Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra počítačových systémů

Diplomová práce

Algebraická kryptoanalýza Baby Rijndael

Bc. Lenka Vábková

Vedoucí práce: prof. Ing. Róbert Lórencz, CSc.

19. února 2016

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 19. února 2016 .....

Odkaz na tuto práci

Vábková, Lenka. *Algebraická kryptoanalýza Baby Rijndael.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

V této práci se zabýváme algebraickou kryptoanalýzou šifry Baby Rijndael. Baby Rijndael je zjednodušená verze nejpoužívanější symetrické blokové šifry AES. Provádíme útoky se znalostí otevřeného a šifrového textu. Postupně jsme útočili na jednu, dvě a čtyři rundy této šifry. Každou rundu si můžeme vyjádřit pomocí soustavy rovnic do stupně maximálně 2. Při útoku se snažíme tuto soustavu vyřešit a k tomu používáme algoritmy XL, XSL, T' a další heuristické postupy. Pro jednu rundu se nám podařilo šifru prolomit. Pro dvě rundy jsme museli znát čtvrtinu bitů klíče a pro čtyři rundy jsme museli znát skoro polovinu bitů klíče.

Klíčová slova Baby Rijndael, Algebraická kryptoanalýza, AES, XL algoritmus, XSL algoritmus

Abstract

In this Theses we deal with algebraic cryptanalysis of Baby Rijndael cipher. Baby Rijndael is a simplified version of the most used symetric block cipher AES. We do the attacks with knowledge of the open text and the cipher text. Gradually, we attack one, two and four rounds of this cipher. We can express each round as a system of equations with maximum degree of terms equal to

two. During attacks we try to solve this system using algorithms XL, XSL, T' and some heuristics. For one round we managed to break the cipher. For two rounds we must know one quarter of bits of the key and for four rounds we must know almost half of bits of the key.

Keywords Baby Rijndael, Algebraic cryptanalysis, AES, XL algorithm, XSL algorithm

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V této práci se budeme zabývat algebraickou kryptoanalýzou šifry Baby Rijndael, což je zjednodušená verze šifry AES. Šifra AES je dnes nejpoužívanější bloková symetrická šifra - tj. šifruje vstupní text po blocích pevně dané délky a pro šifrování a dešifrování se používá stejný tajný klíč (viz. [2]).

Šifra AES (zkratka pro Advanced Encryption Standard) byla vybrána v soutěži instituce NIST (National Institute of Standards and Technology) a standardizována v roce 2001 ([1]), v reakci na slabiny dříve používané šifry DES. Autory vítězného návrhu jsou Vincent Rijmen a Joan Daemen, odtud jméno jejich vítězného návrhu Rijndael.

Během soutěže a v následujících letech tak byla (a stále je) upřena pozornost kryptologů na hledání slabin této šifry. Vzhledem k tomu, že šifru AES lze jednoduše popsat algebraickými operacemi, je jedním z možných způsobů kryptoanalýzy tzv. algebraická kryptoanalýza. Algebraická kryptoanalýza převádí šifry na soustavy rovnic a ty se poté snaží pomocí různých nástrojů vyřešit (více v [3]).

V roce 2002 publikovali Courtois a Pieprzyk ([4]) obecný způsob, jak prolamovat obdobně konstruované šifry pomocí popisu systémem rovnic a jeho následnou linearizací. Konkrétní útok a úspěšná aplikace jejich postupu na AES ale nebyly publikovány.

V následujích pracích (např. [5]) se na Courtoise a Pieprzyka další kryptologové pokoušeli navázat a prozkoumat konkrétní aplikace jejich postupu na kryptoanalýzu AES. Žádné výsledky ale neukázaly praktickou použitelnost těchto postupů a naopak vše nasvědčuje tomu, že AES je odolná i na tyto typy útoků. Neexistence takového útoku ale není důkaz o bezpečnosti této šifry, a proto má smysl dále prozkoumávat možnosti algebraické kryptoanalýzy AES pro lepší pochopení fungování šifry a pro hledání možných slabin.

Za tímto účelem byla v roce 2005 sestrojena Dr. Cliffordem Bergmanem na univerzitě Iowa State University šifra Baby Rijndael. Tato šifra má obdobnou strukturu jako AES, pouze pracuje s menšími vstupy, obdobně zmenšenými operacemi a menším počtem těchto operací.

Cílem této práce je seznámit se se šifrou Baby Rijndael a provést útok pomocí algebraické kryptoanalýzy. Budeme tak hledat slabiny Baby Rijndaelu pomocí převedení šifry, případně jejích částí, do soustav rovnic a hledání řešení těchto soustav.

V první části práce se budeme zabývat strukturou této šifry a jejím převedením na soustavu rovnic, přitom se budeme snažit vytvářet rovnice s co nejmenším stupněm termů. Poté budeme zkoumat algoritmy na řešení soustav nelineárních rovnic pomocí linearizace. Další možný směr výzkumu, kterým by se šlo ubírat, je použití Gröbnerových bází na řešení těchto soustav nelineárních rovnic (viz. např.[6]).

Budeme provádět útok se znalostí otevřeného i šifrového textu a postupně se budeme snažit prolomit jednu, dvě a čtyři rundy.

Kapitola 1

Baby Rijndael

1.1 Popis šifry

Baby Rijndael je zjednodušení AESu, které obsahuje pouze čtyři rundy a v každé rundě čtyři S-boxy (neboli operaci SubBytes popsanou níže). Jeho popis je převzatý z [7] a dimlomové práce [8].

Vstupem je 16 bitů otevřeného textu a 16 bitů klíče. Struktura každé rundy Baby Rijndaelu odpovídá rundě AESu, pouze je prováděna s menšími prvky.

Otevřený text, šifrový text, klíč i průběžné hodnoty reprezentujeme jako matici 2×2, kde každý prvek je čtveřice bitů. Při některých operacích se pracuje se sloupkem této matice jako s 8-mi bitovým vektorem. Nejprve se po bitech přixorují bity klíče na otevřený text (stejně jako při AddRoundKey) a výsledek vstupuje do první rundy. V každé rundě probíhají operace SubBytes, ShiftRows, MixColumns, AddRoundKeys.

SubBytes: Vezme každou ze čtyř čtveřic bitů jako hexadcimální číslo a provede substituci podle substituční tabulky (tabulka 1.1). Viz. obrázek 1.1.

To samé lze také zapsat maticově, pokud si označíme vstupní čtveřici bitů jako u_3, u_2, u_1, u_0 a výstupní čtveřici jako v_3, v_2, v_1, v_0. Budeme pracovat v tělese $\mathrm{GF}(2^4)$, kde můžeme u_3, u_2, u_1, u_0 reprezentovat jako polynom

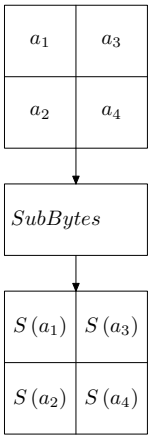
$$u(x) = u_0 + u_1 x + u_2 x^2 + u_3 x^3.$$

Nejprve najdeme multiplikativní inverzi (ozn. $b(x)$) k $u(x)$ v $\mathrm{GF}(2^4)$ vzhledem k ireducibilnímu polynomu $1 + x + x^4$ (inverzní prvek k (0,0,0,0) definujeme jako (0,0,0,0)). Poté s $b(x)$ provedeme afinnní transformaci:

$$\begin{pmatrix} v_3 \\ v_2 \\ v_1 \\ v_0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} b_3 \\ b_2 \\ b_1 \\ b_0 \end{pmatrix} \oplus \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}.$$

Tuto transformaci můžeme vyjádřit také jako:

$$S(u_3, u_2, u_1, u_0) = (x^3 + x^2 + x) \cdot (b_0 + b_1 x + b_2 x^2 + b_3 x^3) + x^3 + x \bmod x^4 + 1.$$

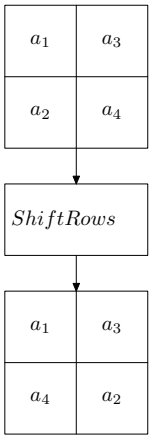


Obrázek 1.1: SubBytes

x	bity x	bity S(x)	S(x)
0	(0,0,0,0)	(1,0,1,0)	a
1	(0,0,0,1)	(0,1,0,0)	4
2	(0,0,1,0)	(0,0,1,1)	3
3	(0,0,1,1)	(1,0,1,1)	b
4	(0,1,0,0)	(1,0,0,0)	8
5	(0,1,0,1)	(1,1,1,0)	e
6	(0,1,1,0)	(0,0,1,0)	2
7	(0,1,1,1)	(1,1,0,0)	c
8	(1,0,0,0)	(0,1,0,1)	5
9	(1,0,0,1)	(0,1,1,1)	7
a	(1,0,1,0)	(0,1,1,0)	6
b	(1,0,1,1)	(1,1,1,1)	f
c	(1,1,0,0)	(0,0,0,0)	0
d	(1,1,0,1)	(0,0,0,1)	1
e	(1,1,1,0)	(1,0,0,1)	9
f	(1,1,1,1)	(1,1,0,1)	d

Tabulka 1.1: Substituční tabulka

ShiftRows: V tabulce průběžných hodnot vymění spodní dva prvky (viz. obrázek 1.2).



Obrázek 1.2: ShiftRows

MixColumns: Oba sloupky tabulky vezme jako 8-mi bitový vektor a přenásobí danou maticí M (viz.obrázek 1.3):

$$M = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

Tato operace se vynechává v poslední rundě.

AddRoundKeys: Ke každé čtveřici bitů přixoruje příslušnou část rundovního klíče (viz.obrázek 1.4).

Ten se vyrábí z původního klíče následující postupem (KeySchedule).

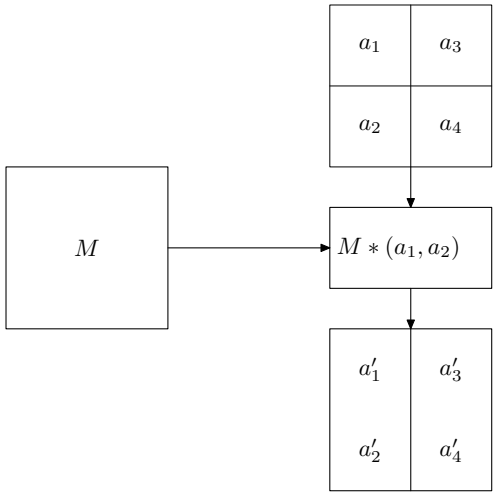
KeySchedule: Vstupní matice klíčů 2×2 rozdělí na sloupce:

$$w_0 = \begin{pmatrix} k_0 \\ k_1 \end{pmatrix} \text{ a } w_1 = \begin{pmatrix} k_2 \\ k_3 \end{pmatrix},$$

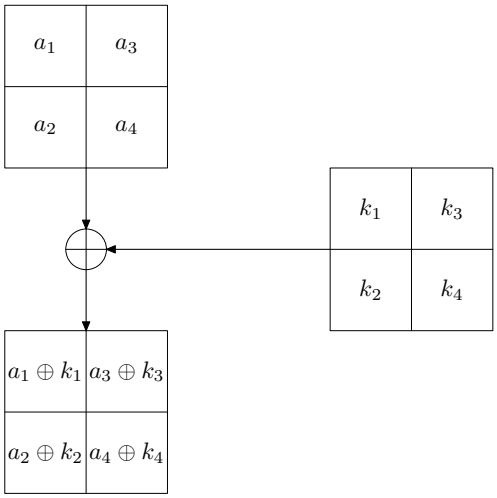
což je klíč, který se xoruje na otevřený text před první rundou. Rundovní klíče pro následující rundy určí jako:

$$w_{2i} = w_{2i-2} \oplus S(reverse(w_{2i-1})) \oplus y_i, \tag{1.1}$$

$$w_{2i+1} = w_{2i-1} \oplus w_{2i}, \tag{1.2}$$

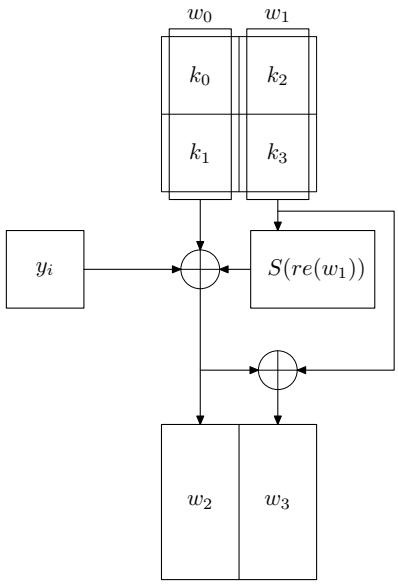


Obrázek 1.3: MixColumns



Obrázek 1.4: AddRoundKey

kde $y_i = \begin{pmatrix} 2^{i-1} \\ 0 \end{pmatrix}$, S označuje substituci podle S-boxu a $i = 1, 2, 3, 4$ je číslo rundy, do které vyráběný klíč vstupuje.



Obrázek 1.5: KeySchedule

1.2 Rovnice

Operace SubBytes je z hlediska algebraické kryptoanalýzy nejzajímavější, protože se jedná o jedinou nelineární součást. Tato operace je funcke, která zobrazuje 16 bitů na 16 bitů, a skládá se ze čtyř S-boxů, tj. čtyř použití funkce, která zobrazuje čtyři bity na čtyři bity. S-box můžeme rozložit na čtyři funkce, které zobrazují čtyři bity na jeden bit, následovně.

Označíme vstupní čtveřici bitů jako u_3, u_2, u_1, u_0 a výstupní čtveřici jako v_3, v_2, v_1, v_0, definujeme čtyři funkce, které pro daný vstup vrátí jeden bit výstupu. Tyto funkce budou mít tvar:

$$\begin{aligned} f_i(u_3, u_2, u_1, u_0) &= c + c_3u_3 + c_2u_2 + c_1u_1 + c_0u_0 + c_{3,2}u_3u_2 + \\ &+ c_{2,1}u_2u_1 + c_{1,0}u_1u_0 + c_{3,1}u_3u_1 + c_{3,0}u_3u_0 + \\ &+ c_{2,0}u_2u_0 + c_{3,2,1}u_3u_2u_1 + c_{2,1,0}u_2u_1u_0 + \\ &+ c_{3,1,0}u_3u_1u_0 + c_{3,2,0}u_3u_2u_0 + c_{3,2,1,0}u_3u_2u_1u_0 = v_i, \end{aligned} \tag{1.3}$$

kde $i = 3, 2, 1, 0$, $c_3, \ldots, c_{3,2}, \ldots, c_{3,2,0}, c_{3,2,1,0}$ jsou koeficinety příslušející po řadě termům $u_3, \ldots, u_3u_2, \ldots, u_3u_2u_0, u_3u_2u_1u_0$ a počítáme modulo 2. Nyní

potřebujeme určit koeficienty $c_3, \ldots, c_{3,2}, \ldots, c_{3,2,1,0}$ a konstanty c u jednotlivých termů. Zároveň víme, že pro každý vstup musí odpovídat hodnota v_i hodnotě bitu příslušného výstupu ze substituční tabulky 1.1.

Ukážeme si postup pro f_3. Budou nás tedy zajímat hodnoty v_3 (první sloupek výstupu v tabulce 1.1).

Postupným dosazováním $(0,0,0,0)$, $(0,0,0,1)$, $(0,0,1,0)$, $\ldots$, $(1,1,1,1)$ za u_3, u_2, u_1, u_0 a příslušného v_3 (tj. nejlevější bit z $\mathrm{S}(u_3, u_2, u_1, u_0)$), získáme:

$$\begin{array}{rcll} f_3(0,0,0,0) &=& c = 1 & \Rightarrow \quad c = 1, \\ f_3(0,0,0,1) &=& c + c_0 u_0 = 1 + c_0 = 0 & \Rightarrow \quad c_0 = 1, \\ f_3(0,0,1,0) &=& c + c_1 u_1 = 1 + c_1 = 0 & \Rightarrow \quad c_1 = 1, \\ f_3(0,0,1,1) &=& c + c_0 u_0 + c_1 u_1 + c_{1,0} u_1 u_0 & \\ & & = 1 + 1 + 1 + c_{1,0} = 1 & \Rightarrow \quad c_{1,0} = 0, \\ & \vdots & & \end{array} \tag{1.4}$$

Vypočtený koeficient vždy dosadíme do další rovnice a tímto způsobem spočítáme všechny koeficienty. Tak získáme funkci f_3. Podobně pro ostatní f_i používáme za v_i postupně všechny i-té bity ze substituční tabulky ($i = 2, 1, 0$). Dostaneme tak tyto 4 funkce, vyjádřené pomocí termů obsahujících u_i:

$$\begin{aligned} f_3(u_3,u_2,u_1,u_0) &= 1 + u_3 + u_1 + u_0 + u_3u_1 + u_3u_0 + u_2u_0 \\ &\quad + u_3u_2u_1 + u_2u_1u_0 + u_3u_1u_0 + u_3u_2u_0 = v_3, \\ f_2(u_3,u_2,u_1,u_0) &= u_3 + u_0 + u_3u_2 + u_3u_0 + u_1u_0 \\ &\quad + u_2u_1u_0 + u_3u_1u_0 = v_2, \\ f_1(u_3,u_2,u_1,u_0) &= 1 + u_3 + u_2 + u_0 + u_3u_1 + u_3u_2 + u_1u_0 + \\ &\quad + u_2u_1 + u_2u_1u_0 + u_3u_2u_0 = v_1, \\ f_0(u_3,u_2,u_1,u_0) &= u_3 + u_1 + u_2u_1 + u_3u_2 + u_3u_2u_1 \\ &\quad + u_3u_1u_0 + u_3u_2u_0 = v_0. \end{aligned} \tag{1.5}$$

1.3 Zjednodušení rovnic

S-boxové funkce můžeme zároveň chápat jako rovnice popisující vztah mezi proměnnými u_i a v_j (viz. 1.3). Pokud bychom chtěli vytvořit rovnice popisující celou šifru použitím uvedených S-boxových funkcí, tím že budeme vždy za vstup do následující rundy dosazovat výstup z předchozí rundy, pak průchodem šifry získáme 16 rovnic se 16ti neznámými s mnoha termy, z nichž některé jsou až stupně 10. Takováto soustava je na dnešní výpočetní techniku příliš složitá, proto budeme používat rovnice pro jednotlivé rundy. Při průchodu každou rundou nám tak přibudou nové rovnice s novými proměnnými. Pro každou rundu ještě navíc nejprve zjednodušíme použité S-boxové funkce (rovnice).

Z rovnic popisujících S-box se chceme zbavit termů stupně tři, proto se je budeme snažit vyjádřit pomocí u_i, v_j a jejich součinů (tj. termů stupně max.

2). Pro každou rovnici popisující S-box 1.5 a termy stupně 3 budeme hledat nahrazení výběrem termů stupně max. 2 následujícím způsobem. Vytvoříme tabulku ohodnocení těchto termů stupně max. 2, ve které budou pro všechny vstupní čtveřice u_3, u_2, u_1, u_0 hodnoty u_i, v_i, $u_i \cdot u_j$, $u_i \cdot v_j$, $v_i v_j$ pro $i, j = 0, \ldots, 3$, viz tabulka 1.2.

Pomocí softwaru Wolfram Mathematica vyřešíme nehomogenní soustavu rovnic, kde levá strana je tato tabulka ohodnocení a pravá strana je ohodnocení součtu termů stupně 3 z rovnice dané funkcí f_i, které chceme nahradit. Řešením bude vektor koeficientů u u_i ,v_i, $u_i u_j$, $u_i v_j$,$v_i v_j$ pro $i, j = 0, \ldots 3$. Použitím vektoru koeficientů u termů stupně max. 2 dostaneme polynom, který nahrazuje termy stupně 3. Stačí tedy nahradit tyto termy tímto polynomem a z každé funkce f_i vytvoříme rovnici s termy stupně max. 2.

Může se stát, že předchozí homogenní soustava nebude mít jedno, ale více řešení, tím získáme více rovnic pro jednu funkci f_i. Také se může stát, že rovnice vzniklá z f_i bude stejná jako rovnice, která vznikla z funkce f_j, proto je dobré vzít sjednocení ze vzniklých rovnic.

V našem případě se zbavíme z první funkce termů: $u_2u_1u_0$, $u_3u_1u_0$, $u_3u_2u_0$ a $u_3u_2u_1$. Z druhé termů: $u_3u_1u_0$, $u_2u_1u_0$. Ze třetí funkce termů: $u_2u_1u_0$, $u_3u_2u_0$. Ve čtvrté funkci nahradíme termy: $u_3u_1u_0$, $u_3u_2u_0$ a $u_3u_2u_1$. Po vynechání stejných rovnic získáme těchto 21 rovnic.

$$
\begin{aligned}
u_3u_1 + u_4u_1 + v_1u_1 + u_1 + u_3 + u_3u_4 + u_4 + u_4v_1 + v_1 + 1 &= 0, \\
u_4u_1 + v_1u_1 + v_2u_1 + u_1 + u_2 + u_2u_3 + u_3 + u_2u_4 + u_3u_4 + v_1 + v_3 &= 0, \\
u_3u_1 + u_1 + u_2 + u_2u_3 + u_3 + u_2u_4 + u_4 + v_1 + u_3v_2 + v_2 + v_3 &= 0, \\
u_3u_2 + u_4u_2 + u_2 + u_1u_3 + u_1u_4 + v_1v_2 + v_2 + v_3 + 1 &= 0, \\
u_1u_2 + u_3u_2 + v_3u_2 + u_1u_3 + u_3 + u_3u_4 + v_1 + v_2 + 1 &= 0, \\
u_2u_1 + u_4u_1 + v_1u_1 + u_1 + u_3 + u_3u_4 + u_4 + v_2 + u_3v_3 &= 0, \\
u_3u_1 + v_1u_1 + u_1 + u_2u_3 + u_2u_4 + u_3v_1 + v_1 + v_2 + v_4 + 1 &= 0, \\
u_2u_1 + v_1u_1 + u_1 + u_2u_3 + u_3 + u_2u_4 + u_2v_2 + v_4 &= 0, \\
u_1u_2 + u_3u_2 + v_1u_2 + u_1u_3 + u_3 + u_3u_4 + u_4 + u_1v_1 + v_3 + v_4 + 1 &= 0, \\
u_1u_2 + u_4u_2 + u_2 + u_3u_4 + v_1 + u_4v_2 + v_2 + v_3 + v_4 &= 0, \\
u_3u_1 + v_1u_1 + v_3u_1 + u_1 + u_2 + u_2u_4 + u_3u_4 + v_1 + v_3 + v_4 &= 0, \\
u_3u_1 + u_4u_1 + u_1 + u_2u_3 + v_1 + v_2 + u_4v_3 + v_4 + 1 &= 0, \\
u_2u_1 + u_4u_1 + u_1 + u_2u_3 + u_3u_4 + v_1v_3 + v_3 + v_4 &= 0, \\
u_4u_2 + u_2 + u_3 + u_1u_4 + u_3u_4 + u_4 + v_2v_3 + v_3 + v_4 + 1 &= 0, \\
u_2u_1 + u_4u_1 + v_4u_1 + u_1 + u_2 + u_2u_3 + v_2 + v_3 + 1 &= 0, \\
u_3u_2 + v_4u_2 + u_2 + u_1u_3 + u_1u_4 + u_1v_1 + v_2 + v_3 + 1 &= 0, \\
u_2u_1 + u_4u_1 + v_1u_1 + u_1 + u_2 + u_2u_3 + v_2 + v_3 + u_3v_4 + v_4 + 1 &= 0, \\
u_4u_1 + v_1u_1 + u_1 + u_2 + u_2u_3 + u_3 + u_2u_4 + u_3u_4 + u_4 + & \\
+v_1 + v_2 + v_3 + u_4v_4 &= 0, \\
u_1 + u_2 + u_2u_3 + u_3 + u_2u_4 + u_3u_4 + u_4 + v_1 + v_2 + v_3 + v_1v_4 &= 0,
\end{aligned}
\tag{1.6}
$$

1	u_1	u_2	u_3	u_4	v_1	v_2	v_3	v_4	u_1u_2	u_1u_3	u_1u_4	u_2u_3	u_2u_4	u_3u_4	u_1v_1	u_1v_2	u_1v_3	u_1v_4	u_2v_1	u_3v_1	u_4v_1	u_4v_4	v_1v_2	v_3v_4
1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	000	0	000	0	00	0	0	1000	0
1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	000	0	000	0	10	0	0	0000	0
1	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	000	0	011	0	00	0	0	0001	1
1	0	0	1	1	1	0	1	1	0	0	0	0	0	1	0	0	0	0	0	000	1	011	1	01	1	0	1100	1
1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	000	0	000	0	00	0	0	0000	0
1	0	1	0	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0	1	110	0	000	1	11	0	1	1010	0
1	0	1	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	010	0	010	0	00	0	0	0000	0
1	0	1	1	1	1	1	0	0	0	0	0	1	1	1	0	0	0	0	1	100	1	100	1	10	0	1	0000	0
1	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	000	0	000	0	00	0	0	0001	0
1	1	0	0	1	0	1	1	1	0	0	1	0	0	0	0	1	1	1	0	000	0	000	0	11	1	0	0011	1
1	1	0	1	0	0	1	1	0	0	1	0	0	0	0	0	1	1	0	0	000	0	110	0	00	0	0	0010	0
1	1	0	1	1	1	1	1	1	0	1	1	0	0	1	1	1	1	1	0	000	1	111	1	11	1	1	1111	1
1	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	000	0	000	0	00	0	0	0000	0
1	1	1	0	1	0	0	0	1	1	0	1	0	1	0	0	0	0	1	0	001	0	000	0	00	0	1	0000	0
1	1	1	1	0	1	0	0	1	1	1	0	1	0	0	1	0	0	1	1	001	1	001	0	00	0	0	0100	0
1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1	101	1	101	1	10	1	1	0101	0

Tabulka 1.2: Tabulka ohodnocení termů pro všechny vstupy

$$u_3u_1 + u_4u_1 + u_1 + u_2 + u_2u_3 + u_3 + u_2u_4 + u_3u_4 + v_1 + v_3 + v_2v_4 = 0,$$
$$u_1u_2 + u_3u_2 + u_4u_2 + u_1u_4 + u_3u_4 + v_1 + v_2 + v_3v_4 + 1 = 0.$$

Další vztahy mezi vstupy a výstupy z S-boxu můžeme získat z jeho vyjádření pomocí násobení polynomů modulo $x^4 + 1$ v tělese (GF 2^4): Je-li $m_0 + m_1x + m_2x^2 + m_3x^3$ polynom vyjadřující inverzní prvek k (u_3, u_2, u_1, u_0) modulo $x^4 + x + 1$ (inverzi k $(0, 0, 0, 0)$ definujeme jako $(0, 0, 0, 0)$), můžeme S-box vyjádřit jako:

$$S(u_3, u_2, u_1, u_0) = (x^3 + x^2 + x) \cdot (b_0 + b_1x + b_2x^2 + b_3x^3) + x^3 + x \bmod x^4 + 1.$$

Neboli

$$\begin{aligned} v(x) &= v_0 + v_1x + v_2x^2 + v_3x^3 = (x^3 + x^2 + x) \cdot (b_0 + b_1x + b_2x^2 + \\ &+ b_3x^3) + x^3 + x \equiv (b_0 + b_1 + b_2 + 1)x^3 + (b_0 + b_1 + b_3)x^2 + \\ &+ (b_0 + b_2 + b_3 + 1)x + b_1 + b_2 + b_3 \bmod x^4 + 1. \end{aligned} \tag{1.7}$$

Z čehož porovnáním koeficientů u jednotlivých řádů x plyne:

$$\begin{aligned} v_3 &= b_0 + b_1 + b_2 + 1, \\ v_2 &= b_0 + b_1 + b_3, \\ v_1 &= b_0 + b_2 + b_3 + 1, \\ v_0 &= b_1 + b_2 + b_3. \end{aligned} \tag{1.8}$$

Z těchto rovnic si vyjádříme

$$\begin{aligned} b_3 &= v_0 + v_1 + v_2 + 1, \\ b_2 &= v_0 + v_1 + v_3, \\ b_1 &= v_0 + v_2 + v_3 + 1, \\ b_0 &= v_1 + v_2 + v_3. \end{aligned} \tag{1.9}$$

Víme, že

$$(b_0 + b_1x + b_2x^2 + b_3x^3)(u_0 + u_1x + u_2x^2 + u_3x^3) \equiv 1 \pmod{x^4 + x + 1}$$

(jsou to inverzní prvky mod $x^4 + x + 1$). Můžeme tedy do této rovnice dosadit za b_i a po roznásobení, upravení a porovnání koeficientů u jednotlivých řádů x dostaneme rovnice (postupně pro řád $x = 0, 1, 2, 3$):

$$\begin{aligned} u_1v_0 &+ u_1v_1 + u_1v_2 + u_2v_0 + u_3v_0 + u_0v_1 + u_2v_1 + u_0v_2 + u_3v_2 + u_0v_3 + \\ &+ u_2v_3 + u_3v_3 + u_1 + u_3 = 1, \\ u_0v_0 &+ u_0v_2 + u_0v_3 + u_1v_0 + u_3v_1 + u_2v_2 + u_3v_2 + u_1v_3 + u_2v_3 + u_0 + \end{aligned} \tag{1.10}$$

$$
\begin{aligned}
& & & +u_1 + u_2 + u_3 = 0, \\
u_1v_0 & + & & u_1v_2 + u_1v_3 + u_0v_0 + u_2v_0 + u_0v_1 + u_3v_2 + u_0v_3 + u_2v_3 + u_3v_3 + \\
& & & +u_1 + u_2 + u_3 = 0, \\
u_0v_0 & + & & u_0v_1 + u_0v_2 + u_1v_0 + u_2v_0 + u_3v_0 + u_1v_1 + u_2v_2 + u_1v_3 + u_2v_3 + \\
& & & +u_3v_3 + u_0 + u_2 + u_3 = 0.
\end{aligned}
$$

Pokud by byl vstup do S-boxu $(0, 0, 0, 0)$, neplatila by první z těchto rovnic, proto ji nebudeme používat a použijeme zbylé tři rovnice. Přidáme je k 21 rovnicím vyjadřujícím S-box 1.6 a provedeme Gaussovu eliminaci. Zjistíme, že pouze 21 rovnic bylo lineárně nezávislých.

Při vytváření rovnic pro celou šifru budeme každým průchodem S-boxu přidávat 4 neznámé, přibude 21 rovnic a dále rundou budou pokračovat nové proměnné. S nimi se provedou operace ShiftRows, MixColumns, AddRoundKey a stanou se vstupem do další rundy (případně šifrovým textem).

1.4 Implementace

Výpočty prováděné v této kapitole, na získání soustavy rovnic popisují S-box jsou implementované v souboru `BabyRijndael.nb` v sekcích `Vytvoření dalších čtyř rovnic pro S-box` a `Snížení stupně u rovnic pro S-box`. Šifrování, dešifrování i expanze klíče jsou realizovány v souboru `BabyRijndael.nb` v sekcích `Funkce potřebné k šifrování, dešifrování a expanzi klíče` a `Funkce na vytváření rovnic`.

Funkce použité pro implementaci operace KeySchedule jsou:

- `word[předpředchozí klíč, předchozí klíč, číslo rundy ve které se použijí]` vrátí dvojici klíčů pro danou rundu, přitom použije funkce `sbox2` a `y`. Tato verze funkce `word` se používá při klasické expanzi klíče, kdy známe hodnoty proměnných *předpředchozí klíč* a *předchozí klíč*.

- `word[předpředchozí klíč, předchozí klíč, jména nových proměnných, číslo rundy ve které se použijí]` vrátí dvojici klíčů pro danou rundu, přitom použije funkce `sbox22` a `y`. Tato verze funkce `word` se používá při algebraické kryptoanalýze, kdy klíče jsou neznámé a vytváříme rovnice popisující vztahy mezi starými proměnnými a novými. Staré proměnné jsou představovány vstupujícími klíči, nové vystupující proměnné si pojmenováváme (*jména nových proměnných*).

- `sbox2[vstup]` vrátí výstupy ze dvou S-boxů, se vstupy *vstup*. Tato funcke se používá při klasické expanzi klíče.

- `sbox22[vstupní proměnné, výstupní proměnné]` do globální proměnné *klicoverce* uloží dvakrát 21 rovnic pro S-box, ve kterých za u_i dosadí

vstupní proměnné a za v_i *výstupní proměnné* a vrátí vektor výstupních proměnných. Tato funkce se používá pro získání rovnic pro klíče při algebraické kryptoanalýze.

- `y[číslo rundy]` vrací vektor $\begin{pmatrix} 2^{\check{c}\acute{i}slo\ rundy-1} \\ 0 \end{pmatrix}$.

Funkce na šifrování jsou:

- `runda[vstup, číslo rundy]` provede operaci `subbytes[vstup]`, na výsledek použije postupně `shiftrows`, `mixcolumns` a `addroundkey` s klíčem daným číslem rundy z množiny *rundovniklice* a vrátí výstup po dané rundě. Tato verze funkce se používá při klasickém šifrování, kdy známe hodnoty vstupu a klíčů.

- `runda[vstup, nové proměnné na výstup z S-boxu, číslo rundy]` provede operaci `subbytes1[vstup, nové proměnné]`, na výsledek použije postupně `shiftrows`, `mixcolumns` a `addroundkey` s klíčem daným číslem rundy z množiny *rundovniklice* a vrátí výstup po jedné rundě. Do proměnné *soustava* uloží rovnice vzniklé při `subbytes1`. Tato verze funkce se používá k vytvoření rovnic při algebraické kryptoanalýze.

- `subbytes[vstup]` vrátí čtyři čtveřice, výstupy ze čtyř S-boxů, které jako vstupy berou části vstupu *vstup*. Tato funkce se používá se při klasickém šifrování, kdy známe hodnoty vstupů.

- `subbytes1[vstupní proměnné, výstupní proměnné]` do globální proměnné *soustava* přidá čtyřikrát 21 rovnic 1.6 (pro čtyři S-boxy), ve kterých za u_i dosadí vstupní proměnné a za v_i výstupní proměnné a vrátí vektor výstupních proměnných. Tato verze se používá k vytvoření rovnic při algebraické kryptoanalýze.

- `shiftrows[matice proměnných]` vrátí matici proměnných s prohozenými prvky v druhém řádku.

- `mixcolumns[matice proměnných]` vrátí vstupní matici přenásobenou danou maticí M.

Dešifrovací funkce jsou `isubbytes`, `ishiftrows`, `imixcolumns` a dělají inverzi k příslušným šifrovacím funkcím.

1.5 Popis šifry AES

Celá kryptoanalýza Baby Rijndaelu má za úkol najít slabinu šifry AES. Proto si zde uvedeme základní definice AESu, podrobný popis najdeme např. v [9]. Šifra Rijndael (=AES) může mít délky bloku otevřeného textu 128, 160, 192, 224 nebo 256 bitů a délky klíčů také 128, 160, 192, 224 nebo 256 bitů (nezávisle na délce bloku). Nejpodobnější Baby Rijndaelu je verze se 128 bity, jak otevřeného textu, tak pracovního bloku, tak i klíče s 10 rundami. Dále tedy budeme pracovat pouze s touto verzí.

V průběhu šifrování se pracuje s bloky velikosti 4x4 byty. Pracovní blok naplníme po bytech otevřeným textem zhora dolů, nejprve nejlevější sloupek a poté postupně další sloupky. Na otevřený text se přixoruje první rudovní klíč (který je také ve formě obdobně vytvořeného pracovním bloku). V každé rundě, stejně jako v Baby Rijndaelu, probíhají operace SubBytes, ShiftRows, MixColumns a AddRoundKey. Také zde se v poslední rundě vynechává operace MixColumns.

SubBytes Nejzajímavější je také zde operace SubBytes, budeme se jí tedy zabývat podrobněji. Operace SubBytes rozdělí vstup na 16 bytů a za každý provede dosazení podle S-boxu, celkem tedy použijeme 16 S-boxů. Každý S-box může být reprezentován tabulkou 16x16 1.3, kde na řádku určeném vyššími (prvními) 4mi bity vstupu a sloupku určeném nižšími (druhými) 4mi bity najdeme prvek, který dosadíme za vstup.

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

Tabulka 1.3: Substituční tabulka pro AES, převzato z [1]

Pro porovnání algebraické kryptoanalýzy potřebujeme zjistit, jakého stupně jsou polynomy vyjadřující S-box. Podobně jako u Baby Rijndaelu můžeme přepisovací tabulku reprezentovat osmi funkcemi, kde výstup každé z nich bude jeden bit výstupu z S-boxu. Označíme-li si vstup do S-boxu $u = (u_7, u_6, u_5, u_4, u_3, u_2, u_1, u_0)$ a výstup $v = (v_7, v_6, v_5, v_4, v_3, v_2, v_1, v_0)$, bude mít tato funkce tvar:

$$\begin{aligned} v_i &= f_i(u_7, u_6, u_5, u_4, u_3, u_2, u_1, u_0) = c + c_0u_0 + \cdots + c_7u_7 + c_{0,1}u_0u_1 + \\ &\quad + \cdots + c_{6,7}u_6u_7 + c_{0,1,2}u_0u_1u_2 + \cdots + c_{5,6,7}u_5u_6u_7 + \cdots + \\ &\quad + c_{0,1,2,3,4,5,6,7}u_0u_1u_2u_3u_4u_5u_6u_7, \end{aligned} \tag{1.11}$$

kde $i = 0, \ldots, 7$, $c_0, \ldots, c_7, c_{0,1}, \ldots, c_{0,1,2}, \ldots, c_{0,1,2,3,4,5,6,7}$ jsou koeficinety příslušející po řadě termům $u_0, u_0u_1, \ldots, u_0u_1u_2, \ldots, u_0u_1u_2u_3u_4u_5u_6u_7$. Postupným dosazováním $u = (0, 0, 0, 0, 0, 0, 0, 0)$ až $u = (1, 1, 1, 1, 1, 1, 1, 1)$ a příslušných bitů výstupu podle substituční tabulky dostaneme pro každé i 156 koeficientů, které definují funkce f_i. Tyto funkce mohou být až stupně 7. Stejně jako u Baby Rijndaelu se budeme chtít zbavit termů stupně většího než 2. Budeme je tedy chtít vyjádřit pomocí termů stupně 1 a 2 s proměnnými u_i a v_i $(i = 7, \ldots, 0)$.

Vytvoříme tabulku 256 x 137 se všemi ohodnoceními proměnných u_i, v_i a jejich součinů u_iu_j, v_iv_j a u_iv_j pro $i, j = 7, \ldots, 0$. Tuto tabulku vezmeme jako homogenní matici, jako pravou stranu k této matici přiřadíme vektor, ve kterém bude ohodnocení součtu částí funkce f_i s vyšším stupněm, a snažíme se najít řešení. Pro každou z funkcí f_i nalezneme 39 rovnic. Jejich sjednocením dostaneme pouze 39 rovnic.

ShiftRows: v bloku 4x4 byty posune i-tý řádek o i pozic doleva $(i = 0, 1, 2, 3)$. To pro algebraickou analýzu znamená pouze to, že se "potkají"jiné proměnné.

MixColumns: každý sloupec se přenásobí konstantní maticí A o velikosti 4x4, která má jako prvky vektory z $GF(2^8)$. Do další funkce půjde tedy lineárně změněný vstup. Matice A zapsaná hexadecimálně:

$$A = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix}.$$

AddRoundKey: na bity pracovního bloku se přixorují příslušné bity klíče.

KeySchedule: z 16 bytů klíče vyrobí 176 bytů klíče pro 10 rund. Probíhá podobně jako u Baby Rijndaelu. Jedinná nelineární operace je průchod S-boxem, který je stejný jako při šifrování.

Kapitola 2

Algoritmy na řešení soustavy nelineárních rovnic

V této kapitole popíšeme algoritmy XL a XSL a metodu T', které se používají na řešení soustav nelineárních rovnic. Podrobný popis těchto algoritmů najdeme v [10] a v [4]. Dále poté uvádíme nově navržené pomocné postupy, které byly použity pro další zjednodušení a řešení soustavy nelineárních rovnic popisujících analyzovanou šifru.

2.1 XL a XSL

V průběhu algebraické kryptoanalýzy budeme potřebovat vyřešit soustavu kvadratických rovnic. V této sekci si ukážeme algoritmus, jak se dá takováto soustava upravit na lineární soustavu, kterou už umíme (snadno) vyřešit.

Nejprve se budeme zabývat algoritmem XL, který byl publikován v roce 2000 na EUROCRYPT [11]. Vstup do tohoto algoritmu bude soustava kvadratických rovnic S nad tělesem K. Jednotlivé rovnice budou tvaru

$$l_k = f_k(x_1, x_2, \cdots, x_n) - b_k = 0$$

pro $k = 1, 2 \cdots, m$. Chceme nalézt řešení $x = (x_1, x_2, \cdots, x_n) \in K^n$ pro dané $b = (b_1, \cdots, b_m) \in K^m$.

Algoritmus 1 (XL algoritmus) vstup: *soustava* $\{l_k\}$, *parametr* $D > 2$

výstup: x

1.Násobení *vynásob všechny rovnice termy* x_{i_1}, $x_{i_1} \cdot x_{i_2}$,$\cdots$, $x_{i_1} \cdot x_{i_2} \cdot \cdots \cdot x_{i_k}$, *kde* $k \leq D - 2$ *a* x_{i_j} *je jedno z* $\{x_1, x_2, \ldots, x_n\}$.

2.Linearizace *všechny nelineární monomy změň na nové proměnné. Proveď Gaussovou eliminaci tak, že všechny monomy s jednou proměnnou* x_1 *budeš eliminovat naposledy.*

3.Vyřeš *Vyřeš rovnici s jednou neznámou (x_1) v tělese K.*

4.Opakuj *Dosaď řešení do soustavy a opakuj.*

Algoritmus XSL ("eXtended Sparse Linearization" nebo "multiply (X) by Selected monomials and Linearize") je upravený XL algoritmus, ve kterém se rovnice násobí pouze vybranými termy. Publikovali ho Nicolas T. Courtois a Josef Pieprzyk v roce 2002 v článku [4]. Zvolíme parametr $P > 1$. Pro každý S-box máme r rovnic tvaru:

$$\sum x_i x_j + \sum x_i + \sum x_j + b = 0.$$

Ty vynásobíme součiny termů z ostatních S-boxů, které mají stupeň maximálně $P - 1$. Poté provedeme linearizaci a dořešíme soustavu stejně jako při XL algoritmu.

2.2 T' metoda

Když potřebujeme více lineárně nezávislých rovnic a přitom nechceme zvýšit počet neznámých, můžeme použít algoritmus s názvem T' metoda viz.[4]. Tím ze soustavy rovnic získáme další platné rovnice, které budou lineárně nezávislé. Vezmeme si soustavu R rovnic a dvě její neznámé, např. x_1, x_2. Označíme si

- T termy dané soustavy,
- $T' = T_{x_1}$ termy soustavy, které i po vynásobení x_1 budou patřit do T,
- $T'' = T_{x_2}$ termy soustavy, které i po vynásobení x_2 budou patřit do T.

Je-li počet rovnic větší nebo roven počtu termů v T_{x_1}, respektive T_{x_2} , můžeme všechny termy z $T - T_{x_1}$ vyjádřit pomocí termů T_{x_1}, respektive T_{x_2}. Toho docílíme provedením Gaussovy eliminace, kde termy z T_{x_1}, respektive T_{x_2}, používáme až nakonec.

Na konci soustavy tak získáme alespoň $R - |T| + |T_{x_1}|$, respektive $R - |T| + |T_{x_2}|$, rovnic s termy pouze z T_{x_1}, respektive T_{x_2}. Označíme si je C_{x_1}, respektive C_{x_2}. Když rovnice z C_{x_1} vynásobíme x_1, dostaneme rovnice s termy z T. Jsou lineárně nezávislé? Určitě jsou lineárně nezávislé na rovnicích s termy z T. Na rovnicích z C_{x_1} budou většinou nezávislé (např. rovnice, která je po přenásobení stejná jako (jiná) rovnice z C_{x_1}, je lineárně závislá na rovnicích z C_{x_1}).

Dále můžeme vzniklé rovnice vyjádřit jen pomocí termů z T_{x_2} (protože mají termy z T) a spolu s C_{x_2} je vynásobit x_2. Vzniknou rovnice s termy z T. Celkem jsme tedy našli maximálně $2(R - |T| + |T_{x_1}|) + R - |T| + |T_{x_2}|$ rovnic. Podobně můžeme použít ostatní proměnné, dokud nezískáme dostatečný počet rovnic.

Při praktickém použití na rovnice vzniklé během analyzování šifry Baby Rijndael je použitelných vzniklých rovnic méně (některé jsou shodné, jiné se modulo 2 vynulují).

2.3 Využití nelineárnosti

Při řešení linearizované soustavy, kterou dostaneme pomocí algoritmů XL/XSL, můžeme při zjednodušování někdy využít původní nelineární vztahy.

Pokud někde narazíme na rovnici, která neobsahuje monomy vzniklé při linearizaci (tedy obsahuje pouze původní proměnné), vyjádříme z této rovnice jednu vybranou neznámou. V ostatních rovnicích z celé soustavy zpětně nahradíme linearizované monomy, které obsahují tuto jednu neznámou, za původní monomy vyššího stupně. V nich poté dosadíme za tuto jednu neznámou její vyjádření. Roznásobíme a linearizujeme.

Např. získáme rovnici $y_{16} + k_{13} + l_2 + l_3 + l_4 + y_{14} + y_{15} = 0$ a můžeme za y_{16} dosazovat $k_{13} + l_2 + l_3 + l_4 + y_{14} + y_{15}$. Po zpětném převedení proměnných m_{y_{16},y_{12},l_2} na kubický monom $y_{16} \cdot y_{12} \cdot l_2$ dosadíme za proměnnou y_{16}. Nově vzniklé termy vyššího stupně po dosazení za y_{16}, např. $k_{13} \cdot y_{12} \cdot l_2$, opět linearizujeme definováním nové proměnné m_{k_{13},y_{12},l_2}.

Správné dosazení za všechny výskyty zajišťuje funkce `opatrnedosad[`a`,`b`]`, která vstup a převede na nelineární tvar, dosadí do něj druhý vstup b, roznásobí (modulo 2) a převede zpět na lineární tvar.

Pokud máme v soustavě rovnici typu $m_{a,b,c} = 1$ pak automaticky $a = 1$, $b = 1$ a $c = 1$ (a,b,c jsou proměnné soustavy v nelineárním tvaru). Podobně je-li $m_{a,b} = 1$, je $a = 1$, $b = 1$ a zároveň pro každé x a proměnnou $m_{a,b,x}$ platí $m_{a,b,x} = x$. Je-li $m_{a,b} = 0$, pak také $m_{a,b,x} = 0$ pro každé x.

2.3.1 Využití částečného řešení

Pokud ani T' metodou nezískáme dostatek rovnic, můžeme soustavu přenásobit některými termy a dostat se tak k rovnicím vyššího řádu. Ani tímto postupem ovšem není zaručen dostatek rovnic, jak bude ukázáno v další kapitole na konkrétním útoku.

Pokud nemáme dostatek rovnic na nalezení unikátního řešení lineární soustavy, můžeme alespoň ze všech řešení lineární soustavy vytvořit všechna řešení nelineárního tvaru soustavy následujícím způsobem.

Mějme kvadratickou (kubickou) soustavu S a její linearizaci, soustavu N. Pomocí funkce LinearSolve získáme partikulární řešení soustavy N a pomocí NullSpace dostaneme nulový prostor soustavy N. Z partikulárního řešení a nulového prostoru můžeme vytvořit množinu vektorů V, jejichž lineární kombinace dává všechna řešení soustavy N. Tato lineární kombinace vektorů V obsahuje zároveň všechna řešení soustavy S (a nějaké další vektory).

Některé souřadnice vektoru z V odpovídají monomům s dvěma (třemi) proměnnými, jiné monomům s jednou proměnnou. Zkrátíme-li v množině V

vektory tak, že z každého vektoru vezmeme jen ty souřadnice odpovídající monomům s jednou proměnnou (to jsou přesně proměnné v S), dostaneme množinu vektorů (ozn. W), jejichž lineární kombinací získáme všechna řešení soustavy S (a nějaké vektory navíc). Postupným dosazením do S z nich získáme všechna řešení soustavy S.

Běžně bychom při hledání řešení S nejprve udělali lineární kombinaci vektorů V, a až poté vybírali souřadnice odpovídající monomům s jednou proměnnou. Ani při obráceném postupu se ale neztratí žádné z řešení S, protože lineární kombinace probíhá nad jednotlivými souřadnicemi.

2.3.2 Využití pravděpodobnosti

V této části si spočítáme, s jakou pravděpodobností bude $m_{a,b} = 0$ pro nějaké proměnné a, b. Pokud je tato pravděpodobnost dostatečně vysoká, můžeme s velkou spolehlivostí dosadit za $m_{a,b}$ nulu a zjednodušit si tak původní soustavu rovnic.

Pokud během řešení soustavy dostaneme rovnice $m_{a,b,z} = 0$, kde z probíhá všechny proměnné (vyjma a, b), tak $m_{a,b} = 0$. Pokud dostaneme pouze některé takové rovnice $m_{a,b,z} = 0$ (z neprobíhá všechny proměnné), bude $m_{a,b} = 0$ s nějakou pravděpodobností, kterou můžeme spočítat.

Nejprve spočítáme pravděpodobnost $m_{a,b} = 0$ pro dvě rovnice typu $m_{a,b,z} = 0$, tedy pro $m_{a,b,c} = 0$ a $m_{a,b,d} = 0$. Převedeme na nelineární tvar. Chceme spočítat pravděpodobnost, že součin $ab = 0$ za podmínky $abc = 0$ a zároveň $abd = 0$, tedy:

$$\begin{aligned} P\left(ab = 0 | abc = 0 \wedge abd = 0\right) &= \frac{P\left(ab = 0 \wedge abc = 0 \wedge abd = 0\right)}{P\left(abc = 0 \wedge abd = 0\right)} = \\ &= \frac{P\left(ab = 0\right)}{P\left(abc = 0 \wedge abd = 0\right)}. \end{aligned} \tag{2.1}$$

Pravděpodobnosti $P\left(ab = 0\right)$ se rovná pravděpodobnosti, že alespoň jedno z a, b je nulové, platí tedy

$$P(ab = 0) = P(a = 0 \wedge b = 0) + P(a = 1 \wedge b = 0) + P(a = 0 \wedge b = 1) = 3 \cdot \frac{4}{16} = \frac{12}{16}.$$

Stačí si tedy dopočítat jmenovatele. Vztahy $abc = 0 \wedge abd = 0$ platí buď v případě, že $ab = 0$ a c, d jsou libovolné (to má pravděpodobnost $\frac{12}{16}$) nebo v případě, že $ab = 1$ a $c = 0$, $d = 0$, což má pravděpodobnost $\frac{1}{16}$. Celkem je tedy

$$P(abc = 0 \wedge abd = 0) = P(ab = 0) + P(ab = 1 \wedge c = 0 \wedge d = 0) = \frac{12}{16} + \frac{1}{16} = \frac{13}{16}.$$

Dostáváme, že pravděpodobnost

$$P(ab = 0 | abc = 0 \wedge abd = 0) = \frac{\frac{12}{16}}{\frac{13}{16}} = \frac{12}{13} = 0,923.$$

Tedy s pravděpodobností $92,3\%$ bude $m_{a,b} = 0$.

Obdobně můžeme odvodit vztah pro n proměnných $a, b, c_1, \ldots, c_{n-2}$, neboli pro $n-2$ rovnic typu $m_{a,b,z} = 0$.

$$\begin{aligned} P(ab = 0 | abc_1 = 0 \wedge \quad \cdots \quad \wedge abc_{n-2} = 0) = & \qquad (2.2) \\ = \quad & \frac{P(ab = 0 \wedge abc_1 = 0 \wedge \cdots \wedge abc_{n-2} = 0)}{P(abc_1 = 0 \wedge \cdots \wedge abc_{n-2} = 0)} = \\ = \quad & \frac{P(ab = 0)}{P(ab = 0) + P(ab = 1 \wedge c_1 = 0 \wedge \cdots \wedge c_{n-2} = 0)}. \end{aligned}$$

Jsou tři způsoby ohodnocení a, b, tak aby $ab = 0$, a pro každý z nich můžeme zbylé proměnné vybrat libovolně (to je 2^{n-2} způsoby z 2^n), máme tedy

$$\begin{aligned} P(ab = 0) \quad = \quad & P(a = 0 \wedge b = 0) + P(a = 1 \wedge b = 0) + P(a = 0 \wedge b = 1) = \\ = \quad & \frac{2^{n-2} + 2^{n-2} + 2^{n-2}}{2^n} = \frac{3 \cdot 2^{n-2}}{2^n}. \end{aligned} \qquad (2.3)$$

Stačí tedy dopočítat

$$P(ab = 1 \wedge c_1 = 0 \wedge \cdots \wedge c_{n-2} = 0) = \frac{1}{2^n}.$$

Celkem tedy dostaneme

$$\begin{aligned} P(ab = 0 | abc_1 = 0 \wedge \ldots \wedge abc_{n-2} = 0) \quad = \quad & \frac{\frac{3 \cdot 2^{n-2}}{2^n}}{\frac{3 \cdot 2^{n-2}}{2^n} + \frac{1}{2^n}} = \\ = \quad & \frac{3 \cdot 2^{n-2}}{3 \cdot 2^{n-2} + 1}. \end{aligned} \qquad (2.4)$$

Pokud bychom chtěli být úplně korektní měli bychom spočítat pravděpodobnost pro p proměnných $a, b, c_1, \ldots, c_{n-2}, d_1, \ldots, d_{p-n}$ a $n-2$ rovnic typu $m_{a,b,z} = 0$. V tomto případě by byly pouze všechny pravděpodobnosti přenásobeny $\frac{2^{p-n}}{2^{p-n}}$, výsledná pravděpodobnost se tedy nezmění.

Můžeme spočítat pravděpodobnosti pro prvních pár n:

n	3	4	5	6	7
P	$\frac{6}{7}$	$\frac{12}{13}$	$\frac{24}{25}$	$\frac{48}{49}$	$\frac{96}{97}$
%	$85,71$	$92,30$	96	$97,95$	$98,96$

Vidíme, že pokud během řešení soustavy dostaneme dvě a více rovnic typu $m_{a,b,z} = 0$, bude s velkou pravděpodobností i $m_{a,b} = 0$.

Kapitola 3

Útoky

3.1 Příprava dvojic otevřený-šifrový text

Jelikož chceme provádět útok se znalostí otevřeného i šifrového textu, zvolíme otevřený text, v tomto případě $5, 4, 2, 1$. Zvolíme klíč a zašifrujeme. Útočit budeme na jednu, dvě a čtyři rundy, potřebujeme tedy určit výstupy z šifrování po první, druhé a čtvrté rundě. Šifrování je naprogramováno po rundách, je tedy snadné pomocí souboru `Utoky.nb` sekce `Příprava dvojic otevřený-šifrový text`, získat všechny tyto šifrové texty. Poté "zapomeneme"klíč a budeme předstírat, že jsme dvojice otevřený-šifrový text získali např. odposlechem komunikace.

3.2 Sestavení rovnic

Způsobem uvedeným v předchozí kapitole umíme pro každý S-box každé rundy získat vztahy pro jeho vstupy a výstupy, 21 rovnic 1.6. Chceme získat rovnice popisující průchod šifrou. Bity klíče, které budeme hledat si označíme $k_1, \ldots, k_{16}$.

Nejprve najdeme rovnice odpovídající operaci KeySchedule. Vstupem a zároveň klíčem, který se bude na otevřený text xorovat před první rundou, jsou bity klíče

$$w_0 = \begin{pmatrix} k_1 \\ k_2 \\ k_3 \\ k_4 \\ k_5 \\ k_6 \\ k_7 \\ k_8 \end{pmatrix} \text{ a } w_1 = \begin{pmatrix} k_9 \\ k_{10} \\ k_{11} \\ k_{12} \\ k_{13} \\ k_{14} \\ k_{15} \\ k_{16} \end{pmatrix}.$$

Do dvou S-boxů v operaci KeySchedule vstupuje

$$reverse(w_1) = (k_{13}, k_{14}, k_{15}, k_{16}, k_9, k_{10}, k_{11}, k_{12})^T .$$

Výstupy S-boxů si označíme $l_1, \ldots l_4$ a $l_5, \ldots l_8$. Uložíme si dvakrát 21 rovnic popisujících vztahy mezi vstupem S-boxu a l_i, tj. přesně rovnice získané v sekci 1.3. V prvním případě za u_i dosadíme $k_{13}, k_{14}, k_{15}, k_{16}$ a za v_j dosadíme l_1, l_2, l_3, l_4, v druhém za u_i dosadíme $k_9, k_{10}, k_{11}, k_{12}$ a za v_j dosadíme l_5, l_6, l_7, l_8. Klíče pro první rundu budou vypadat takto (sčítání bereme samozřejmě modulo 2):

$$w_2 = w_0 \oplus \begin{pmatrix} l_1 \\ l_2 \\ l_3 \\ l_4 \\ l_5 \\ l_6 \\ l_7 \\ l_8 \end{pmatrix} \oplus \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} k_1 + l_1 \\ k_2 + l_2 \\ k_3 + l_3 \\ k_4 + l_4 + 1 \\ k_5 + l_5 \\ k_6 + l_6 \\ k_7 + l_7 \\ k_8 + l_8 \end{pmatrix}, \tag{3.1}$$

$$w_3 = w_1 \oplus w_2 = \begin{pmatrix} k_1 + l_1 + k_9 \\ k_2 + l_2 + k_{10} \\ k_3 + l_3 + k_{11} \\ k_4 + l_4 + 1 + k_{12} \\ k_5 + l_5 + k_{13} \\ k_6 + l_6 + k_{14} \\ k_7 + l_7 + k_{15} \\ k_8 + l_8 + k_{16} \end{pmatrix} \tag{3.2}$$

Vstupem do další rundy KeySchedule budou tyto klíče. Podobně jako v předchozím případě zavedeme nové proměnné, výstupy z S-boxu, $l_{2,1}, l_{2,2}, \ldots, l_{2,8}$. Uložíme si opět rovnice popisující vztahy v S-boxu a spočítáme další klíče pomocí vztahů 1.1:

$$w_{2i} = w_{2i-2} \oplus \begin{pmatrix} l_{i,1} \\ l_{i,2} \\ l_{i,3} \\ l_{i,4} \\ l_{i,5} \\ l_{i,6} \\ l_{i,7} \\ l_{i,8} \end{pmatrix} \oplus \begin{pmatrix} 2^{i-1} \\ 0 \end{pmatrix} \tag{3.3}$$

$$w_{2i+1} = w_{2i-1} \oplus w_{2i}, \tag{3.4}$$

kde $i = 1, \ldots, 4$ označuje číslo rundy, ve které se bude příslušný klíč používat. Postupně dostaneme všechny potřebné klíče (ukládáme je do globální

proměnné *rundovniklice*) a rovnice jejich vztahů z S-boxů použitých v operaci KeySchedule.

Podobně budeme postupovat i při simulaci šifrování. Ke zvolenému otevřenému textu přixorujeme příslušnou část (w_0, w_1) expandovaného klíče. To bude po částech vstup do čtyř S-boxů první rundy, jejich výstupy si označíme $y_1, y_2, y_3, \ldots, y_{16}$. Uložíme si opět rovnice popisující jejich vztahy. S výstupy $y_1, y_2, y_3, \ldots, y_{16}$ provedeme operace ShiftRows, MixColumns a přixorujeme příslušející klíče, tak vznikne vstup do další rundy (viz. obrázek 3.1).

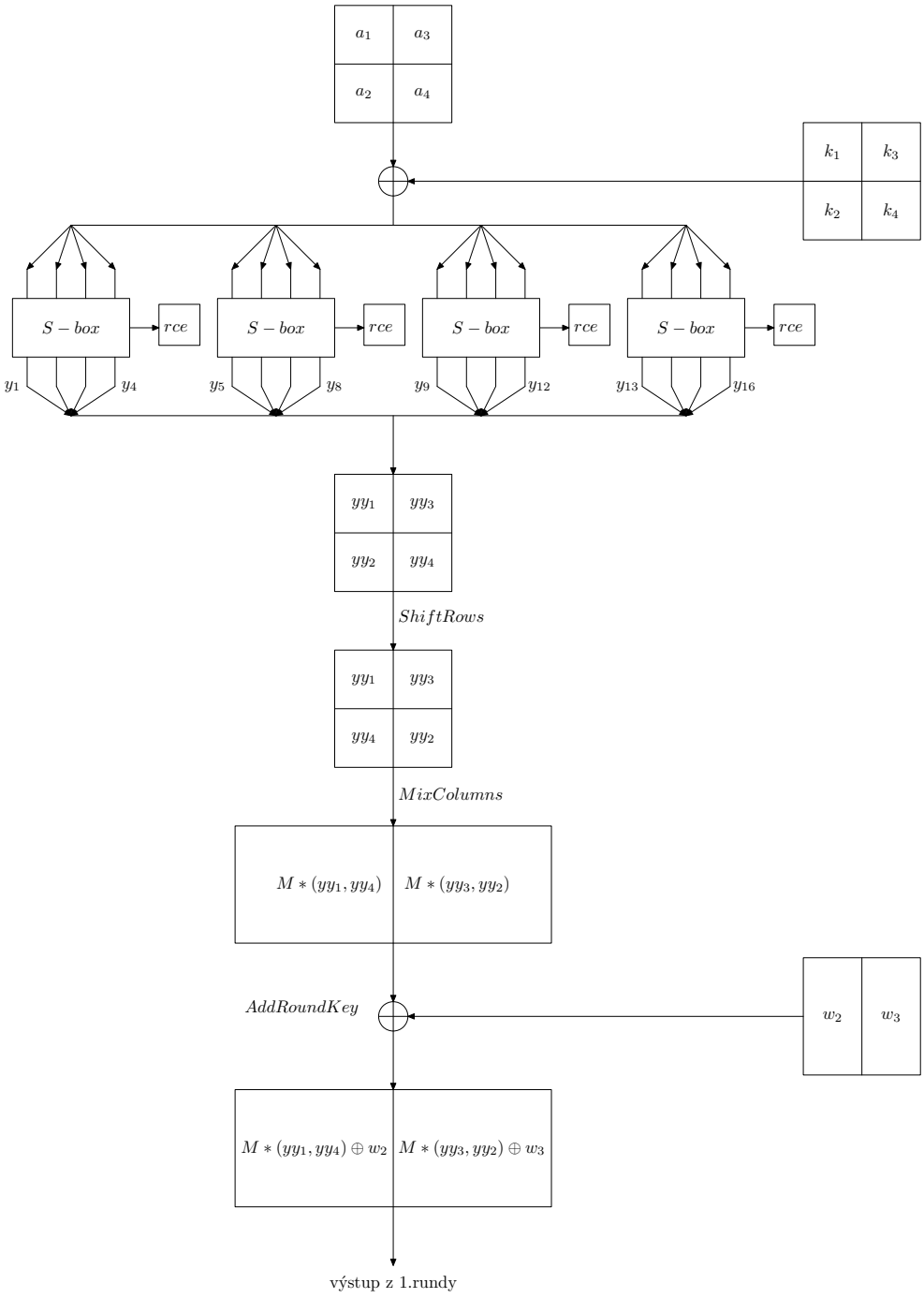
Dále postupujeme obdobně, ve druhé rundě přibudou proměnné $y_{2,1}$, $y_{2,2}, \ldots, y_{2,16}$, ve třetí $y_{3,1}, y_{3,2}, \ldots, y_{3,16}$ a ve čtvrté $y_{4,1}, y_{4,2}, \ldots, y_{4,16}$. Výstup z poslední rundy je zároveň šifrový text, vznikne tak dalších 16 rovnic, které dále použijeme na eliminaci některých neznámých. Vytvoření rovnic pro jednotlivé rundy je naimplementováné v souboru `Utoky.nb` vždy v podsekci `Vytvoření rovnic` u příslušné rundy.

V Mathematice se neúplně dobře pracuje s indoxovanými proměnnými, proto budeme používat dvojité označení:

- k_i nebo ki bity klíče ($i = 1, \cdots, 16$),
- y_i nebo yi výstupy z S-boxů 1.rundy ($i = 1, \cdots, 16$),
- $y_{2,i}$ nebo $y2i$ výstupy z S-boxů 2.rundy ($i = 1, \cdots, 16$),
- $y_{3,i}$ nebo $y3i$ výstupy z S-boxů 3.rundy ($i = 1, \cdots, 16$),
- $y_{4,i}$ nebo $y4i$ výstupy z S-boxů 4.rundy ($i = 1, \cdots, 16$),
- l_i nebo li výstupy z S-boxů při KeySchedule 1.rundy ($i = 1, \cdots, 8$),
- $l_{2,i}$ nebo $l2i$ výstupy z S-boxů při KeySchedule 2.rundy ($i = 1, \cdots, 8$),
- $l_{3,i}$ nebo $l3i$ výstupy z S-boxů při KeySchedule 3.rundy ($i = 1, \cdots, 8$),
- $l_{4,i}$ nebo $l4i$ výstupy z S-boxů při KeySchedule 4.rundy ($i = 1, \cdots, 8$).

Při linearizaci budeme převádět monomy $a \cdot b$ na $m_{a,b}$, případně $a \cdot b \cdot c$ na $m_{a,b,c}$, kde a, b, c jsou proměnné soustavy, kterou linearizujeme. Někdy budeme také převádět zpátky linearizované proměnné $m_{a,b,c}$ na termy abc.

Při generování rovnic používáme funkci `runda[vstup,nové proměnné na výstup z S-boxu, číslo rundy]` (a její podfunkce), která ukládá do globální proměnné *soustava* rovnice z příslušné rundy. Rovnice pro vzniklé při expanzi klíče si ukládáme do globální proměnné *klicoverce*.



Obrázek 3.1: Schéma průchodu proměnných první rundou šifry Baby Rijndael při získávání rovnic

3.3 Útok na jednu rundu

Útok probíhá obdobně jako v [10]. Simulujeme průchod jednou rundou s klíčem $\{k_1, k_2, k_3, \ldots, k_{16}\}$. Dostaneme $6 \cdot 21 = 126$ rovnic se 40-ti neznámými popisujících S-boxy. Šifrový text můžeme použít na eliminování některých neznámých. Chceme, aby výstup z poslední operace byl roven šifrovému textu

$$\{ct_1, ct_2, ct_3, ct_4, ct_5, ct_6, ct_7, ct_8, ct_9, ct_{10}, ct_{11}, ct_{12}, ct_{13}, ct_{14}, ct_{15}, ct_{16}\}.$$

Získáme tak rovnice:

$$\begin{aligned}
k_1 + l_1 + y_1 + y_{15} + y_{16} + y_3 &= ct_1, \\
k_2 + l_2 + y_1 + y_{16} + y_2 + y_4 &= ct_2, \\
k_3 + l_3 + y_1 + y_{13} + y_2 + y_3 &= ct_3, \\
1 + k_4 + l_4 + y_{14} + y_{15} + y_{16} + y_2 + y_4 &= ct_4, \\
k_1 + k_9 + l_1 + y_{11} + y_7 + y_8 + y_9 &= ct_5, \\
k_{10} + k_2 + l_2 + y_{10} + y_{12} + y_8 + y_9 &= ct_6, \\
k_{11} + k_3 + l_3 + y_{10} + y_{11} + y_5 + y_9 &= ct_7, \\
1 + k_{12} + k_4 + l_4 + y_{10} + y_{12} + y_6 + y_7 + y_8 &= ct_8, \\
k_5 + l_5 + y_{13} + y_{15} + y_3 + y_4 &= ct_9, \\
k_6 + l_6 + y_{13} + y_{14} + y_{16} + y_4 &= ct_{10}, \\
k_7 + l_7 + y_1 + y_{13} + y_{14} + y_{15} &= ct_{11}, \\
k_8 + l_8 + y_{14} + y_{16} + y_2 + y_3 + y_4 &= ct_{12}, \\
k_{13} + k_5 + l_5 + y_{11} + y_{12} + y_5 + y_7 &= ct_{13}, \\
k_{14} + k_6 + l_6 + y_{12} + y_5 + y_6 + y_8 &= ct_{14}, \\
k_{15} + k_7 + l_7 + y_5 + y_6 + y_7 + y_9 &= ct_{15}, \\
k_{16} + k_8 + l_8 + y_{10} + y_{11} + y_{12} + y_6 + y_8 &= ct_{16},
\end{aligned} \tag{3.5}$$

To je 16 lineárních rovnic pro 16 neznámých, můžeme tedy vyjádřit y_i a dosadit je do zbylých rovnic. Tak snížíme počet neznámých v soustavě rovnic *soustava* na 24.

3.3.1 Útok pomocí XL algoritmu

Soustavu 126 rovnic s 24 neznámými vzniklou z první rundy si označíme S a vyřešíme ji metodou XL. Stanovíme nejprve $D = 3$. Každou rovnici postupně přenásobíme součiny proměnných do stupně $D - 2 = 1$, tedy jen samotnými proměnnými. Tímto postupem vznikne 3456 rovnic s 24 neznámými. Vzniklou soustavu linearizujeme a dostaneme 2324 proměnných, Gaussovou eliminací získáme dvě řešení lineární soustavy, z nich uděláme lineární kombinaci a dosadíme do S. Zjistili jsme, že vyhovují dvě řešení, z nichž jedno je to zadané (klíč, který jsme "zapomněli").

3.3.2 Útok pomocí XSL algoritmu

Soustavu 126 rovnic s 24 neznámými vzniklou z první rundy si označíme S a vyřešíme ji metodou XSL. Rovnice vzniklé z jednoho S-boxu postupně přenásobíme všemi proměnnými použitými v ostatních S-boxech. To provedeme pro každý S-box. Tímto způsobem získáme 3456 různých (ne nutně lineárně nezávislých) rovnic. Provedeme na ně linearizaci a vznikne lineární soustava rovnic s 2324 neznámými, označíme si ji N. Po Gaussově eliminaci zjistíme, že řešíme stejnou rovnici jako při XL útoku. Tedy i v tomto případě dostaneme jako řešení dva klíče.

3.4 Útoky na dvě rundy

Vezmeme otevřený text, klíč $k_1, k_2, \ldots, k_{16}$, simulujeme šifrování a získáme $12 \cdot 21 = 252$ rovnic se 64 neznámými. Podobně jako při jedné rundě se můžeme pomocí šifrového textu (příslušejícího druhé rundě) zbavit neznámých $y_{2,1}, y_{2,2}, \ldots, y_{2,15}, y_{2,16}$. Získáme tak 252 rovnic se 48 neznámými. Pro přesnější představu rozložení neznámých v rovnicích si vytvoříme tabulku 3.1. Jeden řádek v ní odpovídá rovnicím z jenoho S-boxu a X má ve sloupcích, jejichž proměnné rovnice obsahují. První rundě přísluší *kus[[1]]*,..., *kus[[4]]*, druhé rundě pak *kus[[7]]*,..., *kus[[10]]*, nakonec *kus[[5]]*, *kus[[6]]* (respektive *kus[[11]]*, *kus[[12]]*) přísluší S-boxům z expanze klíče pro 1. (respektive 2.) rundu.

	k_1	y_1	k_5	y_5	k_9	y_9	k_{13}	y_{13}	l_1	l_5	l_{21}	l_{25}
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	k_4	y_4	k_8	y_8	k_{12}	y_{12}	k_{16}	y_{16}	l_4	l_8	l_{24}	l_{28}
kus[[1]]	X	X										
kus[[2]]			X	X								
kus[[3]]					X	X						
kus[[4]]							X	X				
kus[[5]]							X		X			
kus[[6]]					X					X		
kus[[7]]			X				X		X	X	X	
kus[[8]]	X				X				X			X
kus[[9]]	X	X	X					X	X	X	X	X
kus[[10]]		X	X		X		X	X		X	X	X
kus[[11]]	X			X	X	X	X		X		X	X
kus[[12]]	X		X	X		X	X		X	X	X	X

Tabulka 3.1: Tabulka rozmístění proměnných v soustavě pro 2 rundy

3.4.1 Útoky pomocí XL a XSL algoritmu

Zkusíme útok pomocí XL algoritmu. Zde bude počet neznámých 48. Po přenásobení termy z jiných S-boxů, bychom dostali soustavu 13824 rovnic. Po linearizaci se zvýší počet neznámých v lineární soustavě na 18472. To už se nevejde do paměti a Wolfram Mathematica není schopná takto velkou matici vyřešit přímo. Při XSL algoritmu (kdy násobíme pouze proměnnými, které se v daném S-boxu nevyskytují) dostaneme 8736 rovnic se 17256 neznámými. Ani z této soustavy nedostaneme žádné řešení, protože jedna Gaussova eliminace trvá několik desítek minut na běžném pc a na T' metodu bychom jich potřebovali provést odhadem 150.

3.4.2 Útok pomocí XSL algoritmu bez převádění na matici

Funkce `pripravnacko` i `eliminujdleporadi` převádí soustavu rovnic na matici, se kterou pak pracují. To je ale u velkých soustav moc paměťově (i časově) náročné, pokusíme se tedy tomuto převádění na matici vyhnout nebo ho alespoň používat minimálně.

Tento útok začneme jako předchozí, ale vyhneme se převádění soustavy na matici, respektive ho provedeme pouze jednou. Zde je počet neznámých 64, po snížení 48. Po přenásobení termy z jiných S-boxů bychom dostali soustavu 8736 rovnic. Po linearizaci je počet neznámých v lineární soustavě 17256. Můžeme jdnou použít funkci `eliminujdleporadi` a provést Gaussovu eliminaci (s řazením proměnných sestupně podle abecedy). Pokud bychom získali soustavu, která se už nevejde do paměti a Wolfram Mathematica by jí nebyla schopná vyřešit přímo. Mohli bychom se ale pokusit provést Gaussovu eliminaci na menší části této soustavy, a poté z nich zkonstruovat celou soustavu v odstupňovaném tvaru (resp. její matice by měla odstupňovaný tvar, ale v tomto případě pracujeme se soustavou rovnic a ne s její reprezentací v matici).

Rovnice v výsledné soustavě po eliminaci budou lineárně nezávislé, stačí tedy, aby měla soustava stejný počet rovnic jako proměnných, a dostaneme řešení.

Pokud máme málo lineárně nezávislých rovnic, můžeme použít metodu T' na vytvoření nových (převážně) lineárně nezávislých rovnic bez přidávání nových proměnných následovně. Mějme řazení proměnných T sestupně podle abecedy:

$$m_{y_7,y_8,y_9}, m_{y_6,y_8,y_9}, m_{y_6,y_7,y_9}, \ldots, m_{k_1,k_{10},k_{11}}, m_{y_8,y_9}, \ldots$$
$$\ldots, m_{k_{10},k_1}, y_9, y_8, \ldots, k_{12}, k_{11}, k_{10}, k_1.$$

Soustava rovnic má při zachování tohoto řazení termů/proměnných matici v odstupňovaném tvaru.

V metodě T' je potřeba pro každou zvolenou neznámou x_1 vypočítat odstupňovaný tvar celé soustavy, abychom našli rovnice, které obsahují pouze termy z T_{x_1}. V našem případě by to znamenalo pro každou z 36 proměnných

počítat odstupňovaný tvar soustavy velikosti cca 10000. Eliminovat takovou soustavu, by trvalo moc dlouho nebo bychom ji museli dělit na podmatice, ty diagonalizovat zvlášť, a poté je spojovat dohromady. Tomu se můžeme vyhnout, když si uvědomíme, kde se nachází jednotlivé množiny T_x v setříděné T.

Množina proměnných T_{k_1} se nachází v řazení T úplně na konci, protože:

- všechny termy stupně 1 i 2 přenásobené k_1 jsou v T,
- termy stupně 3, které po přenásobení k_1 zůstanou v T, jsou právě ty, co už k_1 obsahují. Takové jsou v řazení T na konci řady termů stupně 3.

Proto nemusíme soustavu nijak upravovat a máme požadovaný tvar pro T_{k_1}. Nachází-li se na konci soustavy rovnice s termy pouze z T_{k_1}, můžeme je přenásobit proměnnou k_1 a přidat k soustavě.

Množina $T_{k_{10}}$ se částečně nachází před T_{k_1} a některé prvky s ní má společné. Začátek setříděné T neobsahuje $T_{k_{10}}$, takže k tomu, aby byla celá soustava v odstupňovaném tvaru podle $T_{k_{10}}$, stačí diagonalizovat část soustavy od rovnice, která jako první obsahuje prvek z $T_{k_{10}}$. Nachází-li se na konci soustavy rovnice s termy pouze z $T_{k_{10}}$, můžeme je opět přenásobit proměnnou k_{10} a přidat k soustavě.

Podobně postupujeme pro další proměnné. Pro prvních pár proměnných je část soustavy, kterou chceme diagonalizovat, dostatečně malá. Když diagonalizujeme jen na tuto část, vyjde výsledná soustava stejně jako diagonalizace celé soustavy.

Tento útok má moc velké paměťové i časové nároky a v rozumném čase jej nebylo možno dokončit.

3.4.3 T' metoda na část soustavy

Jak jsme viděli v předchozí sekci, vznikne i pro XSL útok na dvě rundy příliš mnoho neznámých. Jeden ze způsobů, jak se tomuto vyhnout, je ukryt v T' metodě. Při jejím použití se totiž počet proměnných nezvětšuje, ale počet rovnic ano. Zkusíme začít s rovnicemi odpovídajícími jednomu S-boxu.

Každý S-box v první rundě nám dává 21 rovnic s 8 neznámými, které dávájí po linearizaci (maximálně) 36 proměnných. Množina $T' = T_x$ má 15 prvků pro každé x. Po Gaussově eliminaci (dle T_x) vznikne pouze 1 rovnice, která obsahuje jen termy z T_x. Když ji ovšem vynásobíme proměnnou x, rovnice se vynuluje modulo 2. Tímto způsobem nejsme tedy schopni získat žádné další rovnice. Musíme buď provést útok XSL, nebo přidat nějaké rovnice z jiných S-boxů.

Zkusíme tedy přidat rovnice se společnými proměnnými. V tabulce 3.1 si všimneme, že dvojice rovnic *kus[[4]]* a *kus[[5]]* nebo *kus[[3]]* a *kus[[6]]* obsahuje pouze 12 proměnných. Vezmeme si například *kus[[4]]* a *kus[[5]]* (ozn.P). Tato soustava P bude mít 42 rovnic s 12 proměnnými. Po linearizaci získáme 64

proměnných. Množina $T' = T_x$ má 22 prvků pro každé x, které bylo v obou množinách rovnic, pro ostatní x má $T' = T_x$ 15 prvků. Po eliminaci získáme 42 rovnic, z kterých (při $x = k_{13}$) pouze 4 obsahují jen termy T', ale ani jejich přenásobením nezískáme žádné další rovnice (a zůstaneme na $64 - 42 = 22$ chybějících rovnicích). Při vhodném vyjádření můžeme ze soustavy dostat rovnici:

$$k_{13} + l_2 + l_3 + l_4 + y_{14} + y_{15} + y_{16} = 0,$$

z ní můžeme vyjádřit některou z proměnných a dosadit ji do zbylých rovnic.

Počty rovnic pro dosazení za různé proměnné (po doplnění T' metodou) a počty proměnných v soustavě P po dosazení za příslušnou proměnnou jsou zobrazeny v následující tabulce.

	k_{13}	l_2	l_3	l_4	y_{14}	y_{15}	y_{16}
počet rovnic	50	48	48	48	48	48	48
počet proměnných	65	63	63	63	63	63	63

Je vidět, že ať si vybereme jakoukoli proměnnou, stále nám bude chybět 15 rovnic.

Pokud budeme provádět XSL metodu a přenásobíme všechny rovnice jednoho S-boxu všemi proměnnými druhého S-boxu, získáme po linearizaci a přidání rovnic pomocí XSL 215 rovnic s 230 proměnnými. Stále nám tedy bude 15 rovnic chybět. Můžeme zkusit soustavu P vyřešit pomocí funkce LinearSolve a metody popsané v 2.3.1: získáme 15 lineárně nezávislých řešení soustavy P, jejichž lineární kombinací bychom dostali všechna řešení soustavy P. Z těchto 15ti lineárně nezávislých řešení můžeme vzít pouze souřadnice odpovídající proměnným stupně jedna, a tak dostaneme 12 různých vektorů. Vyzkoušíme, které jejich lineární kombinace (2048) řeší i původní nelineární soustavu, a získáme tak všech 16 řešení této původní nelineární soustavy. Ty (postupně) dosadíme do ostatních rovnic soustavy S a získáme řešení (pomocí funkce Solve) celé soustavy pro 2 rundy.

Těchto 16 řešení jsou přesně vektory, které bychom dostali postupným dosazováním všech možných čtveřic bitů $k_{13}, k_{14}, k_{15}, k_{16}$. Tento útok není tedy lepší než postupné dosazování 4 bitů klíče do soustavy S.

3.4.4 Vylepšení předchozího útoku

Výsledek předchozího útoku nebyl moc dobrý, můžeme ho ale vylepšit. Pomocí XSL algoritmu na P jsme získali po linearizaci soustavu 215 rovnic s 230 proměnnými. Pokud tato soustava obsahuje aspoň dvě rovnice tvaru $m_{a,b,c} = 0$, $m_{a,b,d} = 0$ pro nějaké původní proměnné a, b, c, d, můžeme použít metodu popsanou v 2.3.2. V tomto případě dostaneme rovnice

$$m_{k_{14},k_{13},y_{15}} = 0,\ m_{k_{14},k_{13},l_3} = 0$$

a pravděpodobnost, že $m_{k_{14},k_{13}} = 0$, je $92,3\%$. Zvolíme tedy $m_{k_{14},k_{13}} = 0$ a dořešíme soustavu P. Získáme tak 11 lineárně nezávislých řešení lineární

soustavy, z nichž máme 11 různých redukovaných vektorů. Lineární kombinací těchto vektorů získáme 2048 možných řešení. Po dosazení do soustavy P zjistíme, že 12 řešení ji řeší. Po dosazení do soustavy S získáme jeden klíč (zkontrolujeme, že se jedná o klíč, jímž jsme šifrovali). Pokud bychom nedošli k žádnému klíči, vrátíme se k soustavě před dosazením za $m_{k_{14},k_{13}}$, dosadíme $m_{k_{14},k_{13}} = 1$ (tedy $k_{14} = 1$ i $k_{13} = 1$) a dořešíme.

Podobně můžeme útočit na rovnice vzniklé z třetího S-boxu první rundy a druhého S-boxu z expanze klíče.

3.5 Útok na čtyři rundy

Čtyři rundy Baby Rijndaelu mají 16 šifrovacích S-boxů a 8 S-boxů použitých při expanzi klíče, celkem tedy dostaneme $24 * 21 = 504$ rovnic s 96 neznámými (tuto soustavu si označíme S a její linearizaci N). Přesnější uspořádání proměnných je znázorněno v tabulce 3.2.

Provádět XL nebo XSL na celou soustavu nebo jednu rundu nemá smysl, když víme, že i pro dvě rundy už byly soustavy při těchto útocích moc velké. Musíme tedy začít s nějakou menší částí.

3.5.1 XSL na část soustavy

Všimneme si, že stejně jako u soustavy pro dvě rundy, rovnice příslušející poslednímu S-boxu první rundy (*kusy[[4]]*) a prvnímu S-boxu z expanze klíče (*kusy[[17]]*) obsahují pouze 12 proměnných na 42 rovnic. Soustavu těchto rovnic si označíme P (stejné množství rovnic a proměnných ještě obsahuje sjednocení *kusy[[3]]* s *kusy[[18]]*). Tato soustava P je úplně stejná jako soustava počítaná pro dvě rundy, můžeme tedy použít 16 vektorů řešení, které nám vyšly při útoku na dvě rundy. Tyto vektory řešení soustavy P odpovídají části proměnných soustavy N. Mohli bychom tak za tyto proměnné postupně dosazovat do soustavy N a zkoumat, v kterých případech má soustava N řešení.

Po dosazení má N 84 lineárních proměnných, po použití T' metody získáme 480 rovnic s 2096 neznámými. Zkusíme ji vyřešit pomocí LinearSolve, ale vyjde nám 1616 řešení lineární soustavy N. Po zkrácení vektorů řešení vznikne 83 vektorů, tedy bychom museli vyzkoušet 2^{83} vektorů (respektive všech 2^{12} možných kombinací k_j, které určují ostatní proměnné), což je stejně pracné, jako útok hrubou silou na tuto soustavu.

Pokud bychom chtěli nižší složitost, můžeme znovu vyřešit pouze část soustavy a tu poté dosadit do celé soustavy (metoda je popsaná v 2.3.1). Jak v původní soustavě N, tak i při dosazených $k_{13}, \ldots, k_{16}, l_1, \ldots, l_4, y_{13}, \ldots, y_{16}$, je v rovnicích pro 3. S-box první rundy (*kusy[[3]]*) a 2. S-box z expanze klíče (*kusy[[18]]*) pouze 12 proměnných na 42 rovnic (tuto soustavu si označíme Q).

	k_1 y_1 k_5 y_5 k_9 y_9 k_{13} y_{13} k_4 y_4 k_8 y_8 k_{12} y_{12} k_{16} y_{16}	l_1 l_5 l_4 l_8	y_{21} y_{25} y_{29} y_{213} y_{24} y_{28} y_{212} y_{216}	l_{21} l_{25} l_{24} l_{28}	y_{31} y_{35} y_{39} y_{313} y_{34} y_{38} y_{312} y_{316}	l_{31} l_{35} l_{34} l_{38}	l_{41} l_{45} l_{44} l_{48}
kusy[[1]]	X X						
kusy[[2]]	X X						
kusy[[3]]	X X						
kusy[[4]]	**X** **X**						
kusy[[17]]	**X**	**X**					
kusy[[18]]	X	X					
kusy[[5]]	X X X	X	X				
kusy[[6]]	X X X	X	X				
kusy[[7]]	X X X X	X	X				
kusy[[8]]	X X X X	X	X				
kusy[[19]]	X X	X		X			
kusy[[20]]	X X	X		X			
kusy[[9]]	X X	X	X X	X	X		
kusy[[10]]	X X	X	X X	X	X		
kusy[[11]]	X X		X X	X	X		
kusy[[12]]	X X		X X	X	X		
kusy[[21]]	X			X		X	
kusy[[22]]	X			X		X	
kusy[[13]]	X X	X		X	X X	X	X
kusy[[14]]	X X	X		X	X X	X	X
kusy[[15]]	X X	X		X	X X	X	X
kusy[[16]]	X X	X		X	X X	X	X
kusy[[23]]	X X	X				X	X
kusy[[24]]	X X	X				X	X

Tabulka 3.2: Tabulka rozmístění proměnných v soustavě pro 4 rundy

Chceme tedy vyřešit soustavu Q. Použijeme T' metodu, abychom získali více rovnic. Při jejím použití nám vznikne lineární rovnice

$$y_{12} = k_{12} + l_5 + l_7 + l_8 + y_9 + y_{11}.$$

Dosadíme do Q za y_{12}. Vyřešením soustavy Q pomocí LinearSolve získáme ?15 řešení, ty se po zkrácení redukují na ?8 řešení. Lineární kombinaci těchto vektorů (postupně) dosadíme do Q (v kvadratické formě) a zjistíme, že ?10 z nich ji řeší. Pro každé z nich dopočítáme y_{12}. Máme tedy ?16 různých řešení pro $k_9, k_{10}, k_{11}, k_{12}, l_5, l_6, l_7, l_8, y_9, y_{10}, y_{11}, y_{12}$. Ty můžeme postupně dosadit do celé soustavy. Pro každé řešení soustavy P může nastat libovolné z řešení soustavy Q, celkem musíme vyzkoušet $16 \cdot 16 = 256 = 2^8$ řešení. Dostaneme se tedy ke složitosti srovnatelné s dosazením 8 bitů klíče.

3.5.2 Útok s použitím pravděpodobnosti

Použijeme pravděpodobnostní útok na soustavu P, stejný jako při dvou rundách. Dosadíme tedy za $m_{k_{14},k_{13}}$ nulu. Pravděpodobnost, že je to správně, je $92,3\%$. Kdybychom jednotlivá řešení (celkem je jich 12) dosazovali do celé soustavy, stále budeme mít moc proměnných a málo rovnic. Můžeme ale provést ještě pravděpodobnostní útok na soustavu Q rovnic příslušející 3. S-boxu první rundy a 2. S-boxu z operace KeySchedule. Stejně jako při útoku na dvě rundy dostaneme, že $m_{k_{10},k_9} = 0$ s pravděpodobností $92,3\%$. Dosadíme m_{k_{10},k_9} nulu do celé soustavy N.

Celková pravděpodobnost, že můžeme zároveň dosadit $m_{k_{14},k_{13}} = 0$ i $m_{k_{10},k_9} = 0$ je $0,923 \cdot 0,923 = 0,851929$, což můžeme považovat stále za dostatečně velkou pravděpodobnost.

Jako při dvou rundách bychom chtěli pomocí funkce Solve zjistit, zda má soustava N řešení, když do ní postupně dosazujeme vektory řešení soustav P a Q. To ovšem v tomto případě nebude fungovat, protože je soustava stále moc velká (a má moc proměnných). Můžeme ale dopočítat rovnice pomocí metody T' a když vznikne rovnice jen s lineárními monomy, tak za jeden z nich dosadit do soustavy N (pomocí funkce `vytvorrceadosazuj`). Pokud nedojdeme rovnou k lineárnímu řešení, použijeme Solve na tuto menší soustavu. Nakonec najdeme námi zadaný klíč.

3.6 Implementace

Jednotlivé útoky jsou naprogramovány v souboru `Utoky.nb` v příslušných sekcích. Používáme při nich pomocné funkce:

- `eliminujdleporadi[soustava lineárních rovnic, proměnné ve zvoleném pořadí]` převede vstupní soustavu na matici a provede Gaussovu eliminaci při zachování pořadí proměnných (ve vstupu musí být všechny proměnné, které vstupní soustava obsahuje)

- `pripravnakcko[vstup, kcko]` zjistí množinu T_{kcko} (z T' metody) a provede Gaussovu eliminaci s pořadím proměnných $T \setminus T_{kcko}$, T_{kcko}
- `vytvorrceadosazuj[soustava]` provádí T' metodu se vstupem *soustava*. Když navíc najde rovnici pouze s lineárními členy, tak ji dosadí. Vrací upravenou soustavu a množinu dosazených rovnic (pravidel). Používá funkci `pripravnakcko[soustava, proměnná]` pro různé proměnné.
- `opatrnedosad[soustava, pravidla]` převede oba vstupy do nelineárního tvaru, dosadí do *soustava pravidla*, tu linearizuje a vrátí jako výstup
- `LK[množina vektorů pravidel]` vrátí všechny lineární kombinace vstupních vektorů jako množinu vektorů pravidel
- `vyres[soustava]` pomocí převedení na matici, operace LinearSolve a funkce `LK` vyřeší lineární soustavu, redukuje vektory řešení pouze na souřadnice původních lineárních proměnných, vytvoří všechny lineární kombinace těchto redukovaných řešení a vrátí ty z nich, které řeší nelineární soustavu

Závěr

V této práci jsme prováděli útoky se znalostí otevřeného i šifrového textu na šifru Baby Rijndael (zjednodušenou verzi nejpoužívanější symetrické blokové šifry AES). Nejprve jsme vyjádřili šifru Baby Rijndael pomocí soustavy rovnic stupně maximálně dva. Prozkoumali jsme možnosti řešení této soustavy nelineárních rovnic pomocí známých algoritmů XL, XLS a T'.

Pommocí XL algoritmu se nám povedlo zjistit klíč(e) pro jednu rundu. Pro dvě rundy jsme využili XL a XSL algoritmy pouze na část soustavy, nalezli 16 řešení (vektorů) této částečné soustavy. Poté jsme byli schopni najít klíč zkoušením těchto 16ti částečných řešení dosazováním do původní soustavy. Při využití dodatečně navržených pravděpodobnostních postupů se povedlo omezit počet řešení částečné soustavy na 12 vektorů, ale hledaný klíč (řešení původní soustavy) se v nich nacházel pouze s pravděpodobností $92,3\%$. Útoky pomocí algoritmů XL a XSL na celou soustavu byly příliš prostorově náročné.

K nalezení klíče pro čtyři rundy jsme postupovali obdobně jako při nalezení klíče pro dvě rundy. Po vhodné volbě částí soustavy a použití XL a XSL algoritmů jsme nalezli 256 řešení částečné soustavy. Pro řešení původní soustavy tak bylo potřeba vyzkoušet 256 možných řešení. S využitím pravděpodobnostních postupů se povedlo omezit počet řešení na 144, ale hledaný klíč se v nich nacházel pouze s pravděpodobností $85,1929\%$.

Jakým způsobem by se tyto útoky daly převést na AES? V sekci 1.5 bylo ukázáno, že i S-box šifry AES lze převést na rovnice stupně maximálně dva. Pro každý S-box by nám vzniklo 39 rovnic s 16 proměnnými (po linearizaci bychom dostali 136 proměnných). Pro jednu rundu bychom měli $39 \cdot 8 + 39 \cdot 4 = 468$ rovnic s $128 + 128 + 64 = 320$ proměnnými, po snížení 192 proměnných. Na XL/XSL útok je pravděpodobně toto množství rovnic a proměnných příliš velké.

Literatura

[1] Róbert Lórencz, C.: Pokročilá kryptologie: Symetrická kryptografie, 2013. Dostupné z: https://edux.fit.cvut.cz/archive/B142/MI-KRY/_media/lectures/02/prednaska2.pdf

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

[3] Jureček, M.: Pokročilá kryptologie: Algebraická kryptoanalýza, 2013. Dostupné z: https://edux.fit.cvut.cz/archive/B142/MI-KRY/_media/lectures/06/ac.pdf

[4] Courtois, N. T.; Pieprzyk, J.: Cryptanalysis of Block Cipher with Overdefined System of Equations. In *Advances in Cryptology ASIACRYPT 2002*, Springer Berlin Heidelberg, 2002, s. 267–287. Dostupné z: http://link.springer.com/chapter/10.1007/3-540-36178-2_17

[5] NOVER, H.: Algebraic Cryptanalysis of AES: An Overview. Dostupné z: http://www.math.wisc.edu/~boston/nover.pdf

[6] Stanovský, D.; Barto, L.: *Počítačová algebra*. Praha: MATFYZPRESS, první vydání, 2011, ISBN 978-80-7378-167-5.

[7] Bergman, D.: A Description of Baby Rijndael, 2005. Dostupné z: http://orion.math.iastate.edu/cbergman/crypto/homework/babyr/babyr.pdf

[8] Kokeš, J.: *Kryptoanalýza šifry Baby Rijndael*. Diplomová práce, České vysoké učení technické v Praze, Fakulta informačních technologií, Praha, 2013. Dostupné z: https://dip.felk.cvut.cz/browse/pdfcache/kokesjo1_2013dipl.pdf

[9] Daemen, J.; Rijmen, V.: *The Design of Rijndael: AES-The Advanced Encryption Standard*. Springer-Verlag Berlin Heidelberg, 2002. Dostupné z: http://www.springer.com/jp/book/9783540425809

[10] Kleiman, E.: *The XL and XSL attacks on Baby Rijndael*. Diplomová práce, Iowa State University, Ames, Iowa, 2005. Dostupné z: http://orion.math.iastate.edu/dept/thesisarchive/MS/EKleimanMSSS05.pdf

[11] Courtois, N.; Klimov, A.; Patarin, J.; aj.: Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations. In *Advances in Cryptology EUROCRYPT 2000*, Springer Berlin Heidelberg, 2000, s. 392–407. Dostupné z: http://link.springer.com/chapter/10.1007%2F3-540-45539-6_27

Příloha A

Seznam použitých zkratek

AES Advanced encryption standard

XL eXtended Linearization

XSL eXtended Sparse Linearization

SPN Substitution-Permutation Network

Příloha **B**

Obsah přiloženého CD

- `AK` ..... adresář s implementacemi
  - `BabyRijndael.nb` ..... implementace šifry Baby Rinjdael
  - `Utoky.nb` ..... implementace útoků
  - `AES.nb` ..... výpočet rovnic pro S-box AESu
- `text` ..... text práce
  - `DP.pdf` ..... text této práce ve formátu PDF
  - `DP.ps` ..... text této práce ve formátu PS
  - `DP.tex` ..... zdrojová forma práce ve formátu LaTeX
    - `obrazky` ..... zdrojová forma obrázků ve formátu metapost