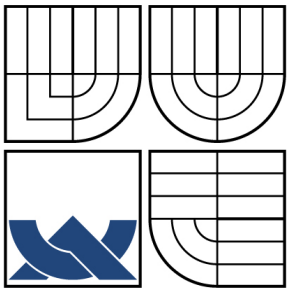
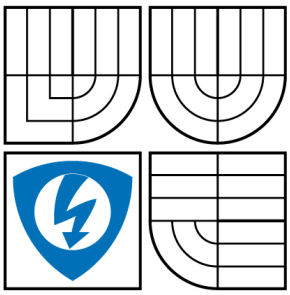


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

POSTRANNÍ KANÁLY V KRYPTOGRAFII

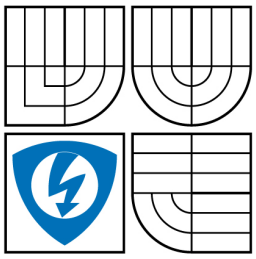
SIDE CHANNELS IN CRYPTOGRAPHY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — LUKÁŠ BUDÍK
AUTHOR

VEDOUCÍ PRÁCE — Ing. ZDENĚK MARTINÁSEK
SUPERVISOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Lukáš Budík	***ID:***	73076
Ročník:	3	***Akademický rok:***	2008/2009

NÁZEV TÉMATU:

Postranní kanály v kryptografii

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte základní útoky postranními kanály na kryptografický modul. Z nastudovaných teoretických vědomostí vypracujte přehled současného stavu problematiky. Navrhněte a realizujte laboratorní úlohu, která by demonstrovala útok vybraným postraním kanálem.

DOPORUČENÁ LITERATURA:

[1] ALFRED J. MENEYES, Paul C. van Oorschot, Scott A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1996

[2] KOCHER, P., JAFFE, J., JUN, B.: Introduction to Differential Power Analysis and Related Attacks, San Francisco, 1998. [.pdf dokument]. Dostupný z WWW: <http://www.cryptography.com/resources/ whitepapers/DPATechInfo.pdf>

[3] ZHOU, YB., FENG, DG.: Side-Channel Attacks: Ten Years After Its Publication and the Impacts on Cryptographic Module Security Testing. State Key Laboratory of Information Security, Institute of Software, Chinese Academy of Sciences, Beijing, 100080, China.

Termín zadání:	9.2.2009	***Termín odevzdání:***	2.6.2009

Vedoucí práce: Ing. Zdeněk Martinásek

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ANOTACE

Tato práce se zabývá problematikou postranních kanálu v kryptografii. Úvodní část popisuje základní členění oboru kryptologie. Dále jsou zmíněné požadavky na kryptografický systém z hlediska bezpečnosti. V další kapitole jsou popsány základní algoritmy a protokoly, které se používají v kryptografii. Druhá polovina práce definuje pojem postranní kanál a jeho jednotlivé typy. Závěr práce demonstruje útok časovým postranním kanálem na algoritmus RSA.

Klíčová slova: kryptoanalýza, kryptografie, kryptografický modul, postranní kanál, RSA

ABSTRACT

This work deals with a problem called sidelong cannals in cryptografy. First part describes basic segmentation of cryptografy branch. In addition to this the document mentions some requirements for cryptology system from security angle. In another section basic algorithms and protocols are described which are used in cryptology. The second part of this work defines conception of sidelong canal and its individual types. Conclusion of this task demonstrates an attack by time-side canal to algorithm RSA.

Keywords: cryptanalysis, cryptograph, cryptographic module**,** side channel, RSA

BUDÍK, L. Postranní kanály v kryptografii. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 49 s. Vedoucí bakalářské práce Ing. Zdeněk Martinásek.

Prohlášení

Prohlašuji, že svou bakalářskou práci na téma Postranní kanály v kryptografii jsem vypracoval samostatně pod vedením vedoucího bakalářské práce s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestně právních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne . . . . . .

. . . . . .

podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Zdeňkovi Martináskovi za užitečnou pomoc a cenné rady při zpracování práce.

V Brně dne . . . .
podpis autora

Obsah

Obsah 5
Seznam obrázků 6
Seznam tabulek 7
1. Úvod 8
2. Základní pojmy 9
2.1. Požadavky na kryptografický systém z hlediska bezpečnosti 9
2.1.1. *Kryptografické algoritmy a protokoly* 10
3. Kryptografický modul a útoky na něj 17
4. Postranní kanály 19
4.1. Časový postranní kanál 20
4.2. Chybový postranní kanál 21
4.3. Napěťově-proudový postranní kanál 24
4.4. Elektromagnetický postranní kanál 26
4.5. Eliminace postranních kanálu 26
5. Experimentální část 27
5.1. Programovací jazyk C++ a program Dev-C++ 27
5.2. Algoritmus RSA 29
5.2.1. Vytvoření klíče 30
5.2.2. Šifrování zprávy 32
5.2.3. Dešifrování zprávy 34
5.2.4. Využití časového postranního kanálu 34
6. Závěr 38
7. Laboratorní úloha 39
8. Literatura 47
Přílohy 49

Seznam obrázků

Obr. 2.1 Kryptografický systém 9
Obr. 2.2 Základní rozdělení 10
Obr. 2.3 Symetrická kryptografie 11
Obr. 2.4 Režim ECB 11
Obr. 2.5 Režim CBC 11
Obr. 2.6 Princip šifrování DES 13
Obr. 2.7 Feistelova šifra 13
Obr. 2.8 Obecný model šifrovaní DES 13
Obr. 2.9 Jeden krok algoritmu IDEA 14
Obr. 2.10 MD5 Hash 15
Obr. 2.11 Asymetrická kryptografie 16
Obr. 2.12 Digitální podpis 16
Obr. 3.1 Útoky na šifru 18
Obr. 4.1 Postranní kanály 19
Obr. 4.2 Časový postranní kanál 20
Obr. 4.3 Algoritmus Square and multiply 21
Obr. 4.4 Chybový postranní kanál 22
Obr. 4.5 Chybový postranní kanál v módu CBC 23
Obr. 4.6 Struktura bloku EB 24
Obr. 4.7 Měření spotřeby kryptografického modulu 24
Obr. 5.1 Struktura jazyka C++ v roce 1998 27
Obr. 5.2 Náhled programu Dev-C++ 28
Obr. 5.3 Ustavení SSL spojení 29
Obr. 5.4 Klíč délky 512 bitů 31
Obr. 5.5 Náhled programu na zpracování RSA 36

Seznam tabulek

Tabulka 1: Úvodní permutace..... 12
Tabulka 2: Srovnání symetrických algoritmů..... 15

1. Úvod

V současnosti používané systémy pro zpracování a přenos dat vyžadují vysoký stupeň kryptografického zabezpečení. Kryptograficky bezpečný systém musí zajistit dostatečné utajení a autentičnost zpracovávaných dat.

Nejmladší útoky kryptoanalytiků se zaměřovaly na prolomení jednoduchého šifrovaného textu. Nejznámější byl útok tzv. hrubou silou, kde se „útočník“ snažil vyzkoušet všechny možné kombinace tajného klíče. Postupem času vzniklo mnoho opatření proti těmto útokům a kryptoanalýza se stávala obtížnější. Postupem času přišli kryptoanalitici na to, že existuje jiný způsob vedení útoku, který využívá tzv. postranních kanálů. První informace o postranním kanálu přináší už rok 1965, kdy vědec P. Wright využil mikrofon k prolomení šifry. Pomocí něj odposlouchával zvuky, které vydával mechanický kryptografický modul.

Postranní kanály vynášejí důležité informace z kryptografického modulu, které při důkladné analýze umožňují prolomit bezpečnostní systém. To přináší nový způsob návrhu bezpečnostního systému. Nezáleží jen na šifře samé, ale také na způsobu jejího použití a realizace. Aby byl celý systém bezpečný, musí být použita kvalitní šifra, která je použita správným způsobem.

Kapitola 2 pojednává o oboru kryptologie, kde jsou popsány základní pojmy, algoritmy a protokoly. Následující kapitola 3 obsahuje popis kryptografického modulu. Kapitola 4 se zaměřuje na postranní kanály v kryptografii. Zde jsou definovány jednotlivé typy a možné způsoby jejich eliminace. Kapitole 5 podrobně popisuje algoritmus RSA, který jsem následně implementoval do jazyka C++. Pomocí časového postranního kanálu, který vzniká u algoritmu Square-and-Multiply při šifrování a dešifrování zprávy, jsem odhalil soukromý klíč.

Na závěr jsem navrhl laboratorní úlohu, která demonstruje útok časovým postranním kanálem

2. Základní pojmy

Problematika probírána v této práci se stává v dnešní době velmi ožehavým tématem. Na kryptografických systémech se zakládá spousta zabezpečovacích mechanismů, které jsou spojeny s našimi každodenními úkony.

Kryptografické systémy zaručují:

- **důvěrnost dat** – utajení informace před neoprávněnými uživateli,
- **integritu dat** – zaručuje, aby data nemohl pozměnit neoprávněný uživatel,
- **autentičnost** – ověření identity dat nebo entity,
- **neporanitelnost** – zabránění popření vykonané věci (např. vytvoření zprávy),
- **kontrola přístupu** – zákaz přístupu neoprávněným entitám k objektu,
- **autorizaci** – vykonávaní činnosti je omezeno pouze na oprávněné subjekty.

Na pochopení celé problematiky si v první kapitole vysvětlíme základní pojmy a principy kryptografie a nejznámější algoritmy.

Základní pojmy:

- **kryptografie** – (z řečtiny – kryptós = skrytý, gráphein = psát) věda, které se zabývá vytvářením šifrovacích systémů,
- **kryptoanalýza** – (z řečtiny – kryptós = hidden, analýein – uvolnit, rozvázat) hlavním úkolem je rozšifrování zprávy bez znalosti příslušného klíče,
- **kryptologie** – společný název pro kryptografii a kryptoanalýzu.

Blokové schéma kryptografického systému, které je vidět na Obr. 2.1:

Obr. 2.1 Kryptografický systém

2.1. Požadavky na kryptografický systém z hlediska bezpečnosti

Pojem kryptografický systém v sobě ukrývá jednotlivé matematické metody, návrhy klíčů, algoritmy, zpracování dat atd.

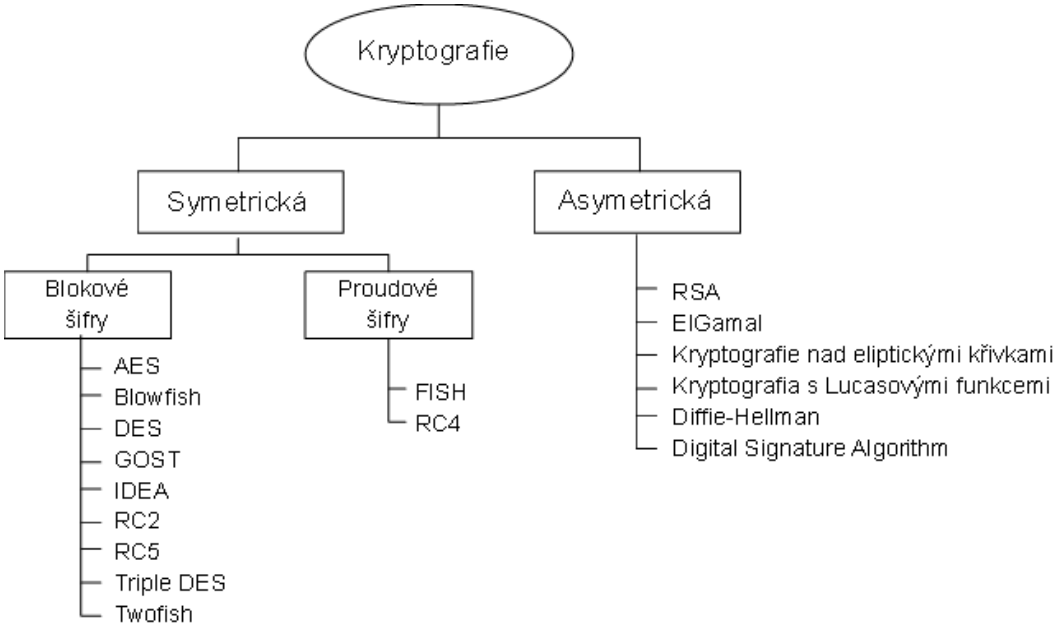
Absolutně bezpečná šifra je Vermanova šifra. Její princip spočívá ve vygenerování unikátního klíče pro každou zprávu v proměnné délce, která odpovídá délce zprávy. To vyžaduje velmi náročné požadavky na kryptografický systém, protože generovat stále unikátní klíč neomezené délky je velmi náročné. V praxi se tedy setkáváme se systémy, které nejsou absolutně bezpečné, ale snaží se splnit následující bezpečností požadavky.

Bezpečností požadavky:

- útok na algoritmus musí být složitý a cenově náročný tak, aby převýšil zabezpečený systém,
- vzít v úvahu současný a budoucí výpočetní výkon útočníka, např. vyzkoušení všech možných klíčů musí být delší než požadovaná doba utajení (útok hrubou silou),
- minimalizovat informace, které jsou potřebné k prolomení systému.

2.1.1. *Kryptografické algoritmy a protokoly*

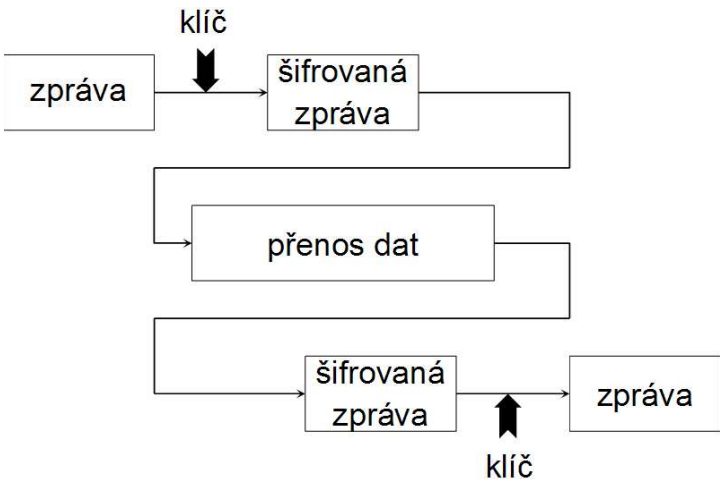
Následující obrázek (Obr. 2.2) znázorňuje základní rozdělení metod kryptografie a nejznámější algoritmy.



Obr. 2.2 Základní rozdělení

Symetrická kryptografie

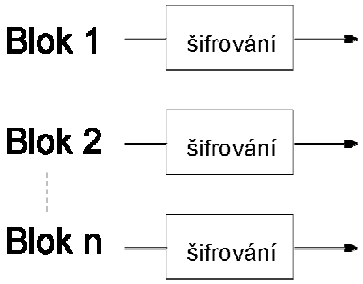
U této metody (Obr. 2.3) se zašifruje zpráva šifrovacím klíčem. Aby příjemce mohl zprávu dešifrovat, musí mu odesílatel sdělit dešifrovací klíč, který je stejný jako šifrovací. Výhoda spočívá v nízké výpočetní náročnosti algoritmu. Příjemce a odesílatel musí zajistit bezpečné předání klíče, z toho vyplývá problém s distribucí klíčů.



Obr. 2.3 Symetrická kryptografie

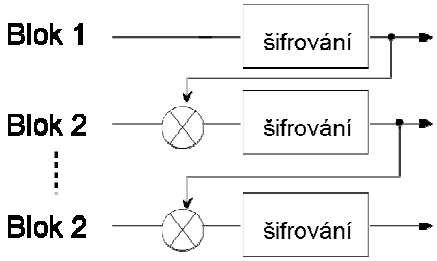
- **blokové šifry** = zpráva se rozdělí na bloky o stejné velikosti, které jsou následně zašifrovány.

  - **Režim ECB (**Obr. 2.4**) –** jednotlivé bloky jsou šifrovány samostatně stejným klíčem. Tento způsob je na ústupu, protože je jednodušší ho prolomit.



Obr. 2.4 Režim ECB

  - **Režim CBC (**Obr. 2.2**)** – následné bloky, než projdou zašifrováním, jsou zpracovány s předchozím blokem.



Obr. 2.5 Režim CBC

- **proudové šifry** = zde se šifruje bit po bitu. Nejčastěji vznikají společným zpracovaním zprávy a klíče pomocí funkce XOR.

- **CFB** – klíč tvoří pseudonáhodná posloupnost. Ta vznikne pomocí blokové šifry předchozího bloku,
- **OFB** – klíč se vytváří v závislosti na inicializačním vektoru a výstupu generátoru.

Algoritmus DES

Patří do skupiny symetrických blokových šifer. Oficiálně byl zaveden v roce 1976. Používá klíč délky 56 bitů. V dnešní době, díky vysokým výpočetním výkonům, je tento algoritmus považován z hlediska bezpečnosti za nedostatečný.

Algoritmus splňuje požadavky:

- Lavinovitost – změna jednoho bitu na vstupu, způsobí změnu výstupních bitů kolem poloviny,
- Úplnost – každý výstupní bit je funkcí všech vstupních bitů,
- Neexistence korelace – neexistuje vztah mezi původním textem a šifrovaným textem. Dále mezi šifrovaným textem a šifrovacím klíčem.

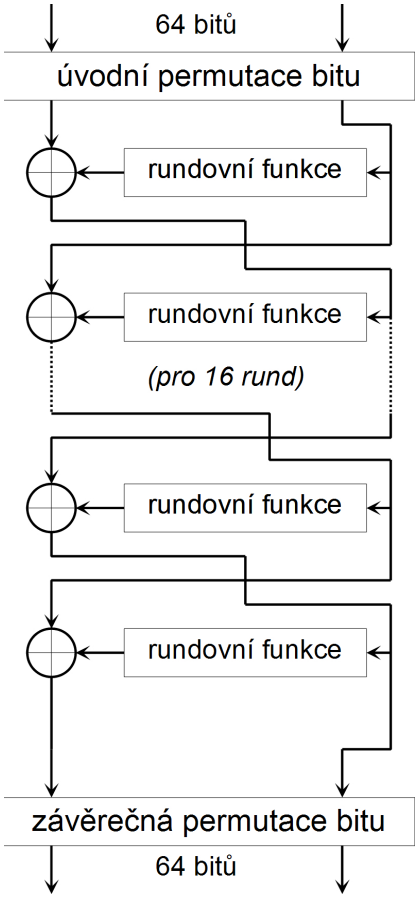
DES šifruje data rozdělená do bloků o velikosti 64 bitů. Na začátku šifrování se musí provést úvodní permutace bloku. Tím se transformuje 64 bitový blok podle Tabulka 1.

Tabulka 1: Úvodní permutace

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

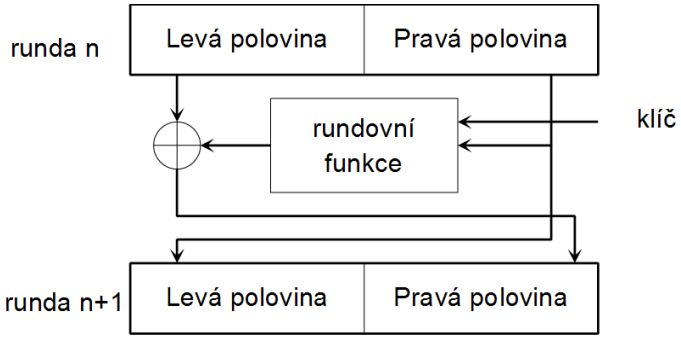
Původní osmibitové slovo slouží jako souřadnice tabulky. První 4 bity označují index řádků a zbylé 4 bity značí index sloupce.

Nyní následuje 16 kroků šifrování tzv. rund (Obr. 2.6). Z původního klíče se vytvoří 16 podklíčů a každý slouží pro jeden krok šifrování.



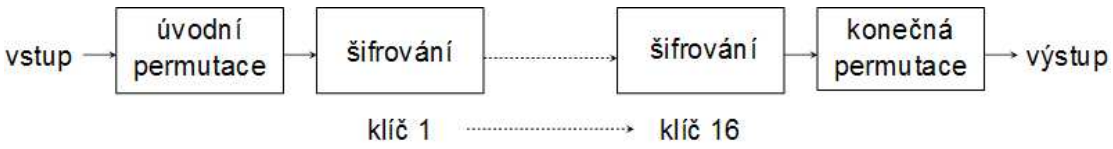
Obr. 2.6 Princip šifrování DES

Pro rundu se používá Feistelova šifra (Obr. 2.7). Na začátku je blok rozdělen na levou a pravou polovinu. V jedné rundě se zašifruje levá polovina bloku a na výstupu se prohodí levá a pravá polovina bloku.



Obr. 2.7 Feistelova šifra

Na konec se provede permutace a vznikne výsledný zašifrovaný blok podle použitého klíče.



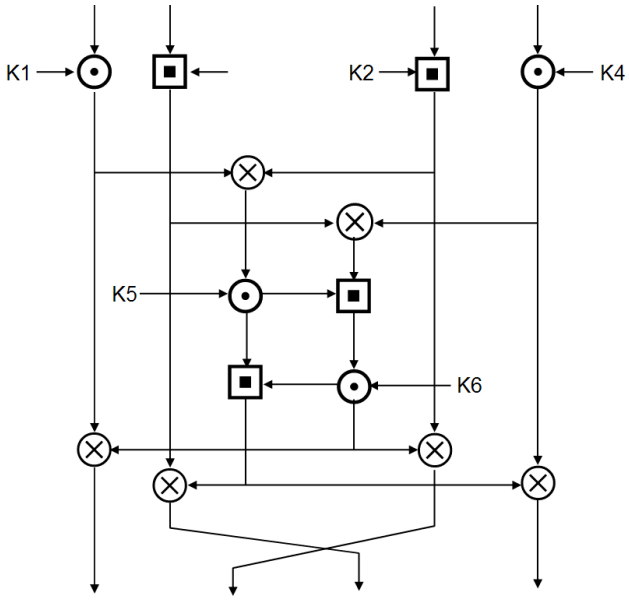
Obr. 2.8 Obecný model šifrovaní DES

Požadavek na zvýšení bezpečnosti vedl ke vzniku algoritmu TripleDES. V podstatě se jedná o trojnásobný DES. TripleDES využívá dvojnásobný (112 bitů) nebo trojnásobný klíč (168 bitů). Bezpečnost se zvýší na úkor rychlosti.

Algoritmus IDEA

Oficiálně byl zaveden v roce 1991 a měl nahradit nedostačující DES. IDEA zpracovává bloky o 64 bitech. K tomu využívá klíč o velikosti 128 bitů. Bezpečnost zaručuje střídání operací z různých skupin, které jsou algebraicky neslučitelné (Obr. 2.9).

Jedná se o operace: bitová nonekvivalence (⊗), sčítání modulo 2^{16} (⊡)a násobení modulo $2^{16}+1$ (⊙).



Obr. 2.9 Jeden krok algoritmu IDEA

Algoritmus AES

Tento algoritmus byl navržen v roce 2000. Pracuje s délkou klíče 128, 192 a 256 bitů. Klíč pracuje s bloky délky 128, 192, 256 bitů. Výhoda algoritmu AES je rychlost šifrování.

Původní šifrovací klíč se rozšíří na expandující klíč, jehož části se následně užívají pro šifrování bloku v jednotlivých rundách. Délka klíče a bloku určuje výsledný počet rund, které se pohybují v hodnotách 10 až 14. Slabiky v blocích se nahradí ekvivalenty z nelineární převodní tabulky. Následně se cyklicky posunou skupiny slabik v závislosti na velikosti šifrovaného bloku. Vytvoří se skupiny po čtyřech slabikách. Ty projdou násobením s definovanou maticí. Tím se ovlivní každá následující skupina slabik. Nakonec se provede operace XOR z části klíče, který se získá z rozšířeného klíče.

Následující Tabulka 2 srovnává výše uvedené algoritmy.

Tabulka 2: Srovnání symetrických algoritmů

	AES	IDEA	3DES
Délka klíče (bity)	128, 192, 256	128	56, 112, 168
Délka bloku (bity)	128, 192, 256	64	64
Operace	XOR, cyklický posun, maticové sčítání a násobení v GF(28)	XOR, cyklický posun, násobení mod $(2^{16})+1$, sčítání mod 2^{16}	XOR, permutace
Klíč	silný	slabý	slabý

Hashovací funkce

Převede dlouhou zprávu do krátké bitové posloupnosti o pevné délce. Funkce je jednocestná a musí se zaručit bezkoliznost. Jednocestnost znamená, že pro y = F (x) můžeme snadno určit y pomocí x, ale opačně to nelze. Nejpoužívanější funkce jsou MAC, MD5 a SHA-1. Hashovací funkce mají své důležité uplatnění v digitálním podpisu.

U digitální podpisu ;se celková zpráva nedělí na jednotlivé bloky, protože účelem není zašifrovat celou zprávu, ale pouze ověřit jestli nedošlo ke změně ve zprávě nebo jestli zpráva pochází od správného odesílatele. Z původní zprávy se udělá pouze otisk pomocí některé hashovací funkce. Tento otisk se potom zašifruje. Na přijímací straně uživatel zprávu odšifruje a zkontroluje, jestli otisky souhlasí. Obr. 2.10 ukazuje, jak se změní výsledný otisk při pouhé změně jednoho znaku.

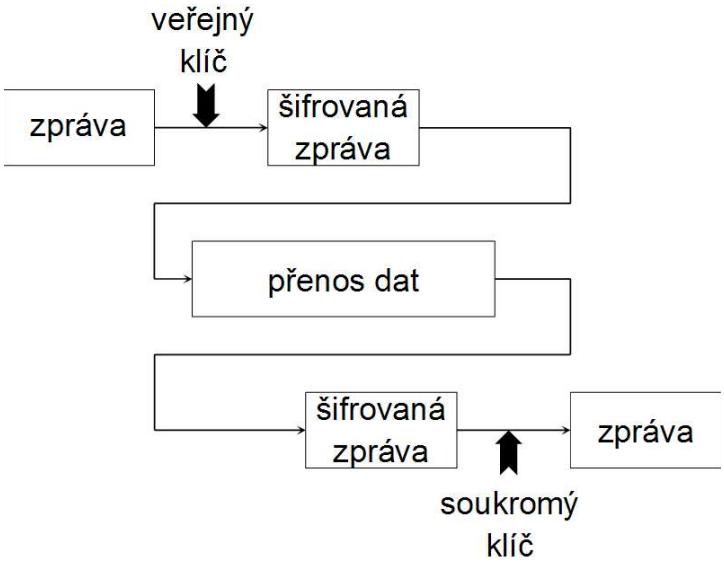
Zpráva: Ahoj, jak se daří.
MD5 Hash: 4bf02aa95ee586d01e529a9462691889

Zpráva: Ahoj, jak se daří!
MD5 Hash: e364c59547ec0741e25731fe96ab6359

Obr. 2.10 MD5 Hash

Asymetrická kryptografie

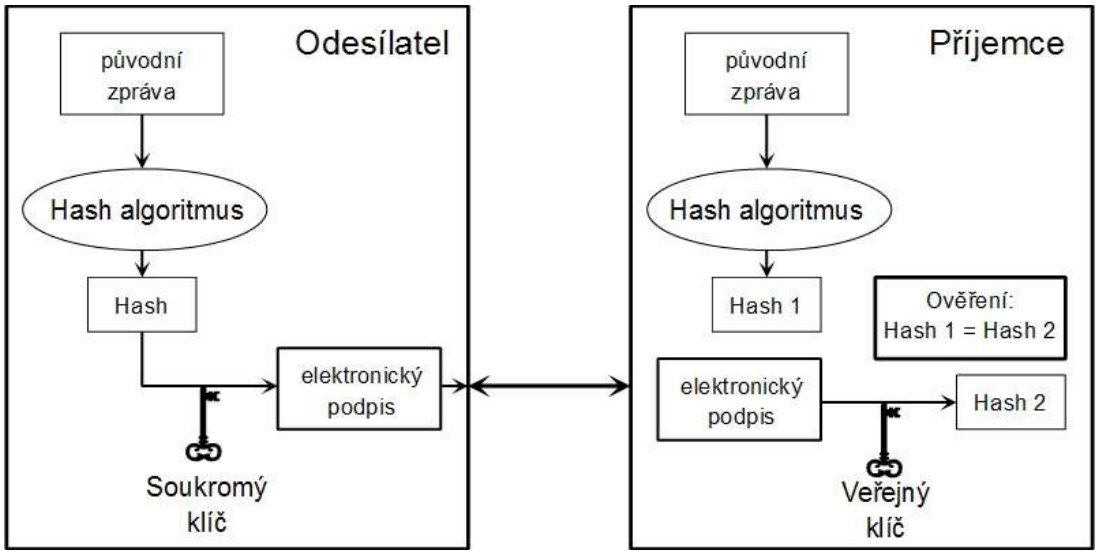
Používá dvojici klíčů veřejný a soukromý. Veřejný klíč zdělí odesílatel příjemci. Ten s jeho pomocí zprávu zašifruje a odešle. Příchozí zprávu lze rozšifrovat pomocí soukromého klíče, který má příjemce. Odpadá zde problém distribuce klíče. Pokud známe pouze veřejný klíč, nelze z něj odvodit soukromý klíč a naopak. K tomu se využívají různé matematické problémy, jejichž obtížnost řešení má exponenciální závislost. V praxi jsou to problémy faktorizace čísel, diskrétního logaritmu a sčítání bodu eliptické křivky.



Obr. 2.11 Asymetrická kryptografie

Nevýhoda asymetrické kryptografie spočívá v složitějším výpočetním algoritmu, krerý je časově náročnější. Nejvíce používaný algoritmus je RSA, který je popsán podrobněji v kapitole 5.2.

Asymetrická kryptogrofie se používá hlavně k zašifrování zprávy (Obr. 2.11) a digitálnímu podpisu u elektronických dokumentů (Obr. 2.12).



Obr. 2.12 Digitální podpis

3. Kryptografický modul a útoky na něj

Kryptografický modul je jeden ze základních kamenů celého bezpečnostního systému. Modul zajišťují základní operace pro funkci kryptografických systémů:

- jsou zde uloženy šifrovací a dešifrovací klíče,
- provádí šifrovací a dešifrovací operace,
- zajišťuje autentizaci a autorizaci.

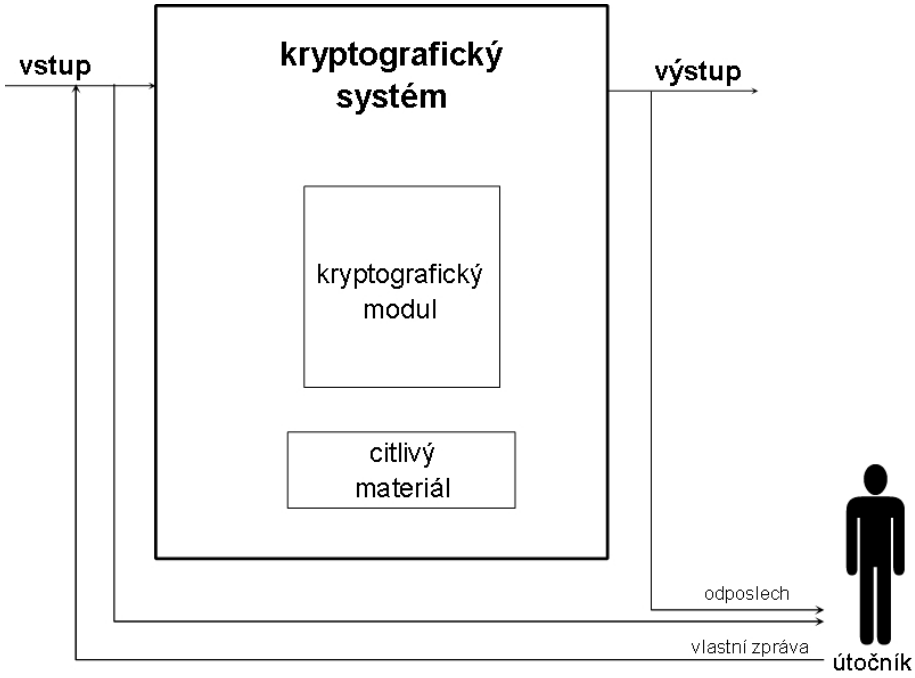
Možné realizace kryptografického modulu:

- softwarový modul,
- hardwarový modul,
- bankomat,
- server,
- elektronické čipy,
- různé karty (telefonní, televizní a atd.).

Útoky na kryptografický modul jsou známy už od jeho vzniku. Nejznámější útoky jsou tzv. útoky přímo na šifrovací algoritmus. Na celý systém se pohlíží jako na matematický model. Zabránění úspěšnosti tohoto útoku vedla k vymýšlení stále složitějších šifrovacích algoritmů.

Útoky na šifru (Obr. 3.1):

- **útok se známou šifrou** – pří znalosti šifrovaného textu, lze na základě opakovatelnosti a pravděpodobnosti odhadnout nešifrovaný text,
- **útok se známým původním textem –** pokud známe část původního textu a šifru, lze rozšifrovat celou zprávu,
- **útok s vybraným otevřeným textem –** na základě zašifrování vlastního textu, lze odhalit šifrovací algoritmus.



Obr. 3.1 Útoky na šifru

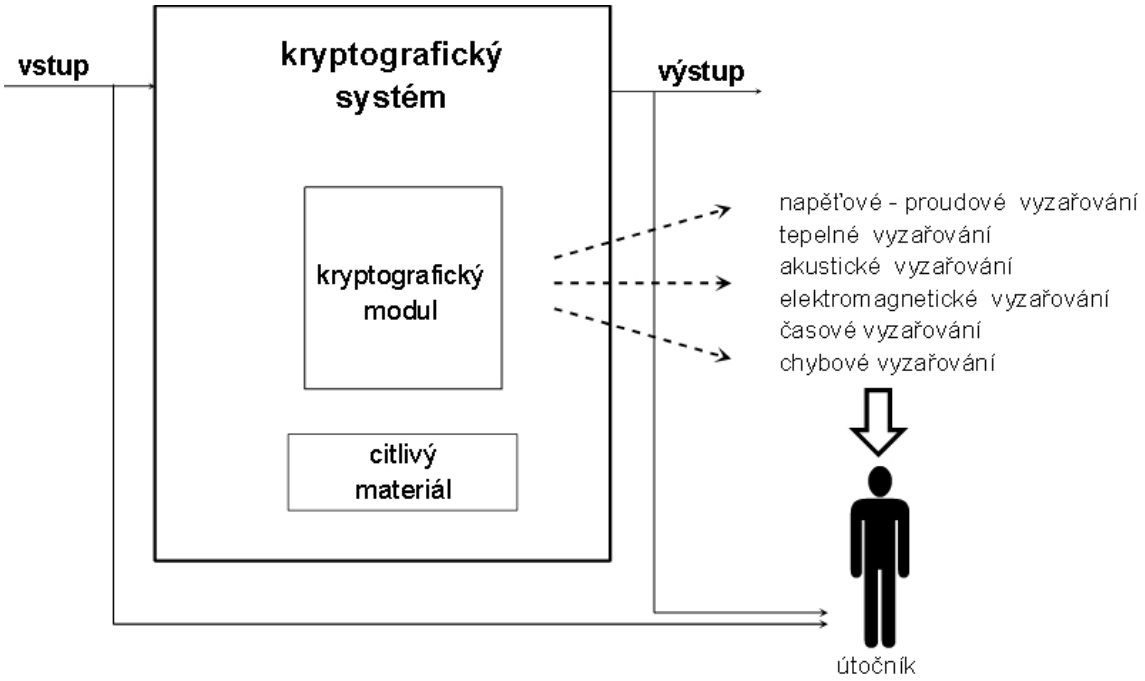
4. Postranní kanály

Druhý typ útoku na kryptografický modul byl objeven v nedávné době. Jedná se o útok pomocí postranních kanálů. Při návrhu kryptografického modulu se tyto kanály těžko odhalují. Ty totiž vznikají, až při samotné implementaci šifrovacího algoritmu. Při své činnosti kryptografický modul produkuje tepelné a jiné záření, spotřebovává výkon ze zdroje atd. Tyto informace mohou být provázány (korelovány) s průběhem operací uvnitř kryptografického modulu. Takto dochází k nežádoucímu vynesení citlivých informací mimo modul.

Z toho vyplývá definice **postranního kanálu (**Obr. 4.1**)** = postranní kanál je každá nežádoucí výměna informace mezi kryptografickým modulem a jeho okolím.

Sledované veličiny kryptografického systému:

- napětí nebo proud,
- tepelné vyzařování,
- akustické signály,
- elektromagnetické vlnění,
- časové závislosti,
- chybové chování.



Obr. 4.1 Postranní kanály

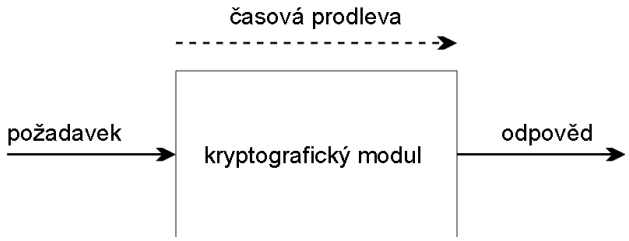
Útoky postranními kanály můžeme rozdělit na dvě části:

a) pasivní útok – zneužívá externí dostupné informace bez jakéhokoliv zásahu do kryptografického modulu. Jedná se o sledování aktivity modulu,

b) aktivní útok – vyžaduje zásah do kryptografického modulu, který pak vytvoří nebo spustí postranní kanál,
   - invazivní – proniknutí k důležité části kryptografického modulu a jeho následná úprava,
   - neinvazivní – sledování úniku informací z odolného zařízení,
   - semi-invazivní – spojení předchozích dvou typů, jedná se o dovolený přístup k modulu.

4.1. Časový postranní kanál

U tohoto postranního kanálu se útočník zaměřuje na dobu, která je potřebná k provedení různých kryptografických operací (Obr. 4.2). Tento časový okamžik může obsahovat důležitou informaci k prolomení algoritmu nebo k zjištění citlivé informace.



Obr. 4.2 Časový postranní kanál

Toto tvrzení vedlo ke vzniku Kocherovy ideji, která jako první využívá časový útok na soukromý klíč RSA, který slouží k odšifrování a podpisu zprávy. Zde se využívá doby trvání, která je potřebná k vypočtení modulární odmocniny pomocí algoritmu „square and multiply“. Ten pracuje s jednotlivými bity soukromého klíče. Z algoritmu (Obr. 4.3) vyplývá, že pokud bit je roven nule, tak výpočet trvá kratší dobu. Pokud je roven jedné, výpočet trvá delší dobu.

$y = (h^b \bmod n)$

– – – – –

$b = b_é, b_1, ...b_{b-1}$

$l = h$

for *i = 1 to b - 1*

$l = l^2 \bmod n$

if $(b == 1)$ then $l = l * h \bmod n$

return l

Obr. 4.3 Algoritmus Square and multiply

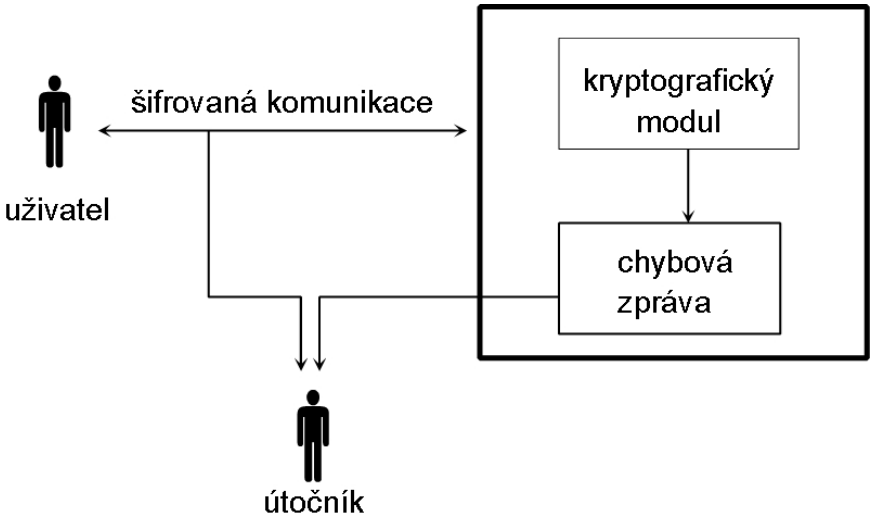
Časový postranní kanál lze nalézt u algoritmů, které používají modulární mocnění:

- DSA,
- Diffieho-Hellmanova protokolu,
- Algoritmus AES,
- DES,
- IDEA,
- RC5 a další.

4.2. Chybový postranní kanál

Chybový postranní kanál (Obr. 4.4) patří k nejvíce nebezpečným postranním kanálům. Zde se pracuje s informacemi, které vzniknou při chybné práci kryptografického modulu. Z počátku se braly tyto chybné zprávy nebo výsledky jako bezcenné. Opak je ale pravdou.

- **Přechodný chybový kanál** – chybový kanál vznikne pouze na požadovaný čas.
- **Stálý chybový kanál** – vytvoření chybového kanálu vede k trvalým následkům. Následující výpočty budou s chybným výsledkem.



Obr. 4.4 Chybový postranní kanál

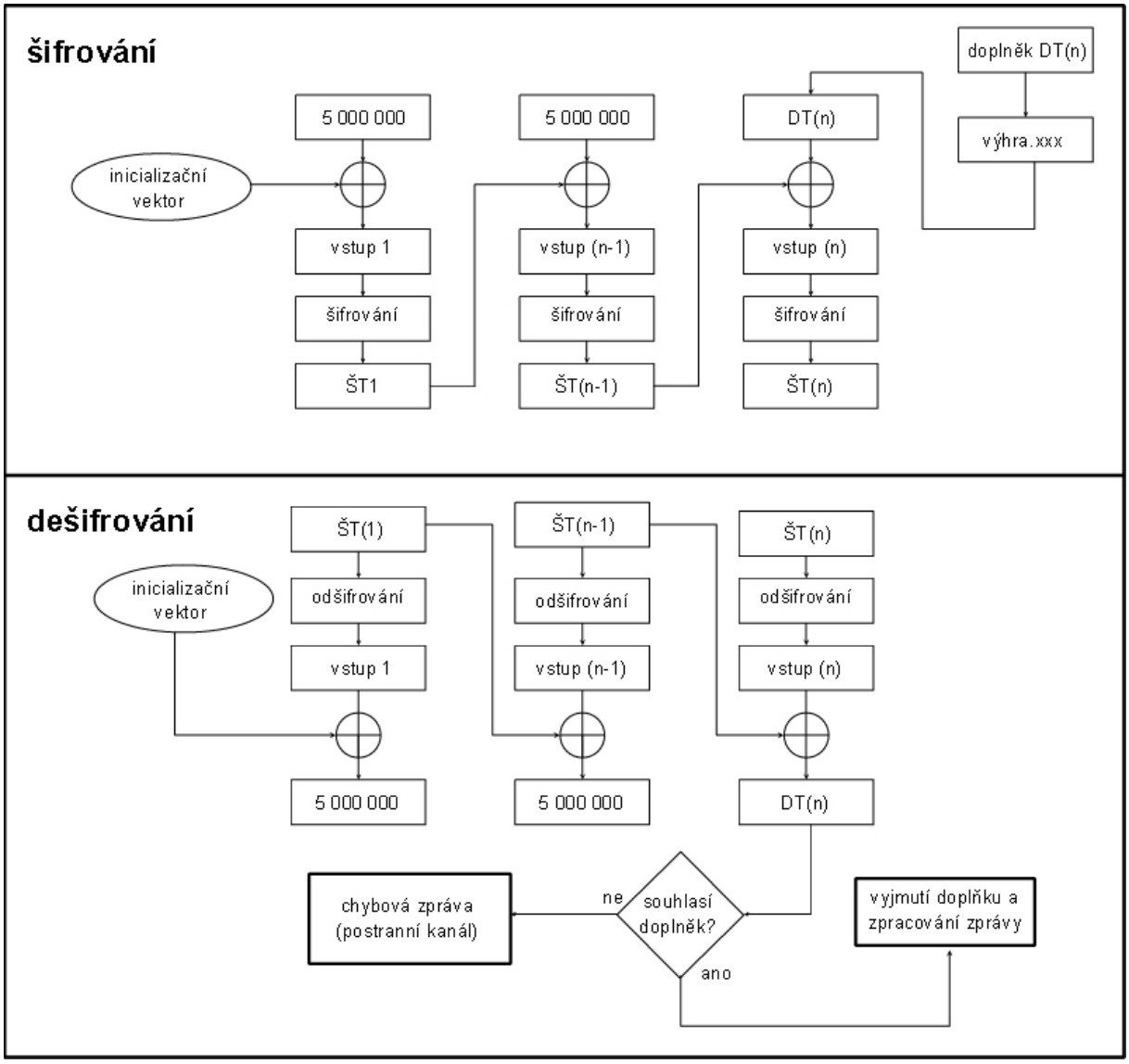
Zabránění nebo ztížení vzniku chybového postranního kanálu závisí na použitém formátu zprávy a zvoleném algoritmu. Chybové zprávy můžou vzniknout při normálním provozu nebo útočníkovým zásahem.

Příklady chybových postranních kanálů

Chybový postranní kanál v módu CBC (Obr. 4.5**)**

U módu CBC se neúplný blok dat doplňuje dalšími bajty. Pokud využijeme k šifrování blokovou šifru, která pracuje s 8 bajty a poslední blok obsahuje pouze 5 bajtů, musí se doplnit o 3 bajty. Pro správnou činnost se musí doplnit také otevřený text, který je už zarovnaný na osmibitové části. Na začátku odšifrovaní se zjistí, kolik bajtů bylo přidáno z posledního bajtu. Pak následuje porovnání doplňků. Pokud se shodují, doplněk se odstraní a zpráva je připravena k zpracování. Neshoda doplňků značí chybu na přenosovém kanálů a přijímací strana informuje odesílatele chybovým hlášením. Těchto hlášení využije útočník k dešifrování zprávy.

Útočník odposlechne nejprve text zašifrované zprávy. Pak posíláním svých zpráv s přídavkem části původní zprávy, pozoruje chybová hlášení. Na jejich analýze dosáhne požadovaného cíle.



Obr. 4.5 Chybový postranní kanál v módu CBC

Chybový postranní kanál v RSA – PKCS #1 ver 1.5

U algoritmu RSA je potřeba vstupní zprávu nejprve zformátovat a upravit tak, aby číslo nebylo větší než modul n. U formátování pomocí PKCS #1 ver 1.5 dokázal Bleichenbacher v roce 1998 existenci postranního kanálu, který umožňuje odhalení původní zprávy. Pokud máme vstupní zprávu Z o velikosti 48 bajtů a modul u RSA má velikost 1 024 bitů, potřebujeme doplnit zprávu do 128 bajtového bloku EB (Obr. 4.6). Pokud na takto pracující dešifrovací přístroj dorazí nesprávný šifrovaný text, odpoví chybovou hláškou. Tato hláška se stává kořistí útočníka. Doba útoku závisí dále na délce modulu, délce bloku D a integritních kontrolách.

EB = 00 || 02 || PS || 00 || D

Legenda: PS = skládá se ze 77 náhodných nenulových bajtů
02 a 00 = bajty
02 = indikuje blok typu 02 (blok pro šifrování)

!!Poslední nenulový bajt zaručuje podmínku EB < n!!

Obr. 4.6 Struktura bloku EB

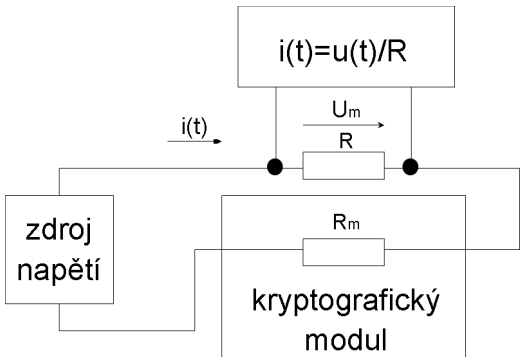
Chybový postranní kanál v RSA – KEM ver 1.5

Aby se zabránilo vzniku postranních kanálů u algoritmu RSA, což má za následek většinou doplňování nebo úprava zprávy, vznikl nový způsob formátovaní zprávy podle schématu RSA – KEM. Vstupní text neprochází žádným formátováním a rovnou se náhodně generuje v délce modulu. Takto vytvořený text vytvoří základ symetrického klíče. Z toho vyplývá, že odpadá zpětná kontrola při odšifrování, a tak i vznik postranního kanálu. Avšak bylo dokázáno [10], že i zde vzniká chybový kanál, který umožní získat otevřený text, ale i soukromý klíč.

4.3. Napěťově-proudový postranní kanál

U tohoto uvedeného postranní kanálu se využívá typická fyzikální vlastnost a to reálná spotřeba. Tato informace může posloužit k odhalení používané operace v algoritmu nebo informaci o kódu. Pokud je kryptografický modul napájen konstantním napětím, můžeme pomocí jednoduchého zapojení (Obr. 4.7) a známého vztahu (4.1) vypočítat jeho spotřebu.

$$p(t) = U \times i(t) \tag{4.1}$$



Obr. 4.7 Měření spotřeby kryptografického modulu

Rozdělení útoku:

- **invazivní** – tento typ útoku vyžaduje zásah do kryptografické modulu, např. odhalení povrchu čipu,

- **neinvazivní** – využívají se informace, které vzniknou při provozu.

Používané metody:

1. **SPA** – Jednoduchá výkonová analýza,
2. **DPA** – Diferenciální výkonová analýza,
3. **HO-DPA –** Diferenciální výkonová analýza vyššího řádu.

Jednoduchá výkonová analýza – SPA

Útočníkovým cílem je sledování spotřeby proudu v systému. Na základě této zjištěné informace lze získat představu o instrukcích. Tato analýza dokáže odhalit rozdíly mezi násobením a umocňováním v RSA. U DESu zase operace permutace a posun.

Diferenciální výkonová analýza – DPA

Analýza DPA spočívá ve dvou krocích. Za prvé shromažduje data a v druhé části je analyzuje. Data se získávají vzorkováním příkonu při činnosti kryptografického modulu. Pro kvalitní vyhodnocení informací musí útočník získat dostatek kryptografických operací, které pracují s klíčem.

Aplikace DPA na algoritmus DES:

- zaznamená se spotřeba proudu z cyklů 1000 operací, které probíhají u DESu,
- útočník musí znát pole zašifrovaných textů T[0..999],
- jedna množina vzorků obsahuje 100 000 bodů,
- data se přepíšou do dvojrozměrného pole S[0..999][0..99999], první index určuje operaci a druhý konkrétní vzorek,
- pomocí funkce D, která je závislá na klíči $D(K_i, T)$, může najít 6 bitů DES klíče, které jsou vstupem jedné z operací DESu,
- vypočte se diferenciální průměrová stopa L[0..63][0..99999] pomocí vztahu:
  - $$L[i][j] = \sum \left[\left(D\left(i, T[k] - \frac{1}{2} \right) (S[k])[j] \right) \right], \qquad (4.2)$$
- následně se určí jedna správná hodnota z L_i(index v poli T). Vypočte se korelace jednotlivých řad v poli L se zaznamenanými příkony. Nejvyšší korelace určí správnou hodnotu,
- postupnými výpočty lze vyhodnotit 48 z 56 bitů klíče.

Diferenciální výkonová analýza vyššího řádu – HO-DPA

Tato analýza se nachází ve fázi testovaní. Princip je založen na vyhodnocování kryptografických podoperací. Požadované informace se získávají z různých zdrojů.

4.4. Elektromagnetický postranní kanál

Tento postranní kanál je znám krátkou dobu a vychází z napěťově-proudového postranního kanálu. To vyplývá z Maxwellových rovnic. Rozdíl spočívá v pozorované veličině. Útočník se snaží změřit elektromagnetické vyzařování kryptografického modulu.

V roce 2001 bylo zveřejněno zjištění, že SSL akcelerátor na sběrnici PCI vyzařoval elektromagnetické pole do vzdálenosti 12,192 m. Ve vzdálenosti 4,572 m bylo možno rozlišit základní kroky výpočtu RSA.

Používané metody:

- **SEMA** – jednoduchá elektromagnetická analýza,
- **DEMA** – diferenciální výkonová analýza.

4.5. Eliminace postranních kanálu

V současné době neexistuje univerzální zabezpečení, které by odstranilo vznik postranních kanálů. V praxi se setkáváme se softwarovou nebo hardwarovou ochranou, která většinou představuje částečné zabránění vzniku postranních kanálu.

Softwarové řešení je založeno na vhodné úpravě algoritmu. Nejdůležitější je vyvarovat se vzniku chyb. Ty lze eliminovat pomocí kontrolních algoritmů např. kontrolního součtu nebo se snažit vyvarovat v algoritmu částí, kde tyto chyby vznikají. U časového postranního kanálu se snažíme zapracovat na době, kterou algoritmus běží. Ta by měla být při různých vstupních parametrech co nejvíce konstantní. Toho můžeme dosáhnout vložením nepotřebného kódu, který pouze upravuje dobu trvání celého algoritmu. Nevýhodou softwarových řešení většinou bývá náročnější řešení kryptografického modulu.

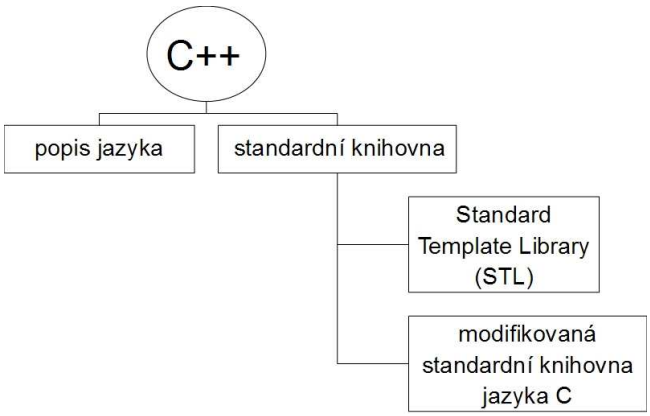
Hardwarová opatření se používají u výkonového a elektromagnetického postranního kanálu. Základním prvkem je zamezení proniknutí a následná manipulace s kryptografickým modulem. K tomu se používají speciální lepidla a kryty, které při poškození provedou znehodnocení čipu. Hodnota elektromagnetického vyzařovaní závisí na provedeném stínění kryptografického modulu. V některých případech se zavádí softwarové nebo hardwarové moduly, které mění unikající měřené fyzikální veličiny. Její hodnota se úmyslně mění, nebo se k ní úmyslně namoduluje šum. Existují také moduly, které neustále sledují množství unikajících chyb, které vyhodnotí a následně rozhodnou, jestli nebyl kryptografický modul napaden.

5. Experimentální část

Tato kapitola se zabývá algoritmem RSA, u kterého je ukázán časový postranní kanál. Pomocí tohoto kanálu lze odhalit soukromý klíč, který slouží k odšifrování zprávy. K realizaci algoritmu RSA a demonstraci útoku časovým postranním kanálem jsem využil programovací jazyk C++ v programu Dev-C++.

5.1. Programovací jazyk C++ a program Dev-C++

Jazyk C++ (Obr. 5.1) se řadí mezi objektově orientovaný programovací jazyk a patří mezi nejrozšířenější programovací jazyky. Vznikl rozšířením jazyka C počátkem osmdesátých let a přidává vysokoúrovňové vlastnosti vyšších jazyků. Jazyk C++ spojuje nízkoúrovňové, tak i vysokoúrovňové vlastnosti.



Obr. 5.1 Struktura jazyka C++ v roce 1998

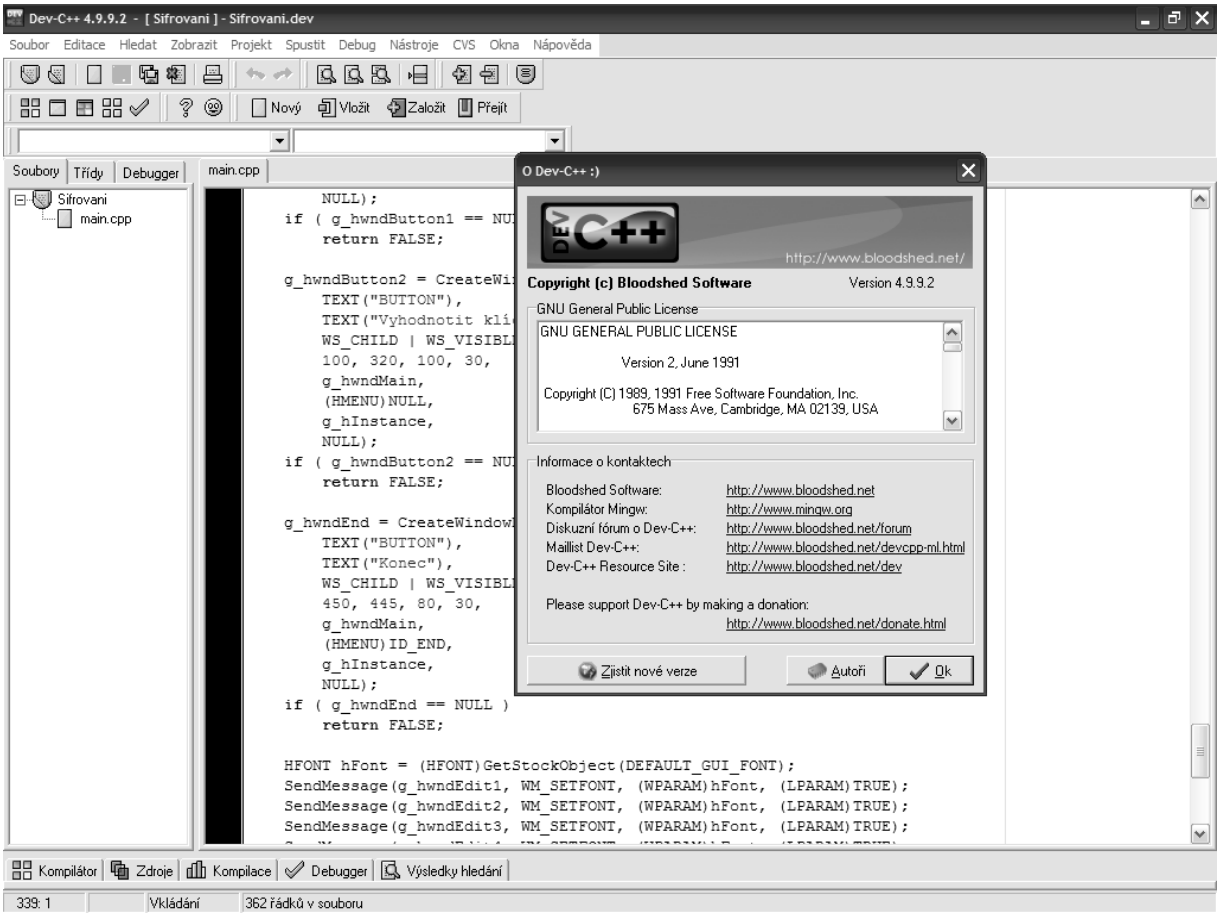
- nízkoúrovňové vlastnosti – používají pro realizaci algoritmů zápis instrukcí procesoru,
- vysokoúrovňové vlastnosti – používají pro realizaci algoritmů postupy bližší lidskému chápání.

Programovací jazyk C lze nazvat jako procedurální jazyk. Data programu vytváří, zpracovává a používá algoritmus. Jazyk C++ zase jako objektově orientovaný jazyk. Zde se vytváří tzv. datové třídy, které popisují souhrn vlastností a jak s nimi pracovat. Výhoda tohoto řešení je dědičnost. Nově vytvořené třídy můžou využívat už předešle vytvořené třídy. Z toho vyplývá, že pro různé proměnné nemusíme psát nový kód.

Jazyk C++ umožňuje vytvoření více stejně pojmenovaných funkcí. Kompilátor na základě počtu a typu parametrů určí jejich správné vykonání.

Program Dev-C++

Pro svoji realizaci jsem zvolil integrované vývojové prostředí (IDE) Dev-C++ (Obr. 5.2). IDE je sada nástrojů, které zjednodušují vývoj aplikace. Program je zdarma a určen pro operační systémy Microsoft Windows.



Obr. 5.2 Náhled programu Dev-C++

Základní charakteristiky IDE:

1. obarvování zdrojového kódu,
2. sbalování a rozbalování zdrojového kódu,
3. rozdělení na projekty, editace a vyhledávání souborů,
4. ladící nástroje, vyhledávání chyb,
5. nápověda zdrojového kódu,
6. seznam použitých objektů,
7. kompilátor a interpreter.

Vlastnosti, které rozšiřuje Dev-C++:

1. rozšíření DevPack,
2. správu zdrojových kódu mezi více uživateli,

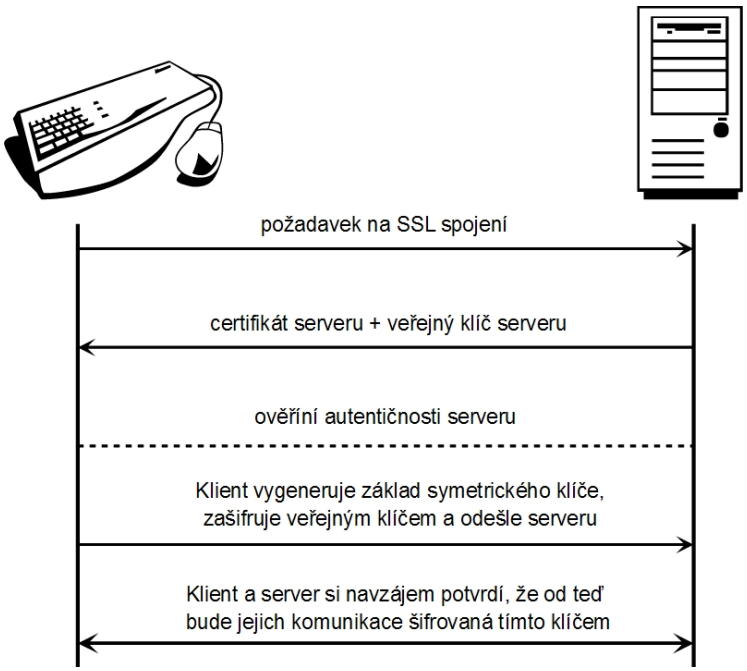
3. seznam TODO,
4. umisťování záložek.

Program Dev-C++ začal vyvíjet Colin Laplace. Poté se ho ujala společnost Bloodshed Software. Vývojové prostředí je vytvořeno v Delphi 6. Na svůj projekt jsem využil poslední vydanou verzi 5.0 beta 9.2 (4.9.9.2), která vyšla v únoru 2005. Program lze rozšířit baličkami vývojových nástrojů a knihoven. Dev-C++ využívá ke své funkci externí programy (např. kompilátor MinGW).

Program lze stáhnout z domovských stránek http://www.bloodshed.net/dev/devcpp.html

5.2. Algoritmus RSA

Tento algoritmus získal název podle počátečních písmen příjmení jeho tvůrců, kteří byli Rivest, Shamir a Adleman v roce 1977. Algoritmus RSA patří mezi asymetrické šifry a je velmi často používán v zabezpečovacích systémech. Jeho nasazení zaručí dobré zabezpečení. Nevýhoda spočívá v jeho nízké rychlosti šifrování. S algoritmem se setkáváme v mobilních telefonech pro bezpečnou autentizaci, bankomatech, digitálním podpisu, protokolu SSL (Obr. 5.3) atd.



Obr. 5.3 Ustavení SSL spojení

Hlavní myšlenka bezpečnosti algoritmu je založena na faktorizaci velkých čísel. To znamená, že není v reálném čase možné rozložit dané číslo na součin mocnin prvočísel.

Používané algebraické operace v RSA:

1. **Modulární aritmetika:**

$$\begin{gathered} a,b,n \in N \\ (a \bmod n + b \bmod n) \bmod n = (a+b) \bmod n \\ (a \bmod n \times b \bmod n) \bmod n = (a \times b) \bmod n \\ \Downarrow \\ a + b = b + a (\bmod n) \end{gathered} \tag{5.1}$$

2. **Malá Fermatova věta (věta o primitivním kořeni):** pokud *p* je prvočíslo a *a* libovolné celé číslo, pak platí:

$$a^p = a (\bmod p) \tag{5.2}$$

   Jestliže *p* nedělí *a*, pak platí:

$$a^{p-1} = 1 (\bmod p) \tag{5.3}$$

3. **Eulerova funkce:** je definována na množině kladných čísel:
   a. Pokud $n \succ 1$ je $\varphi(n)$ rovno počtu kladných celých čísel menších než *n* a nesoudělných s *n*.
   b. Pokud n je prvočíslo pak $\varphi(n) = n - 1.$

4. **Eulerova věta (obecná Fermatova věta):** je-li *a* nesoudělné s *n* pak platí, že

$$a^{\varphi(n)} = 1 (\bmod n). \tag{5.4}$$

5.2.1. Vytvoření klíče

Kvalitně vytvořený klíč tvoří základ bezpečnosti algoritmu RSA. Generování klíčů je výpočetně velmi náročné, protože se pracuje s velkými prvočísly. Hlavní důraz se zaměřuje na délku klíče. Obecně lze říci, když se z dvojnásobí délka klíče, tak doba na generování klíče se zvýší přibližně 16 krát, doba dešifrování 8 krát a šifrování 4 krát. Když tedy budeme klíč používat několik let, musíme vzít v úvahu budoucí výpočetní výkon počítačů.

V dnešní době se používají algoritmy RSA s číslem *n* délky 1024 ($n = 2^{1024}$) a 2048 bitů ($n = 2^{2048}$). Délka 4096 bitů zaručuje do budoucna dostatečnou bezpečnost. Klíč o délce 512 bitů (Obr. 5.4) byl prolomen už v roce 1997 a dnes se už nepovažuje za bezpečný. Dnes osobní počítače dokážou prolomit klíč o délce 256 bitů za pár hodin. Hrozbou pro dnešní klíče do budoucna představují tzv. kvantové počítače, které by prováděli faktorizaci v polynomiálním čase.

0A 66 79 1D C6 98 81 68 DE 7A B7 74 19 BB 7F B0
C0 01 C6 27 10 27 00 75 14 29 42 E1 9A 8D 8C 51
D0 53 B3 E3 78 2A 1D E5 DC 5A F4 EB E9 94 68 17
01 14 A1 DF E6 7C DC 9A 9A F5 5D 65 56 20 BB AB

Obr. 5.4 Klíč délky 512 bitů

Postup vytvoření klíče:

1. Hledáme dvě velká prvočísla *p* a *q*
2. Vypočítáme *n* a *f*

$$n = p \times q \tag{5.5}$$

$$f = (p-1) \times (q-1) \tag{5.6}$$

.

3. Zvolíme veřejný šifrovací klíč *e*, které je nesoudělné s číslem *f* a leží v intervalu:

$$1 \prec e \prec f$$

4. Vypočítáme tajný šifrovací klíč *a*

$$a = e^{-1} \bmod f \tag{5.7}$$

5. Výsledkem je veřejný *(n, e)* a soukromý klíč *(n, a)*

Pokud se vhodně zvolí prvočísla *p* a *q*, nelze určit z veřejného klíče soukromý a naopak. Jestliže prvočísla budou ležet blízko sebe a nebudou dostatečně velké, lze získat soukromý klíč z veřejného následujícím způsobem:

1. Pomocí faktorizace hodnoty *n* se snažíme získat *p* a *q*
2. Vypočítáme *f*

$$f = (p-1) \times (q-1) \tag{5.8}$$

3. Zjistíme zbývající část soukromého klíče

$$a \times e \equiv 1 (\bmod f(n)) \tag{5.9}$$

Můj použitý algoritmus na generování klíče:

```
int isPrime(int k)
{
 int b=1;
 int i;
 for(i=2;i<k;i++)
  if(k%i==0) { b=0; break; }
 return b;
}
void generate()
{
```

```
        int s;
        for(s=0;s<20;s++)
        {
          eb[s]=0;
          ab[s]=0;
        }
        do{
           while(1)
              {
              p=2 + rand()%200;
              q=2 + rand()%200;
             if(p!=q && isPrime(p)==1 && isPrime(q)==1) break;
              }

           n = p * q;
           n1 = (p-1) * (q-1);
           int xx=n1+1;
           for(e=2;e<=xx/2;e++)
               {
               if(xx%e==0) break;
               }
           for (a = 0, s = 0; s != 1; a++)
           s = (s + e) % n1;
         }
      while(p==1 && q==1 && e==1 && a==1 && e==a && q==e && q==a && p==e &&
p==a && x==n);
         int e_pom=e,a_pom=a;
         kb=0;
         rb=0;
            while(e_pom>=1)
         {
                    if(e_pom%2==1) eb[kb]=1; else eb[kb]=0;
                    e_pom/=2; kb++;
         }

         while(a_pom>=1)
         {
                    if(a_pom%2==1) ab[rb]=1; else ab[rb]=0;
                    a_pom/=2; rb++;
         }
         return;
      }
```

5.2.2. Šifrování zprávy

Původní zpráva *X* ve tvaru čísla se šifruje pomocí veřejného klíče *(n, e)* dle vzorce 5. 10, kde výsledkem je opět číslo *Y*. Číslo *X* musí splňovat podmínku $X < n$.

$$Y = X^e \bmod N \tag{5.10}$$

Umocňování takto velkých čísel představuje velmi náročnou operaci. Proto se využívají metody, které tuto náročnou operaci zjednoduší.

1. **Algoritmus Square-and-Multiply**
   a. Exponent *e* se převede na binární číslo

$$e = e_{(k-1)}e_{(k-2)}....e_{(l)}e_{(0)}$$

kde $e_{(k-1)}$ je nejvýznamnější nenulový bit.

b. Postupuje se podle algoritmu:

```
set y = x
for bit j = k - 2 downto 0
begin
 y = y * y mod n   /* square */
 if e(j) == 1 then
 y = y * x mod n  /* multiply */
end
return y
```

Z algoritmu vyplývá, že při zvolení správného exponentu, lze zjednodušit výpočetní náročnost na provedení vztahu 5.10. Např. oblíbené číslo 65537 má binární tvar 0x10001.

Tento algoritmus jsem použil ve svém programu na šifrování zprávy i dešifrování zprávy.

Můj použitý algoritmus na šifrování zprávy:

```
int y = x;
     int j = 0;
     for(j = 1;j <= k-1; j++)
     {
         y = (y*y) % n;
         if(e[j] == 1)
           y = (y*x) % n;
     }
```

2. Chinese Remainder Theorem (CRT) (Čínská věta o zbytcích)

Pomocí tohoto postupu lze čtyřikrát rychleji vypočítat výsledek než podle vztahu 5.10. I když pracuje s více kroky než Algoritmus Square-and-Multiply, je celkově méně nákladná, protože pracuje s menšími exponenty. Zkrácení času šifrování zaručí rozklad na výpočet s menšími moduly:

Princip:

a. rozklad na výpočet s menšími moduly:

$$Y_p \equiv X^e \pmod{p}$$
$$Y_q \equiv X^e \pmod{q}$$

b. hodnotu *e* můžeme snížit pomocí Malé Fermatovy věty vyjádřením

$$e = e_q + k(q-1)$$
$$\Downarrow$$

$$Y_p \equiv X^{e_p} \left(\mathrm{mod}\, p\right)$$
$$Y_q \equiv X^{e_q} \left(\mathrm{mod}\, q\right)$$

c. z toho dostaneme

$$Y = \left[X_p M_p q + X_q M_q p\right] \mathrm{mod}\, n$$

$$\text{kde} \quad \begin{aligned} M_p &= q^{-1} \,\mathrm{mod}\, p \\ M_q &= p^{-1} \,\mathrm{mod}\, q \end{aligned}$$

Jak už bylo řečeno, algoritmus RSA pracuje s čísly. V praxi se většinou šifrují i znaky a celá slova. Znaky v počítači jsou sice vyjádřeny především čísly v osmibitovém rozsahu (0–255). Číslo *n* bývá většinou větší než hodnota 255, a tak se z důvodu bezpečnosti a efektivnosti se každý znak nešifruje zvlášť. Nejběžnějším řešením je rozdělení zprávy do bloků konstantní velikosti. Takto vzniklé bloky se šifrují samostatně. Na přijímací straně se dešifrují a zpětně překódují do původního stavu. Délka bloku se volí v závislosti na hodnotě *n*. Před samotným šifrováním data projdou vhodným slovníkovým komprimačním algoritmem.

5.2.3. Dešifrování zprávy

Dešifrování zprávy má shodný postup stejný jako šifrování. Rozdíl spočívá v použití veřejného klíče *(n, a)*.

Můj použitý algoritmus na šifrování zprávy:

```
int y = x;
int j = 0;

for(j = kb-2; j >= 0; j--)
    {
         y = (y*y) % n;
         if(eb[j] == 1)
         y = (y*x) % n;
    }
```

5.2.4. Využití časového postranního kanálu

Časový postranní kanál vytváří algoritmus Square-and-Multiply v části dešifrování. Operace trvají velmi krátkou dobu, kterou je obtížné změřit. K tomu se využil časovač procesoru, který takto krátkou dobu umožňuje změřit.

Časovač slouží k přesnému měření času v počítači. Může být proveden softwarově jako součást počítačového programu nebo hardwarově. Hardwarové časovače používá především operačním systém k vykonávání multitaskingu. Časovač může být implementován v procesoru, čipsetu nebo ve zvláštním obvodu. Nejstarší používaný časovač v počítačích nese označení Programmable Interval Timer (PIT). Firmy Intel a Microsoft vyvinuly časovač

s označením High Precision Event Timer (HPET), který umožňuje vyšší a přesnější rozlišení času. S tímto časovačem pracují novější operační systémy Windows Vista, Windows Server 2008, Mac OS X, Linux 2.6 a FreeBSD.

Měření v mém programu probíhá tak, že změřím čas na začátku algoritmu. Hodnota se zapíše do proměnné start. Po ukončení jednoho cyklu for se zaznamená konečný čas a zapíše se do proměnné stop. Následně se vypočítá rozdíl hodnot stop a start a uloží do proměnné data[jj+1].

Můj použitý algoritmus na měření jednotlivých cyklů for:

```
for(j = rb-2;  j >= 0; j--)
{
jj=jj+1;
QueryPerformanceFrequency(&ticksPerSecond
QueryPerformanceCounter(&tick);
start = (tick.QuadPart % ticksPerSecond.QuadPart)/1;

     for(g=0;g < 300000; g++){}
     z = z*z % n;
     if(ab[j] == 1)
           {
              z = z*y % n;
              c=jj+1;
               for(g=0;g < 300000; g++){}
           }

    QueryPerformanceFrequency(&ticksPerSecond);
    QueryPerformanceCounter(&tick);
    stop = (tick.QuadPart % ticksPerSecond.QuadPart)/1;

    data[jj+1]= (stop-start);
   }
```

Při tomto řešení jsem musel vyřešit dva klíčové problémy. Při testování programu na různých počítačích vycházely řádově různé hodnoty, v důsledku vstupního kmitočtu na časovači, který se liší typem procesoru. Potom hodnoty na některých počítačích ve výsledném grafu přetekly. Toto jsem ošetřil podělením všech hodnot příslušným číslem.

Můj použitý algoritmus na ošetření výsledných hodnot:

```
w=1;
if (data[c]>130)
{
w=data[c]/130;
for(j = 1;  j <= 20; j++)
data[j+1] = data[j+1]/w;
}
```

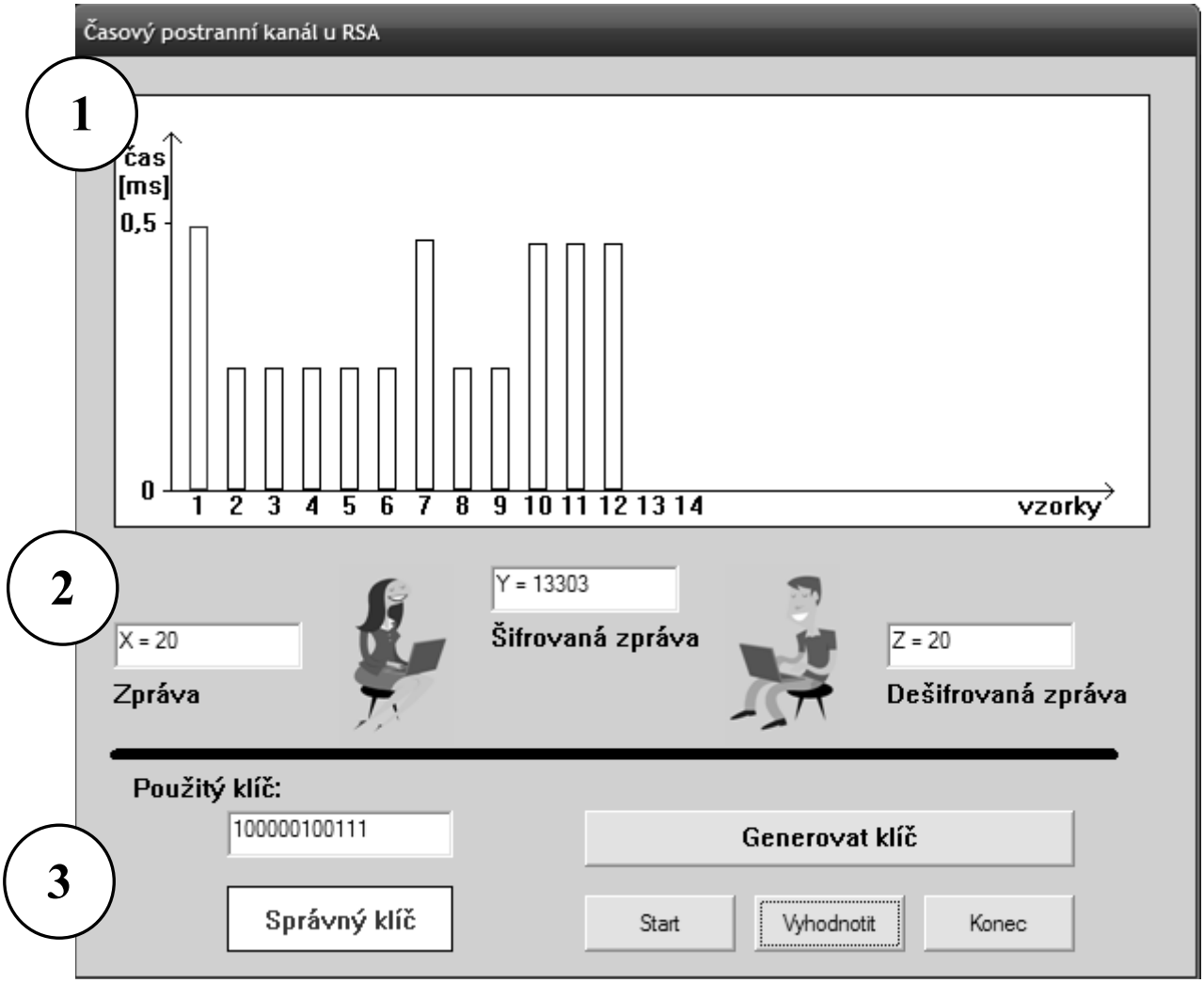
Druhý problém skrýval multitasking. Program je totiž přerušován multitaskingem v závislosti na výkonu a aktuálním vytížení systému, což podstatný vliv na výsledky měření. Výsledky se občas zobrazují s malou chybou a někdy může nastat situace, že změřený čas

některého cyklu neodpovídá správné hodnotě klíče. Aby některé hodnoty nepřetekly, jsou všechny hodnoty v proměnné data[j] otestovány níže uvedeným algoritmem a případně upraveny.

Můj použitý algoritmus na ošetření chybných hodnot:

```
for (j = 1;  j <= 20; j++)
        {
          if (data [j+1]<0) data[j+1]=0;
          if (data [j+1]>200) data[j+1]=200;
        }
```

Algoritmus RSA a následné zobrazení hodnot zprostředkovává grafické uživatelské prostředí (Obr. 5.5), které se skládá ze třech hlavních částí.



Obr. 5.5 Náhled programu na zpracování RSA

V sekci 1 se po stisknutí tlačítka „Start“, vykreslí grafická závislost, které znázorňuje trvání jednotlivých smyček při dešifrování zprávy příslušným vygenerovaným klíčem. Změna klíče se provede stisknutím tlačítka „Generovat klíč“.

Sekce 2 ukazuje komunikaci dvou uživatelů, kteří pro šifrování a dešifrování zprávy využívají algoritmus RSA. Dále je naznačena hodnota původní, šifrované a dešifrované zprávy.

Sekce 3 obsahuje tlačítka pro obsluhu programu. Dále zde probíhá kontrola odečtených bitů soukromého klíče z grafu s originálním klíčem. Pokud je klíč správně odečten z grafu, tak po stisknutí tlačítka „Vyhodnotit“ je uživatel informován hláškou.

6. Závěr

Tato práce se snaží shrnout a objasnit útoky pomocí postranních kanálů. V kapitole 4 jsou přehledně uvedeny typy postranních kanálů. V současné době je nejvíce diskutovaný časový, chybový a výkonový postranní kanál. Pomocí těchto kanálů se podařilo úspěšně vést útok na kryptografický modul. Útok výkonovým postranním kanálem je podrobněji popsán v [5] a [9], chybovým postranním kanálem v [15] a časovým postranním kanál se zabývá moje bakalářská práce. V kapitole 4.5 jsou představeny některá opatření, která je mohou eliminovat. Bohužel zatím neexistuje žádné univerzální opatření, a proto je tato problematika velmi důležitá.

Počet zmíněných postranních kanálů není konečný. Každý nově navržený kryptografický systém může vytvořit další. Některé kanály ještě nebyly odhaleny nebo důkladně prozkoumány. Ať už se jedná o akustický, optický [2] a frekvenční postranní kanál [3].

V části Experimentální část jsem naprogramoval v jazyku C++ algoritmus RSA. U něj jsem pomocí časového postranního kanálu, který vzniká v části dešifrování, odhalil soukromý klíč. Demonstraci tohoto útoku jsem zpracoval do přehledného grafického programu. V budoucnosti lze program rozšířit o další prvky. Program by mohl šifrovat zprávu pomocí symetrických algoritmů. Další rozšíření lze aplikovat na vkládání zprávy, která prochází šifrováním. Ta by mohla být měněna uživatelem, včetně vkládání textu.

V závěru práce jsem navrhl laboratorní úlohu pro studenty, kteří s využitím mého programu se seznámí s problematikou časového postranního kanálu.

7. Laboratorní úloha

Časový postranní kanál u algoritmu RSA

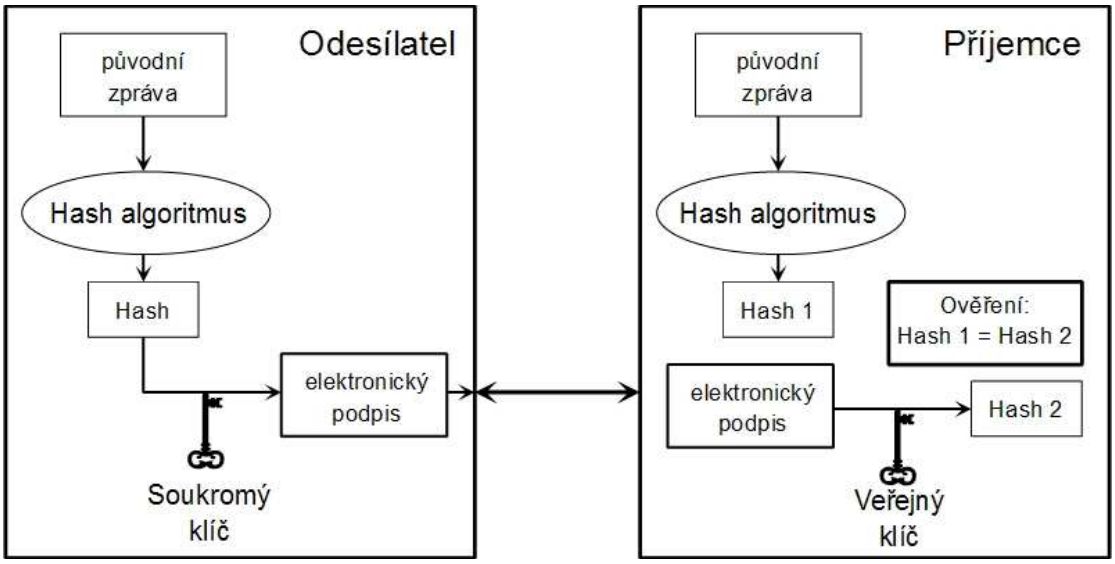
Zadání

1. Prostudujte teoreticky algoritmus RSA a problematiku postranních kanálů.
2. Pomocí programu RSA.exe si nastudované znalosti ověřte.
3. Doplňte zdrojový kód tak, aby probíhalo dešifrování zprávy. Pak si zapište postup na odečítání času.

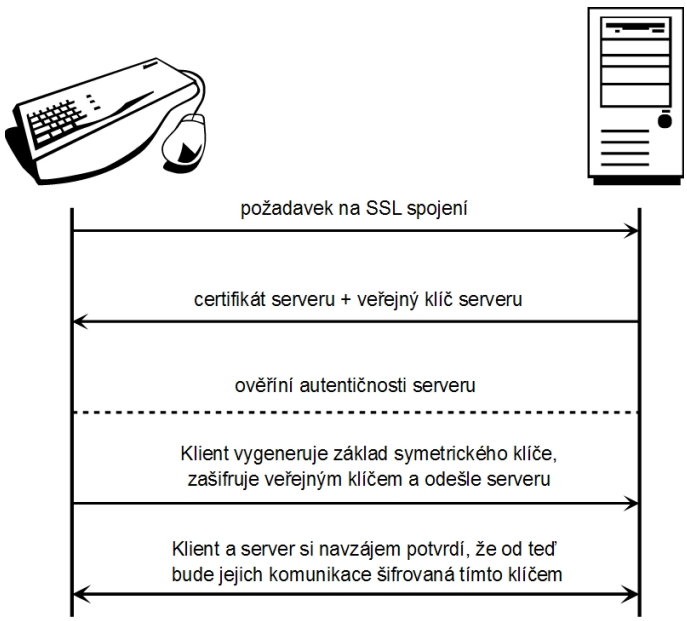
Teoretický rozbor

Algoritmus RSA

Tento algoritmus získal název podle počátečních písmen příjmení jeho tvůrců, kteří byli Rivest, Shamir a Adleman v roce 1977. Algoritmus RSA patří mezi asymetrické šifry a je velmi často používán v zabezpečovacích systémech. Jeho nasazení zaručí dobré zabezpečení. Nevýhoda spočívá v jeho nízké rychlosti. S algoritmem se setkáváme v mobilních telefonech, bankomatech, digitálním podpisu (Obrázek 1.1), protokolu SSL (Obrázek 1.2) atd.



Obrázek 1.1 Digitální podpis



Obrázek 1.2 Sestavení spojení SSL

Hlavní myšlenka bezpečnosti algoritmu je založena na faktorizaci velkých čísel. To znamená, že není v reálném čase možné rozložit dané číslo na součin mocnin prvočísel.

Používané algebraické operace v RSA:

5. **Modulární aritmetika:**

$$\begin{gathered} a,b,n \in N \\ (a \bmod n + b \bmod n) \bmod n = (a+b) \bmod n \\ (a \bmod n \times b \bmod n) \bmod n = (a \times b) \bmod n \\ \Downarrow \\ a + b = b + a (\bmod n) \end{gathered} \tag{1.1}$$

6. **Malá Fermatova věta (věta o primitivním kořeni):** pokud *p* je prvočíslo a *a* libovolné celé číslo, pak platí:

$$a^p = a (\bmod p) \tag{1.2}$$

Jestliže *p* nedělí *a*, pak platí:

$$a^{p-1} = 1 (\bmod p) \tag{1.3}$$

7. **Eulerova funkce:** je definována na množině kladných čísel:
   a. Pokud $n \succ 1$ je $\varphi(n)$ rovno počtu kladných celých čísel menších než *n* a nesoudělných s *n*.

b. Pokud n je prvočíslo pak $\varphi(n) = n - 1$.

8. **Eulerova věta (obecná Fermatova věta):** je-li *a* nesoudělné s *n* pak platí, že

$$a^{\varphi(n)} = 1 (\mathrm{mod}\, n) \tag{1.4}$$

Kvalitně vytvořený klíč tvoří základ bezpečnosti algoritmu RSA. Generování klíčů je výpočetně velmi náročné, protože se pracuje s velkými prvočísly. Hlavní důraz se zaměřuje na délku klíče. Obecně lze říci, když se z dvojnásobí délka klíče, tak doba na generování klíče se zvýší přibližně 16 krát, doba dešifrování 8 krát a šifrování 4 krát. Jestliže budeme používat klíč několik let, musíme vzít v úvahu budoucí výpočetní výkon počítačů.

V dnešní době se používají algoritmy RSA s číslem *n* délky 1024 ($n = 2^{1024}$) a 2048 bitů ($n = 2^{2048}$). Délka 4096 bitů zaručuje do budoucna dostatečnou bezpečnost. Klíč o délce 512 bitů (Obrázek 1.3) byl prolomen už v roce 1997 a dnes se už nepovažuje za bezpečný. Dnešní osobní počítače dokážou prolomit klíč o délce 256 bitů za pár hodin. Hrozbou pro dnešní klíče do budoucna představují tzv. kvantové počítače, které by prováděly faktorizaci v polynomiálním čase.

```
0A 66 79 1D C6 98 81 68 DE 7A B7 74 19 BB 7F B0
C0 01 C6 27 10 27 00 75 14 29 42 E1 9A 8D 8C 51
D0 53 B3 E3 78 2A 1D E5 DC 5A F4 EB E9 94 68 17
01 14 A1 DF E6 7C DC 9A 9A F5 5D 65 56 20  BB AB
```

Obrázek 1.3 Klíč délky 512 bitů

Postup vytvoření klíče:

6. Hledáme dvě velká prvočísla *p* a *q*
7. Vypočítáme *n* a *f*

$$n = p \times q \tag{1.5}$$

$$f = (p - 1) \times (q - 1) \tag{1.6}$$

.

8. Zvolíme veřejný šifrovací klíč *e*, které je nesoudělné s číslem *f* a leží v intervalu:

$$1 \prec e \prec f$$

9. Vypočítáme tajný šifrovací klíč *a*

$$a = e^{-1} \bmod f \tag{1.7}$$

10. Výsledkem je veřejný *(e, n)* a soukromý klíč *(a, n)*

Pokud se vhodně zvolí prvočísla *p* a *q*, nelze určit z veřejného klíče soukromý a naopak. Prvočísla nesmí ležet blízko sebe a nebudou dostatečně velké, lze získat soukromý klíč z veřejného následujícím způsobem:

1. Pomocí faktorizace hodnoty n se snažíme získat *p* a *q*
2. Vypočítáme *f*

$$f = (p-1)\times(q-1) \tag{1.8}$$

3. Zjistíme zbývající část soukromého klíče

$$a \times e \equiv 1 (\mathrm{mod}\, f(n)) \tag{1.9}$$

Původní zpráva *X* ve tvaru čísla se šifruje pomocí veřejného klíče *(e, n)* dle vzorce 1. 10, kde výsledkem je opět číslo *Y*. Číslo *X* musí splňovat podmínku $X < n$.

$$Y = X^e \bmod N \tag{1.10}$$

Umocňování takto velkých čísel představuje velmi náročnou operaci. Proto se využívají metody, které tuto náročnou operaci zjednoduší.

3. Algoritmus Square-and-Multiply

a. Exponent *e* se převede na binární číslo

$$e = e_{(k-1)}e_{(k-2)}....e_{(1)}e_{(0)}$$

kde $e_{(k-1)}$ je nejvýznamnější nenulový bit.

b. Postupuje se podle algoritmu:

```
set y = x
for bit j = k - 2 downto 0
begin
 y = y * y mod n   /* square */
 if e(j) == 1 then
 y = y * x mod n  /* multiply */
end
return y
```

Z algoritmu vyplyvá, že při zvolení správného exponentu, lze zjednodušit výpočetní náročnost na provedení vztahu 1.10. Např. oblíbené prvočíslo 65537 má binární tvar 0x10001.

4. Chinese Remainder Theorem (CRT) (Čínská věta o zbytcích)

Pomocí tohoto postupu lze čtyřikrát rychleji vypočítat výsledek než podle vztahu 1.10. I když pracuje s více kroky než Algoritmus Square-and-Multiply, je celkově méně nákladná, protože pracuje s menšími exponenty. Zkrácení času šifrování zaručí rozklad na výpočet s menšími moduly:

Princip:

a. rozklad na výpočet s menšími moduly:

$$Y_p \equiv X^e \pmod p$$
$$Y_q \equiv X^e \pmod q$$

b. hodnotu *e* můžeme snížit pomocí Malé Fermatovy věty vyjádřením

$$e = e_q + k(q-1)$$
$$\Downarrow$$
$$Y_p \equiv X^{e_p} \pmod p$$
$$Y_q \equiv X^{e_q} \pmod q$$

c. z toho dostaneme

$$Y = \left[X_p M_p q + X_q M_q p\right] \bmod n$$

$$\text{kde} \quad \begin{matrix} M_p = q^{-1} \bmod p \\ M_q = p^{-1} \bmod q \end{matrix}$$

Jak už bylo řečeno, algoritmus RSA pracuje s čísly. V praxi se většinou šifrují i znaky a celá slova. Znaky v počítači jsou sice vyjádřeny především čísly v osmibitovém rozsahu (0–255). Číslo *n* bývá většinou větší než hodnota 255, a tak z důvodu bezpečnosti a efektivnosti se každý znak nešifruje zvlášť. Nejběžnějším řešením je rozdělení zprávy do bloků konstantní velikosti. Takto vzniklé bloky se šifrují samostatně. Na příjmací straně se dešifrují a zpětně překódují do původního stavu. Délka bloku se volí v závislosti na hodnotě *n*. Před samotným šifrováním data projdou vhodným slovníkovým komprimačním algoritmem.

U digitálního podpisu se celková zpráva nedělí na jednotlivé bloky, protože účelem není zašifrovat celou zprávu, ale pouze ověřit jestli nedošlo ke změně ve zprávě nebo jestli zpráva pochází od správného odesílatele. Z původní zprávy se udělá pouze otisk pomocí některé Hashovací funkce. Tento otisk se potom zašifruje. Na příjmací straně uživatel zprávu odšifruje a zkontroluje, jestli otisky souhlasí. Obrázek 1.4 ukazuje, jak se změní výsledný otisk při pouhé změně jednoho znaku.

Zpráva: Ahoj, jak se daří.
MD5 Hash: 4bf02aa95ee586d01e529a9462691889

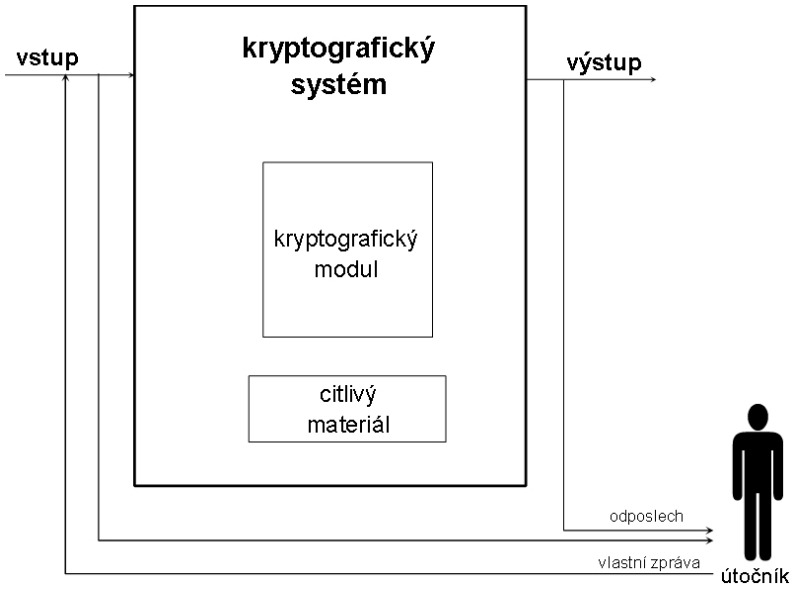
Zpráva: Ahoj, jak se daří!
MD5 Hash: e364c59547ec0741e25731fe96ab6359

Obrázek 1.4 MD5 Hash

Dešifrování zprávy má shodný postup stejný jako šifrování. Rozdíl spočívá v použití veřejného klíče *(a, n)*.

Postranní kanál

Útoky na kryptografický modul jsou známy už od jeho vzniku. Nejznámější útoky jsou tzv. útoky přímo na šifrovací algoritmus (Obrázek 1.5). Na celý systém se pohlíží jako na matematický model. Zabránění úspěšnosti tohoto útoku vedla k vymyšlení složitějších šifrovacích algoritmů.



Obrázek 1.5 Útoky na šifru

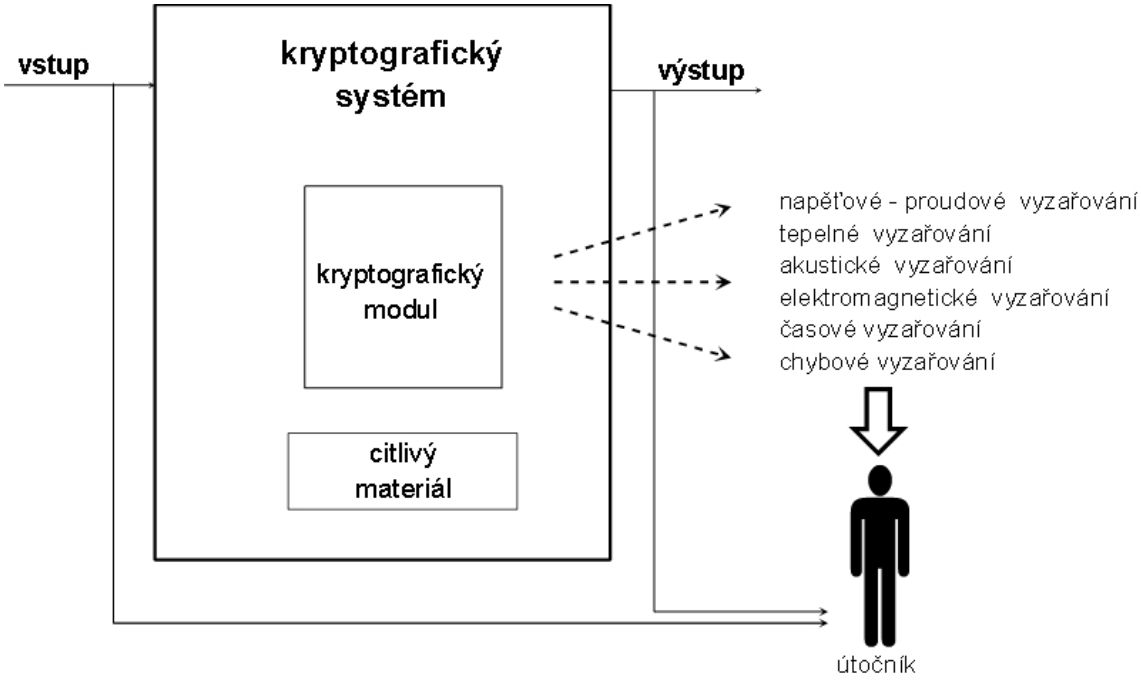
Druhý typ útoku na kryptografický modul byl objeven v nedávné době. Jedná se o útok pomocí postranních kanálů. Při návrhu kryptografického modulu se tyto kanály těžko odhalují. Ty totiž vznikají, až při samotné implementaci šifrovacího algoritmu. Při své činnosti kryptografický modul produkuje tepelné a jiné záření, spotřebovává výkon ze zdroje atd. Tyto informace mohou být provázány s průběhem operací uvnitř kryptografického modulu. Takto dochází k nežádoucímu vynesení citlivých informací mimo modul.

Z toho vyplývá definice **postranního kanálu (Obrázek 1.6)** = je každá nežádoucí výměna informace mezi kryptografickým modulem a jeho okolím.

Sledované veličiny kryptografického systému:

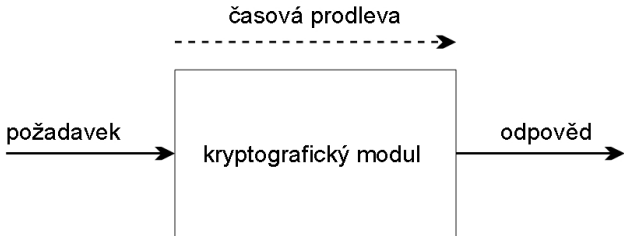
a) napětí nebo proud,
b) tepelné,

c) akustické,
d) elektromagnetické,
e) časové,
f) chybové.



Obrázek 1.6 Postranní kanál

U **časového postranního kanálu** se útočník zaměřuje na dobu, která je potřebná k provedení různých kryptografických operací (Obrázek 1.6). Tento časový okamžik může obsahovat důležitou informaci k prolomení algoritmu nebo k zjištění citlivé informace.



Obrázek 1.6 Časový postranní kanál

Toto tvrzení vedlo ke vzniku Kocherovy ideji, která jako první využívá časový útok na soukromý klíč RSA, který slouží k odšifrování a podpisu zprávy. Zde se využívá doby trvání, která je potřebná k vypočtení modulární odmocniny pomocí algoritmu „square and multiply“. Ten pracuje s jednotlivými bity soukromého klíče. Z algoritmu (Obrázek 1.7) vyplyvá, že pokud bit je roven nule, tak výpočet trvá kratší dobu. Pokud je roven jedné, výpočet trvá delší dobu. Pokud útočník má další informace o vstupním požadavku a algoritmu, tak prolomení je otázkou času.

$y = (h^b \bmod n)$

– – – – –

$b = b_é, b_1, ...b_{b-1}$

$l = h$

for *i = 1 to b - 1*

$l = l^2 \bmod n$

if $(b == 1)$ then $l = l * h \bmod n$

return l

Obrázek 1.7 Algoritmus „Square and Multiply“.

Postup měření

1. Na ploše zapněte program na simulaci časového postranního kanálu u algoritmu RSA. Ikona se jmenuje RSA.exe. Zamyslete se nad tím, jak celý program funguje a vyzkoušejte si, jak pomocí časového postranního kanálu lze zjistit důležitou část soukromého klíče.
2. Otevřete neúplný zdrojový kód programu, který jste si spustili v předchozím bodě. Kód otevřete pomocí programu Dev-C++. Ve zdrojovém kódu úmyslně chybí část algoritmu RSA na dešifrování zprávy. Tuto část dopište do zdrojového kódu. Inspirujte se z teoretického rozboru a části šifrování ve zdrojovém kódu. Vámi upravený zdrojový kód zkompilujte a spustě tlačítkem F9. Ve zdrojovém kódu si najděte a zapište funkci na zaznamenávání času.
3. Z nastudovaných poznatků vymyslete nebo najdete na internetu další konkrétní typy postranních kanálů.

Použité přístroje a pomůcky

Programu Dev-C++

8. Literatura

Seznam použité literatury

[1] PIPER, Fred, MURPHY, Sean. Kryptografie : Průvodce pro každého. Pavel Houser; Pavel Mondschein; Odborná spolupráce Vlastimil Klíma. 2006. vyd. [s.l.] : Dokořán , 2006. 157 s. ISBN 80-7363-074-5

[2] KUN, M.: Optical Time-Domain Eavesdropping Risks of CRT Displays. Proc of the 2002 Symposium on Security and Privacy, Washington, USA, pp.3-18, 2002, ISBN 0-7695-1543-6.

[3] TIU, C., C.: A New Frequency-Based Side Channel Attack for Embedded Systems. Master degree thesis, Deparment of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario, Canada, 2005.

[4] ALFRED J. MENEYES, Paul C. van Oorschot, Scott A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1996

[5] KOCHER, P., JAFFE, J., JUN, B.: Introduction to Differential Power Analysis and Related Attacks, San Francisco, 1998. [.pdf dokument]. Dostupný z WWW:

[6] QUISQUATER, Prof. Jean-Jacques, RIZK, Math. Side channel attacks : State-of-the-art. CRYPTREC. 2002, no. 2002, s. 47. Dostupný z WWW: http://www.ipa.go.jp/security/enc/CRYPTREC/fy15/doc/1047_Side_Channel_report.pdf .

[7] ING. LUDMILA KUNDEROVÁ. Bezpečnost IS/IT [online]. 04 Nov 2008 [cit. 2008-11-20]. Dostupný z WWW: https://akela.mendelu.cz/~lidak/bis/8kryp.htm .

[8] Problémy odolných kryptografických zařízení [online]. Doc. Ing. Daniel Cvrček, Ph.D, 1 [cit. 2008-12-02]. Dostupný z WWW: http://www.fit.vutbr.cz/~cvrcek/cards/karty.html.cs

[9] ING. DANĚČEK, Petr , ING. BŘEZINA, Milan. Útok výkonovým postranním kanálem na hardwarový kryptografický modul. Elektrorevue [online]. 2006, č.

2006/31 [cit. 2008-11-02]. Dostupný z WWW: http://www.elektrorevue.cz/clanky/06031/index.html#Prakticka .

[10] KLÍMA, Vlastimil, ROSA , Tomáš. Na kanálu se pracuje aneb O revolučním objevu v kryptoanalýze. In OpenWeekend 2003. [s.l.] : [s.n.], 2003. s. 41-50.

[11] DOC. ING. BURDA, CSC., Karel. BEZPEČNOST INFORMAČNÍCH SYSTÉMŮ. Brno : [s.n.], 2005. 104 s

[12] VLASTIMIL, Klíma. Symetrická kryptografie. [s.l.] : [s.n.], 2007. 17 s

[13] MANGER, James . A Chosen Ciphertext Attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as Standardized in PKCS #1 v2.0. [s.l.] : [s.n.], 2001. 8 s. Dostupný z WWW: http://www.iacr.org/archive/crypto2001/21390229.pdf . ISBN 3-540-42456-3

[14] KLÍMA, Vlastimil , ROSA, Tomáš. RSA v novém světle : Bezpečnost RSA. Chip. 2001, Prosinec, s. 4.

[15] DANIEL BLEICHENBACHER, Daniel . Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1. [s.l.] : [s.n.], 1998. 12 s. Dostupný z WWW: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.19.8543 . ISBN :3-540-64892-5.

[16] NOUZÁK, Josef. Postranní kanály mikroprocesorů. [s.l.], 2007. 48 s. Vedoucí bakalářské práce Ing. Jan Schmidt PhD.

[17] KLÍMA, Vlastimil. Bezpečné použití RSA : MODERNÍ KRYPTOGRAFICKÉ METODY. Chip. 1.1.2000, č. 11, s. 3

[18] J.-F. DHEM, J.-F. , et al. A practical implementation of the timing attack. UCL Crypto Group Technical Report Series. 1998, no. 1, s. 19.

[19] DOC. ING. KAREL BURDA, CSC., Aplikovaná kryptografie. Přednáška 3, 5 2009

Přílohy

A CD

CD obsahuje:

- Všechny zdrojové kódy programu
- Zdrojový kód pro laboratorní úlohu
- Program RSA.exe
- Elektronickou verzi dokumenty