ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Asymetrický šifrovací algoritmus McEliece
Student:	Bc. Vojtěch Myslivec
Vedoucí:	prof. Ing. Róbert Lórencz, CSc.
Studijní program:	Informatika
Studijní obor:	Počítačová bezpečnost
Katedra:	Katedra počítačových systémů
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

Prostudujte asymetrický šifrovací algoritmus McEliece založený na binárních Goppa kódech. Proveďte rešerši existujících kryptoanalýz algoritmu McEliece a jeho variant. Zvažte metody zabývající se zkrácením velikosti klíčů. Implementujte šifrovací a dešifrovací algoritmy a změřte jejich výpočetní časovou a prostorovou náročnost v závislosti na velikosti klíče.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

prof. Ing. Róbert Lórencz, CSc.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 2. února 2016

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

Diplomová práce

Asymetrický šifrovací algoritmus McEliece

Bc. Vojtěch Myslivec

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

3. dubna 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 3. dubna 2016 ....

Odkaz na tuto práci

Myslivec, Vojtěch. *Asymetrický šifrovací algoritmus McEliece.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

V několika větách shrňte obsah a přínos této práce v češtině. Po přečtení abstraktu by se čtenář měl mít čtenář dost informací pro rozhodnutí, zda chce Vaši práci číst.

Klíčová slova McEliece, asymetrická kryptografie, postkvantová kryptografie, binární Goppa kódy, konečná tělesa, polynomy, Wolfram Mathematica

Abstract

Sem doplňte ekvivalent abstraktu Vaší práce v angličtině.

Keywords McEliece, public-key cryptography, post-quantum cryptography, binary Goppa codes, finite fields, polynomy, Wolfram Mathematica

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Tato práce se zabývá asymetrickým kryptosystémem *McEliece*. Mezi největší přednosti tohoto systému patří jeho odolnost vůči kvantovým počítačům a je tak jedním z vhodných kandidátů pro asymetrickou kryptografii pro postkvantovou dobu.

V prvních kapitolách této práce jsou popsány nezbytné primitivy z oblasti matematiky a teorie kódování, které jsou potřeba pro pochopení a použití kryptosystému McEliece. Jedná se především o počítání s *konečnými tělesy* a *polynomy* (kapitola 1) a binární *Goppa* kódy (kapitola 2).

Kryptosystému *McEliece* se věnuje kapitola 3. Kromě základního popisu generování klíčů a algoritmů pro šifrování a dešifrování je probráno i *Niederreiterovo* schéma – „úprava“ kryptosystému *McEliece* pro získání *digitálního podpisu*. Jsou ukázány slabiny, nevýhody i možné útoky na kryptosystém McEliece a též zmíněna praktická varianta systému odolná vůči těmto aspektům.

V poslední části práce je probrána implementace kryptosystému *McEliece* v softwaru *Wolfram Mathematica* včetně změřených časových složitostí (kapitola 4),.

Kapitola **1**

Obecná algebra

V kapitole jsou probrány definice a algoritmy nutné pro práci s *konečnými tělesy* a *polynomy* nad konečným tělesem. V práci se předpokládá základních znalostí z oblasti *algebry*. Pro tato témata je doporučena literatura [8, 7, 5, 6, 2] (kde lze též najít většinu důkazů následujících vět).

1.1 Základní termíny

Pro ujasnění je uvedena definice tělesa:

Definice 1 (Těleso) *Nechť M je neprázdná množina a $+$ a $\cdot$ binární operace*[1]*. Struktura $T = (M, +, \cdot)$ se nazývá těleso, pokud platí*

1. $(M, +)$ *je* komutativní grupa *(nazývána* aditivní*)*
2. $(M \setminus \{0\}, \cdot)$[2] *je* grupa *(nazývána* multiplikativní*)*
3. *Platí (levý i pravý)* distributivní zákon*:*

$$\forall a, b, c \in M : \big(a(b + c) = ab + ac\big) \wedge \big((b + c)a = ba + ca\big)$$

Těleso, které má konečný počet prvků, se nazývá *konečné těleso*.

Věta 1 *Nechť T je* konečné těleso*, pak jeho počet prvků (*řád*) je p^n, kde p je prvočíslo a $n \in \mathbb{N} \wedge n \geq 1$.*

Číslo p se nazývá *charakteristika*. Navíc platí, že *všechna konečná tělesa* se stejným počtem prvků jsou navzájem *izomorfní*. *Konečné těleso* řádu p^n je tedy dále označováno jako $GF(p^n)$ (z anglického *Gallois field*, dle francouzského matematika *Évariste Galois*).

[1] Pro zjednodušení zápisu je $\cdot$ často vynecháváno.
[2] Prvek *0* je *nulový* (*neutrální*) prvek *aditivní grupy*.

1.2 Reprezentace prvků

Jak bude ukázáno dále, je vhodné prvky tělesa $GF(p^n)$ reprezentovat jako *polynomy* s koeficienty z množiny $\mathbb{Z}_p = \{0, 1, \ldots, p-1\}$, tedy prvek $a \in GF(p^n)$ lze zapsat:

$$A(x) = \sum_{i=0}^{n-1} a_i x^i, a_i \in \mathbb{Z}_p$$

O takovém polynomu říkáme, že je to *polynom nad tělesem $GF(p)$ (řádu maximálně $n-1$)*. Na prvek a je též možné se dívat jako na vektor či n-tici koeficientů a_i:

$$A(x) \cong a \cong (a_{n-1}a_{n-2} \ldots a_0) \cong a_{n-1}a_{n-2} \ldots a_0$$

V této práci se mezi těmito reprezentacemi prvků nadále volně přechází, jak bude v daném kontextu potřeba potřeba[3].

1.3 Operace v tělese $GF(p^n)$

V následujících sekcích jsou probrány operace potřebné pro počítání s tělesy $GF(p^n)$. Konkrétní zvolené algoritmy a jejich implementace je detailně popsána v kapitole 4.

1.3.1 Sčítání

Sčítání v tělese $GF(p^n)$ je definováno stejně jako sčítání polynomů, s tím, že sčítání jednotlivých koeficientů je prováděno *modulo p* (v tělese $GF(p)$:

$$A(x) + B(x) = \sum a_i x^i + \sum b_i x^i = \sum |a_i + b_i|_p \, x^i$$

1.3.2 Násobení

Násobení v tělese $GF(p^n)$ nelze provádět „po složkách“, jako je tomu u sčítání. U takto definované operace by většina prvků neměla (multiplikativní) *inverzi* a nejednalo by se tak o *těleso*.

Při násobení prvků se opět využije jejich reprezentace pomocí polynomů. Výsledkem násobení pak je:

$$A(x) \cdot B(x) = \sum_{i=0}^{n-1} a_i x^i \cdot \sum_{i=0}^{n-1} b_i x^i = \sum_{i=0}^{2n-2} \left| \sum_{j+k=i} a_j \cdot b_k \right|_p x^i$$

Jak je naznačeno, násobení i sčítání koeficientů se provádí *modulo p* (v tělese $GF(p)$.

[3] V některých materiálech se používá i obráceného zápisu $(a_0 a_1 \ldots a_p - 1)$.

Kvůli uzavřenosti násobení v tělese je nutné zavést operaci $A(x) \mod P(x)$, neboli zbytek po dělení polynomu $A(x)$ polynomem $P(x)$. Dále je třeba pro určení tělesa $GF(p^n)$ určit *ireducibilní* polynom, který bude použitý při operaci násobení.

Definice 2 *Polynom $P(x)$ nad tělesem $GF(p)$ je* ireducibilní *právě tehdy, když pro každé dva polynomy $A(x)$ a $B(x)$ nad $GF(p)$ platí:*

$$A(x) \cdot B(x) = P(x) \Rightarrow (deg(A(x)) = 0) \vee (deg(B(x)) = 0)$$

Neboli pro *ireducibilní* polynom platí, že neexistuje rozklad na polynomy nad $GF(p)$ stupně alespoň 1.

Příklad Polynom $x^3 + x + 1$ je nad tělesem $GF(2)$ *ireducibilní*, protože neexistuje jeho rozklad na polynomy stupně alespoň 1.
Polynom $x^2 + 1$ není nad tělesem $GF(2)$ *ireducibilní*, protože:

$$(x+1) \cdot (x+1) = x^2 + |1+1|_2\, x + 1 = x^2 + 1$$

Nyní je možné zavést operaci násobení dvou prvků tělesa jako násobení dvou polynomů *modulo zadaný ireducibilní* polynom:

$$A(x) \cdot B(x) = \sum a_i x^i \cdot \sum b_i x^i = \sum \left| \sum_{j+k=i} a_j \cdot b_k \right|_p x^i \mod P(x)$$

Poznámka Pokud by zvolený $P(x)$ nebyl *ireducibilní*, jednalo by se o *okruh*, nikoliv o *těleso*, protože by neexistovala *multiplikativní inverze* pro některé prvky a navíc by i existovaly tzv. *dělitelé nuly.*

1.3.3 Umocňování

Pro rozšíření operací o opakované násobení je vhodné zavést operaci umocňování.

Definice 3 *Pro prvek a tělesa T a číslo $n \in \mathbb{N}$ je operace umocňování definována následovně:*

$$\begin{aligned} a^0 &= 1 \\ a^n &= \underbrace{a \cdot a \cdot \ldots \cdot a}_{n\text{-krát}} \\ a^{-n} &= \left(a^{-1}\right)^n \end{aligned}$$

Pro efektivní výpočet mocniny prvku je vhodné použít algoritmus *Square-and-Multiply*, kde se dílčí operace „square“ a „multiply“ provádí operací $\cdot$ v daném tělese $GF(p^n)$.

1.3.4 Inverze

Inverzi v *grupě* lze obecně definovat následovně:

Definice 4 (Inverze) *Nechť a je prvkem a $\mathbb{O}$ neutrálním prvkem* grupy $G = (M, \circ)$. *Prvek $\bar{a}$ je* inverzí *prvku a, pokud platí následující rovnice:*

$$a \circ \bar{a} = \mathbb{O}$$

1.3.4.1 Aditivní inverze

Inverze v *aditivní grupě* je značena znaménkem minus „$-$“ a je z definice velmi triviální:

$$|A(x) + (-A(x))|_p = 0 \Rightarrow -A(x) = \sum |-a_i|_p \, x^i$$

Neboli je to aditivní inverze jednotlivých koeficientů *modulo* p (v tělese $GF(p)$).

1.3.4.2 Multiplikativní inverze

Inverze v *multiplikativní grupě* je značena záporným exponentem „$^{-1}$“ či symbolem dělení.

$$\left|A(x) \cdot A(x)^{-1}\right|_p = \left|\frac{A(x)}{A(x)}\right|_p = 1$$

Tuto *multiplikativní inverzi* je třeba počítat *rozšířeným Euklidovým algoritmem pro polynomy* (*EEA*), či případně jinými algoritmy, jako je např. *algoritmus Itoh-Teechai-Tsujii* (*ITT*) [6, 4].

Rozšířený Euklidův algoritmus pro polynomy, stejně jako v modulární aritmetice (neboli pro tělesa $GF(p)$), stojí na nalezení *Bézoutovy rovnosti.* Pro výpočet *EEA* je třeba výpočtu dělení polynomů se zbytkem[4].

1.4 Polynomy nad konečným tělesem

Prvotěleso

[4] Někdy uváděno jako dlouhé dělení.

Kapitola **2**

Lineární kódy

2.1 Kódování

2.2 Lineární kódy

2.2.1 Hammingovy kódy

2.3 Goppa kódy

Ireducibilní binární Goppa kódy

Kapitola **3**

Kryptosystém McEliece

3.1 Asymetrické šifrování McEliece

3.2 Niederreiterovo schéma

3.3 Bezpečnost algoritmů

3.3.1 Typy útoků

3.3.2 Slabiny systému

3.3.3 Existující útoky

3.3.4 Praktická varianta

CCA2-odolná varianta

Kapitola 4

Implementace

Pro implementaci kryptosystému *McEliece* v této práci byl zvolen software *Wolfram Mathematica* [9]. Tento software byl zvolen hlavně díky pohodlnosti některých matematických výpočtů a konstrukcí a také pro přehlednost výstupů.

Při implementaci *kryptosystému* se ukázaly nedostatky softwaru *Mathematica* a bylo nutné zpracovat problematiku (rozšířených) *konečných těles* a *binárních Goppa kódů*. Tyto dvě oblasti byly implementovány přímo v softwaru *Mathematica* tak, aby bylo možné jejich pohodlné použití i v jiných oblastech.

Celková práce byla rozdělena do třech ucelených částí – (binární) *konečná tělesa*, (ireducibilní) *binární Goppa kódy* a *kryptosystém McEliece* –, kde každá z nich lze využít jako *balík* či *knihovna* pro další výpočty. Následující kapitoly popisují jednotlivé části.

4.1 Binární konečná tělesa

Tato kapitola pojednává o implementaci *binárních konečných těles* včetně jejich *rozšíření*. Jsou zmíněny existující řešení v softwaru *Mathematica*, zvolená implementace a popis implementovaných algoritmů.

Ač jsou funkce implementované v co nejobecnějším pojetí, tak je kladen důraz na efektivnost výpočtů vzhledem k *binárním* tělesům – tedy k *tělesům* s charakteristikou 2. Pro *tělesa* s jinou charakteristikou není chování funkcí definováno.

4.1.1 Existující řešení

Pro operace s *konečnými tělesy* v softwaru *Mathematica* byly prostudovány interní funkce pro operace s polynomy a externí balík `FiniteFields`. Vlastnosti těchto řešení jsou popsány v následujících kapitolách.

4.1.1.1 Operace s polynomy

Software *Mathematica* obsahuje funkce pro operace s polynomy nad reálnými (případně i komplexními) čísly. Většina těchto funkcí má volitelnou *možnost*[5] *Modulus*, díky které lze zajistit, aby operace s koeficienty byly prováděny nad celými čísly *modulo* zadané číslo p. Tímto způsobem je možné implementovat operace nad tělesy $GF(p^n)$, nicméně je téměř nemožné tímto způsobem implementovat *rozšířená tělesa* – polynomy nad polynomy.

Pro použití těchto funkcí (např. `ExtendedPolynomialGCD`, je třeba polynomu v úplném tvaru $\sum a_i x^i$ – včetně x^i s tím, že x musí být nedefinovaný *symbol*[6]. Tento požadavek je celkem nepraktický, protože definování této proměnné kdekoliv v programu by vedlo na nemožnost použití těchto funkcí a udržovat si prvky ve formě např. x^6+x^3+x+1 místo 1001011 není pohodlné. Další nevýhoda použití polynomů je, že software *Mathematica* vypisuje polynomy od *nejnižšího* členu po *nejvyšší* (např. $1+x^2+x^4+x^7$), což je obrácený zápis, než je v technické literatuře zvykem.

4.1.1.2 Balík `FiniteFields`

Balík *Balík* v softwaru *Mathematica* je soubor obsahující rozšiřující funkce, které standardně nejsou k dispozici. Balík je možné načíst pomocí funkcí `Needs`, či případně *Get*.

Balík `FiniteFields` obsahuje základní operace pro práci s tělesy $GF(p^n)$. Prvky konečných těles jsou pak určené *seznamem*[7] koeficientů a *hlavičkou*, která určuje do jakého tělesa prvek patří. Výhoda tohoto opatření je, že pro sčítání a násobení je pak možné využít obyčejné symboly operací ($+$, $-$, $*$, $/$) a operace se automaticky provede v daném tělese. Pro parametry p a n je určené jedno těleso $GF(p^n)$ (s jedním konkrétním ireducibilním polynomem) a *seznam* koeficientů prvku se opět píše od nejnižšího řádu po nejvyšší (například polynom x^3+x+1 z tělesa $GF(2^5)$ je zapsán jako $GF[2,5][\{1,1,0,1,0\}]$).

Funkce z balíku `FiniteFields` nejsou dostatečně zdokumentovány, jak je jinak v softwaru *Mathematica* zvykem. Nepodařilo se využít funkcí z tohoto balíku pro operace s *rozšířenými tělesy*.

4.1.2 Zvolené řešení

Existující řešení pro práci s *konečnými tělesy* se ukázala jako nedostačující. Jejich hlavní nevýhodou je nemožnost použití při výpočtech s *rozšířenými tělesy*. Proto bylo implementováno vlastní řešení pro práci s *konečnými tělesy*.

Při implementaci operací nad *konečnými tělesy* bylo dodržováno následující jednotné rozhraní:

[5] Anglicky se tento termín v softwaru *Mathematica* nazývá *Option*.
[6] Jinými slovy proměnná, která nemá definovanou hodnotu.
[7] *Seznamem* se myslí struktura v softwaru *Mathematica* – *List*

- Prvky *konečných těles* jsou reprezentovány *seznamem* koeficientů od nejvyššího po nejnižší.
  U *rozšířených těles* jsou koeficienty opět prvky konečných těles.
  Například polynom $x^3 + x + 1$ je reprezentován seznamem: $\{1, 0, 1, 1\}$
  a polynom $(y + 1)x^2 + (y)$ je reprezentován: $\{\{1, 1\}, \{0, 0\}, \{1, 0\}\}$

- Prvek (seznam koeficientů) může být libovolně dlouhý. V případě potřeby se při výpočtu *redukuje* (ireducibilním) polynomem nebo dorovná *nulovými* koeficienty.

- Počet koeficientů vnitřních prvků (koeficientů) musí být vždy stejný.
  Například prvek $\{\{0, 0\}, \{1\}, \{1, 0\}\}$ není dovolený.

- Jednotlivým funkcím je kromě operandů předáván též i *modul* skládající se z odpovídajících (ireducibilních) polynomů, včetně charakteristiky tělesa. Tento *modul* je definovaný následovně:
  Pro tělesa $GF(p^{n_1})$ je *modul* složen z (ireducibilního) polynomu i_1 stupně n_1 a dané charakteristiky p: $modul_1 = \{i_1, p\}$
  Pro rozšířená tělesa se *modul* skládá z odpovídajícího *polynomu* i_k stupně n_k nad tělesem $GF(p^{n_1 \cdots n_{k-1}})$ a *modulu vnitřního tělesa*:
  $modul_k = \{i_k, modul_{k-1}\}$.

- Všem funkcím se předávají nejdřív *operandy* a poté *modul.*
  Například pro prvky $a, b \in GF(p^{\cdots})$, $m \in \mathbb{N}$ a odpovídající *modul*:
  $krat[a, b, modul]$
  $inverze[a, modul]$
  $mocnina[a, m, modul]$
  $\ldots$

- Pro implementaci operací v *prvotělesech* (tělesech $GF(p^n)$) jsou použité vnitřní funkce softwaru *Mathematica* pro práci s *polynomy.* Implementované funkce pro *prvotělesa* tedy zpravidla obsahují převod ze *seznamu* čísel na *polynom*, zavolání vnitřní funkce pro *polynomy* a převodu zpět na *seznam* koeficientů. Díky těmto vnitřním funkcím je docíleno rychlejšího výpočtu, než kdyby byla použita vlastní implementace nad *seznamy* celých čísel.

- Pro implementaci operací v *rozšířených tělesech* byly implementovány jednotlivé algoritmy operací (popsané níže), jelikož nebylo možné použít pro tyto operace vnitřní funkce softwaru *Mathematica.* Funkce nad *rozšířenými tělesy* zpravidla volají odpovídající funkce ve vnitřních tělesech (například násobení jednotlivých *koeficientů*).

Tato pravidla umožňují pohodlný, jednotný a *rekurzivní* přístup k jednotlivým prvkům a voláním funkcí (druhá složka *modulu* je *modul vnitřního tělesa*, prvky *polynomu* jsou opět *polynomy*, ...).

4.1.3 Implementace operací

V následujících kapitolách je popsána implementace hlavních operací v *konečných tělesech* a použitých algoritmů. Pro další informace je doporučeno nahlédnout do zdrojového kódu a příkladů použití.

V níže uvedených pseudokódech se používá některých prvků ze syntaxe softwaru *Mathematica*:

Zápis	Význam
`foo[bar]`	Volání funkce *foo* s argumentem *bar*
`ham[[i]]`	i-tý prvek seznamu (pole) *ham*

Tabulka 4.1: Prvky syntaxe jazyka softwaru *Mathematica*

4.1.3.1 Sčítání

Jelikož operace sčítání se v jakémkoliv *tělese* provádí po jednotlivých koeficientech *modulo* p, je tato funkce jediná volána místo celkového modulu pouze se zadanou charakteristikou p.

Pro *rozšířená tělesa* funkce rekurzivně volá stejnou operaci sčítání na jednotlivé koeficienty zadaných polynomů až na úroveň *prvotěles* – obyčejných jednorozměrných seznamů. Pro *prvotělesa* funkce používá obyčejné sčítání dvou seznamů modulo p.

Algoritmus 1 Sčítání polynomů

```
function PLUS[a,b,p]                         ▷ Pro GF(p^n), p je prvočíslo
    return Mod[a + b, p]
end function
function PLUS[a,b,p]                         ▷ Pro GF(q^n), q je p^...
    for i ← 1 ... Length[a] do
        c[[i]] ← plus[a[[i]], b[[i]], p]
    end for
    return c
end function
```

4.1.3.2 Redukce polynomu

Redukce polynomu (neboli *modulo* polynom) se používá ve většině dalších funkcí. Tato funkce se volá se dvěma parametry – prvkem a a polynomem (*modulem*) m. Funkce vrátí zbytek polynomu a po dělení polynomem m.

Redukce polynomu pro *rozšířená tělesa* je inspirovaná *Comb metodou* z [3]. K původnímu prvku a se opakovaně přičítá (od nejvyššího řádu) patřičný násobek *polynomu* m tak, aby se daný koeficient a_i rovnal nule (viz příklad níže).

Pro *prvotělesa* se používá interní funkce `PolynomialMod`

Algoritmus 2 Redukce polynomu v tělese s charakteristikou 2

function REDUKUJ[a, $\{m, modul_{vnitrni}\}$]
  $l_a \leftarrow stupen[a] + 1$ ▷ Délka redukovaného polynomu
  $l_m \leftarrow stupen[m]$ ▷ Výsledná délka redukovaného polynomu
   // Převedení m na *monický* polynom
  $koef \leftarrow inverze[m[[1]], modul_{vnitrni}]$ ▷ Inverze nejvyššího koeficientu
  $m \leftarrow krat[koef, m, modul_{vnitrni}]$ ▷ Násobení skalárem

  $m \leftarrow PadRight[m, l_a - l_m]$ ▷ Natáhnutí polynomu na délku a
  for $i \leftarrow 1 \ldots l_a - l_m$ **do**
    $s \leftarrow krat[a[[i]], m, modul_{vnitrni}]$ ▷ Skalární násobek
    $a \leftarrow plus[a, s, 2]$ ▷ Odečtení v binárním tělese
    $m \leftarrow RotateRight[m]$ ▷ Posunutí redukovaného polynomu
  end for

  return a
end function

Příklad Redukce polynomu $x^{12} + x^8 + x^7 + x^5 + x^4 + x^3 + 1$ polynomem $x^4 + x + 1$ (nad tělesem $GF(2)$):

$$1000110111001 \quad \text{mod } 10011 :$$

$$\begin{array}{ccccccccccccc}
1&0&0&0&1&1&0&1&1&1&0&0&1 \\ \hline
1&0&0&1&1&0&0&0&0&0&0&0&0 \\ \hline
0&0&0&1&0&0&1&1&0&0&0&0&0 \\
0&0&0&0&0&1&0&0&1&1&0&0&0 \\
0&0&0&0&0&0&1&0&0&1&1&0&0 \\ \hline
&&&&&&&&&1&1&0&1
\end{array}$$

4.1.3.3 Násobení

Výsledkem násobení dvou polynomů a a b stupně n a m je polynom c stupně $n + m$. Násobení je implementováno tak, že k výsledku c (na počátku je to nulový polynom) se postupně přičítá skalární násobek polynomu b koeficienty polynomu a, který je zároveň *posunutý* o patřičný počet pozic. Využívá se zde faktu, že násobení libovolného *polynomu* $A(x)$ a x^i je posunutí koeficientů polynomu A o i pozic doleva. Výsledný polynom c je následně *redukován* zadaným modulem (viz výše).

Pro *prvotělesa* se používá obyčejného násobení dvou *polynomů* a následné *redukce modulem.*

Algoritmus 3 Násobení prvků

function KRAT[a, b, $\{m, modul_{vnitrni}\}$]
    $p \leftarrow charakteristika[modul]$ ▷ Charakteristika tělesa
    // Natažení na výslednou délku
    $b \leftarrow PadLeft[b, stupen[a] + stupen[b] + 1]$
    $c \leftarrow nulovyPolynom[\ldots]$ ▷ Nulový polynom nad vnitřním tělesem

    for $i \leftarrow stupen \ldots 1$ **do**
        $s \leftarrow krat[a[[i]], b, modul_{vnitrni}]$ ▷ Skalární násobek
        $c \leftarrow plus[c, s, p]$
        $b \leftarrow RotateLeft[b]$ ▷ Posunutí přičítaného polynomu
    end for

    return $redukuj[c]$
end function

Příklad Násobení polynomu $x^3 + x + 1$ polynomem $x^4 + x^2 + 2x + 1$ (nad tělesem $GF(3)$):

$$
\begin{array}{rl}
\multicolumn{2}{l}{1011 \cdot 10121:} \\
1(x^4) & 1\,0\,1\,1\,0\,0\,0\,0 \\
0(x^3) & 0\,0\,0\,0\,0\,0\,0\,0 \\
1(x^2) & 0\,0\,1\,0\,1\,1\,0\,0 \\
2(x^1) & 0\,0\,0\,2\,0\,2\,2\,0 \\
1(x^0) & 0\,0\,0\,0\,1\,0\,1\,1 \\
\hline
 & 1\,0\,2\,0\,2\,0\,0\,1
\end{array}
$$

4.1.3.4 Inverze

Výpočet multiplikativní *inverze* je implementován pomocí *rozšířeného Euklidova algoritmu.* Tento algoritmus se často vizualizuje jako výpočet tabulky po řádkách (viz níže). Ve skutečnosti však pro výpočet dalšího řádku stačí pracovat s hodnotami dvou řádků předešlých. Proto si není nutné udržovat v paměti celou tabulku, ale stačí si udržovat hodnoty dvou řádků a po výpočtu třetího hodnoty posunout.

Výpočet hodnot dalšího řádku tabulky probíhá následovně:

- Hodnoty předchozích řádků jsou:

  Polynomy p_{i-2} a p_{i-1} (na začátku inicializovány na ireducibilní polynom m a *prvek*, ke kterému je hledaná inverze).

  Polynomy k_{i-2} a k_{i-1} (na začátku inicializovány na 0 a 1, respektive *nulový* a *jednotkový polynom*).

- Je spočítán *podíl* q a zbytek p_i pomocí tzv. *dlouhého dělení* polynomu p_{i-2} polynomem p_{i-1}.
- Je spočítán *polynom* $k_i = k_{i-2} - q \cdot k_{i-1}$
- Tyto kroky se opakují, dokud není získán polynom p_i stupně 0 (jinými slovy jediný prvek vnitřního tělesa).
- Výsledná *inverze* se získá jako skalární násobek *polynomu* k_i inverzí (posledního) *koeficientu* polynomu p_i[8].

Inverze v *prvotělese* je implementovaná pomocí interní funkce `PolynomialExtendedGCD`.

Algoritmus 4 Inverze prvků – *Rozšířený Euklidův algoritmus*

function INVERZE[$prvek$, $modul : \{m, modul_{vnitrni}\}$]
    $A \leftarrow m$; $B \leftarrow prvek$
    // Inicializace na *jednotkový* resp. *nulový* polynom z tělesa
    $k_A \leftarrow nulovyPolynom[\ldots]$; $k_B \leftarrow jednotkovyPolynom[\ldots]$
    while $stupen[B] \neq 0$ **do**
        // Výpočet q a C pomocí dlouhého dělení v jednom kroku
        $q \leftarrow A/B$; $C \leftarrow A \mod B$
        $k_C \leftarrow k_A - krat[q, k_B, modul]$
        $A \leftarrow B$; $k_A \leftarrow k_B$
        $B \leftarrow C$; $k_B \leftarrow k_C$
    end while
    // Výpočet koeficientu ve vnitřním tělese
    $koef \leftarrow inverze[Last[C], modul_{vnitrni}]$
    return $krat[koef, k_C, modul_{vnitrni}]$ ▷ Násobení skalárem
end function

Příklad *Rozšířený Euklidův algoritmus* pro výpočet *inverze* polynomu $x^3 + x^2 + 1$ *modulo* $x^6 + x + 1$ (nad tělesem $GF(2)$):

Podíl	Zbytek	Koeficienty	
	1000011	0	1
	1101	1	0
1110	101	−1110	1
11	10	10011	−11
10	1	−101000	111

$\Rightarrow |1101^{-1}|_{1000011} = 101000$

[8] Zde je vidět, že pro výpočet inverze v tělese $GF(q^n)$ je třeba vypočítat inverzi v tělese $GF(q)$.

Poznámka Poslední sloupec tabulky se v algoritmu nepočítá, je zde uveden pouze pro úplnost.

4.1.3.5 Druhá mocnina

Pro prvky tělesa s *charakteristikou* 2 Je výhodné implementovat funkci „na druhou“ díky následujícímu tvrzení:

Tvrzení 1 *Nechť* $A = (a_n \dots a_2 a_1 a_0)$ *je prvek tělesa s* charakteristikou 2, *potom platí:*

$$A^2 = (a_n^2 0 \dots 0 a_2^2 0 a_1^2 0 a_0^2)$$

Důkaz

$$\begin{aligned}
A(x) &= a_n x^n + \dots + a_2 x^2 + a_1 x + a_0 \\
A(x)^2 &= (a_n x^n + \dots + a_2 x^2 + a_1 x + a_0) \cdot (a_n x^n + \dots + a_2 x^2 + a_1 x + a_0) \\
&= a_n x^n \cdot (a_n x^n + \dots + a_2 x^2 + a_1 x + a_0) + \\
&\quad \vdots \\
&+ a_2 x^2 \cdot (a_n x^n + \dots + a_2 x^2 + a_1 x + a_0) + \\
&+ a_1 x \quad \cdot (a_n x^n + \dots + a_2 x^2 + a_1 x + a_0) + \\
&+ a_0 \quad\ \cdot (a_n x^n + \dots + a_2 x^2 + a_1 x + a_0) \\
&= a_n^2 x^{2n} + \dots + a_n a_2 x^{n+2} + a_n a_1 x^{n+1} + a_n a_0 x^n + \\
&\quad \vdots \\
&+ a_n a_2 x^{n+2} + \dots + a_2^2 x^4 + a_2 a_1 x^3 + a_2 a_0 x^2 + \\
&+ a_n a_1 x^{n+1} + \dots + a_2 a_1 x^3 + a_1^2 x^2 + a_1 a_0 x + \\
&+ a_n a_0 x^n + \dots + a_2 a_0 x^2 + a_1 a_0 x + a_0^2 \\
&= a_n^2 x^{2n} + \dots + 2(a_3 a_0 + a_2 a_1) x^3 + 2(a_2 a_0) x^2 + a_1^2 x^2 + 2(a_1 a_0) x + a_0^2 \\
&= \sum_{i=0}^{n} a_i^2 x^{2i} + 2 \sum_{i=1}^{n+1} \sum_{\substack{j<k \\ j+k=i}} a_j a_k \\
&= \sum_{i=0}^{n} a_i^2 x^{2i} \qquad\qquad \cong (a_n^2 0 \dots 0 a_2^2 0 a_1^2 0 a_0^2)
\end{aligned}$$

S využitím tohoto tvrzení je realizace funkce na počítání druhé mocniny triviální:

- Provedení druhé mocniny všech koeficientů.
- Proložení koeficientů polynomu nulovými koeficienty.
- Redukování polynomem (viz výše).

Algoritmus 5 Umocňování na druhou v tělese s charakteristikou 2

```
function naDruhou[ a, {m, modul_vnitrni} ]
    for i ← 1 … Length[i] do
        a[[i]] ← naDruhou[a[[i]], modul_vnitrni]
    end for
    nula ← nulovyPolynom[…]                ▷ Odpovídající nulový koeficient
    a ← Riffle[a, nula]                    ▷ Proloží koeficienty prvkem nula
    return redukujPolynom[a, modul]
end function
```

4.1.3.6 Mocnění

Mocnění *polynomů* je implementováno pomocí algoritmu *Square-and-Multiply* (*SM*). Algoritmus využívá faktu, že libovolnou mocninu lze rozložit na součin mocnin čtverců (^{2},^{4},^{8},…). Konkrétně byla implementována varianta provádějící výpočet od nejvíce významného bitu exponentu[9]. Algoritmus má vstupy polynom a a exponent e. Exponent se vyjádří jako číslo v *binární* soustavě a poté algoritmus provádí cyklus přes bity tohoto rozvoje. V každém kroku se mezivýsledek umocní na druhou a v případě, že je odpovídající bit exponentu 1, přinásobí se původní číslo a.

Algoritmus 6 Umocňování prvku $a^e \mod modul$ – *Square-and-Multiply*

```
function umocni[ a, e, modul ]
    if e = 0 then
        return nulovyPolynom[…]            ▷ Nulový prvek tělesa
    end if
    rozvoj ← IntegerDigits[e, 2]           ▷ Binární rozvoj exponentu
    c ← a                                  ▷ rozvoj[[1]] je vždy 1
    for i ← 2 … Length[rozvoj] do
        s ← naDruhou[c, modul]
        m ← krat[s, a, modul]
        if rozvoj[[i]] = 0 then
            c ← s
        else
            c ← m
        end if
    end for
    return c
end function
```

[9] Uváděna jako *MSB* – z anglického *most significant bit*

Poznámka Takto implementovaný algoritmus je zranitelný vůči odběrové a časové analýze. Pro odolnou implementaci je nutné počítat násobek *vždy* a pokud je daný bit exponentu 1, přiřadit násobek do mezi výpočtu. Pseudokód i reálná implementace je prováděna tímto (bezpečným) způsobem.

Příklad *Square-and-Multiply* pro výpočet $\left(x^3+1\right)^{26}$ *modulo* x^6+x+1 (nad tělesem $GF(2)$):

Op.	Mocnina dek.	Mocnina bin.	Výpočet	Výsledek
	1	1		1001
S	2	1	1000001	10
M	3	11	$10 \cdot 1001$	10010
S	6	110	100000100	1000
S	12	1100	1000000	11
M	13	1101	$11 \cdot 1001$	11011
S	26	11010	101000101	1010

$\Rightarrow |1001^{26}|_{1000011} = 1010$

4.1.4 Možná zlepšení

V této kapitole jsou nastíněny možná zlepšení implementace, která zrychlují výpočet některých operací.

4.1.4.1 Logaritmické tabulky

Pro zrychlení výpočtu násobení a mocnin prvku lze v *konečném tělese* využít faktu, že vždy existuje *primitivní prvek* a převádět tak operace v tělese na operace s celými čísly.

Definice 5 *Nechť α je* generátor multiplikativní grupy *tělesa F. Potom říkáme, že α je* primitivní prvek *tělesa F.*

Důsledek Každý prvek tělesa F – kromě *nulového* prvku *aditivní grupy* – lze vyjádřit jako α^i pro nějaké i.

Důkaz plyne přímo z definice.

Násobení dvou prvků $a = \alpha^{i_a}$ a $b = \alpha^{i_b}$ tak lze převést na součet mocnin *primitivního prvku*:

$$a \cdot b = \alpha^{i_a} \cdot \alpha^{i_b} = \alpha^{i_a+i_b}$$

Podobným způsobem je možné zjednodušit umocňování prvku:

$$a^e = \left(\alpha^i\right)^e = \alpha^{ie}$$

V obou případech je samozřejmě možné použít *Eulerovu větu* a mocniny redukovat *modulo* N, kde N je počet prvků *multiplikativní grupy tělesa* ($N = p^n - 1$ pro těleso $GF(p^n)$). Jakoukoliv operací násobení a mocnění se získá prvek α^{n_c}, kde n_c je celé číslo v rozsahu od 0 do $N - 1$.

Reprezentací prvků pomocí odpovídajících mocnin *primitivního prvku* je tak možné vyhnout se násobení a umocňování prvků v tělese a nahradit ho sčítáním a násobením celých čísel, což je řádově jednodušší. V případě sčítání prvků v tělese je však nutné mít jejich standardní reprezentaci (seznam koeficientů), jelikož se sčítání provádí po jednotlivých koeficientech, respektive bitech. Není možné nahradit sčítání dvou prvků jiné operaci s mocninami *primitivního prvku.*

Pro použití tohoto zrychlení výpočtů je tak nutné připravit v paměti programu překladové *log-* a *antilogaritmické* tabulky pro překlad prvků z jedné reprezentace na druhou.

Ač se tak získá podstatné zrychlení výpočtů v tělese, existuje několik nevýhod tohoto přístupu:

- Je nutné nalézt *primitivní prvek tělesa.*
- Je nutné vygenerovat a uchovat v paměti počítače obě tabulky pro překlad.
  - Tato tabulka lze implementovat pomocí obyčejného pole či seznamu, kde se k daném indexu v seznamu vyskytuje odpovídající hodnota.
  - Pro binární tělesa $GF(2^m)$ je velikost jedné tabulky $O(m2^m)$ (konkrétně $2^m - 1$ hodnot, kde každá je reprezentována m bity).
  - Jelikož je paměťová náročnost *exponenciální*, je možné tyto tabulky uchovávat pouze pro *malá* m (např. 8 či 16, nikoliv však 1024).
- *Nulový prvek* tělesa není možné žádným způsobem zobrazit jako mocninu. Při každé operaci je potřeba s touto skutečností počítat a hlídat jako výjimku.

Tohoto vylepšení se dá využít pro operace ve *vnitřním tělese* $GF(2^m)$, nad kterém jsou postavené polynomy v *binárních Goppa kódech.*

4.1.4.2 Implementace dělení

Dělení prvkem b v *konečném tělese* se převádí na násobení b^{-1}. Pro výpočet *podílu* se tak počítá inverze a následně násobek. Je ale možné implementovat rovnou algoritmus pro dělení.

Algoritmus pro dělení prvku a prvkem b je totožný s algoritmem pro výpočet *inverze* prvku b s tím rozdílem, že je počáteční hodnota koeficientu k_b (viz *EEA* – alg. 4) nastavena na hodnotu a. Výsledkem algoritmu pak bude inverze prvku b vynásobená a, což přesně odpovídá výrazu a/b.

4.2 Ireducibilní binární Goppa kódy

4.3 McEliece

4.4 Měření

Závěr

Literatura

[1] McEliece

[2] Understanding Cryptography

[3] MERCHAN J. G., KUMAR S., PAAR C., PELZL J. *Efficient Software Implementation of Finite Fields with Applications to Cryptography* v Acta Applicandae Mathematicae: An International Survey Journal on Applying Mathematics and Mathematical Applications, Volume 93, Numbers 1-3, pp. 3-32, September 2006. Ruhr-Universitat Bochum, 2006. Dostupné online: `http://www.emsec.rub.de/research/publications/efficient-software-implementation-finite-fields-ap/`

[4] ITT

[5] Přednášky BI-LIN

[6] Přednášky MI-BHW

[7] Přednášky MI-MKY

[8] Přednášky MI-MPI

[9] Wolfram Mathematica

Příloha **A**

Seznam použitých zkratek

EEA *Extended Euclidean Algorithm* – Rozšířený Euklidův algoritmus

GCD *Greatest Common Divisor* – Největší společný dělitel

GF *Gallois field* – konečné těleso

LSB Least Significant Bit/Byte – nejméně významný bit/bajt

MSB Most Significant Bit/Byte – nejvíce významný bit/bajt

S&M algoritmus *Square-and-Multiply*

Příloha B

Obsah přiloženého CD

```
readme.txt ........ stručný popis obsahu CD
exe ............... adresář se spustitelnou formou implementace
src
  impl ............ zdrojové kódy implementace
  thesis .......... zdrojová forma práce ve formátu LaTeX
text .............. text práce
  thesis.pdf ...... text práce ve formátu PDF
  thesis.ps ....... text práce ve formátu PS
```