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

KRYPTOANALÝZA MODERNÍCH KRYPTOGRAFICKÝCH MODULŮ

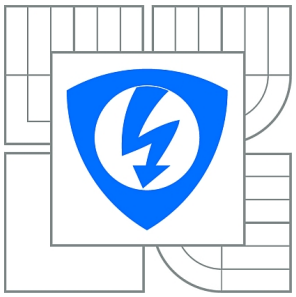
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. ANDRÁS FÖRDŐS
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

KRYPTOANALÝZA MODERNÍCH KRYPTOGRAFICKÝCH MODULŮ

CRYPTANALYSIS OF MODERN CRYPTOGRAPHIC DEVICES

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

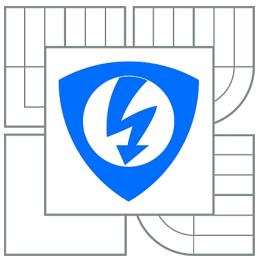
AUTOR PRÁCE — Bc. ANDRÁS FÖRDŐS

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

Student:	Bc. András Fördős	***ID:***	125422
Ročník:	2	***Akademický rok:***	2014/2015

NÁZEV TÉMATU:

Kryptoanalýza moderních kryptografických modulů

POKYNY PRO VYPRACOVÁNÍ:

V rámci diplomové práce prostudujte problematiku proudového postranního kanálu včetně základních metod analýz (jednoduchá a diferenční proudová analýza). Zaměřte se zejména na moderní kryptografické moduly (čipová karta, 32bitové procesory, hardwarové implementace šifrovacích algoritmů, STM32F417 ARM atd.). Z nastudovaných znalostí vyberte vhodné kryptografické moduly (nejméně dva), na které implementujte šifrovací algoritmus AES a proveďte proudovou analýzu v rámci praktické části diplomové práce. 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

ABSTRAKT

Práce se zaměřuje na proudovou analýzu moderních kryptografických modulů. V první části práce je krátký úvod do problematiky proudového postranního kanálu a do základních metod analýz. V textu je popsaný postup porovnání modulů a krátký popis nalezených zařízení. V praktické části byly vybrány celkem dva moduly pro implementaci šifrovacího algoritmu AES-128. První modul představoval čipovou kartu Gemalto .NET v2 a druhý modul představovalo Raspberry Pi. Pro oba moduly byly úspěšně vytvořeny experimentální pracoviště, které umožňovali měření proudové spotřeby algoritmu AES. Na získaných datech byla provedena diferenciální proudová analýza. V závěrečné části práce jsou shrnuty výsledky do tabulek, jsou vidět ukázkové kódy a grafy vytvořené z naměřených hodnot na modulu Raspberry Pi.

KLÍČOVÁ SLOVA

AES, čipová karta, hardwarová implementace, kryptografický modul, mikrokontrolér, proudová analýza, šifrovací algoritmus, Raspberry Pi

ABSTRACT

The thesis focuses on power analysis of modern cryptographic modules. The first part contains a brief introduction to the topic of the power side channel and basic methods of analyzes. The text describes the process of comparison of modules and a short description of devices found. In the practical part two modules has been selected for the implementation of the encryption algorithm AES-128. The first module was the chip card Gemalto .NET v2 and the second one was the Raspberry Pi. A workplace has been created for these modules which allowed to measure the power consumption of the algorithm AES. Differential Power Analysis has been made using the captured results. In its conclusion the work presents the results in tables and samples of source codes. Graphs were made from the results captured on the Raspberry Pi and from the results of the Differential Power Analysis.

KEYWORDS

AES, smart card, hardware implementation, cryptographic device, microcontroller, power analysis, encryption algorithm, Raspberry Pi

FÖRDŐS, András *Kryptoanalýza moderních kryptografických modulů*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2014. 57 s. Vedoucí práce byl Ing. Zdeněk Martinásek, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Kryptoanalýza moderních kryptografických modulů“ jsem vypracoval 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 autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom 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

Děkuji vedoucímu diplomové práce, Ing. Zdeňkovi Martináskovi, Ph.D. za odborné vedení, konzultace, užitečnou metodickou pomoc, trpělivost a cenné připomínky k práci. Chtěl bych poděkovat Réce Tóthové za pomoc při gramatické korektuře této práce. Děkuji své rodině a mé přítelkyni, Denise Sáskové za podporu, povzbuzení a za trpělivost.

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

ÚVOD

V dnešní době je bezpečnost dat, zabezpečení komunikace a šifrování důležitým aspektem nejenom komerční sféry, ale i každodenního života. Krádež citlivých dat, fotek, dokumentů a útoky na kryptografické moduly jako jsou např. i bankovní a kreditní karty je rozšířeným problémem na celém světě. Značná část útoků je zaměřená na analýzu proudového postranního kanálu. Jsou to informace, které unikají z modulů neúmyslně a dají se snadno použít na získání šifrovacího klíče.

Cílem práce bylo prostudovat problematiku proudového postranního kanálu a základní metody analýz. Úkolem bylo zjištění dostupnosti moderních 32bitových kryptografických modulů, čipových karet, mikrokonrolérů a zařízení s hardwarově implementovanými šifrovacími algoritmy. Následně po výběru vhodného modulu bylo úkolem sestavit pracoviště a implementovat algoritmus AES (Advanced Encryption Standard). Na získaných datech pak udělat proudovou analýzu.

Diplomová práce sestává z části teoretické, kde jsou obsaženy základní informace o postranním kanálu, základní informace o metodách analýz. Dále jsou tady popsány kryptografické moduly, výsledky vyhledávání a informace o nalezených modulech. V poslední kapitole teoretické části je uveden krátký popis šifrovacího algoritmu AES.

V praktické části je popis pracovišť, použité programové vybavení a vybrané moduly, kterými jsou čipová karta Gemalto .NET v2 spolu s čtečkou čipových karet Omnikey a Raspberry Pi model B.

V kapitole 2 je popsaná komunikace čipové karty s počítačem. V této kapitole se nachází i programové řešení implementace, aplikace klient a server. V příloze A jsou ukázkové výstupní tabulky a v příloze C jsou ukázkové kódy.

V kapitole 3 je popis modulu Raspberry Pi model B, sestavení experimentální pracoviště, programové vybavení pracoviště, ovladače a implementace algoritmu AES spolu s řídící části měření. Dále jsou v kapitole ukázkové výsledky měření. V další kapitole jsou zpracovány naměřené výsledky a poslední kapitola je věnována závěru.

1 PROUDOVÁ ANALÝZA

Kapitola stručně uvádí postranní kanály a definuje proudovou analýzu. Cílem kapitoly je vybrat vhodný kryptografický modul pro praktickou část, proto se kapitola soustředí na kryptografické moduly.

1.1 Postranní kanál

Postranním kanálem se nazývá každá nežádoucí výměna informací mezi kryptografickým modulem a okolím. Kryptografickým modulem v minulosti bylo považováno zařízení, které mělo jako vstup prostý text a na výstupu pak generoval šifrovaný text, případně opačně. V současné době je známo, že mimo již zmíněných vstupů a výstupů tyto moduly mají i jiné vstupy a výstupy. Kryptografické moduly produkují časové informace, informace o spotřebě (proudový, výkonový postranní kanál), elektromagnetické, chybové informace a jiné. [4, 27]

1.2 Základní metody proudové analýzy

Informace, které unikají z kryptografického modulu jsou v rámci útoku zpracovány a vyhodnocovány, jinými slovy je postranní kanál analyzován. Existují dva základní druhy proudové analýzy. [4, 27]

1.2.1 Jednoduchá proudová analýza

Jednoduchá proudová analýza (SPA - Simple Power Analysis) spočívá v přímém pozorování spotřeby modulu při provádění šifrovací operace. [4, 27]

Samotná spotřeba modulu je však nedostačující pro úspěšnou analýzu. Je potřeba, aby útočník přesně věděl, o jaký šifrovací algoritmus se jedná. Navíc klíč obsažen v zařízení musí mít značný vliv na proudovou spotřebu. Jednoduchá proudová analýza využívá skutečnosti, že různé operace prováděné v kryptografickém modulu generují odlišnou spotřebu proudu. Výhoda SPA spočívá v tom, že stačí velmi malý počet naměřených dat.[13]

Mezi SPA patří i útoky na základě šablonových dat (Template Attacks). V útoku jsou výkonové průběhy charakterizovány pomocí multivariantního normálního rozdělení. Na rozdíl od ostatních typů útoků šablonové útoky se běžně sestávají ze dvou částí: v první je vytvořena šablona a v druhé je šablona použita k útoku. Šablonu je možné vytvořit různými způsoby, např. pomocí zařízení které je útočníkem plně ovladatelné a je stejného typu jako zařízení, na němž bude útok realizován. Na zařízení je možné naměřit proudové průběhy během provádění určité sekvence operací

s různými daty i s různými klíči. Dále seskupením proudových průběhů, které přísluší k určitým párům dat, a klíčů je možno získat šablonu pro každý vytvořený pár. Následně jsou vytvořené šablony a průběhy srovnány s daty a průběhy ze zařízení pod útokem. Výsledky srovnání jsou pravděpodobnosti, nakolik se daná šablona shoduje s měřeným průběhem. Šablona s největší pravděpodobností indikuje klíč.[18]

1.2.2 Diferenciální proudová analýza

Diferenciální proudová analýza (DPA - Differential Power Analysis) je složitějším typem analýzy a je těžší takovým útokům zabránit. Skládají se jak z vizuálních, tak i ze statistických analýz a z metod pro statistickou korekci chyb, aby dokázaly odhalit informace o klíči. Proto je možné klíč odhalit i z průběhů obsahujících velký šum. Sice k provedení DPA je potřeba určité výpočetní techniky, ale je možné proces snadněji automatizovat a pro úspěšnou analýzu stačí jen málo nebo případně i žádná informace o implementaci modulu. Pro úspěšnou analýzu je potřeba naměřit velký počet průběhů spotřeby při šifrování nebo dešifrování pro různá vstupní data. [4, 27]

Důležitým rozdílem mezi SPA a DPA je odlišný způsob analýzy získaných dat. V případě SPA jsou průběhy ve většině případů analyzovány podél časové osy. Útočník se snaží najít určitý vzor, nebo se pokouší najít shodu s určitou šablonou v jediném průběhu. V případě DPA není tvar průběhu podél časové osy důležitý. Během útoku je pozorováno, jak v daném časovém okamžiku závisí proudová spotřeba na zpracovaných datech. Na rozdíl od jednoduché proudové analýzy existuje základní strategie, která se používá u útoků typu DPA. Strategie je rozdělená do pěti kroků:

- Výběr mezivýsledků – mezivýsledek musí být funkce části klíče a známých nekonstantních dat.
- Měření spotřeby – zaznamenání průběhů spotřeby během šifrování nebo dešifrování různých dat.
- Výpočet hypotetických mezivýsledků – pro každou možnou hodnotu klíče s kombinací použitých datových hodnot je spočítaná matice, která je následně použitá k odhalení klíče.
- Mapování hypotetických mezivýsledků na naměřené průběhy spotřeby – výsledky operace jsou hypotetické hodnoty spotřeby, které jsou v posledním kroku použité k odhalení klíče.
- Srovnání hypotetických hodnot spotřeby s naměřenými hodnotami spotřeby – na pozicích, kde jsou tyto průběhy silně závislé na sobě, ve výsledných datech jsou dané hodnoty podstatně vyšší.

Podrobnější informace o problematice proudového postranního kanálu, základních metod analýz a o různých typech útoků je možno najít např. v literatuře [18]

1.3 Kryptografické moduly

V současnosti existuje celá řada kryptografických modulů, jako např. čipové karty, mikrokontroléry, procesory, FPGA (Field-Programmable Gate Array) karty atd. V rámci této práce bylo úkolem najít moderní kryptografické moduly s hardwarově implementovanými šifrovacími algoritmy zaměřenými na algoritmus AES.

Jako výchozí body vyhledávání byly použity dokumentace z konferencií CHES (Workshop on Cryptographic Hardware and Embedded Systems)[11] a CARDIS 2013 (Smart Card Research and Advanced Application Conference).[5]

Na konferencích CHES byly jako testovací moduly často využívány FPGA karty z rodiny Xilinx Spartan nebo Xilinx Virtex, dále pak mikrokontroléry od firmy Atmel, jako např. ATmega128 nebo ATMega-163. Mezi produkty Xilinx nebyl nalezen modul, který by odpovídal zadání této práce, nalezené karty měly jenom softwarovou podporu šifrovacích algoritmů. Vhodný modul byl dál hledán u dalšího výrobce (Atmel). Mezi produkty Atmel byla nalezena rodina mikrokontrolérů SAM4L s 32bitovou procesorovou architekturou ARM Cortex-M4, která má hardwarově implementovaný kryptografický koprocesor AES. Dále na základě zadání byly zkoumány další produkty založené na procesorech ARM a dle doporučení vedoucího práce byly nalezeny moduly Raspberry Pi. Moduly obsahují výkonné, cenově dostupné procesory BCM2835 založeny na architektuře ARMv6 a novější. Moduly Raspberry Pi jsou v dnešní době čím dál tím víc populárnější a to nejenom kvůli cenové dostupnosti ale i kvůli dynamickému vývoji a působivému výkonu. Další výrobce mikrokontrolérů Microchip má mezi produkty mikrokontroléry řady PIC32MZ1024ECM, které odpovídají požadavkům. Jsou to 32bitové zařízení s hardwarově implementovaným kryptografickým koprocesorem podporující AES, 3DES (Triple Data Encryption Algorithm), SHA (Secure Hash Algorithm), MD5 (Message - Digest algorithm) a HMAC (Keyed-Hash Message Authentication Code).

V prezentacích, které jsou dostupné z konference CARDIS 2013, je oblíbeným ukázkovým modulem čipová karta s integrovaným čipem Atmel ATmega-163. Na základě dokumentu [17] na stránce CARDIS 2013 byl zjištěn že v rámci *DPA contest v4* viz [9] byl modul Atmel ATMega-163 smart-card použitý k implementaci AES-256. Tato čipová karta však nemá hardwarově implementovaný kryptografický modul. Dále autoři [16] použili čip MSP430FR5739 pro implementaci AES. Pro další vyhledávání byl použit odkaz na společnost NXP, která vyvíjí a vyrábí mezi mnoha jinými i čipové karty resp. integrované obvody pro čipové karty a taky mikrokontroléry. Mezi produkty byla nalezena rodina integrovaných obvodů pro čipové karty SmartMX a SmartMX2. Vice informací o zmíněních kartách je psáno v kapitole 1.3.2.

1.3.1 Mikrokontroléry

Mezi dostupnými produkty na trhu patří mikrokontroléry rodiny SAM4L od výrobce Atmel. Řada SAM4L patří do rodiny mikrokontrolérů založených na výkonném 32bitovém ARM Cortec-M4 RISC (Reduced Instruction Set Computing) procesoru s rychlostí až 48Mhz. Procesor obsahuje jednotku MPU (Memory Protection Unit) a rychlý, flexibilní ovladač přerušení, aby byla zajištěna podpora moderních operačních systémů bežících v reálném čase. Pro dosažení nízké spotřeby mají zařízení implementovanou technologii picoPower. Pomocí této technologie je zajištěna široká paleta nastavení pro dosažení nejnižší spotřeby a nejvyšší efektivity. Hardwarově implementovaný kryptografický modul podporuje AES s délkou klíče 128 bit v souladu s FIPS (Federal Information Processing Standard). Řada SAM4L nabízí různé možnosti pro připojení periférií jako např. ovladač pro segmentovanou LCD (Liquid - Crystal Display), vestavěná hardwarová dotyková technologie QTouch, ovladač pro USB (Universal Serial Bus) a další. [3]

Mikrokontrolérová rodina od výrobce Microchip pod označením PIC32MZ-EC je založená na 32bitovém MCU s 2MB flash pamětí, audio a grafickým rozhraním s vysokorychlostním USB a s ethernetovým rozhraním. Jádro je typu microAptiv s rychlostí až 200 MHz. Dále je implementovaná jednotka MMU (Memory Management Unit) pro optimální běh operačního systému. Pomocí instrukční sady microMIPS se délka kódu zmenší až o 35%. Vestavěný kryptograficky motor podporuje šifrování/dešifrování a autentizaci AES, 3DES, SHA a další. [29]

Tab. 1.1: Porovnání hlavních parametrů mikrokontrolérů

Parametry	**Řada SAM4L**	**Řada PIC32MZ-EC**
Počet pinů	48-100	64-144
Maximální kmitočet	48 MHz	200 MHz
Program memory	128-512 KB	1024-2048 KB
Data memory	32-64 KB	512 KB
I/O pins	27-75	46-120
Hardwarově implementovaný kryptografický modul	Ano AES-128 bit	Ano AES, 3DES, MD5, HMAC

1.3.2 Čipové karty

Rodina SmartMX obsahuje řady čipů určených pro multifunkční využití s vysokou úrovní bezpečnosti. Čipy obsahují kryptografické koprocesory pro 3DES (64 bit), AES (128 bit)a RSA, ECC (Elliptic Curve Cryptography) (32 bit).[26]

Rodina SmartMX2 se dělí na dvě skupiny čipů dle doporučení.

- SmartMX2-P60 je vylepšená verze rodiny SmartMX. Obsahuje větší paměť, rychlejší procesor, energeticky úsporné, vysokorychlostní kryptografické koprocesory DES, AES, RSA (délka klíče až 4096 bit), ECC.[25]
- SmartMX2-P40 obsahuje čip primárně určený pro bankovnictví, pro zabezpečení elektronických transakcí a další masové finanční nasazení. Implementovaný kryptografický koprocesor podporuje DES, AES (128 bit, 192 bit, 256 bit), RSA (délka klíče až 4096 bit), ECC (délka klíče až 521 bit).[24]

Tab. 1.2: Porovnání hlavních parametrů čipových karet

Parametry	**SmartMX2-P40**	**SmartMX2-P60**
Procesor	16 bit MRK3-SC RISC CPU	8-32 bit High-performance SmartMX2 CPU
EEPROM	13, 40, 72 KB	80, 144 KB
ROM	265 KB	384 KB
RAM	6144 B	8320 B
Koprocesor	DES, AES	2DES/3DES, AES

2 KRYPTOGRAFICKÝ MODUL ČIPOVÁ KARTA GEMALTO

V praktické části práce byl úkolem vybrat kryptografický modul a implementovat šifrovací algoritmus AES. V této kapitole budou popsána zařízení a programová vybavení nutná pro práci s použitými zařízení. Po implementaci bylo úkolem realizovat proudovou analýzu. Pro implementaci byly vybrány moduly dostupné na VUT v Brně.

2.1 Šifrovací algoritmus AES

Původní název algoritmu byl Rijndael a v roce 2001 byl přijat americkým Národním institutem pro standardizaci a technologie jako Advanced Encryption Standard (AES). Jedná se o symetrický šifrovací algoritmus (tentýž klíč slouží k zašifrování i dešifrování dat) s délkou klíče 128, 192 a 256-bitů aplikovaný na bloky délky 128-bitů. Velikost klíče a bloku je nezávislá. Nejprve se vygeneruje šifrovací klíč, následně pak rozšířený klíč, který se pak použije pro šifrování bloku v jednotlivých rundách. Počet rund je přímo závislí na délce klíče a délce bloku. Míra rozšíření se liší v závislosti na délce klíče.[32, 36]

Šifrování probíhá dle následujících kroků:

- Vstupem programu je v tomto případě 16bajtové slovo nazvané jako *Plaintext* a 16bajtový klíč, který je následně rozšířen.
- V prvním kroku je na vstupní slovo aplikován pomocí operace XOR rundovní klíč, který je získán z rozšířeného klíče. Funkce pro přidání klíče se nazývá *addRoundKey()*. Tento krok bývá označen jako nultá runda.
- Vstupem pro další rundu je výstup předchozí rundy. V první rundě je na vstup aplikovaná funkce *Subbytes()*, t.j. hodnota každého bloku je vyměněná dle hodnoty, kterou obsahuje ze substituční tabulky S-Box.
- Dalším krokem je aplikace funkce *shiftRows()*. Tato funkce změní pořadí bloků v určitém řádku dle daného schématu.
- Funkce *mixColumn()* kombinuje hodnoty všech bloků v daném sloupci.
- Posledním krokem v jedné rundě je přidání rundovního klíče pomocí funkce *addRoundKey()*. Tyto kroky se opakují devětkrát.
- Poslední runda se od předchozích liší tím, že je vynechána aplikace funkce *mixColumn()* a rovnou po *shiftRows()* se aplikuje funkce *addRoundKey()*.
- Na výstupu je vrácen zašifrovaný vstupní text.[2]

Dešifrování je provedeno aplikováním inverzních transformací na zašifrovaný výstup.

- Nejprve se na vstup aplikuje v nulté rundě funkce *addRoundKey()*.
- V první rundě jako první krok se provede transformace *inverzShiftRows()*.
- Následně se udělá inverzní substituce *inverzSubbytes()* na základě inverzní substituční tabulky.
- Pomocí operace XOR se přidá rundovní klíč, avšak v tomto případě se bere z konce rozšířeného klíče směrem k začátku.
- Jako poslední krok se provede inverzní operace *inverzMixColumn()*.
- Poslední runda je mírně odlišná od ostatních. Zde se neaplikuje operace *inverzMixColumn()*.
- Po ukončení dešifrování je výsledkem vstupní slovo v původním formátu.[2]

2.2 Popis pracoviště

Pracoviště se skládalo z několika částí, jak je to popsáno níže.

- **Počítač s programovým vybavením** – Počítač byl vybaven operačním systémem Windows 7, dále bylo nainstalováno vývojové prostředí Microsoft Visual Studio 2008 SP1, ovladač Gemalto .NET SDK verze 2.2, který umožní práci s čipovými kartami Gemalto a JCS Suite v3.0 pro práci s kartami Sm@rtCafé.
- **Čtečka čipových karet Omnikey 3121** – slouží k programování podporovaných čipových karet a pomocí čtečky je možné spouštět nahrané programy z karty.[28]
- **Gemalto .NET v2** – čipová karta, na které byl implementován šifrovací algoritmus AES a následně provedena proudová analýza. Programování karty probíhalo v programovacím jazyce C#.[12]
- **Kombinovaná čtečka č.SDI010** – slouží k programování podporovaných čipových karet a pomocí čtečky je možné spouštět nahrané programy z karty.[34]
- **Sm@rtCafé Expert 4.x** – čipová karta pro stejný účel. Na rozdíl od výše zmíněné karty programování probíhá v programovacím jazyce JAVA.

Použitým kryptografickým modulem byla čipová karta Gemalto .NET v2. V dalších částech bude popsáno, co obsahuje tato karta a jak probíhá komunikace s kartou. Dále bude popsaná vlastní implementace šifrovacího algoritmu.

Obr. 2.1: Obrázek pracoviště s počítačem, čtečkou karet a čipovou kartou

2.3 Obsah čipové karty Gemalto .NET v2

Gemalto .NET v2 je programovatelná čipová karta, která byla představená firmou Gemalto v roce 2002. Je to kontaktní typ čipové karty. Na těle karty je viditelná měděná plocha s osmi kontakty, která slouží jak pro datovou komunikaci, tak pro napájení. Rozložení kontaktů a jejich funkce je popsáno v standardu ISO 7816 [15]. Tato část karty obsahuje všechny funkční díly. Čip v kartě je jeden 32bitový mikroprocesor, dále kryptografický koprocesor a tři typy paměti RAM (Random Access Memory), ROM (Read-Only Memory), EEPROM (Electrical Erasable Programmable Read-only Memory). Karta dokáže přijímat data, zpracovat je dle služby, která aktuálně běží a pak vrátit zpracovaná data.

Softwarově je karta navržená pro práci na platformě .NET Framework pro čipové karty s běhovým prostředím CLR (Common Language Runtime). Podporuje několik šifrovacích algoritmů, které jsou popsány v tabulce Tab. 2.1. Pomocí softwarové komponenty ovladače Gemalto .NET SDK v2.2 nazvaného jako Card Explorer můžeme prohlížet, zapisovat a smazat obsah karty. Card Explorer lze spustit buď jako samostatnou aplikaci, nebo jako zásuvný modul v programu Microsoft Visual Studio 2008.

Čipová karta obsahuje souborový systém, jehož vzhled je popsán níže.

- Disk C: složka Gemalto obsahuje aplikace a knihovny od výrobce.

- Disk C: složka Pub je veřejná složka, kam se dají nahrávat aplikace a data uživatele.
- Disk C: složka System obsahuje knihovny, které mají být dostupné pro všechny aplikace, např. knihovny .NET Frameworku pro čipové karty.
- Disk D: složka Pub je veřejná složka obsahující soubor CardConfig.xml ve kterém se nacházejí základní informace o dané kartě.

Tab. 2.1: Podporované kryptografické algoritmy na čipové kartě Gemalto .NET v2

Kryptografický algoritmus	Délka šifrovacího klíče [bit]	
	Minimální	Maximální
DES	64	64
3DES	128	192
Rijndael (AES)	128	256
RSA	256	1024
SHA1	hash	

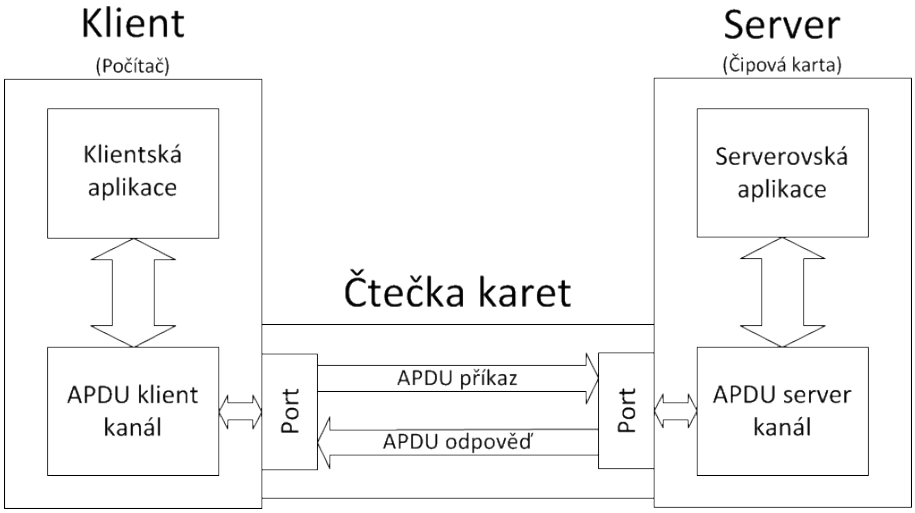
2.4 Komunikace s čipovou kartou

Komunikace u .NET Frameworku pro čipové karty probíhá přes APDU (Application Protocol Data Unit) kanál. APDU mají dvě kategorie typu příkaz a odpověď. Aplikace v počítači představuje stranu klienta a aplikace na čipové kartě stranu serveru. Komunikace probíhá vysláním příkazu přes APDU klientský kanál (čtečka karet) směrem k čipové kartě. Následně přes serverovský kanál karta pošle APDU odpověď směrem k čtečce karet do počítače (ke klientské aplikaci). Blokové schéma komunikace je vidět na obrázku Obr. 2.2. [15]

2.5 Programové řešení – Aplikace klient a server

Ve vývojovém prostředí Visual Studio 2008 byl vytvořen pomocí šablony projekt serverové aplikace se jménem **netCard_Server**. Šablona obsahuje soubor *MyServer.cs*, ve kterém jsou nastavení a metody pro navázaní spojení, registraci kanálu a pro registraci aplikace jako server. Do souboru *MyService.cs* se pak píše kód, který bude po kompilaci sloužit jako aplikace na kartě. Další soubor *nant.build* je konfigurační soubor, který obsahuje definici adresáře, do kterého bude zdrojový kód nahrán a jméno služby v aplikaci.

Pro vytvoření projektu klientské aplikace byla použita jiná šablona a byla pojmenovaná jako **netCard_Client_Console**. Soubor jménem *MyClient.cs* obsahuje



Obr. 2.2: Komunikace s využitím APDU kanálů

metody pro registraci kanálu, referenci na službu v serverové aplikaci, část pro kód volání vzdálené aplikace a odregistrování kanálu. Aby bylo volání vzdálené aplikace úspěšné, je nutno přidat referenci na soubor v projektu serverové aplikace v podadresáři projektu ...\stub\netCard_Server_stub.dll.

2.6 Řešení šifrování

Při psaní kódu nebyla cílem efektivita, ale funkčnost implementace, nacvičení a porozumění, jak funguje šifrovací algoritmus. V implementaci bylo použito AES s délkou klíče 128 bit, a proto počet rund je 10. Od délky klíče je v algoritmu AES přímo závislý počet rund nutných k vykonání celého procesu.

Pro úspěšné šifrování je nutné ze vstupního klíče pomocí expanze udělat rozšířený klíč. Z důvodu komplexnosti algoritmu během seznamovací fáze s algoritmem nebyl tento klíč vygenerován. V literatuře [2] se nachází tabulka, ve které je uveden rozšířený klíč pro vstupní klíč `00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f`. Právě tento klíč byl použitý v programové části a rozdělen na rundovní klíče. Implementace byla realizována dle standardu AES.

Popis vytvořených metod na serverové straně se nachází v příloze B. V aplikaci na straně klienta byl vytvořen následující kód:

```
byte[] result = service.Encrypt();
byte[] resultD = service.Decrypt();
```

Vytvoření polí a následné volání funkcí šifrování a dešifrování

```
for (int i = 0; i < 16; i++){
    Console.WriteLine("{0:x}",result[i]);
}
for (int i = 0; i < 16; i++){
    Console.WriteLine("{0:x}", resultD[i]);
}
Console.ReadKey();
```

Jednoduché cykly byly napsány pro výpis výsledků šifrování a dešifrování v 16kovém formátu aby bylo možné provést vizuální kontrolu výsledků. Získané výsledky jsou shrnuty do tabulek, které jsou vidět v příloze A.

Bylo zašifrováno 16bajtové slovo `00 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff` s klíčem `00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f`. Hodnoty slov při startu jednotlivých rund šifrování se nachází v Tab. A.1. V této tabulce v řádku *runda 10* je výsledek šifrování a má následující tvar `69 c4 e0 d8 6a 7b 04 30 d8 cd b7 80 70 b4 c5 5a`. Po šifrování bylo provedeno dešifrování a výsledkem bylo původní slovo. V tabulce Tab. A.4 jsou obsaženy hodnoty slov během dešifrování. Řádky *runda 2* až *runda 9* jsou identické ale pořadí je inverzní což odpovídá procesu šifrování a následné dešifrování. První a poslední řádek se liší. První řádek se liší z důvody provedené transformace *addRoundKey()*. Poslední řádek je odlišný kvůli aplikace funkce *invShiftRows()* a *invSubBytes()* u dešifrování. V případě Gemalto karty se proudová analýza nebyla provedena.

3 KRYPTOGRAFICKÝ MODUL – RASPBERRY PI MODEL B

Tato kapitola je věnována popisu měření na modulu Raspberry Pi model B. V první podkapitole se nachází podrobnější popis informací o vybraném modulu. V dalších podkapitolách je rozepsán postup sestavení pracoviště, programové vybavení nutné pro realizaci a řízení měření, implementace algoritmu AES a naměřené výsledky.

3.1 Popis modulu

Raspberry Pi je cenově dostupný počítač o velikosti kreditní karty, kterou je možno připojit k monitoru nebo k televizi a jako uživatelský vstup používá klávesnici a myš. Je to malý, ale schopný modul určen pro experimentování v počítačové technice. Zvládne všechno co se dá očekávat od stolního počítače od prohlížení webu přes přehrávání vysoko kvalitního videa, editace dokumentů i hraní počítačových her.[37]

Raspberry Pi je schopen komunikovat s okolím, a proto je využívám pro nejrůznější aplikace jako např. bezpečnostní aplikace, v systémech monitorující počasí, hudební aplikace a mnoho dalších. Je ideálním nástrojem pro výuku dětí, který mají zájem o programování a porozumění jak počítače fungují.[37]

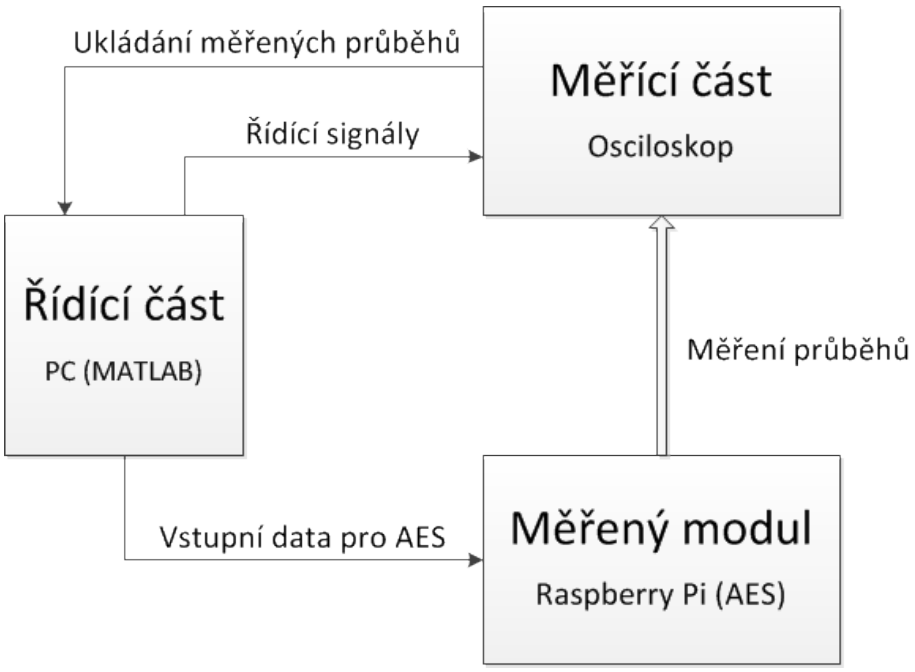
Technické parametry použitého modulu byly následovné:

- Procesor: 700 MHz, ARMv6 rev 7
- Operační paměť: 512 MB RAM
- Periférie:
  - 2 krát USB 2.0 port,
  - Ethernet port,
  - 3.5 mm Jack audio výstup,
  - HDMI (High-Definition Multimedia Interface) konektor,
  - 26pinový GPIO (General Purpose Input/Output) konektor včetně UART (Universal Asynchronous Receiver/Transmitter).
- Datová paměť: paměťová karta od výrobce Kingston Technology, velikost 4 GB

3.2 Experimentální pracoviště

Úvodní krok přípravy pracoviště bylo naplánování měření, implementace algoritmu AES, řízení a zaznamenávání změřených průběhů. Jelikož se jednalo o velký počet změřených hodnot, požadavkem bylo měření automatizovat. Vývojové prostředí

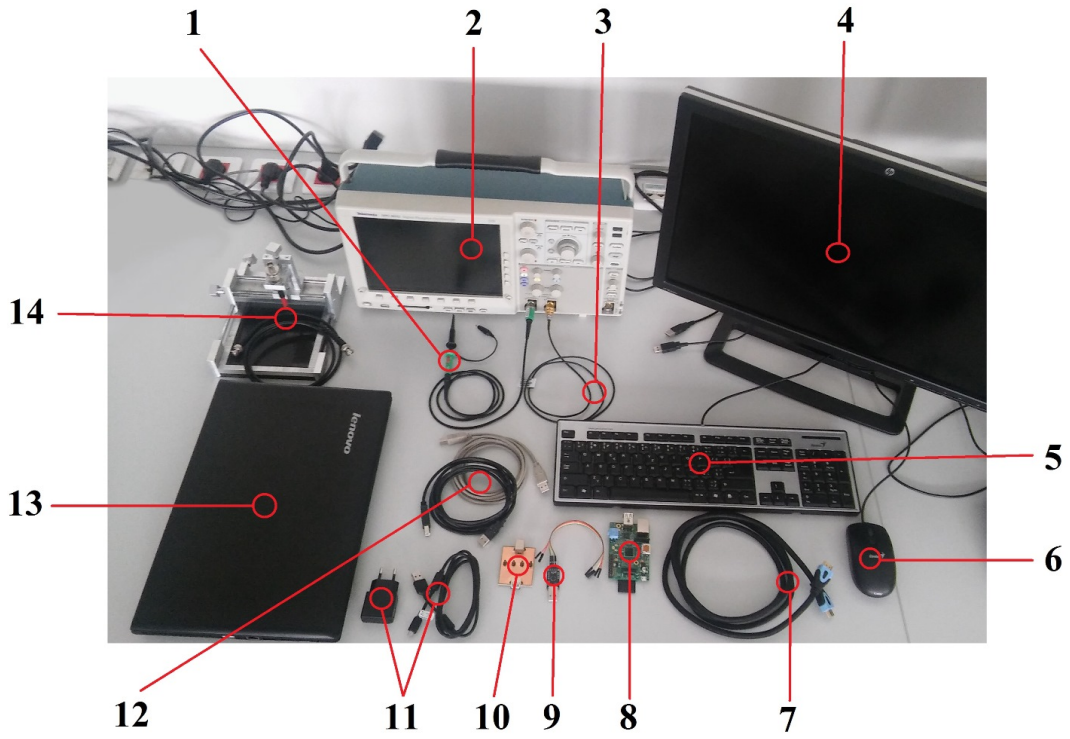
MATLAB byla vhodná volba, protože se dalo doplnit s vhodnými komponenty pro řízení pracoviště a zaznamenávání výstupných hodnot. Části pracoviště je možno znázornit pomocí tří bloků viz. Obr. 3.1.



Obr. 3.1: Blokové schéma pracoviště

Blok označen jako řídící část se sestává z počítače s vývojovým prostředím MATLAB doplněn toolboxem Instrument Control Toolbox pomocí kterého bylo možné plně ovládat osciloskop, který je na obrázku označen jako měřící část. Počítač byl spojen s osciloskopem pomocí kabelu označeným jako USB A-B. Na straně počítače byl připojen do USB portu a do osciloskopu byl připojen na zadním panelu do portu označeným jako *Device*. Blok měřený modul je zařízení Raspberry Pi model B a byl spojen s osciloskopem a s počítačem. S osciloskopem byl spojen pomocí napěťové i proudové (případně elektromagnetické) sondy. Proudová sonda byla napojená na měřící desku, která byla připojená mezi síťovým napájecím konektorem a napájecím konektorem Raspberry Pi. Modul komunikoval s počítačem přes převodník USB na UART. Převodník byl do počítače připojen do USB portu a s Raspberry Pi byl spojen pomocí GPIO pinů (přesné připojení je popsáno v další části práce). Počítač posílal vstupní data pro zpracování a osciloskopem byly změřeny průběhy během šifrování.

Před zahájením měření bylo třeba obdržet potřebné přístroje, kabely a ostatní komponenty. Pracoviště bylo sestaveno z komponent, které jsou vidět na Obr. 3.2. Pod obrázkem jsou dále rozepsaný komponenty podle číselného označení.



Obr. 3.2: Obrázek komponent pro sestavení pracoviště – modul Raspbery Pi

1. Měřící sonda
2. Osciloskop Tektronix DPO 4032
3. Proudová sonda
4. Monitor
5. Klávesnice
6. Myš
7. HDMI kabel
8. Raspberry Pi model B
9. Převodník USB na UART
10. Měřící deska napájecího proudu
11. Napájecí adaptér a micro USB kabel
12. Dva kabely typu USB A-B
13. Řídící stanoviště s programovým vybavením
14. Elektromagnetická sonda se stojanem

3.3 Programové vybavení pracoviště

Kapitola obsahuje informace o postupu instalace programů a ovládačů pro každou součást pracoviště. Součásti jsou rozděleny do podkapitol. Dále jsou popsány nastavení ovladačů a vytváření spojení mezi části z programového hlediska.

3.3.1 Oživení Raspberry Pi model B

Raspberry Pi model B je v základní konfiguraci bez instalovaného operačního systému. Na základě požadavků uživatele je možno vybrat mezi několika distribucemi. V rámci této práce byl vybrán operační systém Raspbian (Debian Wheezy). Jako první krok bylo nutné stáhnout obraz operačního systému, které jsou dostupné z webové stránky [8]. V následujícím kroku byl operační systém nahrán na paměťovou kartu modulu. Návod na instalaci je možno najít na stránkách Raspberry Pi [14]. Pro první start systému je nutné k modulu připojit uživatelské vstupy a výstup tj. klávesnici, myš, monitor a nakonec napájení. Když se systém spustí poprvé, objeví se obrazovka úvodních nastavení. Zde se dá nastavit datum a čas, region případně je možné přidávat uživatele a nastavit v jakém režimu se má systém spustit (režim konzole, režim grafické uživatelské rozhraní GUI). Základní přihlašovací údaje jsou *pi* pro uživatelské jméno a *raspberry* pro heslo. [23].

Po úspěšném spuštění systému bylo potřeba nahrát další nástroje, aby bylo možné s modulem pracovat. Základní verze implementace AES je napsána v programovacím jazyce C#. V rámci měření na modulu Raspberry Pi bude tato verze modifikovaná a doplněná. Aby bylo možné kompilovat a spustit program byl nainstalován nástroj Mono. Mono je implementace s otevřeným kódem pro .NET Framework založená na standardu ECMA (Ecma International – international, private (membership-based) non-profit standards organization for information and communication systems) pro jazyky C# a CLR [20].

Instalace nástrojů a spouštění programů:

- V případě že systém se spustil v GUI spustíme konzoly LX Terminal, je umístěn na vrchní liště ikona  . Do konzole se pak píšu následující příkazy
- Aktualizace systému a knihoven

  ```
  sudo apt-get update
  sudo apt-get upgrade
  ```
- Instalace nástroje Mono

  ```
  sudo apt-get install mono-complete
  ```
- Instalace kompilátoru

  ```
  sudo apt-get install mono-gmcs
  ```
- Po úspěšném provedení předchozích kroků se programy kompilují příkazem

  ```
  gmcs NázevProgramu.cs
  ```
- A následně se spouští příkazem

  ```
  sudo mono NázevProgramu.exe
  ```

Předchozí kroky byly provedeny na základě informací získaných ze zdrojů [31, 30].

Aby bylo možné přijímat data přes UART piny, je nutné modifikovat dvě sou-

bory jménem *cmdline.txt* a *inittab*. Jako první krok je doporučeno soubory zálohovat. Po spuštění LX Terminal do příkazové řádky se napíšu následující příkazy:

```
sudo cp /boot/cmdline.txt /boot/cmdline.bak
sudo cp /etc/inittab /etc/inittab.bak
```

Libovolným textovým editorem spuštěným jako správce se otevře soubor *cmdline.txt*, dále je třeba najít a vymazat část s textem

console=ttyAMA0,115200 kgdboc=ttyAMA0,115200

a soubor uložit. Soubor *inittab* je třeba otevřít obdobně, najít řádek s textem

T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100

a buď řádek vymazat nebo změnit na komentář přidáním znaku # na začátek řádku. Aby se provedené změny aplikovali je potřeba Raspberry Pi restartovat. [6]

3.3.2 Programy a ovladače v řídící stanovišti

Jak již bylo zmíněno, na řídící stanoviště bylo nainstalováno vývojové prostředí MATLAB a toolbox Instrument Control Toolbox, který slouží pro ovládání osciloskopů. Při měření bylo použito osciloskop Tektronix DPO 4032 a proto bylo třeba toolbox doplnit s ovladačem, který je dostupný na webových stránkách MathWorks [35]. Tento ovladač (soubor DPO4032.mdd) byl umístěn do příslušné složky s ovladači. Složka se běžně nachází:

Složka s nainstalovanými programy → MATLAB → toolbox → instrument → instrument → drivers

Osciloskop s prostředím MATLAB komunikuje přes virtuální USB rozhraní. Vytvoření rozhraní spočívá v přiřazení adresy k USB portu kde je osciloskop připojen. Získání adresy se docílí pomocí ovladače typu VISA (Virtual Instrument Software Architecture). Tento ovladač je možné stáhnout z webové stránky společnosti National Instruments a nese označení NI-VISA [22]. Dále byl stažen a nahrán do počítače balík s názvem NI System Configuration 14.5.0, který obsahuje aplikaci NI Measurement & Automation Explorer (zkráceně NI-MAX) pomocí kterého je možné získat virtuální adresy připojených zařízení. NI-MAX je dostupné z adresy [21] Po spuštění aplikace NI-MAX se rozklikne položka:

My System → Devices and Interfaces

Objeví se lišta, kde jsou vidět zařízení spolu s virtuálními adresami. Popis použití získané adresy bude podrobněji popsán v kapitole 3.3.4.

K řídící stanovišti byl připojen převodník USB na UART, který sloužil pro přenos vstupních dat do Raspberry Pi. Převodník je založen na čipu CP2102 od společnosti Silicon Labs a převádí data z USB 2.0 na sériovou komunikaci RS-232 UART tj. vytváří virtuální COM port. Ovladače k převodníku jsou dostupné ze stránek [7].

3.3.3 Implementace AES v Raspberry Pi

Na vybraném modulu byl implementován modifikovaný a rozšířený algoritmus, který byl vytvořen v rámci testovací implementace viz kapitola 2. Pro Raspberry Pi byl celý algoritmus shrnutý do jednoho souboru kvůli jednodušší kompilaci. Byly doplněny metody pro výpočet rozšířeného klíče, které v testovací implementaci nebyly zahrnuty. Tyto metody jsou následující:

```
public static byte[] Expansion()
```

Hlavní metoda, obsahuje postup generace rozšířeného klíče.

```
public static void CipherWord()
```

Metoda pro rozdělení šifrovacího klíče na čtyři části, slová.

```
public static byte[] SubWord(byte[] inputWord)
```

Metoda pro substituci jednotlivých slov pomocí tabulky sBox. Vstupem metody je čtyř bajtové slovo.

```
public static byte[] RotWord(byte[] inputWord)
```

Metoda provádí rotaci slov dle pravidel AES. Vstupem metody je čtyř bajtové slovo.

```
public static byte[] xorRcon(byte[] inputWord, int count)
```

Metoda pro výpočet rundovní konstanty. Dále provádí operaci xor rundovní konstanty a čtyř bajtového slova. Vstupem metody je čtyř bajtové slovo a číslo rundy.

Dále byla přidaná třída která je zodpovědná pro řízení GPIO pinů. Rozložení konektoru GPIO pinů je vidět na obrázku Obr. 3.3, barevně jsou vyznačeny použité piny. Kvůli komplexnosti řízení byla tato třída přebrán ze zdroje [19]. V metodě *Encrypt()* se nachází funkce která je zodpovědná pro nastavění synchronizačního pinu na hodnotu logická 1 (3,3 V)a má tvar:

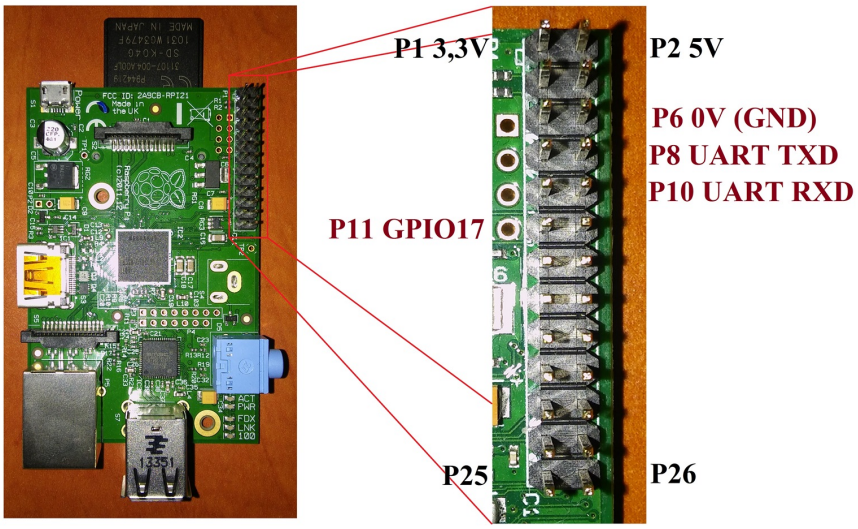
```
gpio.OutputPin(FileGPIO.PinEnum.gpio17, true);
```

Řádek je napsán do kódu přesně před začátkem první rundy šifrování. Další dvě funkce jsou v kódu umístěny po skončení poslední rundy. První má za úkol nastavit synchronizační pin na hodnotu logická 0 (0 V), druhá nastaví všechny piny do úvodního stavu. Funkce mají následující tvar:

```
gpio.OutputPin(FileGPIO.PinEnum.gpio17, false);
gpio.CleanUpAllPins();
```

Byly přidány funkce pro čtení vstupu UART, zde se do modulu dostane vstupní text, který je algoritmem zašifrován. Kvůli automatizaci měření pak byl celý obsah

metody *Main()* obsažen do nekonečného cyklu. Celý soubor s programem je umístěn na přiloženém médiu pod názvem *Program.cs*.



Obr. 3.3: Rozložení GPIO na modulu Raspberry Pi

3.3.4 Řízení měřící části

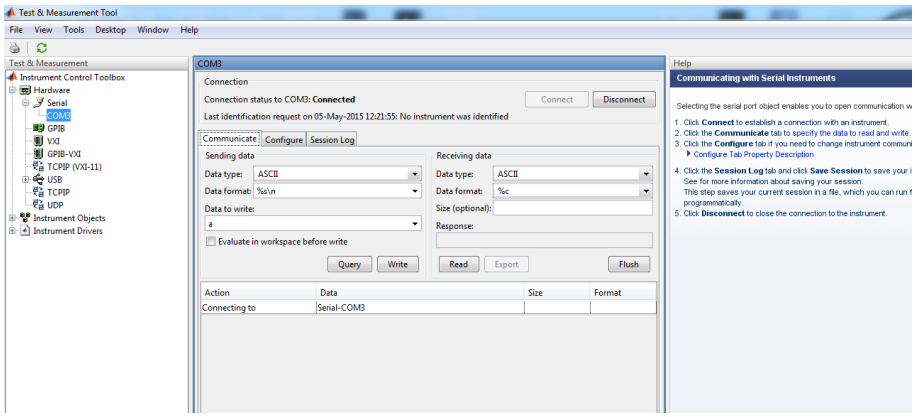
Řízení pracoviště je řešeno pomocí vývojového prostředí MATLAB. Proces se skládá ze dvou částí: generování a posílání vstupních dat na Raspberry Pi, řízení měření na osciloskopu a ukládání naměřeních dat do proměnných.

V prostředí MATLAB byla napsaná funkce, která má za úkol připojit se k převodníku (k virtuálnímu COM portu). Připojení bylo realizováno pomocí nástroje Instrument Control Toolbox. Nástroj se spustí zadáním příkazu `tmtool` do příkazového řádku MATLAB. V levé části okna se nachází lišta s názvem *Test & Measurement* zde se vybere položka:

Hardware → *Serial* → *COM*

V prostřední části se nachází tlačítko *Connect*, kliknutím se naváže spojení s převodníkem viz Obr. 3.4. Níž od tlačítka jsou tři záložky: *Communicate, Configure, Session Log*. V záložce *Configure* se nacházejí nastavení připojení, v rámci měření nebyly změněny. Záložka *Communicate* slouží k zadávání a odesílání dat. Do položky *Data to write* byly zadaný testovací data a kliknutím na tlačítko *Write* byly odeslány. Po odesílání bylo spojení ukončeno kliknutím na tlačítko *Disconnect*. V záložce Session Log byly všechny tyto změny zaznamenány a tlačítkem *Save Session...* ve spodní části okna byl soubor uložen. Soubor byl následně doplněn o řádek který

byl zodpovědný za generování 16ti bajtového náhodného vstupního slova tzv. plain text. Odesílání dat bylo vloženo do for cyklu, který vygenerovaný plain text odesílal po bajtech, aby nedošlo k chybnému přenosu dat. Výsledný soubor je uložen na přiloženém mediu pod názvem *input_plain.m*.



Obr. 3.4: Instrument Control Toolbox s připojeným převodníkem USB na UART

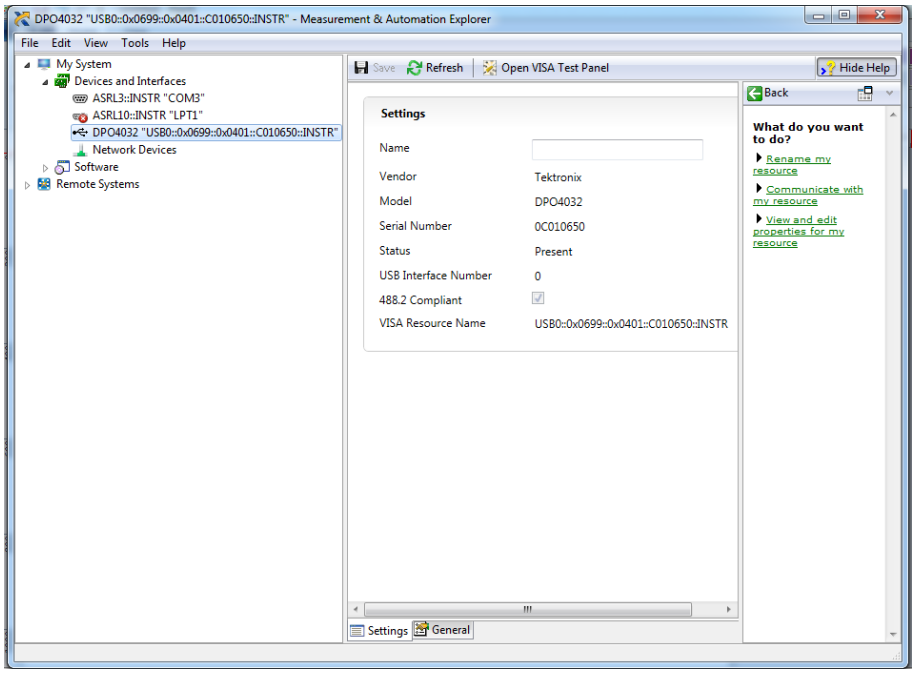
Řízení osciloskopu probíhá obdobně jako v předchozím případě pomocí nástroje Instrument Control Toolbox. Před zahájením komunikace je potřeba získat virtuální adresu připojeného osciloskopu. Pomocí programu NI-MAX je možné zjistit adresy všech připojeních zařízení viz Obr. 3.5. K připojení osciloskopu byla použitá adresa typu VISA, která má tvar *USB0::0x0699::0x0401::C010650::INSTR*. Do příslušné složky byl zkopírován ovladač osciloskopu jak to bylo psáno v kapitole 3.3.2. V nástroji Instrument Control Toolbox se vyhledá ovladač názvem DPO4032.mdd dle cesty:

Instrument Drivers → *MATLAB instrument Drivers* → *DPO4032.mdd*

Kliknutím pravým tlačítkem myší se objeví menu ze kterého se vybere položka *Create Device Object Using Driver ...* následně se objeví nové okno s názvem *Create Device Object*. V okně je vidět tlačítko *Create...*, po kliknutí se objeví okno *New Object Creation*. V části *Define object* z menu *Interface object type* se vybere typ VISA. V části *Configure object creation* z menu *Vendor* se vybere možnost *ni* a do položky *Resource name* se zadává virtuální adresa typu VISA. Po zadání adresy se potvrdí výběr kliknutím *Ok* v obou otevřených oknech. Výběrem:

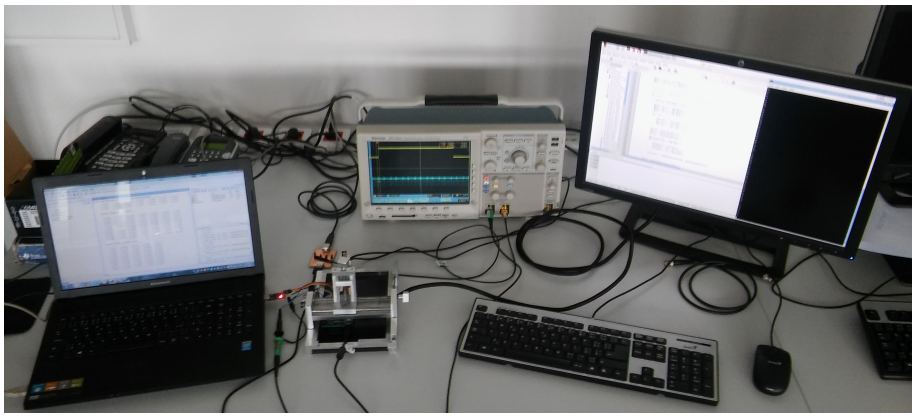
Instrument Objects → *Device Objects* → *scope-DPO4032*

se v prostřední části toolboxu objeví panel s tlačítky *Connect*, *Disconnect* a s třemi záložkami. V záložce *Functions* se dají vybrat funkce osciloskopu. V rámci měření byla použitá funkce *readwaveform* ze skupiny *Waveform group object functions* pro čtení hodnot průběhů. Záložka *Properties* obsahuje nastavení funkcí osciloskopu. Bylo použito nastavení *Firstpoint* (nastaven na 1,0) a *EndingPoint* (nastaven

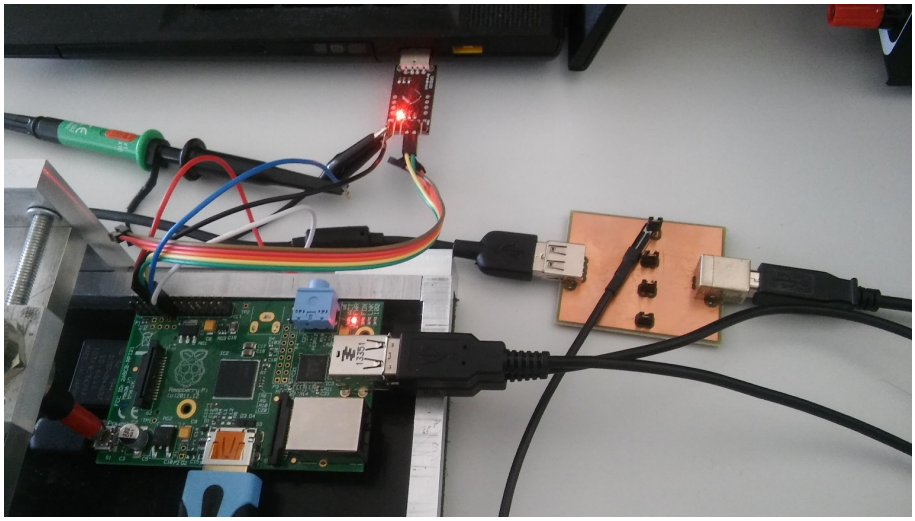


Obr. 3.5: NI-MAX s připojeným hardwarem a virtuální adresy typu VISA

na 100 000,0)ze skupiny *Waveform group object properties* pro nastavení počtu zaznamenaných hodnot. Dále ze skupiny *Acquisition group object properties* byla použita nastavení *state* možnost *stop* pro zastavení měření na čas čtení hodnot a *run* pro spouštění. Ze záložky Session Log obdobně jako v předchozím případě byl uložen soubor a následně modifikován pro potřeby měření. Byly přidány řádky pro ukládání změřených dat a byl rozšířen velikost vstupního bufferu (na 200 000 vzorků) aby bylo možno zaznamenat průběhy s dostatečnou přesností. Nastavení triggeru, rozlišení časové osy a vertikální osy bylo nastaveno manuálně na osciloskopu. Finální verze souboru je uložena na přiloženém médiu pod názvem *savewaveform_v1.m*. Na obrázku Obr. 3.6 je vidět zapojené pracoviště v celku a na obrázku Obr. 3.7 je detailnější pohled na zapojení měřících sond.



Obr. 3.6: Zapojení celé pracoviště



Obr. 3.7: Zapojení měřících sond a převodníku USB na UART

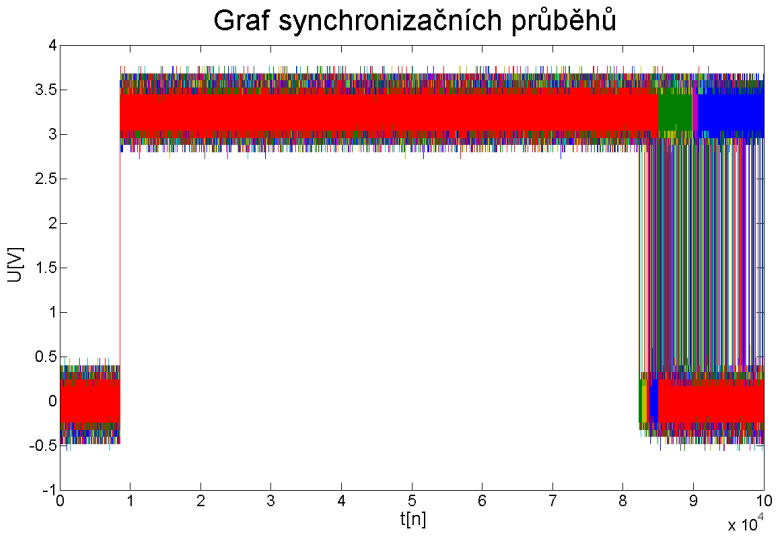
3.4 Naměřené výsledky

V rámci praktické části bylo na modulu Raspberry Pi model B provedeno 500 měření po 100 000 bodech pomocí proudové i pomocí elektromagnetické sondy tj. 1000 změřených průběhů. Během měření byl GUI na Raspberry Pi vypnuto, protože procesy běžící na pozadí se zapnutým GUI nepříznivě ovlivňovali měření. Proudová sonda byla umístěná pomocí plošného spoje do napájecího obvodu (viz Obr. 3.7 vpravo). Elektromagnetická sonda byla pomocí stojanu nastavena na pozici nad napájecím konektorem micro USB (viz Obr. 3.7 levý dolní roh).

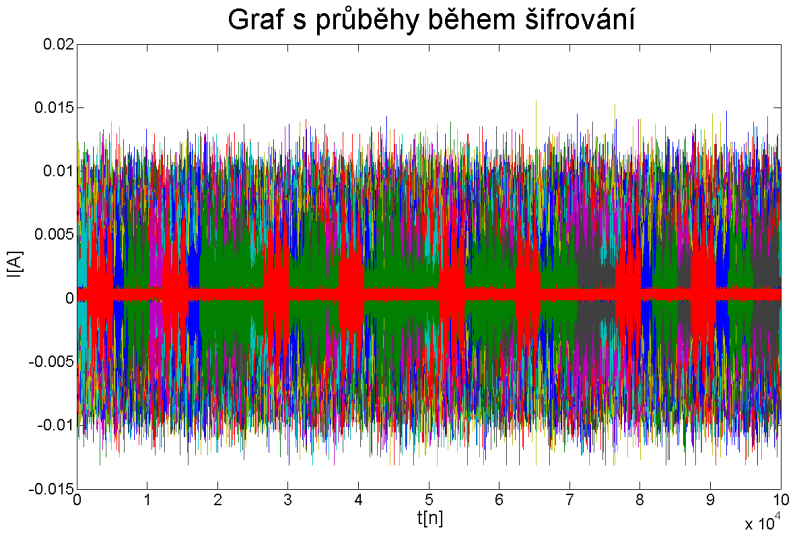
Automatizované měření se provádělo pomocí funkce napsané v prostředí MATLAB s*avewaveform_v1*, která se spustila zadáním příkazu *[input_Data, M1y, M2y] = savewaveform_v1(500, 'USB0::0x0699::0x0401::C010650: :INSTR')* kde input_Data, M1y, M2y jsou výstupní matice pro ukládání vstupních dat (plaintext), synchronizačních průběhů a výstupních dat při šifrování. Jako vstup funkce byly zadány hodnoty, počet měření a virtuální adresa připojeného osciloskopu.

Po změření všech hodnot byly matice manuálně uloženy do formátu *.mat* pro pozdější zpracování. Níže jsou vidět ukázkové grafy vytvořené z naměřených hodnot.

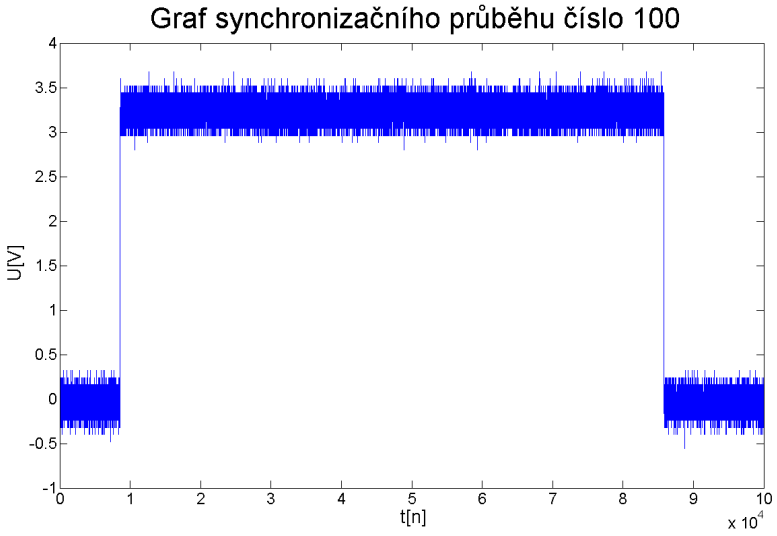
Ukázkové grafy vytvořené z hodnot elektromagnetické sondy



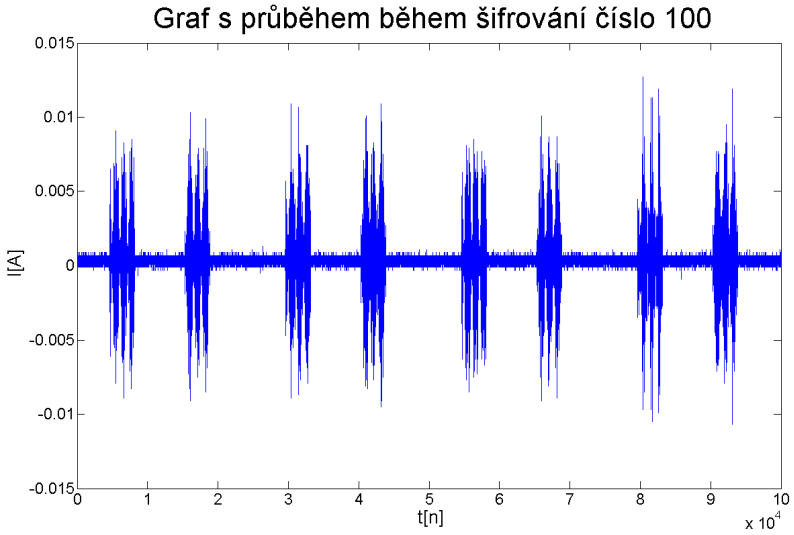
Obr. 3.8: Graf synchronizačních průběhů elektromagnetické sondy



Obr. 3.9: Graf s průběhy během šifrování elektromagnetické sondy

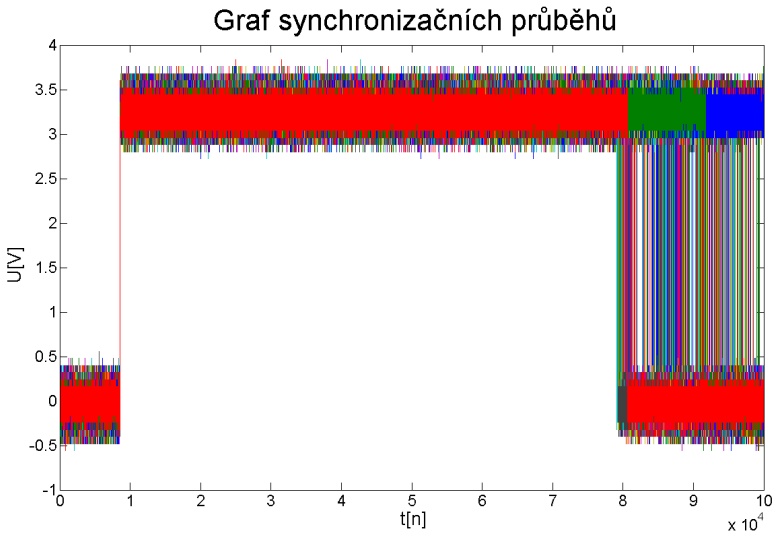


Obr. 3.10: Graf synchronizačního průběhu číslo 100, elektromagnetická sonda



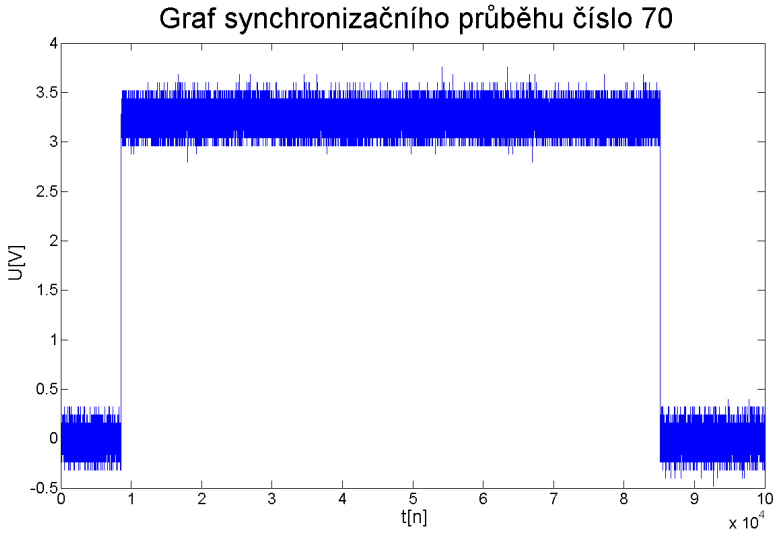
Obr. 3.11: Graf s průběhem během šifrování číslo 100, elektromagnetická sonda

Ukázkové grafy vytvořené z hodnot proudové sondy

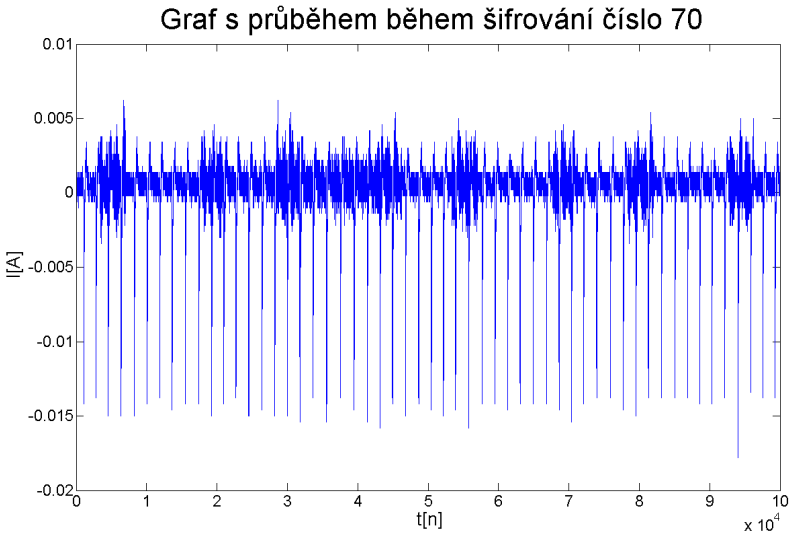


Obr. 3.12: Graf synchronizačních průběhů proudové sondy

Obr. 3.13: Graf s průběhy během šifrování proudové sondy



Obr. 3.14: Graf synchronizačního průběhu číslo 70, proudová sonda

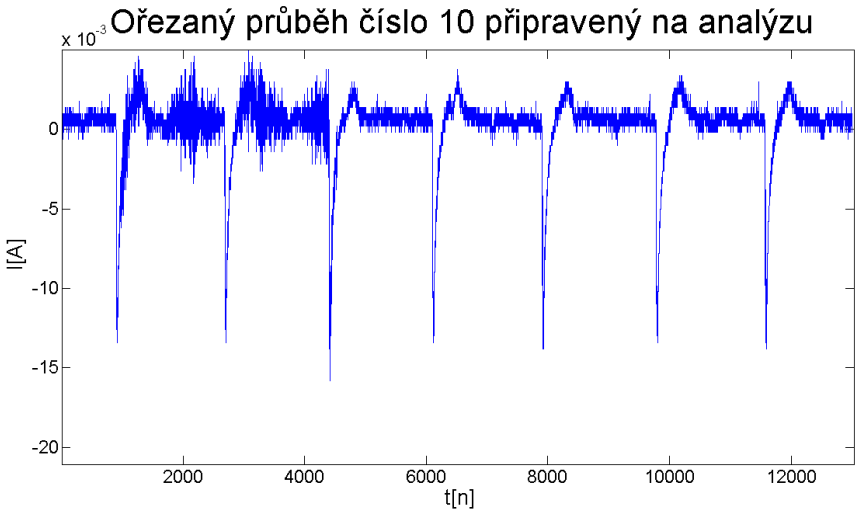


Obr. 3.15: Graf s průběhem během šifrování číslo 70, proudová sonda

4 ZPRACOVÁNÍ VÝSLEDKŮ

Vstupní data pro šifrování byla náhodně generována, a proto průběhy naměřené v rámci praktické části práce mohou být vhodné na diferenciální proudovou analýzu. Pro realizaci diferenciální proudové analýzy byly použité skripty, které byly vytvořené na základě skriptu dostupného ze zdroje [10]. Skript realizuje útok na výstupní data během provádění operace záměny bajtů šifry AES během první rundy. A proto vstupní data skriptu obsahují hodnoty operace přičtení klíče a substituce bajtů. V skriptu jsou zahrnuty dvě metody diferenciální proudové analýzy, a to Kocherova metoda a metoda korelačních koeficientů. V rámci práce byl použitý klíč o délce 16 bajtů, a protože se útok provádí vždy na 1 bajt klíče pro odhalení celého klíče, je útok proveden 16krát.

Před provedením vlastní analýzy bylo nutné průběhy zpracovat. Jelikož pravděpodobně kvůli procesům běžících na pozadí operačního systému jsou délky synchronizačních průběhů a proto i délky průběhů proudových během šifrování proměnné. Z naměřených 500 průběhů byly vybrány ty který byly nejblíž k sobě. V případě měření proudovou sondou se jedná o 136 průběhů a v případě elektromagnetické sondy o 152 průběhů. Po výběru a předběžným ořezáním byly průběhy dlouhé 78 400 bodů, což odpovídá celkové délce šifrovacího procesu tj. 10 rund. Proto byly dále ořezávány na délku 13 000 bodů, což přibližně odpovídá požadované délce pro analýzu. Ukázkový průběh ořezaný k nalýze je vidět na Obr. 4.1 Přesnou délku kvůli výše zmíněnému kolísání délek naměřených průběhů není možné určit.



Obr. 4.1: Ořezaný proudový průběh číslo 10

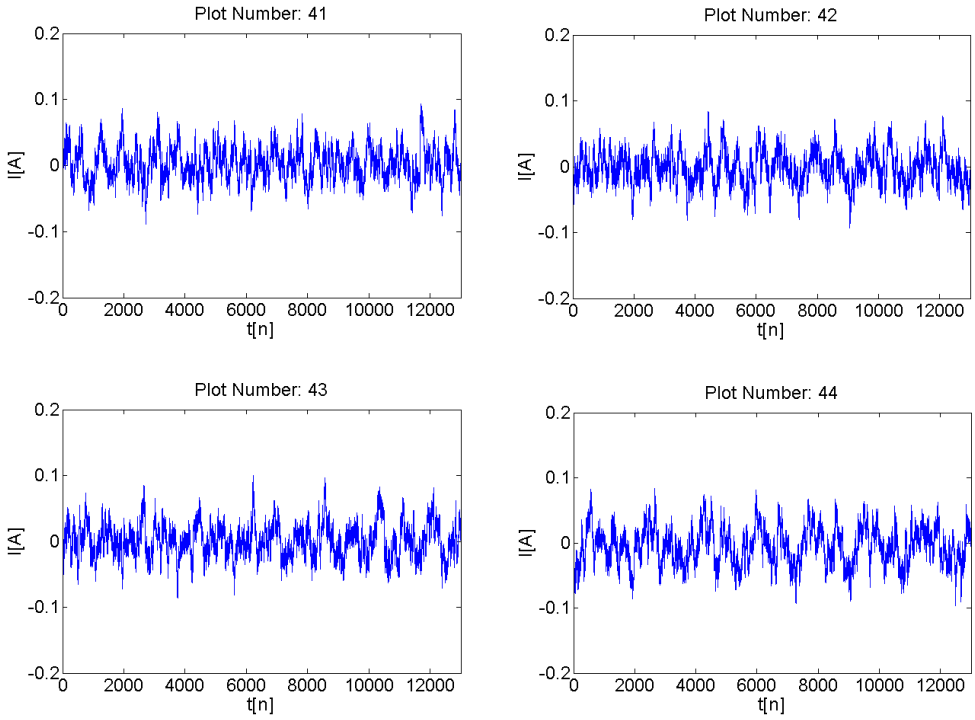
Níže jsou znázorněny výsledky analýzy pomocí modifikovaných skriptů, které byly vytvořeny na základě literatury [10]. Během šifrování byl použitý klíč který je rozepsán v tabulce Tab. 4.1.

Tab. 4.1: Šifrovací klíč použitý během měření na modulu Raspberry Pi

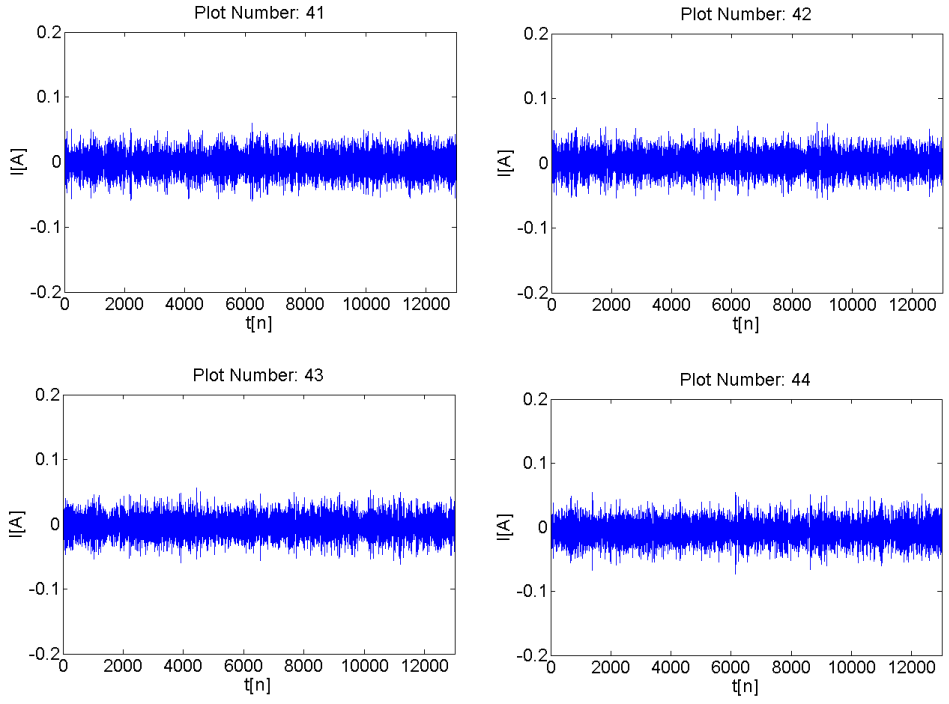
Pořadí bajtu	**Hodnota**	**Pořadí bajtu**	**Hodnota**
1. bajt	43	9. bajt	171
2. bajt	126	10. bajt	247
3. bajt	21	11. bajt	21
4. bajt	22	12. bajt	136
5. bajt	40	13. bajt	9
6. bajt	174	14. bajt	207
7. bajt	210	15. bajt	79
8. bajt	166	16. bajt	60

Jako první byl realizován útok Kocherovou metodou a odhaloval se 5. bajt klíče. Skript je napsán ve formě funkce v prostředí MATLAB a volá se zadáním příkazu *output = kocher(input_plain, traces, sub_bytes)* do příkazového řádku MATLAB. Výstupní parametr *output* o velikosti 256×13 000 bodů obsahuje výsledky analýzy. Každý řádek reprezentuje hodnotu jednoho bajtu a každý bajt může nabývat hodnot 0 až 255. Podle teoretického předpokladu v případě nalezení správného klíče se jeden z průběhů bude značně lišit od ostatních. Vstupní parametry jsou *input_plain* o velikosti 136×16 bajtů a obsahuje hodnoty šifrovaných textů, *traces* (velikost 136×13 000 bodů) obsahuje ořezané průběhy připraveny na analýzu a *sub_bytes* o velikosti 1×256 bajtů který je naplněn hodnotami Sboxu. Hodnota 5. bajtu šifrovacího klíče je známa a podle tabulky Tab. 4.1 je 40. Z výstupní matice skriptu by měl být odlišný průběh s číslem 41 protože první průběh z matice odpovídá hodnotě bajtu 0, druhý k hodnotě bajtu 1 a průběh 41 odpovídá k hodnotě 40. Na obrázku Obr. 4.2 jsou vidět čtyři průběhy pro bajty 40 až 43 pro průběhy změřené proudovou sondou (značení průběhů je Plot Number: 41 až Plot Number: 44) ale ani jeden z průběhů není výrazně odlišný, 5. bajt klíče nebyl odhalen. Na obrázku Obr. 4.3 jsou průběhy pro odhalení obdobného klíče, ale průběhy byly změřeny elektromagnetickou sondou. Protože výsledek analýzy je pro všechny bajty klíče a pro obě sondy podobný, tj. klíč nebyl odhalen, další výsledky Kocherovi metody nejsou prezentovány.

Další metoda byla korelační metoda a odhadoval se 10. bajt klíče. Skript byl napsán jako v předchozím případě ve formě funkce a volá se příkazem *output = korelace(input_plain, traces, sub_bytes, byte_Hamming_weight)*. K vstupním parametrům



Obr. 4.2: Průběhy klíče pro bajty 40 až 43 (průběhy měřené proudovou sondou)

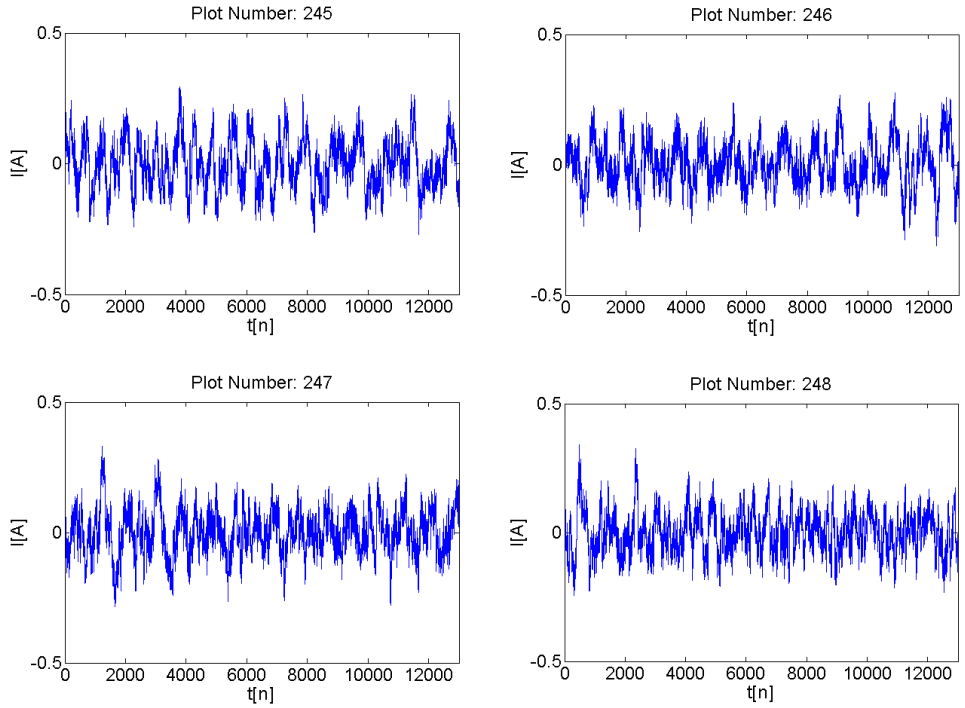


Obr. 4.3: Průběhy klíče pro bajty 40 až 43 (průběhy měřené EM sondou)

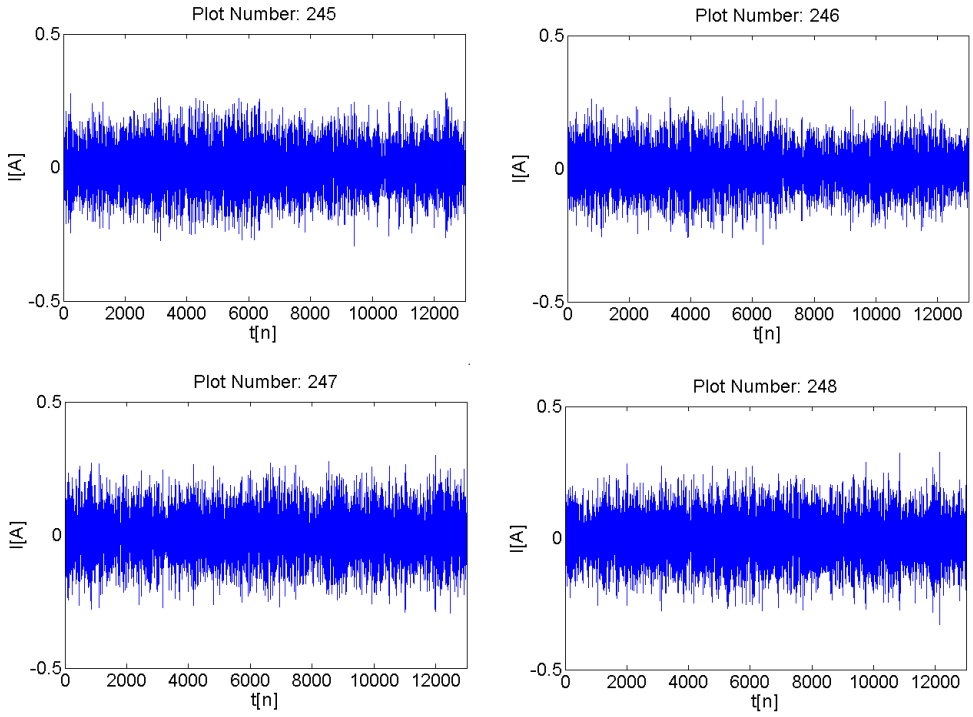
přibila matice *byte_Hamming_weight* která má velikost 1×256 bajtů a obsahuje hodnoty Hammingovi váhy dle standardního modelu. Ostatní parametry se shodují s parametry funkce *kocher()*. Ukázkové výsledky analýzy korelační metodou pro odhalení 10. bajtu klíče jsou vidět na obrázku Obr. 4.4 a Obr. 4.5. Hodnota klíče je podle tabulky Tab. 4.1 247. Podle předpokladu průběh číslo 246 z výstupní matice by musel být značně odlišný od ostatních průběhů, což se nestalo. Hledaný klíč ani v případě korelační metody nebyl odhalen.

Vyhodnocení diferenciální proudové analýzy

Byly realizovány diferenciální proudové analýzy pomocí Kocherovou metodou a metodou korelační. Metody byly použité na průběhy měřené proudovou sondou i na průběhy měřené elektromagnetickou sondou. Analýza ve všech případech byla neúspěšná, ani jeden z předpokládaných výsledných průběhů nebyl odlišný od ostatních. Důvodem neúspěšnosti analýz bylo pravděpodobně kolísání času potřebné k vykonání šifrování vstupních dat. Jelikož se jednalo o složitý modul, na kterém běžel vyvinutý operační systém, procesy běžící na pozadí mohly nepříznivě ovlivnit průběh šifrování.



Obr. 4.4: Průběhy klíče pro bajty 244 až 247 (průběhy měřené proudovou sondou)



Obr. 4.5: Průběhy klíče pro bajty 244 až 247 (průběhy měřené EM sondou)

5 ZÁVĚR

V rámci diplomové práce byla prostudována problematika proudového postranního kanálu, a základní metody analýz. Bylo provedeno vyhledávání mezi dostupnými produkty zaměřené na moduly, které mají hardwarově implementovaný kryptografický modul. Nalezené produkty jsou popsány v kapitole 1.3. Pro praktickou část dle dohody s vedoucím byly vybrány moduly: karta Gemalto .NET v2 s čtečkou Omnikey 3121 a karta Sm@rtCafé Expert 4.x a čtečkou SDI010 a modul Raspberry Pi model B. Byl nainstalován software pro obě karty a byly připojeny obě čtečky karet, ale komunikaci s kartou Sm@rtCafé Expert 4.x se nepodařilo docílit. Do popisu pracoviště jsou zařazena i tato zařízení.

V praktické části byla rozjeta komunikace s .NET kartou a prostudována struktura souborů a nastavění na kartě. Byla vytvořena aplikace na straně serveru (čipová karta) a na straně klienta (počítač). Byl napsán kód v programovacím jazyce C# pro implementaci šifrovacího algoritmu AES s výjimkou části expanze šifrovacího klíče. Tabulka pro rozšířený klíč se vychází z literatury [2] a do programu byla doplněna metoda pro rozdělení rozšířeného klíče na rundovní klíče. Bylo zašifrováno 16bajtové slovo `00 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff` s klíčem `00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f`. Výsledné slovo je vidět v tabulce Tab. A.1 v řádku runda 10 a vypadá následovně `69 c4 e0 d8 6a 7b 04 30 d8 cd b7 80 70 b4 c5 5a`. Po šifrování bylo provedeno dešifrování a výsledkem bylo původní slovo. Hodnoty slova na začátku každé rundy u dešifrování byly shrnuty do tabulky viz A.2. Tato tabulka je kromě prvního a posledního řádku identická s tabulkou šifrování. První řádek se liší kvůli tomu, že v případě šifrování je na ní již provedena transformace *addRoundKey()*. Poslední řádek je odlišný z důvodu aplikace funkce *invShiftRows()* a *invSubBytes()* u dešifrování. V případě Gemalto karty proudová analýza nebyla provedena.

V diplomové práci byl algoritmus implementován i na modul Raspberry Pi model B. Původní program byl modifikován a rozšířen. Byly vynechány části týkající se dešifrování, protože k provedení analýzy nebyly potřebné. Byl doplněn kód pro řízení GPIO pinů na modulu kvůli možnosti synchronizace. Dále byla doplněna metoda pro čtení vstupních dat pro šifrování. Posílání vstupních dat na modul a osciloskop byl řízen z počítače z vývojového prostředí MATLAB. Byly napsaný funkce pro řízení a zaznamenávaní průběhů i pro následné zpracování dat k provedení analýzy. Ukázkové grafy jsou vidět v kapitole 3.4. Byla provedena diferenciální proudová analýza Kocherovou metodou a korelační metodou. K vykonání analýzy byly použity skripty, které byly vytvořeny na základě skriptu dostupného na webových stránkách [10]. Zpracované výsledky jsou vidět v kapitole 4. Analýza byla v obou případech

neúspěšná pravděpodobně z důvodu kolísání času potřebného pro vykonání šifrování. Kolísání bylo pravděpodobně působeno procesy operačního systému běžícími na pozadí. Poslední kapitola je věnovaná závěru práce.

LITERATURA

[1] AES' Galois field. *Sam Trenholme's webpage* [online]. 2014 [cit. 2015-05-18]. Dostupné z: <http://www.samiam.org/galois.html>

[2] Announcing the ADVANCED ENCRYPTION STANDARD (AES). In: *NIST* [online]. November 26, 2001 [cit. 2014-12-14]. Dostupné z: <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>

[3] ATSAM ARM-based Flash MCU: SAM4L Series. In: *Atmel* [online]. 03/2014 [cit. 2014-12-13]. Dostupné z: <http://www.atmel.com/Images/Atmel-42023-ARM-Microcontroller-ATSAM4L-Low-Power-LCD_Datasheet-Summary.pdf>

[4] BAR-EL, Hagai. *Introduction to Side Channel Attacks.* [online]. [cit. 2014-11-09]. Dostupné z: <http://gauss.ececs.uc.edu/Courses/c653/lectures/SideC/intro.pdf>

[5] CARDIS 2013 ORGANIZATION COMMITTEE. *12th Smart Card Research and Advanced Application Conference* [online]. © 2013 [cit. 2014-11-29]. Dostupné z: <http://cardis.sec.t-labs.tu-berlin.de/index.html>

[6] Configure the serials port on the Raspberry Pi. *PrivateEyePi* [online]. [Apr 21, 2015] [cit. 2015-05-18]. Dostupné z: <http://www.projects.privateeyepi.com/home/home-alarm-system-project/wireless-projects/configure-the-serials-ports-on-the-raspberry-pi>

[7] CP210x USB to UART Bridge VCP Drivers. 2015. *Silicon Labs* [online]. [cit. 2015-05-08]. Dostupné z: <https://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx>

[8] Downloads. 2015. *Raspberry Pi* [online]. [cit. 2015-05-08]. Dostupné z: <https://www.raspberrypi.org/downloads/>

[9] *DPA contest v4* [online]. [2013] [cit. 2015-05-19]. Dostupné z: <http://www.dpacontest.org/v4/>

[10] *DPAbook.org: Power Analysis Attacks - Revealing the Secrets of Smartcards* [online]. 2010 [cit. 2015-05-19]. Dostupné z: <http://www.dpabook.org/index.htm>

[11] EISENBARTH, Thomas. *Workshop on Cryptographic Hardware and Embedded Systems* [online]. Poslední aktualizace: 23.10.2014 [cit. 2014-11-29]. Dostupné z: <http://www.chesworkshop.org/index.php>

[12] Gemalto .NET 2.0 Smart Card. In: *Gemalto* [online]. May 2007 [cit. 2014-12-04]. Dostupné z: <http://www.gemalto.com/dwnld/5042_070520_WP_Gemalto_.NET_Certificate_Enrollment_using_MSFT_Certificate_Services.pdf>

[13] HOLEMÁŘ, Jan. *Postranní kanály - vytvoření laboratorní úlohy* [online]. Brno, 2012 [cit. 2014-12-02]. Dostupné z: <https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=69150>. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce Ing. Zdeněk Martinásek.

[14] Installing Operating System Images using Windows. 2015. *Raspberry Pi* [online]. [cit. 2015-05-08]. Dostupné z: <https://www.raspberrypi.org/documentation/installation/installing-images/windows.md>

[15] ISO 7816. *CardWerk: Smarter Card Solution* [online]. Copyright 1999-2014 [cit. 2014-12-04]. Dostupné z: <http://www.cardwerk.com/smartcards/smartcard_standard_ISO7816.aspx>

[16] KERCKHOF, Stéphanie, Francois-Xavier STANDAERT a Eric PEETERS. From New Technologies to New Solutions. In: *CARDIS 2013* [online]. 2013 [cit. 2015-05-19]. Dostupné z: <http://cardis.sec.t-labs.tu-berlin.de/proceedings/CARDIS2013_2.pdf>

[17] LERMAN, Liran, Stephane FERNANDES MEDEIROS, Gianluca BONTEMPI a Olivier MARKOWITCH. A machine learning approach against a masked AES. In: *CARDIS 2013* [online]. 2013 [cit. 2015-05-19]. Dostupné z: <http://cardis.sec.t-labs.tu-berlin.de/proceedings/CARDIS2013_5.pdf>

[18] MANGARD, S., E. OSWALD a T. POPP. *Power Analysis Attacks: Revealing the Secrets of Smart Cards.* Secaucus, NJ: USA:Springer-Verlag New York, Inc., 2007. ISBN 0-387-30857-1.

[19] Mono (C#) anyone? 2012. *Raspberry Pi* [online]. [cit. 2015-05-10]. Dostupné z: <https://www.raspberrypi.org/forums/viewtopic.php?p=88063>

[20] *Mono Project* [online]. 2015. [cit. 2015-05-08]. Dostupné z: <http://www.mono-project.com/>

[21] NI System Configuration 14.5.0. 2015. *National Instruments* [online]. [cit. 2015-05-08]. Dostupné z: <http://www.ni.com/download/ni-system-configuration-14.5.0/5158/en/>

[22] NI-VISA 14.0.1 - ETS. 2015. *National Instruments* [online]. [cit. 2015-05-08]. Dostupné z: <http://www.ni.com/download/ni-visa-14.0.1/5023/en/>

[23] NOOBS Setup. 2015. *Raspberry Pi* [online]. [cit. 2015-05-08]. Dostupné z: <https://www.raspberrypi.org/help/noobs-setup/>

[24] NXP Semiconductors. *NXP chip card IC SmartMX2-P40 for eGov and Banking* [online]. Netherlands, January 2014 [cit. 2014-11-29]. Dostupné z: <http://www.nxp.com/documents/leaflet/939775017483.pdf>

[25] NXP Semiconductors. *NXP secure microcontroller family SmartMX2* [online]. Netherlands, January 2014 [cit. 2014-11-29]. Dostupné z: <http://www.nxp.com/documents/leaflet/75017516.pdf>

[26] NXP Semiconductors. *NXP SmartMX high security microcontroller IC* [online]. Netherlands, January 2014 [cit. 2014-11-29]. Dostupné z: <http://www.nxp.com/documents/leaflet/75017515.pdf>

[27] OBRUČNÍK, Ondřej. *Proudový postranní kanál mikroprocesorů.* Brno, 2010. Dostupné z: <https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=26419>. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Vedoucí práce Ing. Zdeněk Martinásek

[28] OMNIKEY ® 3121 USB Desktop Reader. In: *HID Global* [online]. 2013-09-12 [cit. 2014-12-04]. Dostupné z: <http://www.hidglobal.com/sites/hidglobal.com/files/resource_files/ok-3121-usb-ds-en_0.pdf>

[29] PIC32MZ Embedded Connectivity (EC) Family. In: *Microchip* [online]. © 2013-2014 [cit. 2014-12-13]. Dostupné z: <http://ww1.microchip.com/downloads/en/DeviceDoc/60001191C.pdf>

[30] Raspberry Pi and Mono – Hello World!. 2013. *Dan's Website* [online]. [cit. 2015-05-08]. Dostupné z: <http://logicalgenetics.com/raspberry-pi-and-mono-hello-world/>

[31] Raspberry Pi C# Mono Hello World Setup / Installation / Compile (gmcs) / Execution. 2014. *Hardware Hacks* [online]. [cit. 2015-05-08]. Dostupné z: <http://c-mobberley.com/wordpress/2014/09/26/raspberry-pi-c-mono-hello-world-setup-installation-compile-gmcs-execution/>

[32] Rijndael. In: DAEMEN, Joan a Vincent RIJMEN. *Note on naming* [online]. 9/04/2003 [cit. 2014-12-02]. Dostupné z: <http://csrc.nist.gov/archive/aes/rijndael/Rijndael-ammended.pdf>

[33] Rijndael's mix column stage. *Sam Trenholme's webpage* [online]. 2014 [cit. 2015-05-18]. Dostupné z: <http://www.samiam.org/mix-column.html>

[34] SDI010. *SCM PC-Card GmbH* [online]. © 2013 [cit. 2014-12-04]. Dostupné z: <http://www.scm-pc-card.de/index.php?page=product&function=show_product&lang=de&product_id=230>

[35] Tektronix DPO4032 Oscilloscope. 2015. *The MathWorks, Inc.* [online]. [cit. 2015-05-08]. Dostupné z: <http://www.mathworks.com/matlabcentral/fileexchange/18225-tektronix-dpo4032-oscilloscope>

[36] VALÁŠEK, Michal A. Symetrické šifrování AES/Rijndael v .NET. VALÁŠEK, Michal A. *Aspnet.cz* [online]. 16. 4. 2007 [cit. 2014-12-02]. Dostupné z: <http://www.aspnet.cz/Articles/147-symetricke-sifrovani-aes-rijndael-v-net.aspx>

[37] What is a Raspberry Pi? 2015. *Raspberry Pi* [online]. [cit. 2015-05-07]. Dostupné z: <https://www.raspberrypi.org/help/what-is-a-raspberry-pi/>

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

3DES	Standard šifrování dat trojnásobnou aplikací – Triple Data Encryption Standard
AES	Standard pokročilého šifrování – Advanced Encryption Standard
APDU	Komunikační jednotka mezi čtečkou karet a čipovou kartou – Application Protocol Data Unit
CLR	Běhové prostředí primárně navržen pro čipové karty – Common Language Runtime
DPA	Diferenciální proudová analýza – Differential Power Analysis
ECC	Kryptografie eliptických křivek – Elliptic Curve Cryptography
ECMA	Mezinárodní organizace pro standardizace – International, private (membership-based) non-profit standards organization for information and communication systems
EEPROM	Elektricky mazatelná semipermanentní paměť typu ROM – Electrical Erasable Programmable Read-only Memory
FIPS	Federální standart zpracování informací – Federal Information Processing Standard
FPGA	Programovatelná hradlová pole – Field-Programmable Gate Array
GPIO	Vstupní/Výstupní piny pro všeobecné použití – General Purpose Input/Output
GUI	Grafické uživatelské rozhraní – Graphical User Interface
HDMI	Označení nekomprimovaného obrazového a zvukového signálu v digitálním formátu – High-Definition Multimedia Interface
HMAC	Kombinovaná hašovací funkce– Keyed-Hash Message Authentication Code
LCD	Zobrazovací zařízení, displej z tekutých krystalů – Liquid - Crystal Display
MD5	Specifický typ hašovací funkce – Message-Digest algorithm
MMU	Jednotka pro efektivní řízení paměti – Memory Management Unit

RAM	Paměť s přímým přístupem – Random Access Memory
RISC	Procesor s redukovanou instrukční sadou – Reduced Instruction Set Computing
ROM	Paměť pouze pro čtení – Read-Only Memory
SHA	Bezpečný hašovací algoritmus – Secure Hash Algorithm
SPA	Jednoduchá proudová analýza – Simple Power Analysis
UART	Asynchronní rozhraní pro přenos a příjem dat – Universal Asynchronous Receiver/Transmitter
USB	Univerzální sériová sběrnice – Universal Serial Bus

A UKÁZKA VÝSTUPNÍCH TABULEK

A.1 Tabulky šifrování

Tab. A.1: Hodnoty slova při startu jednotlivých rund

runda 1	00-10-20-30-40-50-60-70-80-90-A0-B0-C0-D0-E0-F0
runda 2	89-D8-10-E8-85-5A-CE-68-2D-18-43-D8-CB-12-8F-E4
runda 3	49-15-59-8F-55-E5-D7-A0-DA-CA-94-FA-1F-0A-63-F7
runda 4	FA-63-6A-28-25-B3-39-C9-40-66-8A-31-57-24-4D-17
runda 5	24-72-40-23-69-66-B3-FA-6E-D2-75-32-88-42-5B-6C
runda 6	C8-16-77-BC-9B-7A-C9-3B-25-02-79-92-B0-26-19-96
runda 7	C6-2F-E1-09-F7-5E-ED-C3-CC-79-39-5D-84-F9-CF-5D
runda 8	D1-87-6C-0F-79-C4-30-0A-B4-55-94-AD-D6-6F-F4-1F
runda 9	FD-E3-BA-D2-05-E5-D0-D7-35-47-96-4E-F1-FE-37-F1
runda 10	69-C4-E0-D8-6A-7B-04-30-D8-CD-B7-80-70-B4-C5-5A

Tab. A.2: Hodnoty slova v jednotlivých rundách po substituci

runda 1	63-CA-B7-04-09-53-D0-51-CD-60-E0-E7-BA-70-E1-8C
runda 2	A7-61-CA-9B-97-BE-8B-45-D8-AD-1A-61-1F-C9-73-69
runda 3	3B-59-CB-73-FC-D9-0E-E0-57-74-22-2D-C0-67-FB-68
runda 4	2D-FB-02-34-3F-6D-12-DD-09-33-7E-C7-5B-36-E3-F0
runda 5	36-40-09-26-F9-33-6D-2D-9F-B5-9D-23-C4-2C-39-50
runda 6	E8-47-F5-65-14-DA-DD-E2-3F-77-B6-4F-E7-F7-D4-90
runda 7	B4-15-F8-01-68-58-55-2E-4B-B6-12-4C-5F-99-8A-4C
runda 8	3E-17-50-76-B6-1C-04-67-8D-FC-22-95-F6-A8-BF-C0
runda 9	54-11-F4-B5-6B-D9-70-0E-96-A0-90-2F-A1-BB-9A-A1
runda 10	7A-9F-10-27-89-D5-F5-0B-2B-EF-FD-9F-3D-CA-4E-A7

Tab. A.3: Hodnoty slova v jednotlivých rundách po transformaci *mixColumn()*

runda 1	5F-72-64-15-57-F5-BC-92-F7-BE-3B-29-1D-B9-F9-1A
runda 2	FF-87-96-84-31-D8-6A-51-64-51-51-FA-77-3A-D0-09
runda 3	4C-9C-1E-66-F7-71-F0-76-2C-3F-86-8E-53-4D-F2-56
runda 4	63-85-B7-9F-FC-53-8D-F9-97-BE-47-8E-75-47-D6-91
runda 5	F4-BC-D4-54-32-E5-54-D0-75-F1-D6-C5-1D-D0-3B-3C
runda 6	98-16-EE-74-00-F8-7F-55-6B-2C-04-9C-8E-5A-D0-36
runda 7	C5-7E-1C-15-9A-9B-D2-86-F0-5F-4B-E0-98-C6-34-39
runda 8	BA-A0-3D-E7-A1-F9-B5-6E-D5-51-2C-BA-5F-41-4D-23
runda 9	E9-F7-4E-EC-02-30-20-F6-1B-F2-CC-F2-35-3C-21-C7
runda 10	V poslední rundě se tato operace neprovádí

A.2 Ukázková tabulka dešifrování

Tab. A.4: Dešifrování: hodnoty slova při startu jednotlivých rund

runda 10	00-11-22-33-44-55-66-77-88-99-AA-BB-CC-DD-EE-FF
runda 9	89-D8-10-E8-85-5A-CE-68-2D-18-43-D8-CB-12-8F-E4
runda 8	49-15-59-8F-55-E5-D7-A0-DA-CA-94-FA-1F-0A-63-F7
runda 7	FA-63-6A-28-25-B3-39-C9-40-66-8A-31-57-24-4D-17
runda 6	24-72-40-23-69-66-B3-FA-6E-D2-75-32-88-42-5B-6C
runda 5	C8-16-77-BC-9B-7A-C9-3B-25-02-79-92-B0-26-19-96
runda 4	C6-2F-E1-09-F7-5E-ED-C3-CC-79-39-5D-84-F9-CF-5D
runda 3	D1-87-6C-0F-79-C4-30-0A-B4-55-94-AD-D6-6F-F4-1F
runda 2	FD-E3-BA-D2-05-E5-D0-D7-35-47-96-4E-F1-FE-37-F1
runda 1	BD-6E-7C-3D-F2-B5-77-9E-0B-61-21-6E-8B-10-B6-89

B POPIS VYTVOŘENÝCH METOD, MODUL GEMALTO

Metody na serverové straně:

```
public byte[] Encrypt()
```

Je to hlavní metoda, která obsahuje metody potřebné pro provedení šifrování. Struktura je v souladu s postupem popsaným výše.

```
public byte[] Decrypt()
```

Metoda obsahuje kód pro provedení dešifrování.

```
public byte[] inputPlainToByte(string[] input)
```

Metoda je použitá pro provedení vstupního slova z typu string[] na typ byte[].

```
public byte[] stringToNumber(string inputString)
```

Je to pomocná metoda, která na vstupu bere číselnou hodnotu jednoho bajtu v typu string a na výstupu vrací dvě čísla typu byte, které slouží k identifikaci pozice v tabulce S-Box.

```
public byte[] sBoxSubstitute(string[] input)
```

Provede substituci vstupního slova, které je typu string[] a vrací pole typu byte[] pro další zpracování.

```
public byte[] sBoxSubByteIn(byte[] input)
```

Na vstupu bere pole typu byte[], provede substituci a vrací pole typu byte[] se substituovanými hodnotami.

```
public byte[] shiftRows(byte[] input)
```

Metoda slouží pro rotování řádků dle pravidel AES.

```
public byte gmul(byte a, byte b)
```

Pomocná metoda, která provádí násobení dvou čísel dle pravidel AES.[1]

```
public byte[] mixColumn(byte[] r)
```

Slouží pro kombinování všech hodnot v určitém sloupci.[33]

```
public byte[] addRoundKey(byte[] input, byte[] expandedKey)
```

Metoda přidává rundovní klíč během šifrování i dešifrování.

```
public byte[] getExpandedKey(int roundNumber)
```

V závislosti na počtu provedených rund vrací hodnotu rundovního klíče z expandovaného klíče.

`public static byte[] invShiftRows(byte[] input)`

Metoda je použita u dešifrování a slouží pro rotování řádků dle pravidel AES.

`public byte[] invMixColumn(byte[] r)`

Inverzní metoda provede míchaní hodnot v určitém sloupci.

`public byte[] invsBoxSubByteIn(byte[] input)`

Substituční metoda, která používá inverzní tabulku S-Box.

C UKÁZKA VYBRANÝCH ZDROJOVÝCH KÓDŮ

Hlavní metoda pro šifrování

```
public byte[] Encrypt()
        {
        string[] inputPlain = { "00", "11", "22", "33", "44",
"55", "66", "77", "88", "99", "aa", "bb", "cc", "dd",
"ee", "ff" };
        string[] key = {"00", "01", "02", "03", "04", "05",
"06", "07", "08", "09", "0a", "0b", "0c", "0d", "0e", "0f"};

        byte[] result = new byte[16];
        byte[] resultStart = new byte[16];
        byte[] resultSR = new byte[16];

        resultStart = inputPlainToByte(inputPlain);
        resultStart = addRoundKey(resultStart,
getExpandedKey(0));

       for (int x = 0; x < 9; x++)
       {
           result = sBoxSubByteIn(resultStart);
           resultSR = shiftRows(result);

           byte[] split0 = new byte[4];
           byte[] split1 = new byte[4];
           byte[] split2 = new byte[4];
           byte[] split3 = new byte[4];

           for (int i = 0; i < 4; i++)
           {
                split0[i] = resultSR[i];
                split1[i] = resultSR[i + 4];
                split2[i] = resultSR[i + 8];
                split3[i] = resultSR[i + 12];
           }
```

```
            split0 = mixColumn(split0);
            split1 = mixColumn(split1);
            split2 = mixColumn(split2);
            split3 = mixColumn(split3);

            byte[] afterMix = new byte[16];
            for (int i = 0; i < 4; i++)
            {
                  afterMix[i] = split0[i];
                  afterMix[i + 4] = split1[i];
                  afterMix[i + 8] = split2[i];
                  afterMix[i + 12] = split3[i];
            }
      resultStart = addRoundKey(afterMix,
getExpandedKey(x+1));

      }

      result = sBoxSubByteIn(resultStart);
      resultSR = shiftRows(result);
      resultStart = addRoundKey(resultSR,
getExpandedKey(10));

      finalOutput = resultStart;

      return finalOutput;
}
```

Metoda pro rotování dle pravidel AES

```
public byte[] shiftRows(byte[] input){

      byte pom = 0;
      byte[] inMatrix = new byte[16];

      inMatrix = input;

      pom = inMatrix[1];
      inMatrix[1] = inMatrix[5];
      inMatrix[5] = inMatrix[9];
```

```
        inMatrix[9] = inMatrix[13];
        inMatrix[13] = pom;

        pom = inMatrix[2];
        inMatrix[2] = inMatrix[10];
        inMatrix[10] = pom;
        pom = inMatrix[6];
        inMatrix[6] = inMatrix[14];
        inMatrix[14] = pom;

        pom = inMatrix[15];
        inMatrix[15] = inMatrix[11];
        inMatrix[11] = inMatrix[7];
        inMatrix[7] = inMatrix[3];
        inMatrix[3] = pom;

        return inMatrix;
}
```

Metoda pro kombinování hodnot jednotlivých bloků v jednom sloupci.

```
public byte[] mixColumn(byte[] r){

        byte[] a = new byte[4];
        for (int c = 0; c < 4; c++)
        {
            a[c] = r[c];
        }

         r[0]=(byte)(gmul(a[0],2) ^ gmul(a[3],1) ^
gmul(a[2],1) ^ gmul(a[1],3));

         r[1]=(byte)(gmul(a[1],2) ^ gmul(a[0],1) ^
gmul(a[3],1) ^ gmul(a[2],3));

         r[2]=(byte)(gmul(a[2],2) ^ gmul(a[1],1) ^
gmul(a[0],1) ^ gmul(a[3],3));

         r[3]=(byte)(gmul(a[3],2) ^ gmul(a[2],1) ^
gmul(a[1],1) ^ gmul(a[0],3));
```

```
        return r;
}
```

Pomocná metoda pro násobení dvou čísel dle pravidel AES

```
public byte gmul(byte a, byte b){
        byte p = 0;
        byte help0x80 = 128;

        byte hi_bit_set;
        for (int counter = 0; counter < 8; counter++)
        {
        if ((b & 1) == 1)
           p ^= a;
           hi_bit_set = (byte)(a & help0x80);
           a <<= 1;
           if (hi_bit_set == 0x80)
              a ^= 0x1b;
              b >>= 1;
        }
        return p;
}
```