reviews/market/endpoint-protection-platforms/

[18] Hamilton E. Link and William D. Neumann. Clarifying obfuscation: Improving the security of white-box des. In Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume I - Volume 01, ITCC '05, pages 679–684, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2315-3. doi: 10.1109/ITCC.2005.100. URL http://dx.doi.org/10.1109/ITCC.2005.100.: .

[19] Hiroki Yamauchi, Yuichiro Kanzaki, Akito Monden, Masahide Nakamura, and Ken ichi Matsumoto. Software obfuscation from crackers' viewpoint. In Proceedings of the 2nd IASTED international conference on Advances in computer science and technology (ACST 2006), pages 286–291, Anaheim, CA, USA, 2006. ACTA Press: .

[20] I. Sorokin. Comparing files using structural entropy. Journal in Computer Virology, 7(4), 2011: .

[21] James Xiao and Yongxin Zhou. Generating large non-singular matrices over an arbitrary field with blocks of full rank. Cryptology ePrint Archive, Report 2002/096, 2002. http://eprint.iacr.org/: .

[22] Jason Lee: A C++ library of encryption algorithms. 2013, [Online, accessed 15.03.2019].
URL https://github.com/calccrypto/Encryptions

[23] K. Butler, N. Scaife, H. Carter and P. Traynor: *"CryptoLock (and Drop it): Stopping Ransomware Attacks on User Data"*. Internation Conference on Distributive Computing Systems„ 2016.

[24] L. Robert: “We Live Security”. 22 Dec 2016, [Online].

[25] Lee, J.; Lee, J.; Hong, J.: How to Make Efficient Decoy Files for Ransomware Detection? 09 2017, s. 208–212, doi:10.1145/3129676.3129713.

[26] Marc Beunardeau, Aisling Connolly, Remi Geraud, and David Naccache. 2016. White-box cryptography: Security in an insecure environment. IEEE Security Privacy 14, 5 (2016), 88–92.: .

[27] Matthias Jacob, Dan Boneh, and Edward W. Felten. Attacking an obfuscated cipher by injecting faults. In Joan Feigenbaum, editor, Digital Rights Management Workshop, volume 2696 of Lecture Notes in Computer Science, pages 16–31. Springer, 2002. ISBN 3-540-40410-4. URL http://dblp.uni-trier.de/db/conf/ccs/ccsdrm2002. htmlJacobBF02: .

[28] Mohamed Karroumi. Protecting white-box AES with dual ciphers. In Proceedings of the 13th international conference on Information security and cryptology, ICISC'10, pages 278–291, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-24208-3. URL http://link.springer.com/chapter/10.1007%2F978-3-642-24209-0_19: .

[29] Nolen Scaife, Henry Carter, Patrick Traynor, and Kevin R.B. Butler. 2016. CryptoLock (and Drop It): Stopping Ransomware Attacks on User Data. In Proc. 36th Int. Conf. Distributed Computing Syst. (ICDCS '16): .

[30] Olivier Billet, Henri Gilbert, and Charaf Ech-Chatbi. Cryptanalysis of a white box AES implementation. In Proceedings of the 11th international conference on Selected Areas in Cryptography, SAC'04, pages 227–240, Berlin, Heidelberg, 2005. Springer-Verlag. ISBN 3-540-24327-5, 978-3-540-24327-4. doi: 10.1007/978-3-540-30564-4_16: .

[31] R. Lyda and J. Hamrock. Using entropy analysis to find encrypted and packed malware. IEEE Security and Privacy, 5(2), 2007: .

[32] R. Perdisci, A. Lanzi, and W. Lee. Classification of packed executables for accurate computer virus detection. Pattern recognition letters, 29(14), 2008: .

[33] SHAMIR, Adi a Nicko VAN SOMEREN. Playing hide and seek with stored keys. 1998. Dostupne z: https://www.cs.jhu.edu/~astubble/600.412/s-c-papers/keys2.pdf: .

[34] Stanley Chow, Phil Eisen, Harold Johnson, and Paul C. van Oorschot. 2003. A White-Box DES Implementation for DRM Applications. In Proc. ACM Workshop on Digital Rights Manage.(DRM '02): .

[35] Stanley Chow, Philip Eisen, Harold Johnson, and Paul C. Van Oorschot. 2003. White-Box Cryptography and an AES Implementation. In Proc. Int. Workshop Select. Areas in Cryptography (SAC '02): .

[36] Tim Kerins and Klaus Kursawe. A cautionary note on weak implementations of block ciphers. In 1st Benelux Workshop on Information and System Security (WISSec 2006), page 12, Antwerp, BE, 2006: .

[37] Vadim Kotov Mantej Singh Rajpal: *Understanding Crypto-Ransomware*. Bromium, Nov 2014.

[38] Victor Shoup: NTL: A Library for doing Number Theory. [Online, accessed 20.02.2019].
URL https://www.shoup.net/ntl/

[39] Wil Michiels and Paul Gorissen. Mechanism for software tamper resistance: an application of white-box cryptography. In Proceedings of the 2007 ACM workshop on Digital Rights Management, DRM '07, pages 82–89, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-884-8. doi: 10.1145/1314276.1314291. URL http://doi.acm.org/ 10.1145/1314276.1314291: .

[40] Wyseur, B.: *White-Box Cryptography*. Boston, MA: Springer US, 2011, ISBN 978-1-4419-5906-5, s. 1386–1387, doi:10.1007/978-1-4419-5906-5_627.
URL https://doi.org/10.1007/978-1-4419-5906-5_627

[41] Wyseur, Brecht. White-Box Cryptography. 2009. Dostupné z: https://www.cosic.esat.kuleuven.be/publications/thesis-152.pdf: .

[42] Wyseur, Brecht. White-box cryptography: Hiding keys in software. 2012. Dostupne z: http://whiteboxcrypto.com/files/2012$_m$*isc.pdf* : .

[43] Yaying Xiao and Xuejia Lai. A secure implementation of white-box AES. In Computer Science and its Applications, 2009. CSA '09. 2nd International Conference on, pages 1–6, 2009. doi: 10.1109/CSA.2009.5404239: .

[44] Yin Jia, TingTing Lin, and Xuejia Lai. 2016. A generic attack against white box implementation of block ciphers. In Proc. Int. Conf. Comput. Inform. and Telecommun. Systems (CITS '16): .

[45] Yoni De Mulder, Brecht Wyseur, and Bart Preneel. Cryptanalysis of a perturbated white-box AES implementation. In Guang Gong and Kishan Chand Gupta, editors, INDOCRYPT, volume 6498 of Lecture Notes in Computer Science, pages 292–310. Springer, 2010. ISBN 978-3-642-17400-1. URL http://dblp.uni-trier.de/db/conf/ indocrypt/indocrypt2010.htmlMulderWP10: .

[46] Yoni De Mulder, Peter Roelse, and Bart Preneel. Cryptanalysis of the Xiao - Lai whitebox AES implementation. In Lars R. Knudsen and Huapeng Wu, editors, Selected Areas in Cryptography, volume 7707 of Lecture Notes in Computer Science, pages 34–49. Springer, 2012. ISBN 978-3-642-35999-6. URL http://dblp.uni-trier.de/db/conf/ sacrypt/sacrypt2012.htmlMulderRP12: .