VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

REALIZACE ÚTOKU NA MASKOVANÝ ŠIFROVACÍ ALGORITMUS

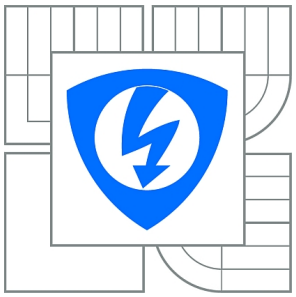
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. RADKA JAKUBÍKOVÁ
AUTHOR

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

REALIZACE ÚTOKU NA MASKOVANÝ ŠIFROVACÍ ALGORITMUS

POWER ANALYSIS ATTACK ON MASKED AES IMPLEMENTATION

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

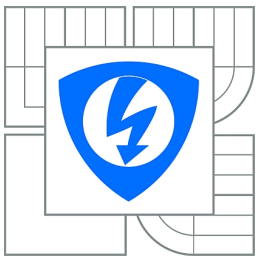
AUTOR PRÁCE — Bc. RADKA JAKUBÍKOVÁ

AUTHOR

VEDOUCÍ PRÁCE — Ing. ZDENĚK MARTINÁSEK, Ph.D.

SUPERVISOR

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Studentka: Bc. Radka Jakubíková ***ID:*** 125463
Ročník: 2 ***Akademický rok:*** 2014/2015

NÁZEV TÉMATU:

Realizace útoku na maskovaný šifrovací algoritmus

POKYNY PRO VYPRACOVÁNÍ:

V rámci diplomové práce nastudujte problematiku proudové analýzy. Seznamte se se soutěží DPA contest 4 http://www.dpacontest.org a nastudujte a zprovozněte všechny potřebné nástroje pro realizaci a vyhodnocení útoku (soutěž v4.2). Podrobně nastudujte použitou metodu maskování algoritmu AES. S pomocí proudových průběhů navrhněte a realizujte kompletní útok na toto maskovací schéma a odešlete útok do soutěže na zhodnocení. Dosažené výsledky přehledně zpracujte.

DOPORUČENÁ LITERATURA:

[1] Mangard, S.; Oswald, E.; Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security). Secaucus, NJ, USA:Springer-Verlag New York, Inc., 2007, ISBN 0387308571.

[2] Kocher, P. C.; Jaffe, J.; Jun, B.: Differential Power Analysis. In CRYPTO ’99: Proceedings of the 19th Annual International Cryptology Conference onAdvances in Cryptology, London, UK: Springer-Verlag, 1999, ISBN 3-540-66347-9, s. 388–397.

Termín zadání: 9.2.2015 ***Termín odevzdání:*** 26.5.2015

Vedoucí práce: Ing. Zdeněk Martinásek, Ph.D.
Konzultanti diplomové práce:

doc. Ing. Jiří Mišurec, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Autor diplomové práce nesmí při vytváření diplomové práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

ABSTRAKT

Šifrovací algoritmy jsou dnes běžně používaným zabezpečovacím prvkem. V některých situacích je algoritmus provozován na speciálním modulu, aby nedocházelo k útokům pomocí internetu. Ovšem i na šifrovací modul může být zaútočeno a to pomocí útoku postranním kanálem. Díky nejrůznějším typům analýz proudových průběhů modulu se tajná data ocitají v nebezpečí. Útoky postranním kanálem využívají především znalosti šifrovacího algoritmu a jednoduché nebo diferenciální proudové analýzy. Diplomová práce se zaměřuje na útok, který by bylo možné uskutečnit diferenciální proudovou analýzou pro data uveřejněná v soutěži DPA Contest. Práce popisuje nejen různé techniky analýz a typy útoků, ale také novou implementaci DPACv4.2, pro kterou zároveň vypracovává korelační analýzu. Na jejím základě je diskutován možný typ útoku pro DPACv4.2.

KLÍČOVÁ SLOVA

šifrovací modul, útok postranním kanálem, diferenciální proudová analýza, jednoduchá proudová analýza, korelační analýza, soutěž DPA

ABSTRACT

The cryptographic algorithms are commonly used as a security item today. In some situations, the special device is used to run the cryptographic algorithm, so the data are protected against the attack from the internet. Naturally, the attack can be loaded on the device as well using the side channel attack. The data are under the great danger, because nowadays plenty of power consumption analyses exist. The side channel attack uses knowledge about the cryptographic algoritm and simple or differential analysis. The diploma thesis focuses on the differential power analysis attack for the data published under the DPA contest. This thesis covers different types of analyss and attacks, and describes the new DPACv4.2 implementation. The correlation analysis is presented for the DPACv4.2 and the possible attack is discussed at the conclusion.

KEYWORDS

cryptographic device, side channel attack, differential power analysis, simple power analysis, correlation analysis, DPA contest

JAKUBÍKOVÁ, Radka *Realizace útoku na maskovaný šifrovací algoritmus*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2014/2015. 55 s. Vedoucí práce byl Ing. Zdeněk Martinásek, PhD.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Realizace útoku na maskovaný šifrovací algoritmus“ jsem vypracovala samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autorka uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušila autorská práva třetích osob, zejména jsem nezasáhla nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědoma následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Ráda bych poděkovala vedoucímu diplomové práce panu Ing. Zdeňku Martináskovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

(podpis autora)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .
(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Bezpečnost dat je dnes velice důležitá. K ochraně se využívá nejrůznějších šifrovacích algoritmů. Nejpoužívanějším algoritmem je AES (Advanced Encryption Standard). Jako americký standard je v popředí zájmu odborníků, kteří se snaží o jeho vylepšení. Na druhou stranu jsou vylepšovány i útoky proti němu.

AES může být implementován také na šifrovacím modulu, který může podléhat útoku postranním kanálem. Jedná se o útok analyzováním proudové spotřeby modulu. Nejčastěji používané analýzy jako Jednoduchá proudová analýza – SPA nebo Diferenciální proudová analýza – DPA budou v práci detailněji popsány. Zároveň budou uvedeny různé typy útoků založené na těchto analýzách.

Útok postranním kanálem je jedním z nejobávanějších útoků na šifrovací moduly. Pokud budeme předpokládat, že je na něm implementován algoritmus AES, je často k útoku použita metoda DPA. Pro objektivní porovnání úspěšnosti útoku byla zahájena tzv. DPA soutěž – DPA Contest (DPAC). Jejím cílem je ohodnotit různé útoky na stejná data. Před půl rokem byla zveřejněna nejnovější verze – 4.2, která přichází s novou vylepšenou implementací AES algoritmu. Ta obsahuje více opatření proti útoku nežli předešlé verze. Třetí kapitola pojednává o všech těchto změnách a zároveň popisuje způsob zpracování dat, která lze pro útok použít.

Cílem práce je analyzovat DPACv4.2, popsat změny oproti předchozí verzi, instalovat potřebné programy pro práci s daty DPACv4.2, analyzovat proudové průběhy zveřejněné v soutěži a lokalizovat v nich zajímavé body. K samotnému útoku nebylo z výpočetní náročnosti analýz a časových důvodů přistoupeno, v závěrečné části práce bude diskutován možný způsob útoku.

1 KRYPTOGRAFIE

Kryptografie je nauka o implementaci speciálních algoritmů, za pomoci kterých je možné dosáhnout utajení nejrůznějších zpráv. Tento pojem spadá pod celý vědní obor, který se nazývá kryptologie. Druhou disciplínou kryptologie je kryptoanalýza, jejímž cílem je rozluštit šifry a zjistit tak sílu kryptografických algoritmů.

Cílem kryptografie je použití takových matematických vzorců a funkcí, které slibují splnění těchto tří bodů: důvěrnost, integritu a autentičnost přenášených dat. Tyto algoritmy se pak používají v procesu šifrování. Vstupními parametry procesu je zpráva a kryptografický klíč. Výstupem se stává šifrovaný text, který vznikl aplikací kryptografického algoritmu na vstupní hodnoty. Opačný proces pak nazýváme dešifrování. V dnešní době jsou používané šifrovací algoritmy veřejně známé a přístupné, k zjištění původní zprávy je potřeba najít správný klíč.

Kryptografické systémy

- Symetrická kryptografie – vysílací a přijímací strana sdílí jeden tajný klíč. Tento druh systému je typický pro dvoubodový spoj. Využívá se například ve vojenství. Nejznámějším a nejvíce používaným algoritmem je Advanced Encryption Standard (AES), podrobnější popis v příloze A. Jedná se o blokovou šifru pro niž jsou velikosti bloků dat pevně stanoveny. AES je schopný využívat šifrovací klíče délky 128, 192 a 256-ti bitů na šifrování a dešifrování 128-mi bitového toku dat.
- Asymetrická kryptografie – obě komunikující strany používají jiný klíč. Při šifrování používá vysílací strana tzv. veřejného klíče, jehož princip není utajován. Jeho dešifrování je časově a technicky náročné. Přijímací strana následně zprávu dešifruje svým soukromým klíčem. Mezi nejpopulárnější systémy tohoto druhu dnes patří Rivest-Shamir-Adleman (RSA) algoritmus. Podstata zabezpečení je v obtížnosti rozkladu násobku dvou velkých prvočísel. [21]

Každý šifrovací algoritmus je v dnešní době znám, víme jeho postup a funkce. Úroveň jeho zabezpečení se hodnotí podle toho, jak obtížné je zjistit kryptografický klíč z veřejně dostupných dat, například z původní zprávy nebo z šifrovaného textu. Za výpočetně bezpečný algoritmus považujeme takový, k jehož prolomení potřebujeme velký výpočetní výkon, v praxi nedostupný.

Obyčejný stolní počítač dostačuje svým výpočetním výkonem k využívání šifrovacích algoritmů. Nevýhodou je jeho připojení k síti Internet, díky které může být software poškozen nejrůznějšími druhy virů. Z tohoto důvodu není počítač vhodným prostředkem k ukládání takto důležitých a cenných informací jako jsou šifrovací klíče. Uzavřené platformy nabízejí větší bezpečnost pro uchovávání těchto cenných

informací. Obecně takové zařízení nazýváme kryptografický modul. Může jím být například USB – Universal Serial Bus token nebo bezkontaktní Radio Frequency Identification – RFID čip. [14]

Při používání kryptografických modulů musíme opět uvažovat o možném útoku. Útočník může sledovat přenos mezi modulem a výpočetním zařízením a na základě těchto dat získat kryptografický klíč. K ohodnocení úrovně zabezpečení je nutné zjistit, jaké znalosti o modulu útočník vlastní. Je tedy zřejmé, že kryptografický modul se nesmí spoléhat pouze na utajení vlastního algoritmu.

1.1 Útoky na kryptografické moduly

V posledních letech se objevily nejrůznější způsoby útoků na výše zmíněné kryptografické moduly. Techniky, jak získat potřebné informace, se od sebe zásadně liší ve své časové náročnosti, vybavení, znalosti útočníka a v neposlední řadě jde také o cenu použitého útočného systému. Útoky na kryptografické moduly lze kategorizovat několika způsoby. Nejčastěji se setkáváme s rozdělením podle dvou kritérií. Prvním z nich je posouzení útoku podle aktivity [14]:

- Aktivní útok – znamená využití různých prostředků na ovlivnění chování kryptografického modulu. Díky tomu je útočník schopný zjistit tajný klíč.
- Pasivní útok – je realizován pouhým sledováním fyzických změn na kryptografickém modulu. Útočník nezasahuje přímo do procesu, který se děje uvnitř modulu, ale snaží se klíč odhalit pomocí jeho chování, které může pozorovat.

Druhým kritériem pro kategorizaci útoků je rozhraní, které je využito pro útok. Může se jednat o fyzické nebo logické rozhraní kryptografického modulu. Rozlišujeme tyto útoky [14]:

- Invazivní útoky – jedná se o nejefektivnější druh útoku, který existuje. V podstatě zde nejsou stanoveny žádné limity. Útočník se v první řadě snaží dostat pod kryt kryptografického modulu a tak získat přístup k jednotlivým komponentům modulu. Poté může použít různých typů sond na pozorování datových přenosů. Tuto část útoku můžeme považovat za pasivní. Jakmile útočník zasáhne do signálových přenosů a tím ovlivní funkci modulu, můžeme útok nazývat aktivní. Z popisu vyplývá, že se jedná o velmi účinný útok. Na druhou stranu je velmi obtížné jej realizovat neboť potřebné vybavení je příliš drahé.
- Semi-invazivní útoky – u tohoto druhu se opět setkáváme s úplným odkrytím kryptografického modulu, ale již se nejedná o přímé spojení se základní deskou. Cílem pasivního útoku je přečíst obsah paměťových článků. Úkolem aktivního útoku se stává přímé ovlivnění funkce modulu. Chybné informace mohou být zavedeny do zařízení například světelnými paprsky, elektromagnetickým po-

lem nebo rentgenovým zářením. Semi-invazivní útoky nevyžadují tak složité a drahé vybavení jako je tomu u invazivních typů, ale stále jsou tyto útoky náročné na čas a znalosti.

- Neinvazivní útoky – útočník sleduje pouze rozhraní modulu, vůbec nezasahuje do stavby modulu. V tomto případě nedochází ke spojení se základní deskou, k jejímu měření nebo k zjišťování paměti modulu. Pro útok jsou využity pouze přímo dostupná rozhraní. Modul není vystavený stálým změnám a tak tento útok nezanechává žádné stopy. Neinvazivní útok je velmi nebezpečný, protože může být realizován běžně dostupnými prostředky. Stejně jako předchozí útoky jej můžeme dělit na aktivní a pasivní. Pasivním se stává tehdy, pokud útočník měří proudovou spotřebu modulu, elektromagnetické pole nebo časový průběh při procesu šifrování. Na základě naměřených veličin pak útočník může zjistit kryptografický klíč. Tyto útoky jsou známy pod pojmem útok postranním kanálem. Dělíme je podle měřené veličiny: časový postranní kanál, proudový postranní kanál a útok elektromagnetickým kanálem. Neinvazivní útok je aktivní v případě změny některého parametru, který je důležitý pro správnou činnost modulu, jako je napájecí napětí, časový průběh nebo teplota zařízení.

Práce se dále zabývá pouze útoky pomocí proudového postranního kanálu. Jedná se o analýzu proudové spotřeby modulu, která je získána při výměně dat mezi kryptografickým modulem a jeho okolím. Proudová spotřeba není konstantní a vyjadřuje práci logických buněk, ze kterých je složen obvod v kryptografickém modulu. Proudovou analýzou je možné zjistit jaké operace modul prováděl.

2 TYPY ÚTOKŮ POMOCÍ PROUDOVÉ ANALÝZY

Útok pomocí proudové analýzy využívá informace o okamžitém stavu proudové spotřeby kryptografického modulu v závislosti na právě zpracovávaných datech. Jak musí útočník postupovat, aby našel kryptografický klíč modulu, je popsáno v následující kapitole.

2.1 Jednoduchá proudová analýza

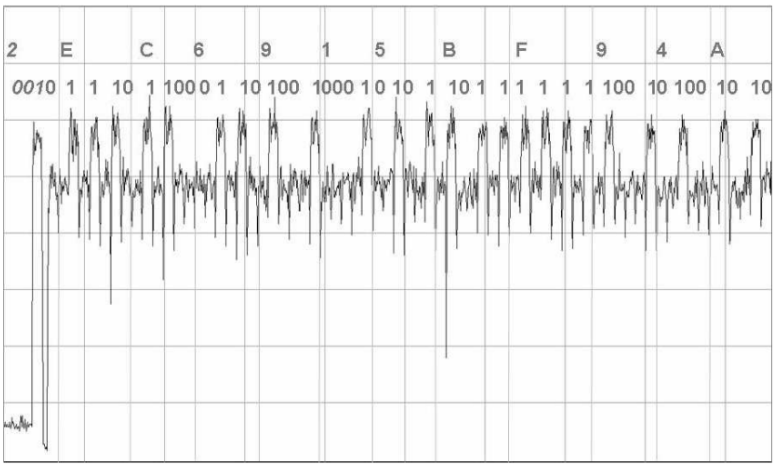
Simple Power Analysis (SPA) attack neboli útok pomocí jednoduché proudové analýzy spočívá v odhalení šifrovacího klíče přímo z proudového průběhu kryptografického modulu. Útočník potřebuje znát jednu nebo více proudových stop pro daná vstupní data a zároveň přesnou implementaci kryptografického algoritmu v modulu. Pokud zná pouze jeden průběh, je možné využít komplexních statistických metod.

Každý algoritmus, který je spuštěn na kryptografickém modulu, je vypočítáván v jednotlivých sekvencích. Jednotlivé operace jsou přeloženy do instrukcí, které provádí šifrovací modul. Existují případy, kdy lze přímo z naměřeného proudového průběhu modulu odhadnout sled jeho operací. Jako příklad je na obr. 2.1 uveden průběh proudu naměřený při výpočtu RSA algoritmu „square and multiply“ 2.1.

Algoritmus 2.1 Algoritmus RSA - square and multiply

```
x, n, c = (c_{k-1}, c_{k-2}, ..., c_1, c_0)   // x - zpráva, n - modul, c - šifrovací klíč
z = 1                                          // z - výstup
for i = k - 1 to 0 do
    z = z^2 mod n                              // operace square
    if c_i = 1 then
        z = (z * x) mod n                      // operace multiply
    end if
end for
Return z
```

Lze jednoduše rozlišit dva výrazné vzory v grafu 2.1 – vysoká a nízká úroveň. Pokud víme, že algoritmus RSA „square and multiply“ 2.1 provádí umocňování pro každý bit, a jestliže se objeví bit s hodnotou 1, počítá se také násobení, není těžké odvodit jakým výpočtům odpovídají jednotlivé úrovně proudu. Nižší úroveň značí operaci umocňování – odpovídá bitu s hodnotou 0, a vyšší úroveň operaci násobení – bit s hodnotou 1. [11]



Obr. 2.1: Graf proudové spotřeby RSA „square and multiply“ [11]

Z příkladu je patrné, že algoritmus obsahující pouze násobení a umocňování, je náchylný na SPA útok. Dalším ze známých algoritmů, které je touto metodou možné prolomit, může být Data Encryption Standard – DES [5].

2.1.1 Útoky pomocí šablon

Metoda je založena na závislosti zpracovávaných dat na proudovém průběhu kryptografického modulu. Útok je složen se dvou částí. V první, tzv. profilující, si útočník připraví šablonu – profiluje zařízení, k tomu potřebuje vlastnit identické zařízení. Druhou fází je útok, při kterém je zaznamenán proudový průběh. Tento nový průběh je porovnán s šablonou vytvořenou v profilující fázi. Cílem je najít nejpodobnější průběh a zjistit tak použitý klíč, případně jinou informaci, pro kterou jsou šablony připraveny (např. Hammingovu váhu vybrané operace).

Šablona představuje statistické vlastnosti každého proudového průběhu, jenž existuje pro všechny možné operace a klíče. Pokud použijeme model vícerozměrného normálního rozdělení, je možné šablonu charakterizovat jako vektor středních hodnot s kovarianční maticí $(\mathbf{m}, \mathbf{C})$.

Pokud je n různých operací a pro každou z nich je naměřeno p proudových průběhů označených $t_1, \ldots, t_p$, vektor středních hodnot může být určen následovně:

$$\mathbf{m} = \frac{1}{p} \sum_{j=1}^{p} t_j, \tag{2.1}$$

přitom všechny průběhy t_j patří výpočtu jedné operace. Druhou částí šablony je kovarianční matice šumu, která představuje kovarianci mezi více než dvěma náhodnými hodnotami:

$$\mathbf{C}(u, v) = cov(N_u, N_v), \tag{2.2}$$

kde N je vektor šumu pro proudový průběh jedné operace.

Vyhodnocení nejlepšího proudového průběhu, tedy nalezení takové šablony, která nejlépe odpovídá naměřeným datům při útoku, může být následující:

- Pro každou šablonu, jenž je určena vždy pro jeden klíč, se vypočítá proudový průběh šumu N_i, který odpovídá útoku používající stejný klíč jako šablona: $N_i = \mathbf{m_i} - t$.
- Pravděpodobnost N_i pro všechny šablony lze vypočítat pomocí kovarianční matice každé šablony:

$$p(N_i) = \frac{1}{\sqrt{(2\pi)^n |\mathbf{C_i}|}} \cdot \mathrm{e}^{\frac{1}{2} N_i^T \mathbf{C}_i^{-1} N_i}. \tag{2.3}$$

Teoreticky by měla šablona s nejvyšší pravděpodobností odpovídat průběhu šumu. Pokud víme, že každá šablona je určena vždy pro jeden klíč, můžeme z vybrané šablony s nejvyšší pravděpodobností odečíst i tajný klíč, který byl při šifrování použit. [19]

2.2 Diferenciální proudová analýza

K nejvíce používaným útokům patří útoky pomocí Diferenciální proudové analýzy neboli Differential power analysis (DPA) attacks. Tato metoda nevyžaduje znalost kryptografického modulu, pouze potřebuje k prolomení klíče několik proudových průběhů, čímž se liší od SPA útoku.

Dalším velkým rozdílem mezi SPA a DPA útokem je způsob zpracovávání proudových průběhů. Při použití SPA útoku, pozoruje útočník průběh podél časové osy, snaží se vyhledat vzor a přiřadit ho šabloně jednoho proudového průběhu. V případě DPA útoku analyzuje útočník pouze část proudového signálu a sleduje jeho závislost na toku dat. Útočník tedy potřebuje znát velké množství proudových průběhů, aby mohl analyzovat proudovou spotřebu modulu v daném čase, jako funkci zpracovávaných dat. Dále kapitola pojednává o DPA útoku a jeho základním postupu, který se skládá z pěti kroků [14], jednoduchý útok lze vyzkoušet pomocí demo verze uvedené v [7]:

1. Volba mezivýsledku kryptografického algoritmu

Na začátku DPA útoku je třeba vybrat mezivýsledek právě vykonávaného šifrovaného algoritmu. Tento výsledek musí být funkcí $f(d, k)$, kde proměnná d zastupuje známá vstupní data a druhá proměnná k vyjadřuje malou část šifrovacího klíče. Mezivýsledek, který splňuje tyto požadavky pak může sloužit k odhalení proměnné k, tedy části klíče například prvního bajtu.

2. Měření proudové spotřeby modulu

Ve druhém kroku měří útočník proudovou spotřebu kryptografického modulu při procesu šifrování a dešifrování různých datových bloků D. Zároveň musí útočník znát odpovídající data d, která jsou použita při výpočtu mezivýsledku v kroku prvním. Známá data zapisujeme do vektoru $\mathbf{d} = (d_1, \ldots, d_D)$, kde d_i znamená i-tý průběh šifrovacího nebo dešifrovacího procesu.

Proudový průběh je útočníkem zaznamenáván při každém průběhu šifrování nebo dešifrování. Tyto záznamy odpovídají datovým blokům d_i jako $\mathbf{t'}_i = (t_{i,1}, \ldots, t_{i,T})$, kde T značí délku proudové stopy. Proudová spotřeba je zapsána pro každý datový blok D, z čehož vyplývá zápis proudových stop do matice $\mathbf{T}$, která má velikost $D \times T$. Velmi důležitým bodem je správně spárovat datové bloky a proudovou spotřebu modulu, tedy aby odpovídal naměřený proudový signál stále stejným šifrovacím operacím. Tohoto výsledku může útočník dosáhnout správným nastavením měřicího zařízení nebo pomocí nejrůznějších programů.

3. Určení hypotézy mezivýsledků

V následujícím kroku útočník hledá nejdříve tzv. hypotézy šifrovacího klíče. Možnosti, které útočník má, se zapisují do vektoru $\mathbf{k} = (k_1, \ldots, k_K)$, kde index K představuje všechny volby klíčů k. Na základě známých dat d a hypotetických klíčů k může útočník vypočítat hypotetický mezivýsledek $f(d, k)$ pro všechny existující datové bloky D a všechny hypotézy šifrovacího klíče. Výsledkem je matice $\mathbf{V}$ rozměrů $D \times K$. Následující výpočet popisuje první krok v obrázku 2.2:

$$v_{i,j} = f(d_i, k_j); \quad i = 1, \ldots, D; \quad j = 1, \ldots, K. \tag{2.4}$$

Sloupec j matice $\mathbf{V}$ odpovídá mezivýsledkům, které byly vypočítány pro hypotézu klíče k_j. Každý sloupec matice $\mathbf{V}$ představuje výpočet mezivýsledků kryptografickým modulem pro odpovídající datový blok. V modulu je použita hodnota klíče, která je zároveň prvkem vektoru $\mathbf{k}$. Klíč, který byl modulem využit označíme indexem ck. Cílem DPA útoku se stává zjištění sloupce matice $\mathbf{V}$, který byl modulem zpracovaný během procesu šifrování a dešifrování. Na základě této informace může útočník najít klíč, použitý kryptografickým modulem, jenž odpovídá hodnotě k_{ck}.

4. Vzájemný vztah mezivýsledků a proudové spotřeby

Útočník musí dále určit vztah mezivýsledků a proudové spotřeby modulu – namapuje hypotetické výsledky z matice $\mathbf{V}$ na matici $\mathbf{H}$, hodnot hypotetických výsledků proudové spotřeby. K tomuto kroku využívá útočník simulace proudové spotřeby

modulu. Na konci je tedy každému hypotetickému mezivýsledku $v_{i,j}$ přiřazena právě jedna hodnota hypotetické proudové spotřeby $h_{i,j}$.

Úspěšnost DPA útoku velmi závisí na znalostech útočníka, jak dalece rozumí kryptografickému modulu a jaké má zkušenosti. Čím více se útočníkova simulace přibližuje skutečným hodnotám proudových signálů, tím má také útočník větší šance na úspěch. Nejčastějšími modely, které jsou využívány pro mapování matice $\mathbf{V}$ na $\mathbf{H}$, jsou Hammingova vzdálenost[1] a Hammingova váha[2].

5. Porovnání hypotetických a naměřených hodnot proudové spotřeby

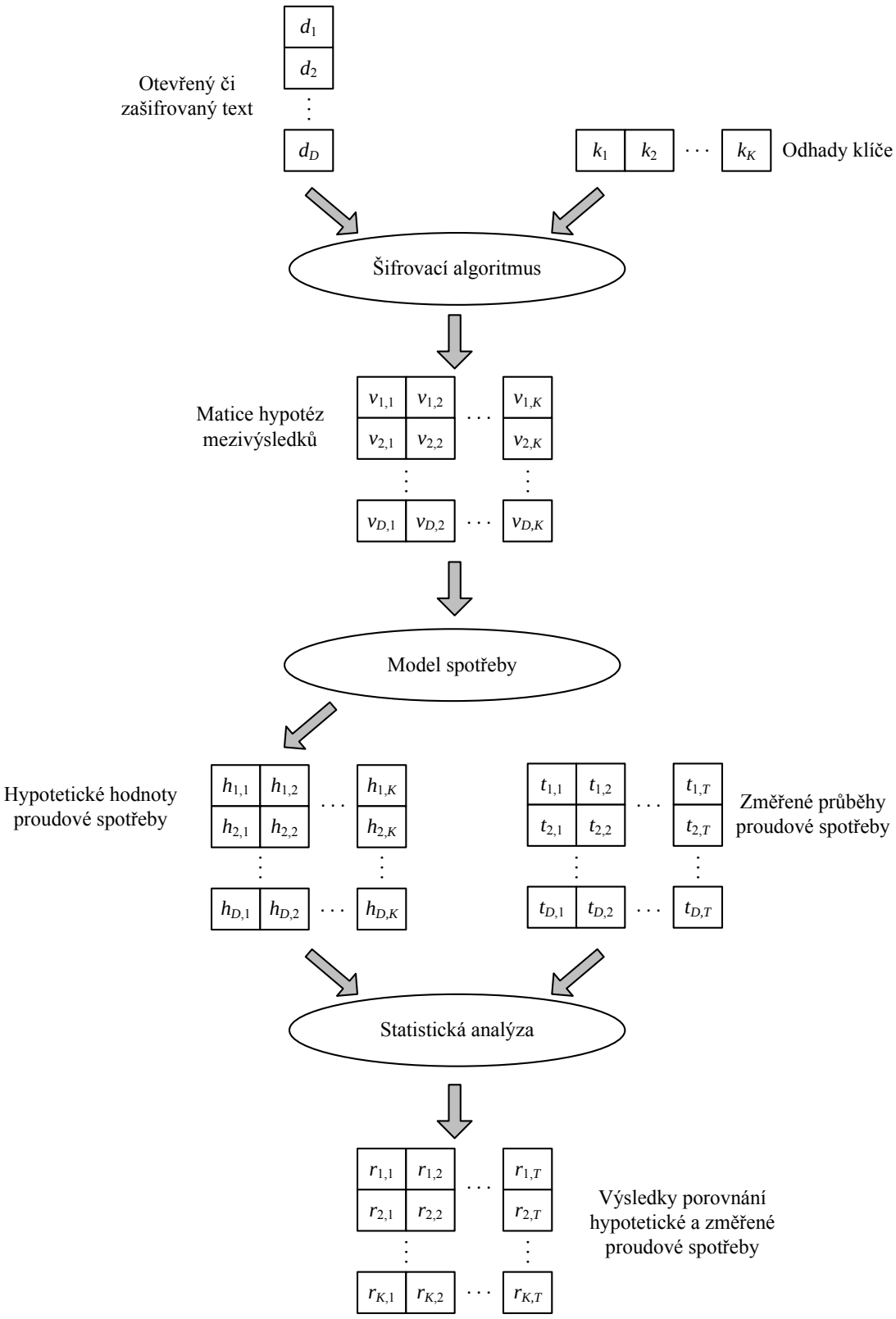
V závěrečném kroku provádí útočník porovnání hypotetických hodnot proudové spotřeby s naměřenými hodnotami proudových signálů pro každý hypotetický klíč. Jedná se o komparaci hodnot sloupce $\mathbf{h}_i$ matice $\mathbf{H}$ se sloupcem $\mathbf{t}_j$ matice $\mathbf{T}$(stopy signálové spotřeby modulu). Výsledek se ukládá do matice $\mathbf{R}$ o velikosti $K \times T$. Metody srovnávání těchto hodnot jsou popsány dále v této kapitole. Ovšem všechny algoritmy mají stejnou vlastnost, čím více se shodují hodnoty h_i a t_j, tím větší je hodnota $r_{i,j}$.

Kryptografický modul počítá v různých krocích šifrovacího procesu mezivýsledek $\mathbf{v}_{ck}$. Tudíž naměřené hodnoty proudového signálu závisí na těchto výsledcích v daných bodech měření. Tyto body označujeme *ct*. Pozice $\mathbf{t}_{ct}$ tedy obsahuje hodnotu proudové spotřeby závislou na hodnotě $\mathbf{v}_{ck}$. Hodnota $\mathbf{h}_{ck}$ v matici $\mathbf{H}$ byla simulována na základě $\mathbf{v}_{ck}$. Proto sloupce $\mathbf{h}_{ck}$ a $\mathbf{t}_{ct}$ mají velmi silnou vazbu. Tyto dvě hodnoty vedou k největší hodnotě v matici $\mathbf{R}$, $r_{ck,ct}$. Útočník tedy může ve výsledku najít pouze nejvyšší hodnotu matice $\mathbf{R}$ a tím tak dojít ke správnému klíči, který byl kryptografickým modulem použit.

Pokud se útočníkovi nepodaří vyhledat nejvyšší hodnotu matice $\mathbf{R}$, protože všechny její hodnoty jsou velmi podobné, musí naměřit více proudových průběhů. Jestliže má útočník málo záznamů a informací o proudové spotřebě a také malé zkušenosti, nemůže DPA útok nikdy úspěšně realizovat.

[1] Hammingova vzdálenost určuje počet míst, ve kterých se dvě zprávy liší. V modulu se jedná o počet přechodů mezi 0 a 1 a opačně. Výsledný počet přechodů se využívá k popisu proudové spotřeby modulu.

[2] Hammingova váha definuje počet nenulových míst ve dvojkové posloupnosti. V případě modulu to znamená, že proudová spotřeba je úměrná počtu datových bitů s logickou 1.



Obr. 2.2: Blokový diagram znázorňující DPA útok, 3.-5.krok [22]

2.2.1 Útoky založené na korelačním koeficientu

Korelace je pojem ze statistiky, který značí vzájemný lineární vztah dvou proměnných, nejčastěji X a Y. Míra korelace je pak vyjádřena korelačním koeficientem $\rho(X, Y)$, jenž může nabývat hodnot od $\langle -1; 1 \rangle$. Krajní případy korelačního koeficientu:

- $\rho(X, Y) = -1$ - závislost veličin je nepřímá, čím více se zvýší hodnoty v jedné skupině znaků, tím více se ve druhé skupině hodnoty sníží,
- $\rho(X, Y) = 0$ - mezi veličinami neexistuje lineární závislost, i přesto zde může být nějaký druh závislosti jenž není možné zapsat lineární funkcí,
- $\rho(X, Y) = 1$ - jedná se o přímou závislost veličin, nazýváme ji také jako dokonalá korelace.

Vzorec pro vyjádření korelačního koeficientu:

$$\rho(X, Y) = \frac{cov(X, Y)}{\sigma_X \sigma_Y}, \tag{2.5}$$

kde platí, že druhé momenty náhodných veličin X a Y jsou konečné a $cov(X, Y)$ označuje kovarianci X a Y. Při dosazení konkrétních hodnot$((x_1, y_1), (x_2, y_2), \dots)$ do předchozího vzorce 2.5, dostáváme výběrový korelační koeficient:

$$r = \frac{\sum_{i=1}^{n} (x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2 \cdot \sum_{i=1}^{n} (y_i - \bar{y})^2}}. \tag{2.6}$$

V případě DPA útoku je korelační koeficient použit k určení lineární závislosti mezi sloupci $\mathbf{h}_i$ a $\mathbf{t}_j$ a výsledkem je matice $\mathbf{R}$ odhadovaných korelačních koeficientů. Každá hodnota $r_{i,j}$ je odhadnuta na základě prvků datových bloků h_i a t_j a výsledek můžeme psát jako úpravu vzorce 2.6:

$$r_{i,j} = \frac{\sum_{d=1}^{D} (h_{d,i} - \bar{h}_i) \cdot (t_{d,j} - \bar{t}_j)}{\sqrt{\sum_{d=1}^{D} (h_{d,i} - \bar{h}_i)^2 \cdot \sum_{d=1}^{D} (t_{d,j} - \bar{t}_j)^2}}, \tag{2.7}$$

kde $\bar{h}_i$ a $\bar{t}_j$ představují střední hodnoty sloupců $\mathbf{h}_i$ a $\mathbf{t}_j$. [14]

2.2.2 Útoky založené na rozdílu středních hodnot

Základní myšlenkou tohoto druhu útoku je určení vzájemného vztahu matice $\mathbf{H}$ a $\mathbf{T}$. Matice $\mathbf{R}$ obsahuje binární hodnoty nul a jedniček. Tyto sekvence čísel jsou funkcí vstupních dat d a hypotetického klíče k_i. Útočník může zhodnotit správnost klíče k_i pomocí rozdělení matice $\mathbf{T}$ na dvě skupiny. První skupina bude složena z řádků hodnot proudových průběhů, řádky matice $\mathbf{T}$, které odpovídají pozicím ve vektoru $\mathbf{h}_i$ s hodnotami rovny nule. Všechny zbývající řádky matice $\mathbf{T}$ budou tvořit druhou skupinu. Následně je možné určit střední hodnotu řádků. Výsledek střední hodnoty

první skupiny si útočník zapíše do vektoru $\mathbf{m'}_{0i}$ a výsledek druhé skupiny do vektoru $\mathbf{m'}_{1i}$. Hypotetický klíč je určen správně, pokud je mezi hodnotami vektoru $\mathbf{m'}$ velký rozdíl v určitém časovém okamžiku.

Rozdíl mezi hodnotami $\mathbf{m'}_{0i}$ a $\mathbf{m'}_{1i}$ značí korelaci mezi vektorem $\mathbf{h}_{ck}$ a některým ze sloupců matice $\mathbf{T}$, to znamená, objevuje se zde lineární závislost mezi hodnotami h a t. Pro všechny ostatní vztahy je rozdíl středních hodnot víceméně nulový. Pokud se útočník setká s tím, že jeho výsledky rozdílů vektoru $\mathbf{m'}_{0i}$ a $\mathbf{m'}_{1i}$ jsou všechny blízké nule, znamená to pro něj, že jeho hodnoty hypotetického klíče nejsou správné.

Výsledkem této metody pro DPA útok je matice $\mathbf{R}$. Každý řádek matice $\mathbf{R}$ odpovídá rozdílu středních hodnot $\mathbf{m'}_{0i}$ a $\mathbf{m'}_{1i}$ pro jeden hypotetický klíč. Matice je určena pomocí těchto vztahů:

$$m_{1i,j} = \frac{1}{n_{1i}} \cdot \sum_{l=1}^{n} h_{l,i} \cdot t_{l,j}, \tag{2.8}$$

$$m_{0i,j} = \frac{1}{n_{0i}} \cdot \sum_{l=1}^{n} (1 - h_{l,i}) \cdot t_{l,j}, \tag{2.9}$$

$$m_{1i,j} = \sum_{l=1}^{n} h_{l,i}, \tag{2.10}$$

$$m_{0i,j} = \sum_{l=1}^{n} (1 - h_{l,i}), \tag{2.11}$$

$$\mathbf{R} = \mathbf{M_1} - \mathbf{M_0}, \tag{2.12}$$

ve kterých je n počet řádků matice $\mathbf{H}$. [14]

2.2.3 Útoky založené na vzdálenosti středních hodnot

Tato metoda přináší určitá vylepšení oproti dříve zmíněné metodě rozdílu středních hodnot. Útočník při tomto druhu útoku počítá i s odchylkami středních hodnot. Metoda je založena na testu určujícím, zda jsou střední hodnoty pro dvě statistická rozdělení stejná či nikoliv.

Útočník v tomto případě opět rozdělí matici $\mathbf{T}$ na dvě skupiny, jako tomu bylo u předchozího útoku 2.2.2, ale v dalším kroku neprovede rozdíl středních hodnot, místo toho použije test na porovnání vzdálenosti středních hodnot. Výsledkem se opět stává matice $\mathbf{R}$, jejíž prvky jsou vypočítány následujícím způsobem:

$$r_{i,j} = \frac{m_{1i,j} - m_{0i,j}}{s_{i,j}}, \tag{2.13}$$

kde prvek $s_{i,j}$ zastupuje směrodatnou odchylku pravděpodobnostního rozdělení dvou skupin. [14]

3 DPA CONTEST

Kryptografické moduly bývají častěji vystaveny možnému útoku, který nazýváme útok postranním kanálem. Útočník používá k odhalení klíče různých metod, viz 2.2. Jednou z nejpoužívanějších se stává DPA útok. Kryptografové se snaží najít taková opatření, aby zabránili co možná nejvíce útokům postranními kanály. Z toho důvodu se snaží vystavit kryptografické moduly nejrůznějším typům útoků, aby zjistili, kde jsou ještě slabá místa použité implementace.

Ke zlepšení kryptografického algoritmu slouží několik výzkumů. Také existuje tzv. DPA soutěž (DPA contest), do které se mohou kryptoanalytici zapojit a vyzkoušet tak účinnost svého útoku, pomocí DPA metod, na proudové průběhy postranního kanálu, které jsou zveřejněny pořadateli soutěže. Tento typ soutěže vznikl v důsledku možnosti objektivně porovnat různé typy útoků. Pořadatelé soutěže tedy umožnili kryptoanalytikům přístup k šifrovaným datům, klíčům a zašifrovaným datům. Pokud různé útoky použijí stejná data, je možné je mezi sebou objektivně porovnat.

První verze DPA soutěže byla zveřejněna roku 2008 a zaměřila svůj útok na nechráněný algoritmus DES. Ve druhé verzi se jednalo o útok na nechráněný AES a třetí verze vznikla s cílem zjistit nejlepší nastavení při měření proudových toků. Čtvrtá verze DPA soutěže vyšla v roce 2013 a cílila na algoritmus AES-256 chráněný maskováním RSM – Rotating Sbox Masking, spuštěný na mikrokontroleru ATMEL AVR-163. [8]

3.1 RSM - Maskování pomocí rotace Sboxu

Běžnou ochranou šifer v softwarovém provedení je maskování a tzv. shuffling – „míchání“ [20]. Dalším opatřením může být i tzv. hiding – „skrývání“, který je používán méně. Hiding způsobuje, že je únik informací stejnoměrný a nezávislý na zpracovávaných datech. Shuffling je jednoduchým opatřením, jehož funkcí je náhodně přeházet operace při šifrování. Maskování používá náhodnou hodnotu – masku, která je přičtena k tajným datům.

V roce 2012 navrhli autoři [17] jednoduché opatření proti útoku postranním kanálem – Rotating Sbox Masking, který je typem Low-entropy masking scheme = LEMS – maskovací schéma s nízkou mírou entropie. Původně byl RSM navržen pro hardwarové implementace [17], ale později byl vylepšen také pro softwarové použití. Pořadatelé soutěže DPA contest se rozhodli právě pro RSM protiopatření, protože jej mají vědečtí odborníci v rámci DPAC již nastudován.

Cílem maskování je útočníkovi znemožnit odhad mezivýsledku kryptografického modulu z naměřené proudové spotřeby. Hodnoty na sobě nebudou již závislé,

protože maska vnese do mezivýsledků náhodnost. Při použití této metody nedochází ke změnám proudové spotřeby modulu.

Maska RSM je typu aditivního booleanovského maskování, jež používá operaci XOR a staticky maskovanou tabulku Sbox, konkrétněji viz priloha A.1. Proces maskování je přidán k algoritmu AES následně:

1. Je náhodně vygenerováno 16 masek M_i, $i = [0, 15]$, které jsou veřejné. Na počátku každého šifrování je zároveň náhodně vygenerován tzv. offset, který může nabývat hodnot $\langle 0; 15 \rangle$. Každý bajt i z pole stavů je maskován odpovídající maskou $M_{offset+i}$ pomocí operace XOR, kde součet *offset+i* odpovídá funkci *mod16*.

2. Sbox je nahrazen 16-ti maskovanými Sboxy, jež jsou předpočítány v prvním kroku:
$$MaskedSubBytes_i(X) = SubBytes(X \oplus M_i) \oplus M_{i+1}, \tag{3.1}$$
kde X je byte z pole stavů.
3. Maskované bajty jsou následně kompenzovány pomocí funkce XOR:
$$MaskCompensation_{offset} = Mask_{offset} \oplus MixColumns(ShiftRows(Mask_{offset})) \tag{3.2}$$
4. Pro poslední rundu je kompenzace odlišná, protože zde neprobíhá operace MixColumns():
$$MaskCompensationLastRound_{offset} = Mask_{offset} \oplus ShiftRows(Mask_{offset}). \tag{3.3}$$

Operace MaskedSubbytes() nejdříve používá 8 Sbox tabulek se sudými indexy a poté zbývajících 8 s lichými. Funkce $Mask_{offset}$ aplikuje šestnáct maskovaných bajtů na každý bajt z pole stavů podle vypočítaného indexu. Sada použitých masek pro DPACv4 je:

$$\begin{aligned} M_{i\in[0,15]} = \{&\text{0x00, 0x0f, 0x36, 0x39, 0x53, 0x5c, 0x65, 0x6a,} \\ &\text{0x95, 0x9a, 0xa3, 0xac, 0xc6, 0xc9, 0xf0, 0xff}\}. \end{aligned} \tag{3.4}$$

Upravený pseudokód pro použití masky je uveden algoritmem 3.1. Pokud bychom odstranili řádky psané modře a operaci MaskSubBytes() nahradili pouze SubBytes() dostali bychom opět kód pro základní algoritmus AES. [6]

Algoritmus 3.1 Maskování algoritmu AES-256 v DPA Contest V4.

Vstup: Otevřený text X; 16 bajtů X_i, kde $i \in \langle 0, 15 \rangle$,
Expanze klíče; 15 128bitových konstant $\mathsf{RoundKey}[r]$, kde $r \in \langle 0, 14 \rangle$

Výstup: Šifrovaný text X; 16 bajtů X_i, kde $i \in \langle 0, 15 \rangle$,

1: Náhodně generovaný $\mathsf{offset} \in \langle 0, 15 \rangle$ s rovnoměrným rozdělením.
2: $X = X \oplus \mathsf{Mask}_{\mathsf{offset}}$
3: // Maskování otevřeného textu
4: // Všechny rundy kromě poslední
5: **for** $r \in \langle 0, 12 \rangle$ **do**
6: $X = X \oplus \mathsf{RoundKey}[r]$
7: // AddRoundKey
8: **for** $i \in \langle 0, 15 \rangle$ **do**
9: $X_i = \mathsf{MaskedSubBytes}_{\mathsf{offset}+i+r}(X_i)$
10: **end for**
11: $X = \mathsf{ShiftRows}(X)$
12: $X = \mathsf{MixColumns}(X)$
13: $X = X \oplus \mathsf{MaskCompensation}_{\mathsf{offset}+1+r}$
14: **end for**
15: // Poslední runda
16: $X = X \oplus \mathsf{RoundKey}[13]$
17: **for** $i \in \langle 0, 15 \rangle$ **do**
18: $X_i = \mathsf{MaskedSubBytes}_{\mathsf{offset}+13+r}(X_i)$
19: **end for**
20: $X = \mathsf{ShiftRows}(X)$
21: $X = X \oplus \mathsf{RoundKey}[14]$
22: // Odmaskování šifrovaného textu
23: $X = X \oplus \mathsf{MaskCompensationLastRound}_{\mathsf{offset}+14}$

3.2 DPA verze 4.2

3.2.1 Shrnutí útoků provedených ve verzi DPACv4

Od zahájení DPACv4 v červenci roku 2013 bylo přijato a ohodnoceno 28 útoků. Všechny útoky a jejich výsledky je možné prohlédnout na [8]. Je možné je rozdělit do dvou skupin: profilující a neprofilující útoky. Některé z profilujících útoků jsou velmi efektivní, protože jejich implementace dokázala zjistit klíč již z prvního proudového průběhu. Neprofilující útoky potřebovaly minimálně čtrnáct proudových průběhů k získání šifry. V dalším textu budou rozebrány neprofilující typy útoků.

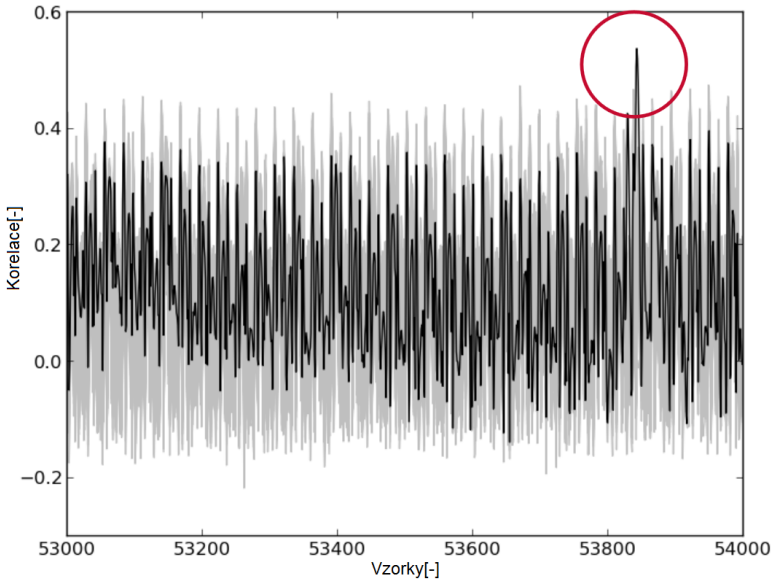
Útok navrhovaný Moradim [16] a jeho týmem je typu jednorozměrného korelačního proudového útoku, jímž poukazuje na zranitelnost díky základní designové chybě. Problém nastává v okamžiku, kdy vstup Sboxu x_i maskován m_i je přepsán ve stejném registru výstupem y_i maskovaným m_{i+1}. V registru tedy probíhá následující výpočet:

$$(x_i \oplus y_i) \oplus (m_i \oplus m_{i+1}). \tag{3.5}$$

V RSM je maska m_i a m_{i+1} vyvážená, ale složená maska z m_i a m_{i+1} již vyvážená není a způsobuje únik informací prvního řádu. Ten může vést k jednoduchému jednorozměrnému CPA.

Kanghong a jeho tým odhalil klíč ve dvou krocích z 69 proudových průběhů. V prvním kroku se útočník snaží uhodnout offset, který byl použit pro všechny klíče. Při tomto postupu bylo zjištěno, že velikost Hammingovy váhy pro masku $m_0 \oplus m_{15}$ je 8, zatímco Hammingova váha pro ostatní kombinace masek je 4. Tento rozdíl je možné pozorovat v proudovém průběhu a díky dočasnému lokálnímu maximu lze odhadnout velikost offsetu. V druhém kroku může útočník použít jednorozměrný CPA na všechny proudové průběhy se stejným offsetem a tak zjistit tajný klíč. Tuto zranitelnost objevili i další účastníci soutěže, princip získání offsetu byl přitom vždy jiný.

Stejný problém DPACv4 našel i tým Kutznera, ale zároveň zveřejnil útok, který využil jiné slabosti. Poukázal na vlastnost jenž je označována jako konstantní rozdíl RSM masek. Autoři zjistili, že rozdíl mezi maskami m_i a m_{i+8} je stále stejný, tedy $m_i \oplus m_{i+8}$ je konstantní. Klíč je možné nalézt pomocí korelačně vylepšeného srážkového útoku tzv. first-order correlation enhanced collision attack. Ve své podstatě tento útok nejdříve najde všechny proudové průběhy ve kterých dva Sboxy kolidují, následně je vypočítána korelace mezi dvěma časovými body S_i a S_{i+8} a na závěr je vybrána největší korelace, která znamená srážku - „collision“ a tedy správný klíč, v obrázku 3.1 označeno červenou kružnicí [12]. Jinými slovy, maska použitá Sboxem S_0 v první rundě je stejná jako v poslední rundě použitá Sboxem S_7. Do DPACv4 bylo přihlášeno hned několik dalších útoků, jenž využili předpovídané sekvence operací AES. [4] [13]



Obr. 3.1: Výsledek korelační analýzy vylepšeného srážkového útoku. [12]

V implementaci DPACv4 se objevily tyto 4 základní nedostatky:

1. Maska m_i a m_{i+1} je vyvážená, ale při operaci XOR dochází k jejich nevyváženosti.
2. Masky m_0 a m_{15} mají větší Hammingovu vzdálenost než ostatní masky, což způsobuje únik hodnoty offsetu.
3. V každé rundě algoritmu se offset konstantně zvyšuje a to umožňuje použití tzv. collision attack – srážkový útok.
4. Nezměněné a předvídatelné pořadí operací může také vést ke collision attack nebo k tzv. second order CPA – druhý stupeň útoku postranním kanálem, využívá k útoku druhého momentu – variance. [13]

3.2.2 Opatření pro vylepšení implementace DPACv4

V následujícím textu jsou popsána opatření, která byla učiněna pro verzi DPACv4.2 (vylepšená DPACv4), vedoucí k omezení zjištěných úniků informací v předešlé implementaci, jak bylo zmíněno výše.

Původní kód DPAv4 je psán v jazyku C a kompilován pomocí knihovny avr-gcc (veřejně dostupná knihovna pro Atmel AVR mikrokontrolér) k získání kódu v jazyku assembly (jazyk symbolických adres). V kódu jazyku C nenajedeme přímý výpočet součtu XOR hodnot m_i a m_{i+1}, ale při kompilaci pomocí avr-gcc se mohou tyto výpočty objevit. Knihovna avr-gcc opakovaně používá několik registrů pro optimalizaci designu. Uvažujme, že $x \oplus m_i$ je v registru a následuje hodnota $y \oplus m_{i+1}$, proudový průběh postranního kanálu bude v dalším kroku odpovídat hodnotě $x \oplus y \oplus m_i \oplus m_{i+1}$, jenž je nevyvážená hodnota.

Přímou cestou, jak odstranit tento nedostatek, je upravení kódu v jazyku symbolických adres, ovšem to je velmi zdlouhavý úkol, který může být doprovázen chybami. Obvyklý postup je napsat pouze makra kódu. Další opatření, které je třeba v tomto bodě udělat, je nastavení registru modulu na nulu vždy před každým vstupem nové hodnoty.

Většina útoků, která se účastnila DPACv4, využila úniku hodnoty offsetu pomocí rozdílné Hammingovy vzdálenosti (masky m_0 =0x00 a m_{15}=0xff mají Hammingovu vzdálenost 8). Jakmile útočník zná offset, může seřadit proudové průběhy podle stejného offsetu. Stejné hodnoty offsetu znamenají stejné hodnoty masek. Pokud jsou masky konstantní, považuje se implementace za AES bez maskování.

Jako řešení je navrženo použití náhodného offsetu pro každý Sbox. I když používáme náhodné offsety, základní výběr 16-ti masek zůstane nepozměněný. Náhodný offset je aplikován použitím náhodného pole s 16-ti nezávislými indexy. Toto pole slouží k nezávislému přiřazení masek k jednotlivým Sboxům. V této implementaci je možné použít stejnou masku k několika Sboxům. Tímto krokem lze předejít i tzv. collision attack (correlation enhanced collision attack), protože masky budou náhodně přiřazovány Sboxům, již nedojde ke stejnému rozdílu mezi maskami m_i a m_{i+8}. S tímto opatřením by mohlo souviset navýšení náročnosti výpočtu, protože množina masek (MaskCompensation) je velmi objemná na uložení v paměti. Řešením se stává výpočet masek za běhu algoritmu, což přináší jen časovou ztrátu, která je oproti navýšení náročnosti přijatelnější.

Byly zveřejněny také útoky vyšších řádů (second order CPA), které vyvolávají nutnost posílit úroveň zabezpečení DPACv4. Jednou z možností by mohla být úprava maskovacího schématu. Na druhou stranu je možné kombinovat různé typy zabezpečení a tak hlídat i navýšení náročnosti výpočtu na přijatelné úrovni. Pořadatelé DPACv4.2 nakonec zvolili možnost přidání další úrovně zabezpečení a vybrali tzv. shuffling – míchání. Jelikož hlavním cílem CPA útoku se stává první a poslední runda, je třeba zamíchat pořadí Sboxů pouze pro tyto dvě rundy. Pro výběr indexů Sboxů je použita permutace, ta způsobí náhodný výběr Sboxů pro první a poslední rundu a pro ostatní rundy jsou Sboxy volány stále stejně, nejdříve sudé a pak liché.

3.2.3 Implementace DPAv4.2

Tabulka 3.1 ukazuje srovnání implementace původní a vylepšené. Hodnoty v tabulce zahrnují i cenu operace KeyExpansion(), ovšem neuvažuje se s cenou generátoru náhodných čísel. Poslední sloupec – overhead vyjadřuje procentuální navýšení/pokles upravené implementace od původní. Z tabulky také vyplývá, že implementace je po úpravě kódu v assembly rychlejší. Operace maskování je prováděna ve speciál-

ním pořadí aby se tak předešlo některým horizontálním útokům[1], například klíč je nejdříve maskován náhodnou maskou předtím než je maskován nezašifrovaný text. Kód vylepšeného algoritmu je uveden v alg. 3.2.

Tab. 3.1: Porovnání cen implemetace DPAv4 a nové upravené verze [3]

Architektura	Původní verze	Upravená verze	Overhead
Velikost kódu (bajty)	11136	17847	60%
RAM (bajty)	8	12	50%
Počet cyklů	113600	16004	-86%

[1]Horizontální útok - je aplikován na stejné části tajné informace, která je použita v několika operacích, v průběhu celého algoritmu.

Algoritmus 3.2 Vylepšená implementace AES pro DPACv4.2

Vstup: Otevřený text X; 16 bajtů X_i, kde $i \in \langle 0, 15\rangle$,
Expanze klíče; 15 16-ti bajtových konstant RoundKey$[r]$, kde $r \in \langle 0, 14\rangle$
16 masek, maska je tvořena 8 bity, označeny Mask[]

Výstup: Šifrovaný text X; 16 bajtů X_i, kde $i \in \langle 0, 15\rangle$,

1: // Náhodně generovaný offset(s rovnoměrným rozložením) – 16 hodnot po 4 bitech
2: // Náhodně generované dvě shuffle funkce(s rovnoměrným rozložením) – Shuffle0 a Shuffle13:$[0, 15] \rightarrow [0, 15]$
3: RoundKey[0] $\leftarrow$ RoundKey[0] $\oplus$ Mask[offset[]]
4: // Všechny rundy kromě poslední
5: **for** $r \in \langle 0, 12\rangle$ **do**
6: $X = X \oplus$ RoundKey$[r]$ // AddRoundKey
7: **if** $r = 0$ **then**
8: **for** $i \in$ Shuffle0$([0, 15])$ **do**
9: $X_i =$ MaskedSubBytes$_{\text{offset[i]}+r}(X_i)$
10: **end for**
11: **else**
12: **for** $i \in [0, 15]$ **do**
13: $X_i =$ MaskedSubBytes$_{\text{offset[i]}+r}(X_i)$
14: **end for**
15: **end if**
16: $X =$ ShiftRows(X)
17: $X =$ MixColumns(X)
18: **for** $i \in [0, 15]$ **do**
19: MaskCompensation[i] =
20: MixColumns(ShiftRows(Mask[offset[i]+(r+1)])) $\oplus$ Mask[(offset[i]+(r+1))]
21: **end for**
22: $X = X \oplus$ MaskCompensation[]
23: **end for**
24: $X = X \oplus$ RoundKey[13] // Poslední runda
25: **for** $i \in$ Shuffle13$([0, 15])$ **do**
26: $X_{\text{shuffle}}[i] =$ MaskedSubBytes$_{\text{offset[i]}+13}(X_i)$
27: **end for**
28: $X =$ ShiftRows(X)
29: $X = X \oplus$ RoundKey[14]
30: **for** $i \in [0, 15]$ **do** // Odmaskování šifrovaného textu
31: MaskCompensationLastRound[i] = ShiftRows(Mask[offset[i]+14])
32: **end for**
33: $X = X \oplus$ MaskCompensationLastRound[]

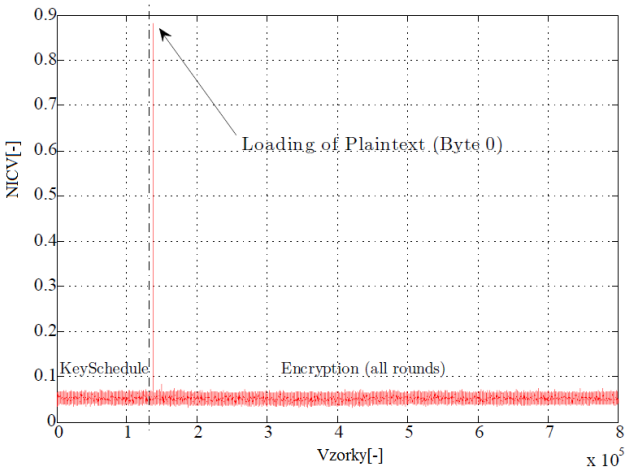
Cílem nové implementace DPAv4.2 bylo odstranit možné úniky prvního řádu. Ke kontrole byla použita detekční technika Normalized Inter-Class Variance (NICV) [2]. Tato metoda je schopna detekovat možné jednorozměrné úniky (first-order attack), je nezávislá na modelu a je počítána s ohledem na veřejně dostupné informace jako je nezašifrovaný nebo šifrovaný text. Vzorec pro určení NICV:

$$NICV = \frac{\mathsf{Var}[\mathbb{E}[Y|X]]}{\mathsf{Var}[Y]}, \tag{3.6}$$

kde Y odpovídá proudovým průběhům a X je vybraná část nezašifrovaného nebo šifrovaného textu.

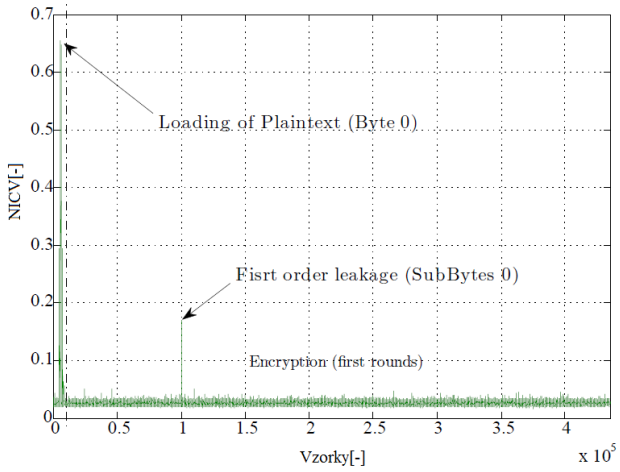
Vypočítaná NICV pro naměřené proudové průběhy v DPACv4.2 vzhledem k bajtu otevřeného textu je zobrazena na obr.3.2. Z grafu je zřejmé, že nedochází k žádnému jednorozměrnému úniku v části operace SubBytes() ani dále v průběhu šifrování. Z obrázku lze vyčíst pouze dvě velké špičky v inicializační části AddRoundKey(), které mohou znamenat únik. Pomocí metody CPA útoků bylo zjištěno, že tyto špičky odpovídají načítání nešifrovaného textu do různých částí paměti kryptografického modulu. Je jasné, že tato data neobsahují žádné informace o klíči a tak nejsou náchylná na útok.

V obrázku 3.3 je možné pozorovat NICV pro data z dřívější verze DPACv4. Zde byl únik, spojený s nešifrovaným textem, zřejmý již v části algoritmu SubBytes(). Tento únik je již velmi citlivý na možný útok.

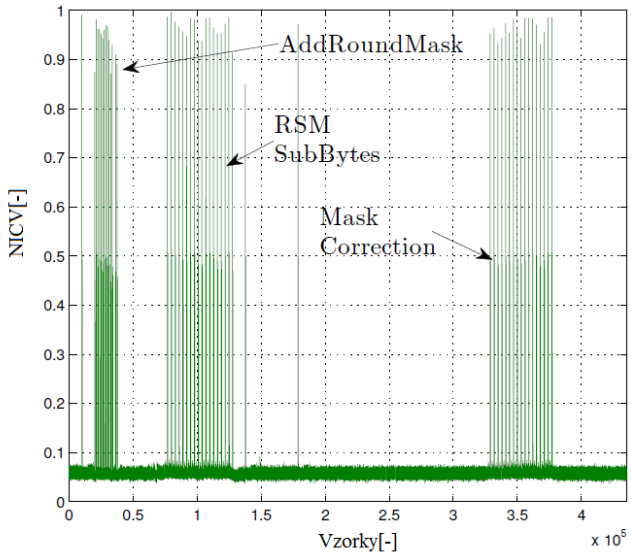


Obr. 3.2: NICV pro nově navrženou implementaci [3].

Offset je další hodnota, která je náchylná na útok. Určení NICV s ohledem na 4 bity offsetu, které byly použity v původní implementaci, nám umožňuje pozorovat 48 výrazných špiček viz obr.3.4. Tyto únikové body představují místa načítání indexů 4 bitů offsetu, které slouží k adresování maskovacích tabulek. Pokud únik nastává v bodě přenosu tajné informace, stačí pouze jeden takový bod a je možné odhalit celý klíč.



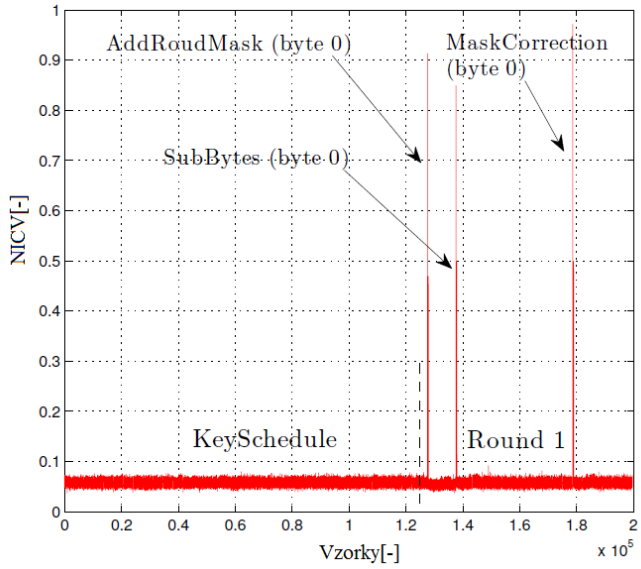
Obr. 3.3: NICV pro původní implementaci DPAv4 [3].



Obr. 3.4: NICV počítáno pro první 4 bity offsetu pro původní implementaci [3].

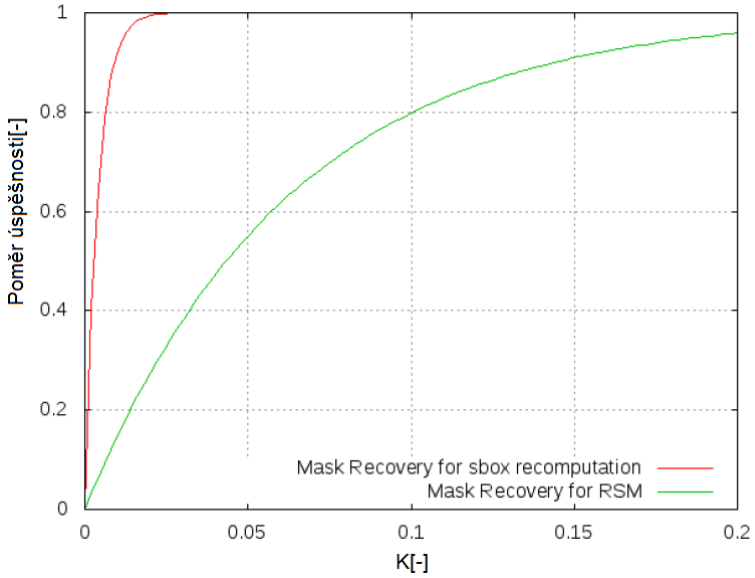
Vypočítaná NICV s ohledem na první 4 bity offsetu z 16-ti pro novou implementaci je na obr.3.5. Tři špičky signálu představují načítání indexu, který je použit ke čtení masky, Sboxu a maskovací korekční tabulky z paměti modulu. Tyto informace jsou náchylné na útok, protože určují bajt masky, který je použit k maskování klíče. Avšak není možné již na tento typ úniku aplikovat tzv. horizontální útok, protože každý bajt masky je vybrán jiným 4-bitovým offsetem.

Následující odstavec se zabývá porovnáním plně entropického Sbox maskování s vylepšenou RSM implementací, popsanou výše. V případě entropického Sbox maskování je 256 různých využitelných úniků, u RSM je jich pouze 16. Horizontální útok je v případě RSM schopný odhalit masku 16-ti bajtů stavů ze stavové matice, zatímco horizontální útok na přepočet Sboxu může objevit masku pouze jednoho bajtu stavu. V nové implementaci je použito náhodného offsetu k maskování kaž-



Obr. 3.5: NICV počítáno pro první 4 bity offsetu pro novou implementaci [3].

dého bajtu a tak není nadále možné využití horizontálního útoku. Pro takový typ útoku by bylo nutné odhadnout 16^{16} hodnot, což by bylo výpočetně velmi náročné. Použití míchání (shuffling) náročnost útoku ještě zvyšuje, protože by bylo nutné odhadnout 16! možných stavů nešifrovaného textu. Na obr.3.6 je vidět rozdíl poměru úspěšnosti odhalení masky mezi vylepšeným RSM a entropickým Sbox maskováním. Poměr úspěšnosti je dán vztahem $SR = 1 - \mathrm{e}^{-n \times k}$, kde n zastupuje počet průběhů a k je exponent prvního stupně. [3]



Obr. 3.6: Rozdíl poměru úspěšnosti odhalení masky pro vylepšenou implementaci s RSM a původní implementací [3]

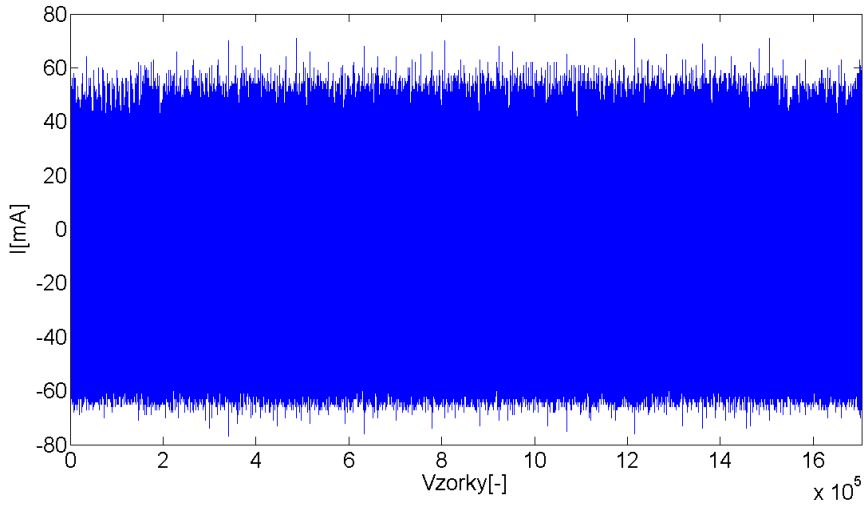
3.2.4 Zpracování proudových průběhů

DPA soutěž je přístupná na webových stránkách [8], kde je možné najít všechny potřebné informace o soutěži a jak se této soutěže zúčastnit. Na stránkách je možné najít všechny proudové průběhy, které byly naměřeny na mikrokontroleru ATMega-163. Tyto průběhy jsou uloženy po 1000 průbězích v souborech typu ZIP. Každý tento soubor má velikost 850MB, celkem je zpřístupněno 32 těchto souborů. Každý průběh má 1 704 402 vzorků. Zde se liší verze DPAv4 a DPAv4.2. Starší verze měla v jednom průběhu 435 002 vzorků. Rozdíl lze pozorovat na grafech 3.7 a 3.8.

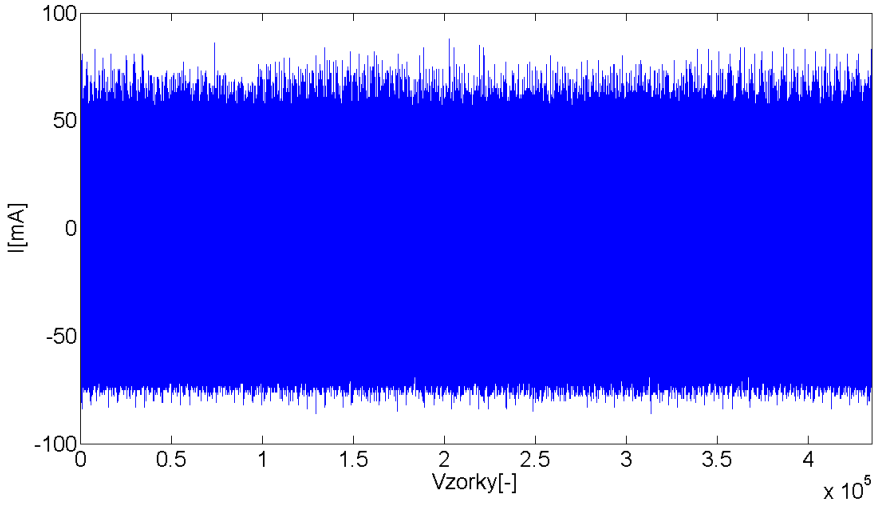
Dalším důležitým dokumentem je „index_file“. Ten obsahuje informace o klíči, otevřeném textu, šifrovaném textu, offsetu, jménu adresáře, ve kterém se proudové průběhy nacházejí, a jméno proudového průběhu. Přesně takto za sebou informace navazují na každém řádku v index_file, například:

```
1 8FCB29D88CCD2751E7F29B72F8BFB7FB EA4BA83CD2D8CB0B095B53A2D09B17B4
2 7A24A8271074BD6E0AACB0148C8AFA85 AB5D4F761C28E039 1A9406F3B2857CED
3 A85E0B321693F4D7 k00 DPACV42_00000.trc.bz2
```

Pro spuštění útoku je potřeba nástroj „Attack-Wrapper“, který slouží k načítání proudových průběhů, informací z index_file a spuštění útoku. Výsledky předává ke zpracování nástroji ComputeResults. ComputeResults vyhodnocuje úspěšnost útoku. Nástroj Attack-Wrapper je možné spustit na operačním systému Linux, Windows nebo MAC OS X. V každém systému se instaluje a spouští jinak. V následujícím textu bude popsán postup pro použití nástroje Attack-Wrapper v prostředí Linux.



Obr. 3.7: Proudový průběh pro DPACv4.2



Obr. 3.8: Proudový průběh pro DPACv4

Pro instalaci v OS Linux je nutné mít nainstalovaný kompilační balíček *g++* a knihovnu *libbz2*. Po stažení všech potřebných souborů a instalaci balíčků, je možné instalovat i nástroj Attack-Wrapper následujícími příkazy:

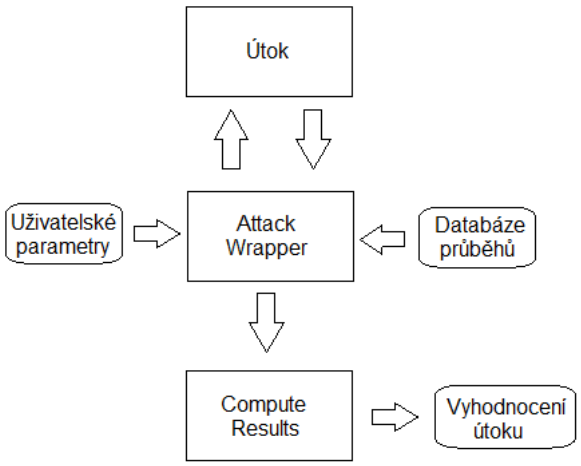
```
tar xzf attack_wrapper-2.1.1.tar.gz
cd attack_wrapper-2.1.1
./configure
make
```

Tento nástroj komunikuje s programem, který provádí útok. Na obrázku 3.9 je graficky znázorněna spolupráce Attack-Wrapper a programu. Spouštění nástroje Attack-Wrapper se provádí v příkazové řádce terminálu příkazem:

```
attack_wrapper -d DPA_contest2 -x index_file -f -e v4_2 fifo
```

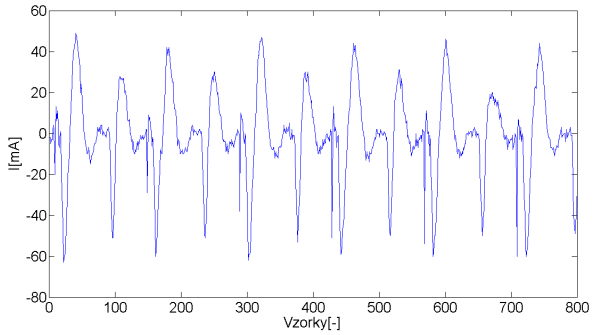
kde parametr *d* je adresář s proudovými průběhy, *x* je index_file, *f* použití FIFO módu a nakonec výběr verze DPA soutěže, v tomto případě DPAv4.2. Pro nápovědu o všech možných parametrech slouží příkaz:

```
attack_wrapper --help
```

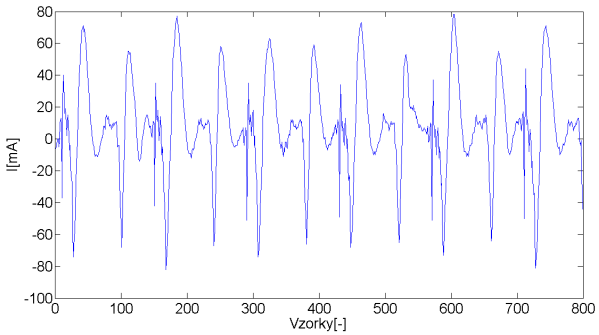


Obr. 3.9: Diagram funkce nástroje Attack-Wrapper. [10]

Po spuštění načítání dat v terminálu se spustí útok. Program, který obsahuje útok, může být vytvořen v jakémkoli z těchto jazyků: C, C++, Perl, Python, Matlab a jiné. V těchto programech lze také proudový průběh zpracovávat. Na obr. 3.10 lze vidět porovnání proudových průběhů pro obě poslední verze soutěže, je zobrazeno pouze 800 prvních vzorků.



(a) pro nově navrženou implementaci



(b) pro původní implementaci DPAv4

Obr. 3.10: Proudové průběhy DPA contest – prvních 800 vzorků jednoho průběhu

4 ANALÝZA PROUDOVÝCH PRŮBĚHŮ

V závěru předchozí kapitoly bylo popsáno, jakým způsobem se pracuje s daty, která jsou uveřejněna pod DPACv4.2. Proudové průběhy, které jsou dostupné, je potřeba analyzovat, aby bylo možné vybrat vhodný typ útoku. V následujícím textu je rozebrán postup analýzy a všechny její výsledky, které byly v rámci práce získány. Závěrem je diskutován možný způsob útoku v rámci DPACv4.2. Uvedená data byla zpracována v simulačním prostředí MATLAB.

4.1 Korelační analýza

Korelace vyjadřuje vzájemný vztah dvou proměnných, jak je popsáno v 2.2.1. Pomocí korelační analýzy proudových průběhů jsme schopni najít zajímavé body (špičky v korelační analýze), které určují, kdy kryptografický modul pracoval s daty, které vypočítáme z hypotetických hodnot. Korelují se známá data, naměřená proudová spotřeba, s hypotetickými, odhad klíče.

Pro korelaci tedy volíme jako hodnotu X proudový průběh, který je zveřejněn v DPACv4.2. Hodnota Y je hypotetická proudová spotřeba vypočítaná pomocí vybrané operace z algoritmu AES pro data známého šifrovaného textu, masek, offsetu a odhad klíče. Hypotetická data můžeme počítat pro různé kroky v algoritmu a pomocí korelace hledat, ve kterých momentech došlo k jejich použití při šifrování. Podrobněji popsáno v [15].

Zvolený postup pro analýzu dat byl následující:

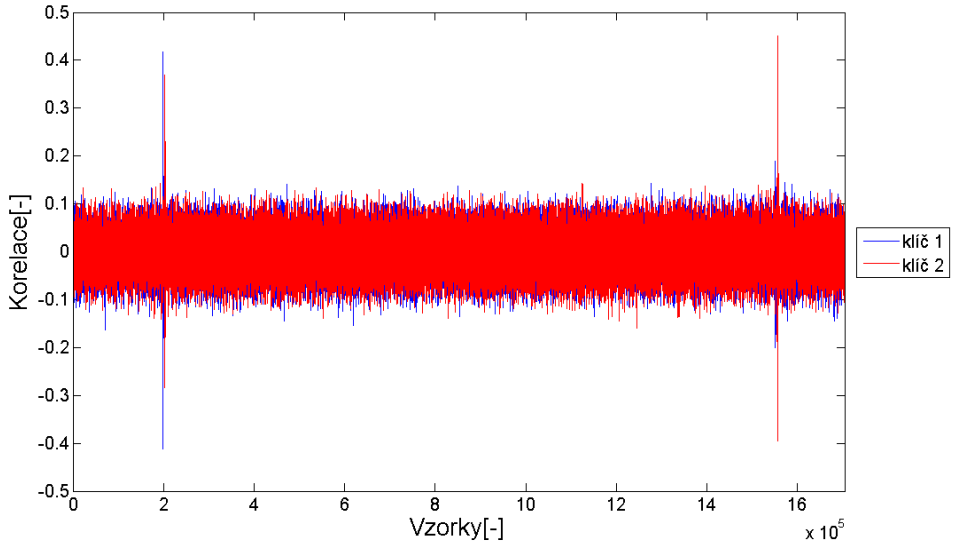
- uložení všech známých dat pomocí nástroje Attack-Wrapper (otevřený text, klíč, offset, proudový průběh a hodnoty vektoru *shuffle*),
- vytvoření kódu pro výpočet korelační analýzy, která využívá známá a hypotetická data,
- výpočet korelace pro jednotlivé kroky algoritmu, takto vypadaly jednotlivé operace v použitém kódu:

```
1    after_keyMask (p,:) = bitxor(key, M_OFF(1,mod(klic,16)+1));
2
3    after_AddRoundKey (p,:) = bitxor(plaintext(p,k_new), ...
        bitxor(key, M_OFF(1,mod(klic,16)+1)));
4
5    after_sbox (p,:) = bitxor(SubBytes( plaintext(p,k_new) ...
        +1), bitxor(key, M_OFF(1,mod(klic,16)+1)));
```

- vyhodnocení pomocí zobrazení výpočtů v grafu.

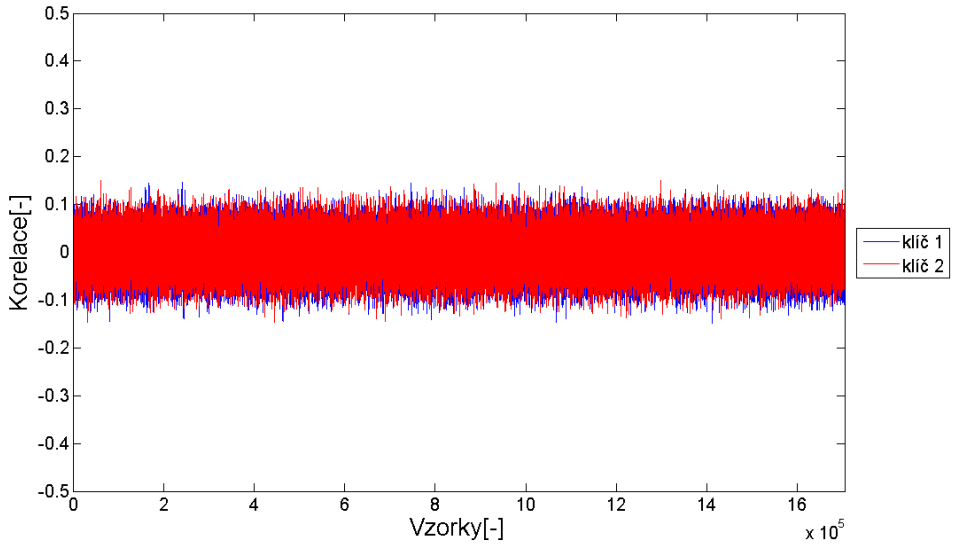
Na základě předchozí zkušenosti s DPAC v [22] byla zvolena korelace postupně pro jednotlivé fáze upravené implementace AES v DPACv4.2, viz algoritmus 3.2. Nejdříve byla analýza vytvořena pro maskování klíče, poté operaci AddRoundKey() a poslední se zaměřila na výsledek SubBytes().

První korelační analýza, která byla v průběhu práce vytvořena, se zabývala hledáním zajímavých bodů při užití operace $Key \oplus Mask(offset_{new})$. Tento výpočet bral v úvahu odhad klíče, tedy 0 až 255, a hledal možnou shodu se skutečnými proudovými průběhy. Z výsledné korelační matice se vybírá pouze jeden řádek, který obsahuje nejvyšší hodnoty – představuje největší míru korelace. Tento řádek vždy odpovídá nějakému klíči z matice odhadovaných klíčů a teoreticky by měl odpovídat bajtu klíče, pro který korelaci počítáme. V grafu 4.1 je vidět průběh korelační analýzy pro první a druhý bajt klíče. Dvě špičky v grafu představují místo, kde došlo ke zpracovávání dat vypočítaných pro maskování klíče. Výsledný klíč, který byl nalezen pomocí korelace (klíč 1=192, klíč 2=85), neodpovídal skutečnému (klíč 1=143, klíč 2=203).

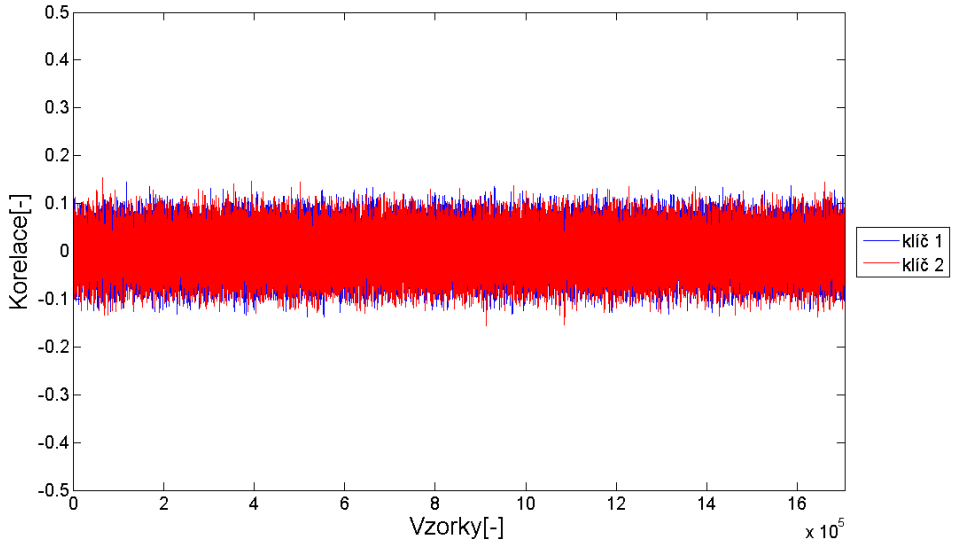


Obr. 4.1: Korelační analýza pro operaci maskování klíče, použití hodnoty $offset_{new}$. Výsledky korelace (klíč 1=192, klíč 2=85) neodpovídaly skutečným hodnotám (klíč 1=143, klíč 2=203).

Stejný postup byl proveden pro operace AddRoundKey() a SubBytes(), po řadě šestý a devátý řádek v kódu 3.2. Výsledky jsou v grafech 4.2 a 4.3, kde je vždy vykreslena korelace pro dva klíče. Ani v jednom případě nejsou vidět žádné korelační špičky, pomocí korelace tedy nebyly nalezeny žádné výskyty hypotetických hodnot při šifrování. Klíč, který byl určen pomocí korelace, neodpovídal skutečnému.



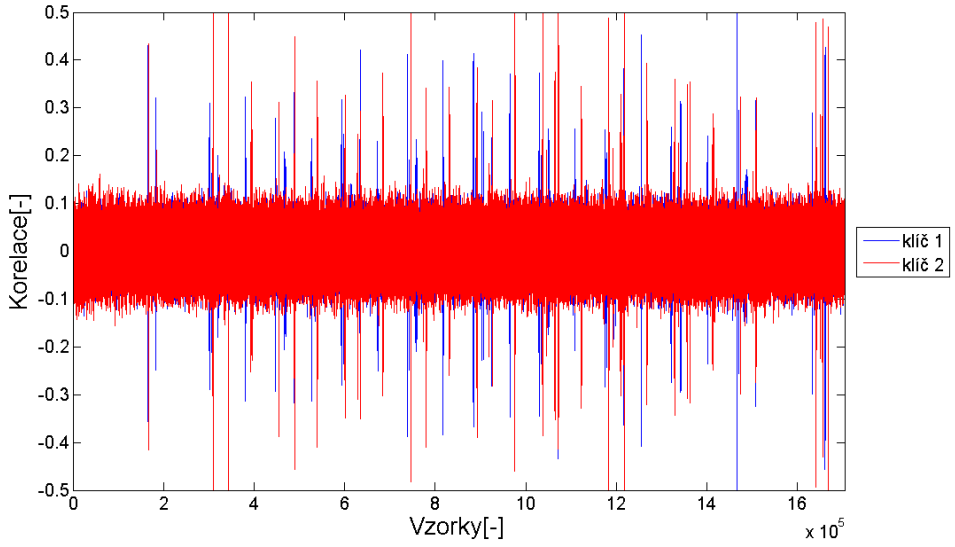
Obr. 4.2: Korelační analýza pro operaci AddRoundKey(), použití hodnoty $offset_{new}$. Výsledky korelace (klíč 1=91, klíč 2=171) neodpovídaly skutečným hodnotám (klíč 1=143, klíč 2=203).



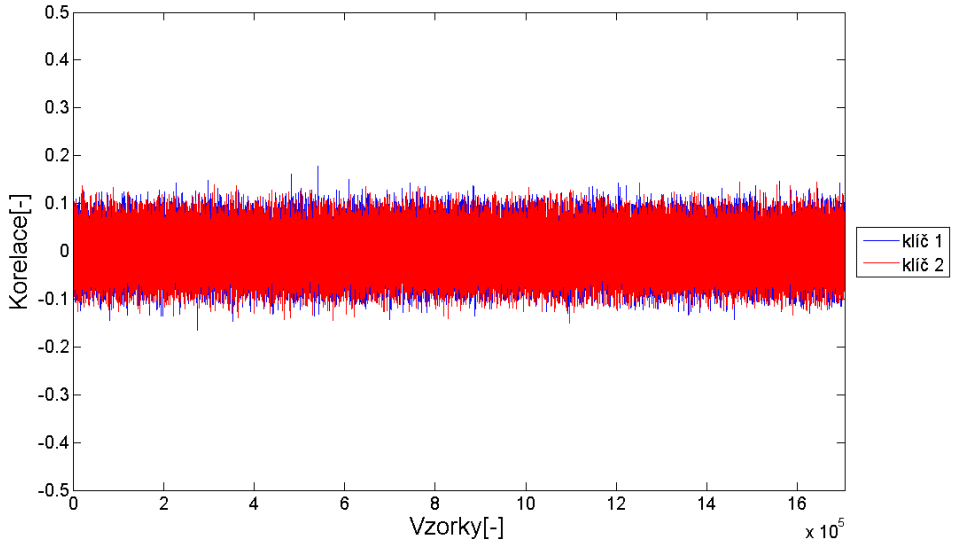
Obr. 4.3: Korelační analýza pro operaci SubBytes(), použití hodnoty $offset_{new}$. Výsledky korelace (klíč 1=132, klíč 2=192) neodpovídaly skutečným hodnotám (klíč 1=143, klíč 2=203).

Po mnoha výpočtech a úpravách kódu pro korelační analýzu zde byla objevena chyba a to v použité masce. V nové upravené implementaci DPACv4.2 je v prvním kroku použita maska, která se vybírá podle hodnot uložených ve vektoru offset, vygenerovaný v inicializační fázi. Indexy masek odpovídají přesně popořadě hodnotám vektoru offsetu, protože v první části algoritmu není offset ovlivněn náhodným

přeskládáním neboli operací *shuffle*. K tomuto kroku dochází až na 9. řádku algoritmu 3.2. Korelační analýza popsaná výše používá indexy masek po přeskládání offsetu – $offset_{new}$, platí pouze pro kroky maskování klíče a AddRoundKey(), operace SubBytes() již používá indexy z vektoru – $offset_{new}$.



Obr. 4.4: Korelační analýza pro operaci maskování klíče, použití hodnoty $offset$. Výsledky korelace (klíč 1=106, klíč 2=52) neodpovídaly skutečným hodnotám (klíč 1=143, klíč 2=203).

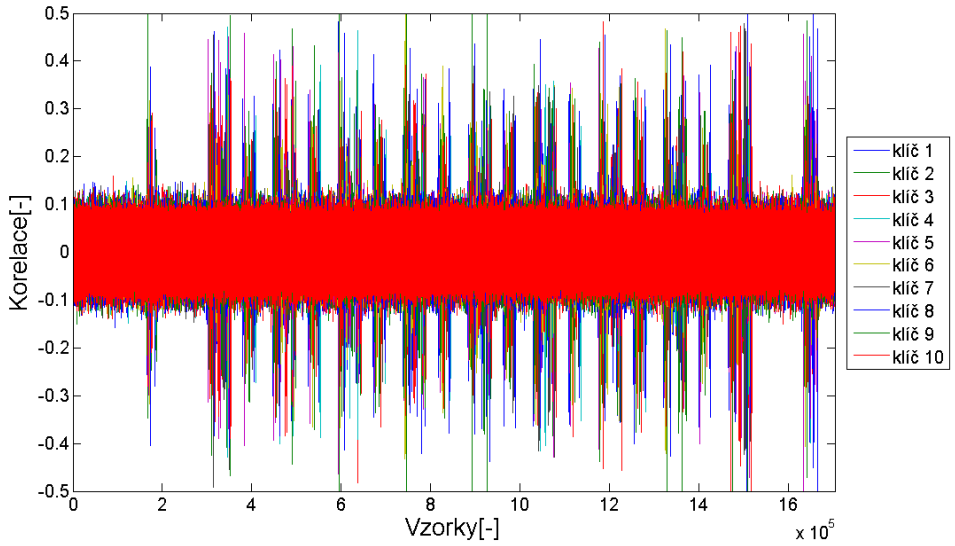


Obr. 4.5: Korelační analýza pro operaci AddRoundKey(), použití hodnoty $offset$. Výsledky korelace (klíč 1=51, klíč 2=14) neodpovídaly skutečným hodnotám (klíč 1=143, klíč 2=203).

Upravený kód na konci odstavce ukazuje nejdůležitější část, která byla změněna, a pomocí ním získaná korelace je uvedena v grafech 4.4 a 4.5. Pozorujeme již výraznou změnu výsledku především pro operaci $Key \oplus Mask$. Objevilo se již více špiček – zajímavých bodů. Tyto body by mohly pomoci při útoku. Jak je vidět z grafů pro operaci AddRoundKey(), není velký rozdíl v porovnání s předchozími výsledky, které používaly hodnotu $offset_{new}$.

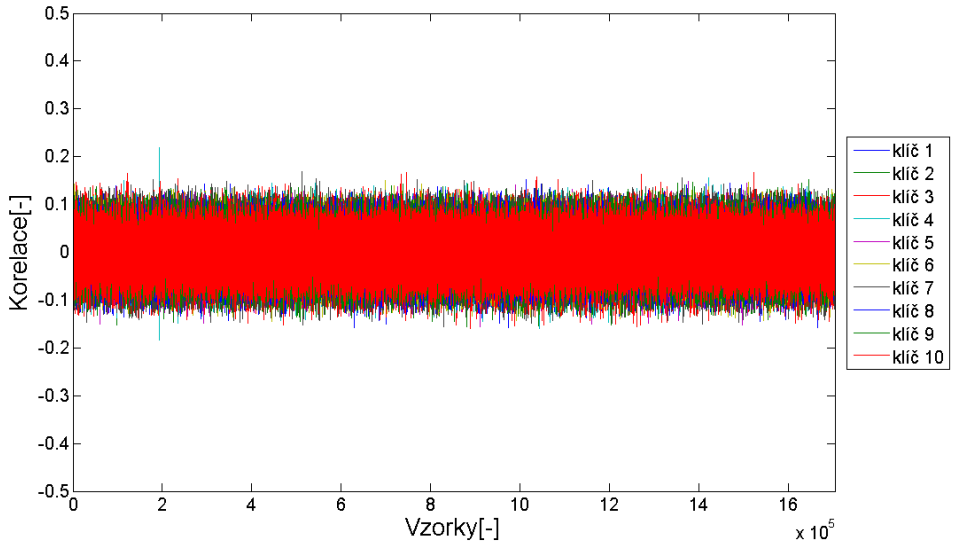
```
1  offset = offset(p,klic);                    //výběr offsetu pro vybraný ...
       klíč a proudový průběh p
2  M_OFF= mask(offset + 1, : );        //přeskládání masek podle ...
       hodnoty offsetu
```

Kód pro určení klíče pomocí hypotetických hodnot je výpočetně náročný, protože pracuje s maticemi velkých rozměrů. Proudový průběh má 1704402 vzorků a při korelaci s maticí hypotetických hodnot (256×1000 - proudových průběhů je pro každý klíč naměřeno přesně 1000) je velikost korelační matice 256×1704402. Z důvodu velkého objemu dat, byla korelace počítána pro části po cca 300000 vzorcích proudového průběhu. Následně byly dílčí výsledky spojeny do jednoho pro daný klíč. Je tedy zřejmé, že získání konečného výsledku je i časově náročné.

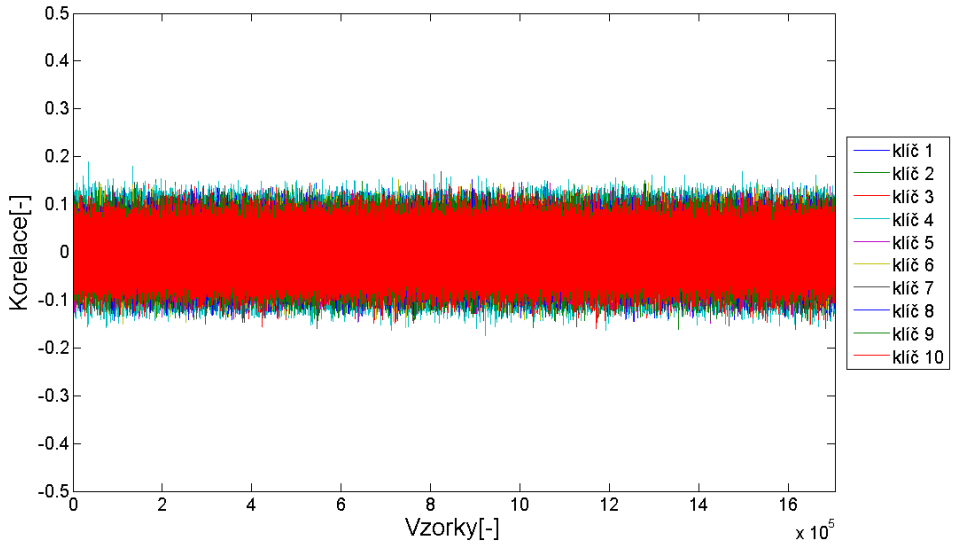


Obr. 4.6: Korelační analýza pro známý klíč, operace maskování klíče.

Vypočítané klíče neodpovídaly klíčům použitých při šifrování, bylo proto přistoupeno ke korelační analýze, která brala v úvahu pouze data známá, tedy skutečné hodnoty. Všechna taková data je možné uložit pomocí nástroje Attack-Wrapper a následně využít v upraveném kódu. Ten redukuje matici pro hypotetický klíč pouze na jeden sloupec, ve kterém jsou uloženy výsledky pro vybranou operaci algoritmu AES a známý klíč. Po korelaci s naměřenými průběhy je výsledkem vektor.



Obr. 4.7: Korelační analýza pro známý klíč, operace AddRoundKey().



Obr. 4.8: Korelační analýza pro známý klíč, operace SubBytes().

Nová verze kódu byla aplikována opět na stejné operace $Key \oplus Mask$, AddRoundKey() a SubBytes(). Všechny výsledky jsou uvedené v grafech 4.6, 4.7 a 4.8, kde je lze porovnat. Pro přehlednost bylo použito pouze deset klíčů (klíč $1, \ldots, 10 = 143, 203, 41, 216, 140, 205, 39, 81, 231, 242$). Výsledky korelační analýzy pro operace AddRoundKey() a SubBytes() se velmi podobají výsledkům počítaných pro hypotetické hodnoty. Ani v jednom z těchto grafů nelze pozorovat špičky – zajímavé body. Pro tyto operace byla korelace hypotetické a skutečné proudové spotřeby minimální i pro správný odhad klíče. Je to způsobeno dobrou implementací šifrovacího algoritmu nebo může být chyba v použitém algoritmu pro korelační analýzu. Z výsledků

pro operaci maskování klíče lze vyčíst zajímavé body, které by mohly být využity při útoku.

4.2 Diskuze útoku pro DPACv4.2

Jedním z velice efektivních útoků na AES je například DPA útok pomocí šablon. Tento útok pracuje na stejném principu jako SPA útok pomocí šablon, jak je popsáno v 2.1.1. Vytváření šablon je shodné a hledání největší pravděpodobnosti shody použitého a hypotetického klíče k_j ve všech proudových průbězích $\mathbf{T}$, počet řádků je D – odpovídající všem vstupním datům, je následující:

$$p(k_j|\mathbf{T}) = \frac{(\prod_{i=1}^{D} p(\mathbf{t}_i|k_j)) \cdot p(k_j)}{\sum_{l=1}^{K}((\prod_{i=1}^{D} p(\mathbf{t}_i|k_l)) \cdot p(k_l))}. \tag{4.1}$$

Vztah 4.1 je základem pro DPA útok pomocí šablon. Největší pravděpodobnost $p(k_j|\mathbf{T})$ pro $j = 1, \ldots, K$, určuje klíč k_j jako nejlepší odhad skutečného klíče použitého šifrovacím modulem. [18]

Velikost kovarianční matice šablony ovlivňuje výpočetní výkon. Je lepší tuto matici zmenšit a to především výběrem vzorků proudových průběhů, které obsahují nejvíce informací o tajném klíči. Jak je popsáno v [9], mohou být tyto body zájmu – Point of interest, vybrány pomocí techniky: body s maximální variancí, analýza hlavních komponent – principal component analysis (PCA) nebo body, ve kterých je rozdíl průměrů proudových průběhů velký.

Inspirací pro útok na maskovaný algoritmus AES může být [18], kde je pro vytváření šablon využito operací, které pracují s hodnotami masek. Body zájmu jsou vybrány jako vzorky z proudových průběhů, kdy došlo k operaci s maskou. Pro nalezení těchto bodů by pro případ útoku na DPACv4.2 mohlo být využito například korelační analýzy, jejíž výsledky byly popsány výše, nebo jedné z technik zmíněných v [9] a jako operace pracující s maskou by mohla být zvolena $Key \oplus Mask$. Pokud by bylo možné tímto způsobem určit, jaká maska byla použita, dal by se zjistit i vektor offsetů.

5 ZÁVĚR

Celá práce je zaměřena na útoky postranním kanálem a to konkrétně proudovou analýzou. Proudová analýza studuje proudovou spotřebu kryptografického modulu v závislosti na jeho činnosti.

Jsou uvedeny dvě základní analýzy a to jednoduchá – SPA a diferenciální – DPA. Každá z analýz představuje pro modul bezpečnostní riziko, protože na jejich principu pracuje několik typů útoků. Velmi časté jsou útoky například pomocí korelačního koeficientu nebo šablon.

Pro objektivní porovnání úspěšnosti nejrůznějších útoků slouží tzv. DPA Soutěž. Ta byla zahájena již v roce 2008. Dodnes bylo vydáno několik jejích verzí, které cílily na různé šifrovací algoritmy. Poslední verzí je 4.2, která zveřejňuje upravenou implementaci algoritmu AES. V předchozí verzi bylo jako opatření proti úniku informací zvoleno maskování AES. Hodnota tzv. offsetu, více popsáno v 3.1, byla ovšem velmi jednoduše všemi účastníky získána. Z toho důvodu byla implementace vylepšena hned několika kroky, především bylo přidáno tzv. míchání – shuffling. Podrobněji jsou tyto změny popsány a zdůvodněny v části 3.2.

Hlavním cílem práce bylo seznámit se s novou implementací DPACv4.2 a po analýze dat uskutečnit útok. Proudové průběhy, které jsou zveřejněny spolu s šifrovanými daty, na webových stránkách DPACv4.2, byly analyzovány pomocí zvolené korelační analýzy. Byl proveden výpočet korelace skutečných dat (známých dat) s hypotetickými. Výpočet se zaměřoval postupně na tři důležité výpočty algoritmu AES, a to: maskování klíče, operaci AddRoundKey() a SubBytes(). Výsledky všech analýz jsou přehledně zpracované v kapitole 4.1. Nejvyšší korelační špičky se objevují při operaci maskování klíče. U operací AddRoundKey() a SubBytes() pozorujeme minimální korelace mezi hypotetickou hodnotou a naměřenou proudovou spotřebou, i když byl použit správný odhad klíče. Může to být způsobeno dobrou implementací algoritmu nebo skrytou chybou v kódu pro výpočet korelační analýzy.

Z důvodů výpočetně a časově náročných analýz nebyl útok kompletně realizován. Jeho možný postup je diskutován v závěru poslední kapitoly 4.2. Navrhováno je použití útoku pomocí šablon, který by mohl určit hodnotu offsetu a na základě jeho znalosti a znalosti použité masky při operaci SubBytes() odhalit i vektor míchání – shuffle. Inspirací by zároveň mohl být popis již uskutečněných útoků v rámci DPACv4.2, který data předzpracuje a z nich odhadne hodnoty offsetu a vektoru shuffle. Na tomto základě použije útok pomocí šablon nebo korelačního koeficientu.

LITERATURA

[1] *ADVANCED ENCRYPTION STANDARD*. FIPS PUB 197. National Institute of Standards and Technology (NIST), 2001. Dostupné z: <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>

[2] BHASIN, Shivam, Jean-Luc DANGER, Sylvain GUILLEY a Zakaria NAJM. NICV: Normalized Inter-Class Variance for Detection of Side-Channel Leakage. Dostupné z: <http://eprint.iacr.org/2013/717.pdf>

[3] BHASIN, Shivam, Nicolas BRUNEAU, Jean-Luc DANGER, Sylvain GUILLEY a Zakaria NAJM. Analysis and Improvements of the DPA Contest v4 Implementation. 2014, s. 18. Dostupné z: <http://www.dpacontest.org/v4/data/v4_2/article_implem_dpav42.pdf>

[4] CLAVIER, Christophe, Benoit FEIX, Georges GAGNEROT, Mylene ROUSSELLET a Vincent VERNEUIL. Improved Collision-Correlation Power Analysis on First Order Protected AES. V: *Lecture Notes in Computer Science*. Springer, 2011, s. 1-17. Dostupné z:<https://hal.inria.fr/inria-00633527/document>

[5] *DATA ENCRYPTION STANDARD*. FIPS PUBS 46-3. U.S. DEPARTMENT OF COMMERCE, 1999. Dostupné z: <http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf>

[6] DIGITAL ELECTRONIC SYSTEMS RESEARCH GROUP. Description of the masked AES of the DPA contest v4. 2013, s. 5. Dostupné z: <http://www.dpacontest.org/v4/data/rsm/aes-rsm.pdf>

[7] *Dpabook.org* [online]. 06 .07 .2009 [cit. 2014-12-08]. Dostupné z: <http://www.dpabook.org/about/index.htm>

[8] *DPA contest* [online]. 2008, 8.10.2014 [cit. 2014-12-08]. Dostupné z: <http://www.dpacontest.org/home/>

[9] EL AABI, Moulay Abdelaz, Sylvain GUILLEY a Philippe HOOGVORST. *Template Attacks with a Power Model* [online]. 2007 [cit. 2015-05-17]. Dostupné z: <https://eprint.iacr.org/2007/443.pdf>

[10] HNATH, William a Jordan PETTENGILL. *Diferential Power Analysis Side-Channel Attacks in Cryptography*. Worcester, 2010. Dostupné z: <http://users.wpi.edu/ martin/MQP/hnathpettengill.pdf>. Bachelor project. Faculty of Worcester Polytechnic Institute.

[11] JOYE, Marc a Francis OLIVIER.: Side-Channel Analysis. In Encyclopedia of Cryptography and Security (2nd Ed.), 2011, s. 1198-1204. Dostupné z:<http://dx.doi.org/10.1007/978-1-4419-5906-5_516>

[12] KUTZNER, Sebastian a Axel Y. POSCHMANN. *On the Security of RSM*. 2014, 24 s. [cit. 3.5.2015]. Dostupné z: <https://www.cosade.org/cosade14/presentations/session6_c.pdf>

[13] KUTZNER, Sebastian a Axel POSCHMANN. On the Security of RSM - Presenting 5 First and Second-order Attacks. V: *Lecture Notes in Computer Science*. Springer, 2014. Dostupné z:<http://link.springer.com/chapter/10.1007%2F978-3-319-10175-0_20#page-1>

[14] MANGARD, Stefan, Elisabeth OSWALD a Thomas POPP. *Power analysis attacks: revealing the secrets of smart cards*. New York: Springer, 2007, 337 s. ISBN 978-0-387-30857-9.

[15] MARTINÁSEK, Zdeněk. *Kryptoanalýza postrannými kanály*. Brno, 2013. Dostupné z: <https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=62844>. Dizertační práce. VUT Brno. Vedoucí práce Václav Zeman.

[16] MORADI, Amir, Sylvain GUILLEY a Annelie HEUSER. Detecting Hidden Leakages. V: *12th International Conference on Applied Cryptography and Network Security*. Springer, 2014, s. 1-19. Dostupné z: <https://eprint.iacr.org/2013/842.pdf>

[17] NASSAR, Maxime, Youssef SOUISS, Sylvain GUILLEY a Jean-Luc DANGER. RSM: a Small and Fast Countermeasure for AES, Secure against 1st and 2nd-order Zero-Offset SCAs. V: *Design Automa- tion and Test in Europe*. 2012. Dostupné z:<https://hal.archives-ouvertes.fr/hal-00666337/document>

[18] OSWALD, Elisabeth a Stefan MANGARD. Template Attacks on Masking—Resistance Is Futile. V: *Lecture Notes in Computer Science*. Springer, 2007, s. 243–256. Dostupné z: <https://choucroutage.com/Papers/SideChannelAttacks/ietifs-2011-mangard.pdf>

[19] RECHBERGER, Christian a Elisabeth OSWALD. Practical Template Attacks. V: *Lecture Notes in Computer Science*. Springer, 2004, s. 443–457. Dostupné z: <http://www.researchgate.net/profile/Elisabeth_Oswald/publication/225145994_Practical_Template_Attacks/links/0c960535c9d6760e9a000000.pdf>

[20] RIVAIN, Matthieu, Emmanuel PROUFF a Julien DOGET. *Higher-order Masking and Shuffling for Software Implementations of Block Ciphers* [online]. [cit. 2015-05-17]. Dostupné z: <http://eprint.iacr.org/2009/420.pdf>

[21] ŠILHAVÝ, Pavel a Karel NĚMEC. *Datová komunikace*. VUT, FEKT Brno, 2011.

[22] ZAPLETAL, Ondřej. *Klasifikátory proudových otisků*. Brno, 2014. Dostupné z: <https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=85416>. Diplomová práce. VUT Brno. Vedoucí práce Zdeněk Martinásek.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

AES Advanced Encryption Standard

DPA Difrerential Power Analysis

DPAC Difrerential Power Analysis Contest

LEMS Low-entropy masking scheme

RAM Random-access memory

RFID Radio Frequency Identification

RSA Rivest-Shamir-Adleman algorithm

RSM Rotating Sbox Masking

SPA Simple Power Analysis

USB Universal Serial Bus

SEZNAM PŘÍLOH

A ADVANCED ENCRYPTION STANDARD

Již v devadesátých letech bylo jasné, že stávající šifrovací algoritmus DES bude muset být v budoucnu nahrazen nějakým lepším algoritmem. Americký ústav NIST (National Institute of Standards and Technology) začal v roce 1997 s vývojem nového algoritmu pod názvem AES – Advanced Encryption Standard. Cílem amerického ústavu bylo vyvinout nový šifrovací systém pomocí otevřené soutěže. Po několika kolech soutěže a dlouhých testování byl v roce 2000 vybrán algoritmus Rijndael pro standard AES.

Z počátku byl AES používán americkými úřady na zabezpečení dokumentů, které obsahovali pouze citlivá data. V roce 2003 bylo vydáno schválení o použití AES také pro tajná data, a tak získat úroveň zabezpečení označené jako tajné. Do úrovně velmi tajné se mohou řadit data, která byla zašifrována AES za pomocí šifrovacího klíče větší než 128 bitů. DES a 3-DES byl zcela nahrazen novějším AES, který se posléze stal nejpoužívanějším šifrovacím algoritmem ve všech pracovních odvětvích.

Následující text popisuje princip AES algoritmu, který je možné použít pro šifrování datového bloku o velikosti 128 bitů s velikostí klíče 128, přičemž velikost klíče může být i 192 nebo 256 bitů. Podle velikosti šifrovacího klíče používáme zkratky algoritmu jako je AES-128, který používá 128-mi bitový klíč, a stejně tak i AES-192 a AES-256. Rozdíl mezi těmito typy AES je ve zvyšujícím se počtu rund (opakování) a v rozložení klíče. AES-128 přijímá vstupní data vždy jako matici o velikost 4×4, kde každý prvek matice představuje jeden bajt. Tato vstupní matice je převedena na tzv. pole stavů, kde je každý stav – bajt označen indexy. První index značí číslo řádku a druhý pak číslo sloupce. S bajty ve stavovém poli se poté provádějí různé operace. Jednou z nich je operace XOR, která se provádí funkcí *mod2*. Po šifrovacím procesu je stavové pole převedeno do výstupní matice, opět velikosti 4×4 bajty. Velikost datového bloku a klíče se v AES-128 shoduje a počet rund je 10, rozdíly u dalších dvou typů jsou zpracovány v tabulce A.1. [1]

Tab. A.1: Porovnání typů AES [1]

	Délka klíče	Velikost bloku	Počet rund
AES-128	128 bitů	128 bitů	10
AES-192	192 bitů	128 bitů	12
AES-256	256 bitů	128 bitů	14

Bajty v poli stavů se při průchodu rundou zpracovávají čtyřmi procesy. Z pseudokódu, uvedeného níže, je možné vyčíst, že proces se při každé rundě opakuje

– bajty prochází transformacemi SubBytes(), ShiftRows(), MixColumns() a AddRoundKey(), jen poslední runda je výjimkou a vynechává se transformace MixColumns(). [1]

```
1    AES-128(byte in [16] , byte out [16] , word w[44])
2        byte state [4,4] ;
3        state = in;
4        AddRoundKeyCstate, w[0,3])
5        for
6            round = 1 step 1 to 9
7            SubBytes(state)
8            ShiftRows(state)
9            MixColumns(state)
10           AddRoundKey(state, w[round*4,(round+1)*4-1])
11       end
12       SubBytes(state)
13       ShiftRows(state)
14       AddRoundKey(state, w[40,43])
15       out = state;
```

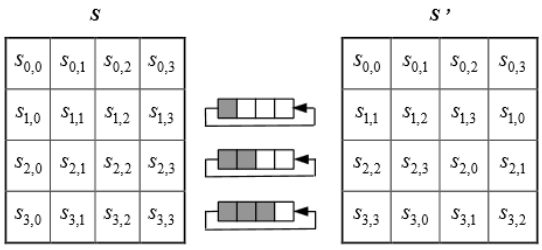
A.1 Operace SubBytes()

Tato bajtová transformace je založena na použití speciální substituční tabulky, která se nazývá S-box A.4. Důležité v tomto případě je, že se jedná o nelineární bajtovou substituci, jenž probíhá nezávisle pro každý bajt.

Substituce neboli nahrazení bajtu $s(1,1) = 35$ ze stavového pole se provede nalezením průniku třetího řádku a pátého sloupce v substituční tabulce. Výsledek bude bajt $s'(1,1) = 96$.

A.2 Operace ShiftRows()

Bajty se při této transformaci cyklicky posunují v posledních třech řadách stavového pole pokaždé s jiným krokem (*offset*). První řádek se označuje nulový a není ovlivněn posunováním. V druhém řádku se posunují všechny bajty o jeden doleva a tedy bajt na pozici $(1,0)$ bude přesunut na pozici $(1,3)$. V případě třetího řádku se jedná o posunutí doleva o dvě místa a na čtvrtém řádku o tři místa. Názorně je transformace vykreslena na obrázku A.1.



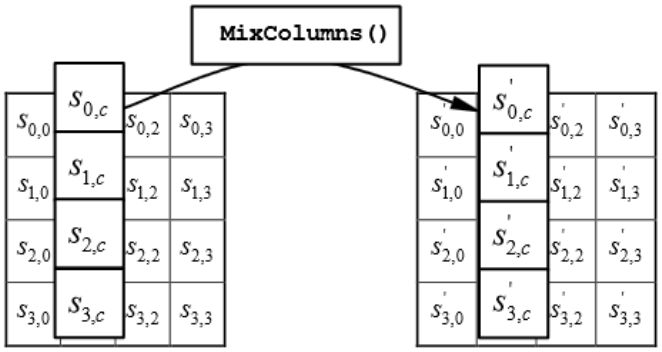
Obr. A.1: Operace ShiftRows() [1]

A.3 Operace MixColumns()

Sloupce stavového pole jsou považovány za polynom čtvrtého stupně. Každý sloupec je vynásoben pevně daným polynomem $a(x)$: $a(x) = 03x^3 + 01x^2 + 01x + 02$. Operace násobení:

$$s'(x) = a(x) \oplus s(x), \tag{A.1}$$

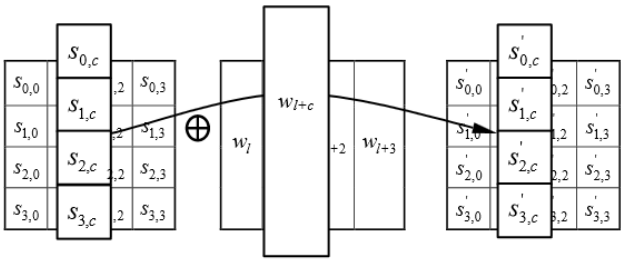
názorně na obrázku A.2.



Obr. A.2: Operace MixColumns() [1]

A.4 Operace AddRoundKey()

Tato transformace se nachází jak v inicializační části algoritmu tak i v závěru každé rundy. Jedná se o výpočet funkce XOR (tedy *mod2*) bajtu ze stavového pole a bajtu klíče pro danou rundu. V první fázi, tedy fázi inicializační, používá AddRoundKey() šifrovací klíč A.3.



Obr. A.3: Operace AddRoundKey(), kde hodnota l odpovídá vztahu: $l = runda * sloupec$ [1]

A.5 Operace Key Expansion()

Operace slouží ke generování klíčů v jednotlivých rundách. V první fázi, inicializační, proběhne pouhé posunutí jednotlivých bajtů v matici klíče stejným způsobem jako je tomu v operaci ShiftRows(). V dalších krocích, rundách, jsou klíče extrahovány z již expandovaného klíče v inicializační fázi. K této transformaci se využívá funkcí SubWord(), RotWord a konstanty Rcon[i], jejich podrobný postup je uveden v (článek).

A.6 Dešifrování

Výše popsané operace se používají i v procesu dešifrování, ovšem jejich pořadí je jiné. Některé z transformací jsou označené předponou „Inv“ znamenající „opačný“ – operace pracují inverzně.

První krok, inicializační runda, je stejná jako u šifrování - provádí se pouze operace AddRoundKey(). Postup algoritmu v následujících rundách je: InvShiftRow(), InvSubByte(), AddRoundKey() a InvMix Column(). Poslední runda opět nepoužívá Mixcolumn() a její postup je následující: InvShiftRow(), InvSubByte() a AddRoundKey().

		y														
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Obr. A.4: Substituční tabulka – Sbox [1]

B OBSAH DVD

- Složka „Text“ obsahuje elektronickou verzi diplomové práce.
- Složka „Korelační analýza“ obsahuje všechny kódy, které byly použity pro jednotlivé korelační analýzy, a zároveň všechny výsledky, které byly v práci zpracovány.
- Složka „attack_wrapper-2.1.1“ obsahuje softwarové nástroje pro realizaci útoku a získání naměřených dat, zároveň kód k uložení naměřených dat.