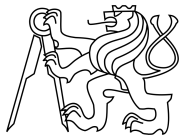
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Výměna klíčů založená na isogeniích supersinguláních eliptických křivek
Student:	Klára Drhová
Vedoucí:	Ing. Jiří Buček
Studijní program:	Informatika
Studijní obor:	Teoretická informatika
Katedra:	Katedra teoretické informatiky
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

Prostudujte matematické základní principy aritmetiky na eliptických křivkách a seznamte se s principy tvorby isogenií supersinguláních eliptických křivek a jejich využitím při výměně klíčů [1,2]. Prozkoumejte stávající implementaci výměny klíčů používající isogenia supersinguláních eliptických křivek [3]. Implementujte vzorovou aplikaci výměny klíčů založené na isogeniích supersinguláních eliptických křivek s cílem názorně prezentovat základní principy z [1]. Vyberte vhodnou implementační platformu, např. jazyk C/C++ s použitím vhodných knihoven.

Seznam odborné literatury

[1] JAO, David a DE FEO, Luca. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies [online]. 2011 [vid. 11. prosince 2015]. Dostupné z: http://cacr.uwaterloo.ca/techreports/2011/cacr2011-32.pdf

[2] DE FEO, Luca, JAO David, a PLUT, Jérome. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. Journal of Mathematical Cryptology. 2014, vol. 8, no. 3, s. 209-247.

[3] DE FEO, Luca, JAO, David, PLUT, Jérome a kolektiv. Quantum-Resistant Cryptosystems from Supersingular Elliptic Curve Isogenies [online repozitář] 2015 [vid. 11. prosince 2015]. Dostupné z: https://github.com/defeo/ss-isogeny-software

L.S.

doc. Ing. Jan Janoušek, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 14. ledna 2016

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra teoretické informatiky

Bakalářská práce

Výměna klíčů založená na isogeniích supersingulárních eliptických křivek

Klára Drhová

Vedoucí práce: Ing. Jiří Buček

15. května 2016

Poděkování

Ráda bych poděkovala vedoucímu své bakalářské práce, panu Ing. Jiřímu Bučkovi, za cenné rady, připomínky, ochotu a čas, které mi věnoval. Dále bych chtěla poděkovat panu Ing. Ivo Petrovi, Ph.D., za přínosné konzultace týkající se eliptických křivek.

Velký dík patří mé rodině, která mě ochotně a bez přestání podporuje celý můj život a v neposlední řadě bych velice ráda poděkovala svému příteli za podporu, trpělivost a za to, že je.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracovala samostatně a že jsem uvedla 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 15. května 2016 .....

Odkaz na tuto práci

Drhová, Klára. *Výměna klíčů založená na isogeniích supersingulárních eliptických křivek*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Tato bakalářská práce se zabývá implementací výměny klíčů založené na isogeniích supersingulárních eliptických křivek, která se řadí do postkvantové kryptografie. S příchodem kvantových počítačů se dnes používané šifrovací systémy stanou prolomenými a budou nahrazeny novými. Postkvantová kryptografie nabízí takové kandidáty. K implementaci byl vybrán dnes velmi rozšířený programovací jazyk C++ spolu s matematickou knihovnou PARI. Byla vytvořena fungující aplikace realizující výměnu klíčů, jejíž největším přínosem je jednoduchý návrh, který ilustruje základní principy algoritmu výměny klíčů a umožňuje jeho snadné porozumění.

Klíčová slova isogenium, supersingulární eliptické křivky, výměna klíčů, Diffie–Hellman, postkvantová kryptografie, C++, knihovna PARI

Abstract

This thesis deals with the implementation of the key exchange based on supersingular elliptic curves isogenies, which belongs to the post-quantum cryptography. With the advent of quantum computers, cryptographic systems that are used today will become broken and will be replaced with the new ones. Post-quantum cryptography offers such candidates. Very widespread programming language C++ was selected for the implementation, along with the PARI – mathematical library. A working application was created that performs key exchange. Its biggest benefit is its simple design, which illustrates the basic principles of the key exchange algorithm and which allows to understand the algorithm easily.

Keywords isogeny, supersingular elliptic curves, key exchange, Diffie–Hellman, post-quantum cryptography, C++, PARI library

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V dnešním světě je počítačová bezpečnost aktuálnějším tématem než kdykoli předtím. Příchod Internetu umožnil jeho uživatelům přístup k nejrůznějším informacím a téměř všudypřítomné Wi-Fi sítě a 3G sítě umožňují se připojit téměř odkudkoli. Zabezpečení komunikace je pro všechny samozřejmostí. S příchodem kvantových počítačů se ovšem mnoho běžně používaných kryptosystémů veřejného klíče může stát zranitelných a dnes běžně používané šifry asymetrické kryptografie se mohou stát nepoužitelnými.

Výzkum využití kvantové mechaniky v informatice započal v 80. letech 20. století. Peter Shor představil v roce 1994 dva algoritmy pro kvantový počítač, z nichž jeden dokáže faktorizovat celá čísla a druhý spočíst diskrétní logaritmus. Oba tyto algoritmy pracují v polynomiálním čase. To znamená, že s příchodem univerzálních kvantových počítačů bude u šifrovacích systémů založených na problému diskrétního logaritmu či na problému faktorizace možné snadno vypočíst k danému veřejnému klíči klíč privátní. Tím získáme možnost s pomocí vypočteného privátního klíče rozšifrovat libovolnou zprávu zašifrovanou příslušným veřejným klíčem. Dnes hojně využívané kryptosystémy veřejného klíče postavené na těchto matematických problémech, jakými jsou například RSA, Diffie–Hellman, ElGamal a další, budou prolomeny a stanou se nepoužitelnými.

Postkvantová kryptografie se zabývá kryptografickými algoritmy, pro které není znám způsob útoku pomocí kvantového počítače. Mezi kryptosystémy založené na těchto algoritmech patří například NTRU[1], McEliece a kryptosystém založený na isogeniích supersingulárních eliptických křivek. Poslední zmíněný kryptosystém má oproti předchozím výhodu v tom, že poskytuje Forward Secrecy a tím zamezuje zpětnému dešifrování minulých zpráv v případě odtajnění hlavního dlouhodobého klíče.

Ve své bakalářské práci se zabývám studiem matematických základů nutných pro pochopení fungování kryptosystému založeného na isogeniích super-

[1] N-th degree truncated polynomial ring

singulárních eliptických křivek a dále implementací výměny klíčů užívající tohoto kryptosystému.

Abych mohla výměnu klíčů založenou na isogeniích supersingulárních eliptických křivek náležitě popsat, je nezbytné nejprve pochopit potřebné matematické základy. Jedná se především o teorii týkající se algebraických struktur, základní teorii týkající se vektorových prostorů a teorii spojenou s eliptickými křivkami a isogenii eliptických křivek. Sekce 2.2, 2.3 a 2.4 obsahují úvod do těchto oblastí.

V sekci 2.6 popisuji algoritmus výměny klíčů založené na isogeniích eliptických křivek. V návaznosti na popis algoritmu se v sekci 2.7 zabývám stávající implementací výměny klíčů používající isogenia supersingulárních eliptických křivek.

V sekci 2.8 porovnávám postkvantové algoritmy výměny klíčů s klasickými algoritmy.

V kapitole 3 se zabývám návrhem a realizací aplikace vykonávající výměnu klíčů založenou na isogeniích supersingulárních eliptických křivek. Nejprve se věnuji výběru vhodného programovacího jazyka a vhodných knihoven, dále popisuji implementaci samotné aplikace.

V kapitole 4 se věnuji samotnému testování vytvořené aplikace, která byla popsána v kapitole 3.

Kapitola 1

Cíl práce

Cílem rešeršní části mé bakalářské práce je prostudovat potřebné základy nutné k pochopení výměny klíčů založené na isogeniích supersingulárních eliptických křivek. Toto zahrnuje prostudování základních principů aritmetiky na eliptických křivkách, dále seznámení se s principy tvorby isogenií supersingulárních eliptických křivek a jejich využitím při výměně klíčů [1, 2]. Na toto navazuje prostudování algoritmu výměny klíčů používající isogenia supersingulárních eliptických křivek a prozkoumání stávající implementace [3].

Cílem praktické části mé bakalářské práce je implementovat vzorovou aplikaci výměny klíčů založené na isogeniích supersingulárních eliptických křivek s cílem názorně prezentovat základní principy z [1].

Kapitola 2

Analýza

V této kapitole se věnuji matematické teorii potřebné k pochopení výměny klíčů založené na isogeniích supersingulárních eliptických křivek. Pojmy **supersingulární eliptická křivka** a **isogenium** se zabývám v sekci 2.4. Dále se v této kapitole zaměřuji na samotný algoritmus výměny klíčů a analyzuji stávající implementaci této výměny. Na konci této kapitoly porovnávám postkvantové a klasické algoritmy výměny klíčů.

2.1 Použité značení

V této sekci uvádím přehledný souhrn veškerého použitého značení v této bakalářské práci.

$\mathbb{Z}$	Obor celých čísel
$\mathbb{N}$	Obor přirozených čísel $1, 2, 3, \ldots$
$\mathbb{R}$	Obor reálných čísel
$\mathbb{C}$	Obor komplexních čísel
$\exists$	Existuje
$p \mid q$	p dělí q
$p \nmid q$	p nedělí q
$M \subset V$	M je vlastní podmnožina V
$M \subseteq V$	M je podmnožina V
$x \in M$	x leží v množině M, množina M obsahuje prvek x
$M \times N$	Kartézský součin množin M a N
$a \equiv b \pmod{m}$	Prvek a je kongruentní b modulo m
$\gcd(a, b)$	Největší společný dělitel čísel a a b
$[x]$ nebo $[x]_{\mathcal{R}}$	Třída prvků ekvivalentních s x
$f : G \to H$	Zobrazení množiny G do množiny H
a^{-1}	Inverzní prvek k a
e	Neutrální prvek grupy
0 a 1	Neutrální prvek grupy $(R, +)$ a $(R, \cdot)$

$\#G$	Řád grupy
$G[m]$	Množina m-torzních bodů grupy G
$\ker f$	Jádro homomorfizmu f
T^*	Multiplikativní grupa tělesa T
$[L : K]$	Stupeň L nad K
$GF(q)$	Galoisovo či konečné těleso řádu q
$\mathbb{F}_q$	Konečné těleso řádu q
$\mathbb{Z}/p\mathbb{Z}$	Množina celých čísel modulo p
$\mathbb{Z}_p^\times$	Modulární multiplikativní grupa
$p(x)$	Polynom s proměnnou x
$\deg p(x)$	Stupeň polynomu $p(x)$
$K[x]$	Komutativní okruh polynomů nad okruhem K
(a)	Nejmenší ideál okruhu R obsahující prvek $a \in R$
$[a]$ nebo $a + J$	Rozkladová třída okruhu R vzhledem k ideálu J pro $a \in R$
R/J	Faktorokruh okruhu R vzhledem k ideálu J
$\overline{T}$	Algebraický uzávěr tělesa T
$\langle M \rangle$	Lineární obal množiny vektorů M
$\dim V$	Dimenze vektorového prostoru V
$E(T)$	Eliptická křivka nad tělesem T
O	Bod v nekonečnu eliptické křivky
$M_{A,B}$	Montgomeryho tvar eliptické křivky
Δ	Diskriminant Weierstrassovy rovnice eliptické křivky
j nebo $j(E)$	J-invariant eliptické křivky E
$P \oplus Q$	Součet bodů P a Q na eliptické křivce
$\ominus P$	Opačný bod bodu P na eliptické křivce
$P \ominus Q$	Součet bodu P a $\ominus Q$
$[m]P$	$\begin{cases} \text{Součet } m \text{ bodů } P \text{ pro } m > 0 \\ \text{Součet } m \text{ bodů } \ominus P \text{ pro } m < 0 \\ O \text{ pro } m = 0 \end{cases}$
$\#E$	Řád eliptické křivky
$E[m]$	m-torzní podgrupa eliptické křivky E
R^3	Množina všech uspořádaných trojic, kde prvky jsou z okruhu R
$P_2(R)$	Projektivní rovina nad okruhem R
$x = (x_1, x_2, x_3)$	Homogenní souřadnice bodu x na eliptické křivce
$(x_1 : x_2 : x_3)$	Třída všech prvků ekvivalentních s (x_1, x_2, x_3)
$f(x)$	Racionální funkce proměnné x nad $\mathbb{C}$
$\mathrm{div}(f)$	Divizor funkce f
$e_m(P, Q)$	Weilovo párování bodů P a Q řádu dělícího m
$M \cong N$	M je izomorfní s N
$x \in_R M$	x je zvoleno rovnoměrně náhodně z množiny M
$\|a\|_m$	a modulo m

2.2 Základní algebraické struktury

Nejprve je nutné uvést definice a věty týkající se algebraických objektů. Více informací k daným matematickým oblastem lze získat z [4, 5, 6], odkud byly definice převzaty.

Definice 2.2.1. Relací $\mathcal{R}$ na množině M nazýváme libovolnou podmnožinu kartézského součinu $M \times M$. Relaci $\mathcal{R}$ na množině M nazýváme **ekvivalencí** na množině M, právě když platí následující podmínky:

i) reflexivita: pro každé $x \in M$ platí $(x, x) \in \mathcal{R}$,

ii) symetrie: pokud $(x, y) \in \mathcal{R}$, pak $(y, x) \in \mathcal{R}$,

iii) tranzitivita: pokud $(x, y) \in \mathcal{R}$ a $(y, z) \in \mathcal{R}$, potom $(x, z) \in \mathcal{R}$.

Množinu všech prvků ekvivalentních s $x \in M$ značíme

$$[x] := \{y \in M \mid (x, y) \in \mathcal{R}\},$$

případně $[x]_{\mathcal{R}}$, chceme-li zvýraznit, o jaké relaci na množině M mluvíme, a nazýváme ji **třídou prvků ekvivalentních s** x.

Definice 2.2.2. Množinu G vybavenou jednou binární operací $\cdot : G \times G \to G$, která splňuje podmínky

i) asociativity, tj. $a \cdot (b \cdot c) = (a \cdot b) \cdot c$ platí pro každé $a, b, c \in G$,

ii) existence neutrálního prvku $e \in G$ vůči grupové operaci $\cdot$, tj. $a \cdot e = e \cdot a = a$ pro každé $a \in G$,

iii) existence inverzních prvků, tj. ke každému $a \in G$ existuje prvek $a^{-1} \in G$ takový, že $a \cdot a^{-1} = a^{-1} \cdot a = e$ pro každé $a \in G$,

nazýváme **grupou** (*group*).

Pokud navíc pro každé $a, b \in G$ platí rovnost $a \cdot b = b \cdot a$, potom grupu G nazýváme **abelovskou** (nebo komutativní). Grupa G je **konečná**, má-li množina G konečný počet prvků. Počet prvků konečné grupy G nazýváme **řádem grupy** G (*order of the group*) a značíme ho symbolem $\#G$.

Značení. Buď G grupa. Pro celé číslo $n \in \mathbb{Z}$ a $g \in G$ zavedeme následující značení:

$$g^n = \begin{cases} \overbrace{g \cdot g \cdot \cdots \cdot g}^{n \text{ členů}}, & \text{pokud} \quad n > 0 \\ e, & \text{pokud} \quad n = 0 \\ \overbrace{g^{-1} \cdot g^{-1} \cdot \cdots \cdot g^{-1}}^{|n| \text{ členů}}, & \text{pokud} \quad n < 0 \end{cases}$$

Definice 2.2.3. Buď $(G, \cdot)$ grupa a H podmnožina G. Pokud H spolu s operací $\cdot$ tvoří grupu, pak o grupě $(H, \cdot)$ mluvíme jako o **podgrupě** grupy $(G, \cdot)$ (*subgroup*).

Definice 2.2.4. Nechť G je grupa. **Řádem prvku** $a \in G$ rozumíme nejmenší přirozené číslo n takové, že $a^n = e$. Pokud takové číslo neexistuje, pak řekneme, že řád a je nekonečný.

Poznámka. Řád neutrálního prvku v jakékoliv grupě je 1. Žádný další prvek takový řád nemá.

Věta 2.2.1. Buďte G grupa řádu $n \in \mathbb{N}$ a $a \in G$. Potom $a^n = e$.

Poznámka. Řád prvku je dělitelem řádu grupy.

Definice 2.2.5. Buď G grupa a $m \in \mathbb{N}$. Množinu všech prvků $g \in G$ řádu dělícího m označíme symbolem $G[m]$ a nazýváme **množinou m-torzních bodů**. Přesněji klademe

$$G[m] = \{g \in G \mid g^m = e\}.$$

Definice 2.2.6. Zobrazení $f : G \to H$ grupy $(G, \cdot)$ do grupy $(H, \odot)$ splňující $f(a \cdot b) = f(a) \odot f(b)$ pro každé $a, b \in G$ nazýváme **homomorfizmem** grup G a H (*group homomorphism*).

Pokud navíc platí $H = G$, pak f nazýváme **endomorfizmem** (*endomorphism*).

Homomorfismus f, který je bijektivní (prostý a na), nazýváme **izomorfizmem** G a H (*isomorphism*).

Definice 2.2.7. Grupy, mezi kterými existuje izomorfizmus, se nazývají **izomorfní**.

Definice 2.2.8. Jádrem homomorfizmu (*kernel*) $f : G \to H$, kde G a H jsou grupy, nazýváme množinu

$$\ker f := \{a \in G \mid f(a) = e\},$$

kde e je neutrální prvek H.

Definice 2.2.9. Grupu G nazýváme **cyklickou** (*cyclic group*), pokud existuje prvek $a \in G$ takový, že pro libovolné $b \in G$ existuje celé číslo $n \in \mathbb{Z}$ splňující $b = a^n$. Tento prvek a pak nazýváme **generátorem** grupy G.

Definice 2.2.10. Okruhem (*ring*) R nazýváme množinu R se dvěma binárními operacemi $+ : R \times R \to R$ a $\cdot : R \times R \to R$ splňujícími

i) R je abelovská grupa vůči $+$,

ii) operace $\cdot$ je asociativní,

iii) platí distributivita násobení vůči sčítání, tj. pro každé $a, b, c \in R$ je

$$a \cdot (b + c) = (a \cdot b) + (a \cdot c) \quad \text{a} \quad (a + b) \cdot c = (a \cdot c) + (b \cdot c).$$

Značení. Neutrální prvek grupy $(R, +)$ budeme značit 0.

Značení. Buď R okruh. Pro přirozené číslo $n \in \mathbb{N}$ a $r \in R$ zavedeme následující značení:

$$nr = \overbrace{r + \cdots + r}^{n \text{ členů}}$$

Definice 2.2.11. Buď R okruh. Pokud existuje přirozené číslo n takové, že $nr = 0$ pro každé $r \in R$, pak nejmenší takové n nazýváme **charakteristikou** okruhu R (*characteristic*). O takovém okruhu R pak říkáme, že má kladnou charakteristiku. Pokud žádné takové n neexistuje, pak o R mluvíme jako o okruhu s charakteristikou 0.

Definice 2.2.12. Zobrazení $f : R \to S$ okruhu R do okruhu S nazýváme **homomorfizmem** (*homomorphism*) právě tehdy, když platí

$$f(a + b) = f(a) + f(b) \quad \text{a} \quad f(ab) = f(a)f(b)$$

pro každé $a, b \in R$. **Jádrem homomorfizmu** (*kernel*) okruhů nazýváme množinu

$$\ker f = \{a \in \mathbb{R} \mid f(a) = 0 \in S\}.$$

Pokud navíc platí, že f je bijektivní (prosté a na), pak f nazýváme **izomorfizmem** R a S (*isomorphism*).

Definice 2.2.13. Okruhy, mezi kterými existuje izomorfizmus, se nazývají **izomorfní**.

Definice 2.2.14. Okruhy můžeme dále klasifikovat podle následujících dodatečných vlastností.

i) Okruh nazýváme **okruhem s jedničkou** (*ring with identity*), pokud v něm existuje neutrální prvek vůči násobení $\cdot$. Tento prvek budeme značit 1.

ii) Okruh nazýváme **komutativní** (*commutative*), pokud je násobení $\cdot$ komutativní.

iii) Okruh nazýváme **oborem integrity** (*integral domain*), pokud je komutativním okruhem s jedničkou $1 \neq 0$ a z rovnosti $ab = 0$ plyne $a = 0$ nebo $b = 0$ (často též říkáme, že v oboru integrity neexistují netriviální dělitelé nuly).

iv) Okruh nazýváme **okruhem s dělením** (*division ring*), pokud jeho nenulové prvky spolu s operací násobení $\cdot$ tvoří grupu.

Definice 2.2.15. Komutativní okruh s dělením nazýváme **tělesem** (*field*). **Multiplikativní grupu** tělesa T označujeme symbolem T^*, tedy $T^* = (T \setminus \{0\}, \cdot)$.

Těleso T je **konečné**, má-li množina T konečný počet prvků.

Poznámka. Měli bychom si uvědomit, že každé těleso je zároveň i okruhem, takže definice týkající se okruhů se vztahují i na tělesa.

Definice 2.2.16. Buď T těleso. Podmnožinu K množiny T, jež spolu s operacemi z tělesa T opět tvoří těleso, nazýváme **podtělesem** tělesa T (*subfield*). O tělese T pak také mluvíme jako o **rozšíření** tělesa K (*extension*). Pokud $K \neq T$, pak K nazýváme **vlastním podtělesem** tělesa T.

Definice 2.2.17. Buď L rozšíření tělesa K. Uvažujeme-li L jakožto vektorový prostor nad K a má-li tento prostor konečnou dimenzi, pak L nazýváme **konečným rozšířením** tělesa K (*finite extension*). Dimenze tohoto vektorového prostoru se nazývá **stupněm** L nad K a značí se $[L : K]$.

Věta 2.2.2. Průnik všech podtěles tělesa T je opět tělesem a nazýváme ho **prvopodtělesem** tělesa T.

Věta 2.2.3. Konečné těleso má prvočíselnou charakteristiku.

Věta 2.2.4. Buď T konečné těleso. Potom T má p^n prvků, kde p je prvočíselná charakteristika tělesa T a přirozené číslo n je stupeň T nad svým prvopodtělesem. Počet prvků tělesa nazýváme **řádem tělesa** (*order of finite field*).

Definice 2.2.18. Těleso $\mathbb{Z}/p\mathbb{Z}$, p prvočíslo, je izomorfní se známým tělesem

$$(\{0, 1, \ldots, p-1\},\ + \bmod p,\ \cdot \bmod p)$$

a nazýváme ho **Galoisovým tělesem** $GF(p)$ řádu p (*Galois field of order p*).

Značení. Konečné těleso řádu p^n značíme $GF(p^n)$ nebo $\mathbb{F}_{p^n}$.

Definice 2.2.19. Číslo $a \in Z_p^\times$ nazveme **kvadratické reziduum modulo** p (*quadratic residue modulo p*), pokud má rovnice

$$x^2 \equiv a \pmod{p}$$

řešení. V opačném případě jej nazveme **kvadratické nereziduum modulo** p (*quadratic nonresidue modulo p*).

Poznámka. Množina $\{k \in \mathbb{Z} \mid 1 \leq k \leq n-1,\ \gcd(k, n) = 1\}$ s operací násobení modulo n tvoří grupu, kterou značíme symbolem $\mathbb{Z}_n^\times$ a mluvíme o ní jako o modulární multiplikativní grupě.

Definice 2.2.20. Nechť T je konečné těleso. Prvek $a \in T$ nazveme **čtvercovým prvkem** (*square element*), pokud má rovnice

$$x^2 = a$$

v T řešení. V opačném případě jej nazveme **nečtvercovým prvkem** (*non-square element*).

Definice 2.2.21. Buď R libovolný okruh. **Polynomem** (*polynomial*) nad R nazýváme formální výraz tvaru

$$p(x) = \sum_{k=0}^{n} \alpha_k x^k = \alpha_0 + \alpha_1 x + \cdots + \alpha_n x^n,$$

kde n je nezáporné celé číslo, $\alpha_k \in \mathbb{R},\ k = 0, 1, \ldots, n$, jsou koeficienty polynomu $p(x)$ a x formální proměnná.

Nejvyšší n, pro které platí $\alpha_n \neq 0$, nazýváme **stupněm** polynomu (*degree*) a značíme $\deg p(x)$. Jsou-li všechny koeficienty $\alpha_0, \ldots, \alpha_n$ nulové, nazýváme $p(x)$ **nulovým polynomem** a jeho stupeň nedefinujeme.

Definice 2.2.22. Buď K okruh. Potom množina polynomů s koeficienty z tohoto okruhu spolu s operacemi sčítání a násobení definovanými jako

$$\sum_{i=0}^{n} a_i x^i + \sum_{i=0}^{n} b_i x^i = \sum_{i=0}^{n} (a_i + b_i) x^i$$

$$\left(\sum_{i=0}^{n} a_i x^i \right) \cdot \left(\sum_{i=0}^{m} b_i x^i \right) = \sum_{i=0}^{n+m} \left(\sum_{j+k=i} a_j b_k \right) x^i$$

tvoří **komutativní okruh polynomů nad okruhem** K. Tento okruh značíme $K[x]$.

Definice 2.2.23. Buď $p(x) \in K[x]$ stupně alespoň 1. Řekneme, že $p(x)$ je **ireducibilní** nad K, jestliže pro každé dva polynomy $a(x)$ a $b(x)$ z $K[x]$ platí

$$a(x) \cdot b(x) = p(x) \Rightarrow (\deg a(x) = 0 \ \vee \ \deg b(x) = 0).$$

Věta 2.2.5. Dělení polynomu zavedeme takto:

Buď $g(x) \neq 0$ polynom nad tělesem T. Potom pro každé $f(x) \in T[x]$ existují jednoznačně určené polynomy $q(x), r(x) \in T[x]$ splňující

$$f(x) = q(x)g(x) + r(x), \quad \text{kde } \deg r(x) < \deg g(x).$$

Polynom $r(x)$ nazveme zbytkem po dělení polynomu $f(x)$ polynomem $g(x)$ a polynom $q(x)$ nazveme částečný podíl.

Věta 2.2.6. Nechť $a(x)$, $p(x)$ jsou dva polynomy nad tělesem T. Počítání **modulo polynom** zavedeme takto:

$$a(x) \pmod{p(x)} = \text{ zbytek po dělení } a(x) \text{ polynomem } p(x).$$

Definice 2.2.24. Podmnožinu S okruhu R nazýváme **podokruhem** (*subring*) okruhu R pokud S vůči operacím $+$ a $\cdot$ definovaným na R tvoří okruh.

Definice 2.2.25. Podmnožinu J množiny R nazýváme **ideálem** (*ideal*) okruhu R pokud je J podokruhem okruhu R a pro každé $a \in J$ a $b \in R$ platí $ab \in J$ a $ba \in J$.

Definice 2.2.26. Buď R komutativní okruh. Nejmenší ideál okruhu R (ve smyslu inkluze) obsahující prvek $a \in R$ značíme symbolem (a). Symbolicky můžeme tuto definici zapsat následovně

$$(a) = \bigcap_{\substack{J \text{ ideál } R \\ a \in J}} J.$$

Ideál J okruhu R se nazývá **hlavní ideál** (*principal ideal*), pokud existuje $a \in R$ splňující $J = (a)$. V tomto případě také říkáme, že ideál J je **generovaný** prvkem a.

Věta 2.2.7. Je-li R komutativní okruh a $a \in R$, pak $(a) = \{ra + na \mid r \in R, n \in \mathbb{Z}\}$. Pokud R navíc obsahuje jedničku, pak $(a) = \{ra \mid r \in R\}$.

Věta 2.2.8. Nechť J je ideálem okruhu R. Potom můžeme množinu R rozložit na disjunktní **rozkladové třídy** okruhu R vzhledem k ideálu J. Pro tyto třídy $[a]$, $a \in R$, platí

$$[a] = \{a + j \mid j \in J\}.$$

Rozkladové třídy můžeme též značit $[a] = a + J$.

Věta 2.2.9. Množina rozkladových tříd okruhu R vzhledem k ideálu J spolu s operacemi

$$(a + J) + (b + J) := (a + b) + J,$$
$$(a + J) \cdot (b + J) := ab + J,$$

kde $a, b \in R$, tvoří okruh.

Poznámka. Výše uvedené operace můžeme zapsat též jako:

$$[a] + [b] := [a + b],$$
$$[a] \cdot [b] := [ab].$$

Definice 2.2.27. Množina rozkladových tříd okruhu R vzhledem k ideálu J s operacemi zavedenými v předchozí větě se nazývá **faktorokruh** (*factor ring, quotient ring*) okruhu R vzhledem k ideálu J a značí se R/J.

Věta 2.2.10. Je-li $f(x) \in T[x]$, potom okruh rozkladových tříd $T[x]/(f(x))$ je tělesem, právě když $f(x)$ je ireducibilní polynom nad tělesem T.

Poznámka. Příkladem takového tělesa může být například $\mathbb{F}_{191}[X]/(X^2+1)$, kde $\mathbb{F}_{191}[X]$ je komutativní okruh polynomů nad tělesem $\mathbb{F}_{191}$ a X^2+1 je polynom ireducibilní nad $\mathbb{F}_p$.

Definice 2.2.28. Prvek b z tělesa T nazýváme **kořenem** (*root*) polynomu $f(x) \in T[x]$ pokud $f(b) = 0$.

Následující věty a definice přejímám z jiného zdroje [7].

Definice 2.2.29. Buď S rozšířením tělesa T a $a \in S$. Řekneme, že prvek a je **algebraický** nad T, pokud existuje polynom z $T[x]$, jehož je a kořenem. V opačném případě se prvek a nazývá **transcendentní** nad T. Je-li každý prvek tělesa S algebraický nad T, hovoříme o **algebraickém rozšíření**.

Definice 2.2.30. Těleso T se nazývá **algebraicky uzavřené** (*algebraically closed*), pokud každý polynom z $T[x]$ stupně ≥ 1 má v tělese T alespoň jeden kořen.

Definice 2.2.31. Řekneme, že S rozšíření tělesa T je **algebraický uzávěr** tělesa T, pokud je S algebraicky uzavřené těleso a zároveň je algebraickým rozšířením tělesa T. Algebraický uzávěr tělesa T značíme $\overline{T}$.

2.3 Vektorové prostory

Pro pochopení algoritmu výměny klíčů založené na isogeniích supersingulárních eliptických křivek je nutné uvést několik definic týkajících se vektorových (lineárních) prostorů. Zdrojem, ze kterého čerpám, je [5].

Definice 2.3.1. Buďte V množina, T těleso a mějme dvě operace $+ : V \times V \to V$ a $\cdot : T \times V \to V$. V nazýváme **vektorovým prostorem nad tělesem** T pokud platí následující podmínky.

i) $(V, +)$ je abelovská grupa,

ii) pro každé $x \in V$ platí $1 \cdot x = x$.

iii) (kompatibilita násobení) pro každé $\alpha, \beta \in T$ a $x \in V$ platí $\alpha \cdot (\beta \cdot x) = (\alpha\beta) \cdot x$,

iv) (distributivita) pro každé $\alpha, \beta \in T$ a $x, y \in V$ platí

$$(\alpha + \beta) \cdot x = (\alpha \cdot x) + (\beta \cdot x), \quad \alpha \cdot (x + y) = (\alpha \cdot x) + (\alpha \cdot y).$$

Definice 2.3.2. Lineární kombinací vektorů $x_1, x_2, \dots x_n \in V$ nazýváme vektor

$$\sum_{k=1}^{n} \alpha_k x_k \in V,$$

kde $\alpha_1, \alpha_2, ..., \alpha_n \in T$ jsou koeficienty lineární kombinace. Lineární kombinaci nazýváme **triviální** pokud $\alpha_k = 0$ pro každé $k = 1, 2, \dots, n$. V opačném případě mluvíme o **netriviální** lineární kombinaci.

Definice 2.3.3. Vektory $x_1, x_2, \dots, x_n$ nazýváme lineárně nezávislé, právě když pouze jejich triviální lineární kombinace dává 0. Tj. když z rovnosti

$$\sum_{k=1}^{n} \alpha_k x_k = 0, \quad \alpha_k \in T, \quad k \ = 1, 2, \dots, n,$$

plyne $\alpha_k = 0$ pro každé $k = 1, 2, \dots, n$.

Definice 2.3.4. Lineárním obalem množiny vektorů $M \subset V$ nazýváme množinu všech lineárních kombinací vektorů z množiny M a značíme $\langle M \rangle$. Platí tedy

$$\langle M \rangle = \left\{ \sum_{k=1}^{n} \alpha_k x_k \;\middle|\; n \in \mathbb{N}, \quad x_k \in M, \quad \alpha_k \in T, \quad k \ = 1, 2, \dots, n \right\}.$$

O množině $M \subset V$ říkáme, že **generuje** V, právě když $\langle M \rangle = V$.

Definice 2.3.5. Bází vektorového prostoru V nazýváme libovolnou uspořádanou n-tici $(x_1, x_2, \dots, x_n)$ vektorů z V, pro něž platí

i) množina $x_1, x_2, ..., x_n$ generuje V,

ii) vektory $x_1, x_2, ..., x_n$ jsou lineárně nezávislé.

Číslo n nazýváme **dimenzí** prostoru V. Pokud pro každé $n \in \mathbb{N}$ ve V existuje n lineárně nezávislých vektorů, říkáme že V má **nekonečnou dimenzi**. Dimenzi prostoru V značíme $\dim V$.

2.4 Eliptické křivky a isogenium

V této kapitole uvádím několik definic a vět týkajících se eliptických křivek a isogenií. Definice byly převzaty z [4, 5].

Definice 2.4.1. Zjednodušená Weierstrassova rovnice eliptické křivky:
Eliptickou křivkou nad Galoisovým tělesem $GF(p)$, $p \geq 3$, nazýváme množinu

$$E(GF(p)) = \Big\{ (x, y) \mid x, y \in GF(p), \quad y^2 = x^3 + ax + b \Big\} \cup \{O\},$$

kde $a, b \in GF(p)$ splňují $4a^3 + 27b^2 \neq 0$.

Definice 2.4.2. Zobecněná Weierstrassova rovnice:

Eliptickou křivkou (*elliptic curve*) E nad konečným tělesem $GF(q^k)$, q je prvočíslo, $k \geq 1$, nazýváme všechna řešení (x, y) rovnice

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \quad x, y \in GF(q^k),$$

spolu s bodem v nekonečnu O. Po koeficientech $a_1, \ldots, a_5, a_6$ se vyžaduje, aby splňovaly podmínku $\Delta \neq 0$, kde

$$\begin{aligned} &b_2 = a_1^2 + 4a_2, \quad b_4 = 2a_4 + a_1a_3, \quad b_6 = a_3^2 + 4a_6, \\ &b_8 = a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2, \\ &\Delta = -b_2^2b_8 - 8b_4^3 - 27b_6^2 + 9b_2b_4b_6. \end{aligned}$$

Poznámka. Uvažujeme-li zjednodušenou Weierstrassovu rovnici eliptické křivky ve tvaru $E : y^2 = x^3 + ax + b$, pak můžeme takovou křivku vyjádřit i ve tvaru tzv. **Montgomeryho křivky**, kterou můžeme vyjádřit rovnicí $M_{A,B} : By^2 = x^3 + Ax^2 + x$. Oba tvary křivky jsou mezi sebou navzájem převoditelné [9].

Definice 2.4.3. Veličinu Δ z předchozí definice nazýváme **diskriminantem** Weierstrassovy rovnice (*discriminant*). Veličinu j definovanou následovně

$$j = \frac{(b_2^2 - 24b_4)^3}{\Delta}$$

nazýváme **j-invariantem** eliptické křivky (*j-invariant*).

Poznámka. Pro každou charakteristiku tělesa existuje jen konečný počet možných j-invariantů supersingulárních eliptických křivek.

Definice 2.4.4. Sčítání $\oplus$ na eliptické křivce E zadané pomocí zobecněné Weierstrassovy rovnice je definováno následovně. Bod O se chová jako neutrální prvek vůči $\oplus$. Buďte $P, Q \in E$, $P = (p_1, p_2)$ a $Q = (q_1, q_2)$. Potom

i) pokud $p_1 = q_1$ a $p_2 + q_2 + a_1q_1 + a_3 = 0$, pak $P \oplus Q = O$,

ii) jinak pokud $p_1 \neq q_1$, klademe

$$\lambda = \frac{q_2 - p_2}{q_1 - p_1} \qquad \nu = \frac{p_2q_1 - q_2p_1}{q_1 - p_1},$$

a pokud $p_1 = q_1$, klademe

$$\lambda = \frac{3p_1^2 + 2a_2p_1 + a_4 - a_1p_2}{2p_2 + a_1p_1 + a_3} \qquad \nu = \frac{-p_1^3 + a_4p_1 + 2a_6 - a_3p_2}{2p_2 + a_1p_1 + a_3}$$

a potom

$$P \oplus Q = (r_1, r_2) = \left(\lambda^2 + a_1\lambda - a_2 - p_1 - q_1,\ -(\lambda + a_1)r_1 - \nu - a_3\right).$$

Věta 2.4.1. Buď E eliptická křivka. Operace sčítání zavedená v předchozí definici má následující vlastnosti:

i) (neutrální prvek) $P \oplus O = O \oplus P = P$ pro každé $P \in E$.

ii) (inverze) pro každé $P \in E$, $P = (p_1, p_2)$, je $P \oplus (\ominus P) = O$, kde $\ominus P = (p_1, -p_2 - a_1 p_1 - a_3)$.

iii) (asociativita) $(P \oplus Q) \oplus R = P \oplus (Q \oplus R)$ pro každé $P, Q, R \in E$.

iv) (komutativita) $P \oplus Q = Q \oplus P$ pro každé $P, Q \in E$.

Poznámka. Jinak řečeno, množina E spolu s operací $\oplus$ tvoří abelovskou grupu.

Značení. Pro body P, Q na eliptické křivce E zavedeme následující značení:

$$P \ominus Q = P \oplus (\ominus Q)$$

Značení. Pro $m \in \mathbb{Z}$ a $P \in E$ budeme používat následující značení:

$$[m]\,P = \begin{cases} \overbrace{P \oplus \cdots \oplus P}^{m \text{ členů}}, & \text{pokud} \quad m > 0 \\ \overbrace{\ominus P \ominus \cdots \ominus P}^{|m| \text{ členů}}, & \text{pokud} \quad m < 0 \\ [0]\,P = O, & \text{pokud} \quad m = 0 \end{cases}$$

Definice 2.4.5. Počet bodů na eliptické křivce E včetně bodu v nekonečnu O nazýváme **řádem či kardinalitou eliptické křivky** (*order or cardinality of elliptic curve*) a značíme $\#E$.

Definice 2.4.6. Nechť E_1 a E_2 jsou eliptické křivky. Potom **isogeniem** (*isogeny*) z E_1 do E_2 nazýváme homomorfizmus $\phi : E_1 \to E_2$ splňující $\phi(O) = O$.

V celé této práci se zabývám pouze tzv. separabilními isogenii (*separable isogeny*), pro něž platí, že jejich **stupeň** je roven velikosti jejich jádra [1]. Definici separabilního isogenia můžete nalézt například v [4].

Definice 2.4.7. Nechť E je eliptická křivka a $m \in \mathbb{N}$. **M-torzní podgrupa** eliptické křivky E (*m-torsion subgroup*) označovaná jako $E[m]$ je množina bodů eliptické křivky E řádu dělícího m,

$$E[m] = \{P \in E \mid [m]\,P = O\}.$$

Bod $Q \in E[m]$ nazýváme **m-torzním bodem** (*m-torsion point*).

Definice 2.4.8. Nechť E je eliptická křivka nad tělesem K s charakteristikou p. Eliptickou křivku E nazýváme **supersingulární** (*supersingular elliptic curve*) právě tehdy, když pro nějaké $r \geq 1$ platí

$$E[p^r] = O.$$

Poznámka. Jinak řečeno, eliptická křivka je supersingulární právě tehdy, když množina bodů řádu p je triviální.

Poznámka. Pokud platí pro nějaké $r \geq 1$ výše uvedená podmínka, potom platí tato podmínka pro všechna taková r.

Definice 2.4.9. Ekvivalentní definice supersingulární křivky
Eliptická křivka $E(GF(p^l))$, p prvočíslo, $l \in \mathbb{N}$, se nazývá **supersingulární**, pokud platí

$$\#E(GF(p^l)) \equiv 1 \pmod{p}$$

Definice 2.4.10. Buď R okruh s dělením. Uvažme množinu $R^3 \setminus \{(0,0,0)\}$ vybavenou relací ekvivalence mezi jejími prvky x, y definovanou předpisem

$$x \sim y \Leftrightarrow (\exists r \in R)(x = ry).$$

Množinu tříd ekvivalentních prvků značíme $P_2(R)$ a nazýváme **projektivní rovinou** nad R. Třídu všech prvků ekvivalentních s $(x_1, x_2, x_3) \in R^3 \setminus \{(0,0,0)\}$ značíme $(x_1 : x_2 : x_3)$.

Každá třída ekvivalence představuje přímku v původním trojrozměrném prostoru R^3, tj. pokud $x = (x_1, x_2, x_3) \in R^3$ je nenulový, pak

$$(x_1 : x_2 : x_3) = \{rx \mid r \in R\}.$$

Každá třída $[x]$ je jednoznačně udána trojicí (x_1, x_2, x_3), kterou nazýváme **homogenními souřadnicemi** příslušného bodu.

Definice 2.4.11. Racionální funkcí (*rational function*) f jedné proměnné x nad tělesem $\mathbb{C}$ nazýváme podíl polynomů nad $\mathbb{C}$ s proměnnou x,

$$f(x) = \frac{a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n}{b_0 + b_1 x + b_2 x^2 + \cdots + b_m x^m},$$

kde $a_0, a_1, \ldots, a_n, b_0, b_1, \ldots, b_n \in \mathbb{C}$. Předpokládejme, že polynomy v čitateli a jmenovateli lze faktorizovat. Potom lze f upravit do tvaru

$$f(x) = \frac{a(x-\alpha_1)^{e_1}(x-\alpha_2)^{e_2}\cdots(x-\alpha_r)^{e_r}}{b(x-\beta_1)^{d_1}(x-\beta_2)^{d_2}\cdots(x-\beta_s)^{d_s}},$$

kde lze předpokládat, že $\alpha_1, \ldots, \alpha_r, \beta_1, \ldots, \beta_s$ jsou vzájemně různá (jinak bychom vždy mohli některé členy krátit). Koeficienty $\alpha_1, \ldots, \alpha_r \in \mathbb{C}$ nazýváme **nulami** (*zeros*) f, $\beta_1, \ldots, \beta_s \in \mathbb{C}$ nazýváme **póly** (*poles*) a exponenty $e_1, \ldots, e_r, d_1, \ldots, d_s$ pak násobnostmi příslušné nuly nebo pólu.

Nuly, póly a násobnosti racionální funkce f můžeme zaznamenat pomocí tzv. **divizoru** f, který definujeme jako formální sumu

$$\mathrm{div}(f(x)) = [e_1]\alpha_1 + [e_2]\alpha_2 + \cdots + [e_r]\alpha_r - [d_1]\beta_1 - [d_2]\beta_2 - \cdots - [d_r]\beta_r.$$

Uvažme nyní eliptickou křivku E nad jistým tělesem K. Buď nyní $f(x, y, z)$ racionální funkce definovaná na E, tj. $(x : y : z) \in P_2(K)$ uvažujeme z E. Body, kde je čitatel f nulový (v K) opět nazýváme nulami a body kde je nulový jmenovatel (v K) nazýváme póly a opět mluvíme o jejich násobnosti. Tudíž i v tomto případě definujeme divizor funkce f jako formální výraz

$$\mathrm{div}(f) = \sum_{P \in E} [n_P]P,$$

kde $n_P \in \mathbb{Z}$ je násobnost pólu/nuly i s příslušným znaménkem. Protože se zabýváme divizory racionálních funkcí je tato suma nulová pouze pro konečný počet bodů z E.

Definice 2.4.12. Buďte $P, Q \in E[m]$, tj. P a Q splňují $[m]P = [m]Q = O$ na eliptické křivce E. Nechť f_P a f_Q jsou racionální funkce na E splňující

$$\mathrm{div}(f_P) = [m]P - [m]O \quad \text{a} \quad \mathrm{div}(f_Q) = [m]Q - [m]O.$$

Weilovo párování bodů P a Q je definováno předpisem

$$e_m(P, Q) := \frac{f_P(Q \oplus S)}{f_P(S)} \Big/ \frac{f_Q(P \ominus S)}{f_Q(\ominus S)},$$

kde $S \in E$ je libovolný bod na křivce E splňující $S \notin \{O,\ P,\ \ominus Q,\ P \ominus Q\}$. Tímto způsobem je definováno zobrazení

$$E(GF(p))[m] \times E(GF(p))[m] \to GF(p)^*.$$

Následující definice a věty jsou převzaty z jiných zdrojů [10, 11, 12].

Definice 2.4.13. Nechť E je eliptická křivka dána rovnicí $y^2 = x^3 + ax + b$ definovaná nad konečným tělesem $\mathbb{F}_{p^n}$. Dále mějme $0 \neq d \in \mathbb{F}_{p^n}$ nečtvercový prvek. Řekneme, že eliptická křivka $E^{(d)}$ je **kvadratickým twistem** křivky E, pokud je dána rovnicí

$$y^2 = x^3 + ad^2x + bd^3.$$

Poznámka. Z definice j-invariantu vyplývá, že křivka E i její **kvadratický twist** $E^{(d)}$ mají shodný j-invariant.

Věta 2.4.2. (Mestre) Nechť E je eliptická křivka definovaná nad prvočíselným tělesem $\mathbb{F}_p$ a $\#E = p + 1 - t$, $t \in \mathbb{Z}$. Označme $E^{(d)}$ eliptickou křivku, která je kvadratickým twistem křivky E. Pak pro $\#E^{(d)}$ platí

$$\#E^{(d)} = p + 1 + t.$$

Ekvivalentně můžeme toto interpretovat jako skutečnost, že součet bodů E a $E^{(d)}$ je konstantní a platí

$$\#E + \#E^{(d)} = 2(p+1).$$

Věta 2.4.3. Nechť $\mathbb{F}_q$ je konečné těleso, kde $q = r^2$ je sudá mocnina nějakého prvočísla. Potom existují supersingulární eliptické křivky E a její kvadratický twist $E^{(d)}$ nad $\mathbb{F}_q$, pro které platí

$$E(\mathbb{F}_q) \cong (\mathbb{Z}/(r-1)\mathbb{Z})^2 \quad \text{a} \quad E^{(d)}(\mathbb{F}_q) \cong (\mathbb{Z}/(r+1)\mathbb{Z})^2$$

2.5 Další teorie

Kromě matematické teorie uvedené v sekcích 2.2, 2.3 a 2.4 je potřeba uvést ještě některé definice a věty týkající se jiných matematických oblastí.

Definice 2.5.1. Číslo $a \in \mathbb{Z}$ nazveme **čtvercovým číslem** (*square number, perfect square*), pokud má rovnice

$$x^2 = a$$

v $\mathbb{Z}$ řešení. V opačném případě jej nazveme **nečtvercovým číslem** (*non-square number*).

Definice 2.5.2. Číslo $a \in \mathbb{Z}$ nazveme **bezčtvercovým číslem** či **číslem beze čtverců** (*square-free number*), pokud pro všechna prvočísla p platí $p^2 \nmid n$.

Definice 2.5.3. [13] Celé nenulové číslo Δ nazveme **fundamentálním diskriminantem** (*fundamental discriminant*), pokud $\Delta \equiv 1 \pmod 4$ a Δ je beze čtverců, nebo pokud je Δ dělitelné čtyřmi, $\Delta/4 \equiv 2 \pmod 4$ nebo $\Delta/4 \equiv 3 \pmod 4$ a $\Delta/4$ je beze čtverců.

Dále ještě v této práci používám pojem **polynom Hilbertovy třídy**, jehož defnici a způsob výpočtu můžete nalézt v [14].

2.6 Algoritmus výměny klíčů

Výměna klíčů založená na isogeniích supersingulárních eliptických křivek je obdobou Diffie–Hellmanova schématu (dále jen D-H) výměny klíčů. První myšlenka založit bezpečnost výměny klíčů na isogeniích supersingulárních eliptických křivek pochází z roku 2011 a zveřejnili ji David Jao a Luca De Feo [1].

2.6.1 Základní algoritmus

V této podsekci předpokládám dvě komunikující strany, které nazývám pro zjednodušení Alice a Bob.

Mějme prvočíslo p dáno vzorcem $l_A^{e_A} \cdot l_B^{e_B} \cdot f \pm 1$, kde l_A a l_B jsou malá prvočísla, e_A a e_B jsou přirozená čísla a f je kofaktor zvolený tak, aby p bylo prvočíslo.

Příklad 1. Mějme $l_A = 2$, $l_B = 3$, $e_A = 6$, $e_B = 1$ a $f = 1$. Potom $p = 2^6 \cdot 3^1 \cdot 1 - 1 = 191$.

Dále potřebujeme konečné těleso $\mathbb{F}_{p^2}$ a supersingulární eliptickou křivku E_0 definovanou nad tímto tělesem, jejíž kardinalita bude $(p \mp 1)^2 = (l_A^{e_A} \cdot l_B^{e_B} \cdot f)^2$

Příklad 2. Jedna z možných supersingulárních křivek s kardinalitou $\#E_0 = (2^6 \cdot 3^1 \cdot 1)^2 = 36\ 864$ je

$$E_0 : Y^2 = X^3 + (152z + 187)X + (35z + 81),$$

kde pro koeficienty zjednodušené Weierstrassovy rovnice platí $a = 152z + 187, b = 35z + 81 \in \mathbb{F}_{p^2}$.

Alice (resp. Bob) nalezne libovolnou bázi $\{P_A, Q_A\}$ (resp. $\{P_B, Q_B\}$) generující $E_0[l_A^{e_A}]$ (resp. $E_0[l_B^{e_B}]$), tedy $\langle P_A, Q_A \rangle = E_0[l_A^{e_A}]$ (resp. $\langle P_B, Q_B \rangle = E_0[l_B^{e_B}]$). Bázi si navzájem obě strany vymění.

Poznámka: Oba body báze Alice P_A i Q_A musí mít řád $l_A^{e_A}$ a musí být nezávislé. Analogická podmínka platí i pro Bobovu bázi.

Příklad 3. Jedna z možných bází generujících $E_0[l_A^{e_A}]$ je

$$\{P_A, Q_A\} = \{(120z + 175 : 174z + 15 : 1), (12z + 28 : 43z + 142 : 1)\}.$$

Pro $E_0[l_B^{e_B}]$ to může být například báze

$$\{P_B, Q_B\} = \{(24z + 36 : 6z + 183 : 1), (139z + 23 : 3z + 149 : 1)\}.$$

Dále Alice zvolí dva náhodné prvky $m_A, n_A \in_R \mathbb{Z}/l_A^{e_A}\mathbb{Z}$ tak, aby alespoň jeden nebyl dělitelný číslem l_A. Následuje výpočet isogenia $\phi_A : E_0 \to E_A$ s jádrem $K_A := \langle [m_A]P_A + [n_A]Q_A \rangle$. Alice také musí vypočíst obrazy Bobovy báze $\{P_B, Q_B\}$ ve svém isogeniu, tedy $\{\phi_A(P_B), \phi_A(Q_B)\} \subset E_A$. Stejné kroky provede i Bob.

Poznámka: Velikost jádra a tedy i stupeň vypočteného isogenia Alice je vždy $l_A^{e_A}$. To je způsobeno tím, že jsme m_A a n_A volili tak, aby alespoň jeden tento prvek nebyl dělitelný číslem l_A. Tedy i prvek $[m_A]P_A + [n_A]Q_A$ bude mít řád $l_A^{e_A}$. Stejně i pro Boba.

Příklad 4.

$$\begin{aligned} m_A &= 0, \\ n_A &= 17, \\ [m_A]P_A + [n_A]Q_A &= (110z + 103 : 68z + 87 : 1), \\ E_A : Y^2 &= X^3 + (4z + 129)X + (87z + 190), \\ \phi_A(P_B) &= (124z + 172 : 15z + 61 : 1), \\ \phi_A(Q_B) &= (95z + 62 : 10z + 75 : 1). \end{aligned}$$

$$\begin{aligned} m_B &= 1, \\ n_B &= 2, \\ [m_B]P_B + [n_B]Q_B &= (62z + 55 : 126z + 71 : 1), \\ E_B : Y^2 &= X^3 + (121z + 158)X + (152z + 76), \\ \phi_B(P_A) &= (142z + 185 : 20z + 161 : 1), \\ \phi_B(Q_A) &= (162z + 32 : 41z + 73 : 1). \end{aligned}$$

Poté Alice pošle Bobovi svou eliptickou křivku E_A spolu s oběma obrazy bodů Bobovy báze $\phi_A(P_B), \phi_A(Q_B) \in E_A$.

Po obdržení $E_B, \phi_B(P_A), \phi_B(Q_A) \in E_B$ od Boba Alice vypočte nové isogenium $\phi'_A : E_B \to E_{AB}$ mající jádro $K'_A := \langle [m_A]\phi_B(P_A) + [n_A]\phi_B(Q_A) \rangle$. Bob provede to samé na své straně.

Příklad 5.

$$\begin{aligned} [m_A]\phi_B(P_A) + [n_A]\phi_B(Q_A) &= (13z + 53 : 163z + 13 : 1), \\ E_{AB} : Y^2 &= X^3 + (23z + 176)X + (181z + 189). \end{aligned}$$

$$\begin{aligned} [m_B]\phi_A(P_B) + [n_B]\phi_A(Q_B) &= (185z + 6 : 160z + 107 : 1), \\ E_{BA} : Y^2 &= X^3 + (23z + 176)X + (181z + 189). \end{aligned}$$

Alice a Bob poté mohou použít společný j-invariant křivek jako sdílený tajný klíč.

$$\begin{aligned} E_{AB} = \phi'_B(\phi_A(E_0)) = \phi'_A(\phi_B(E_0)) = E_{BA} = \\ = E_0/_{\langle [m_A]P_A + [n_A]Q_A, [m_B]P_B + [n_B]Q_B \rangle} \end{aligned}$$

Příklad 6.

$$\begin{aligned} j(E_{AB}) &= 104z + 23, \\ j(E_{BA}) &= 104z + 23. \end{aligned}$$

Tento algoritmus výměny klíčů mezi dvěma stranami je založen na následujícím výpočetním problému:

Problém. Supersingulární výpočetní Diffie–Hellmanův problém [2]
(*Supersingular Computational Diffie–Hellman (SSCDH) problem*)
Nechť $\phi_A : E_0 \to E_A$ je isogenium, jehož jádro je rovno $\langle [m_A]P_A + [n_A]Q_A \rangle$ a nechť $\phi_B : E_0 \to E_B$ je isogenium, jehož jádro je rovno $\langle [m_B]P_B + [n_B]Q_B \rangle$, kde čísla m_A, n_A (resp. m_B, n_B) jsou vybrána náhodně z $\mathbb{Z}/l_A^{e_A}\mathbb{Z}$ (resp. $\mathbb{Z}/l_B^{e_B}\mathbb{Z}$) a nejsou obě dělitelná l_A (resp. l_B).
Za pomoci křivek E_A, E_B a bodů $\phi_A(P_B), \phi_A(Q_B), \phi_B(P_A), \phi_B(Q_A)$ nalezněte j-invariant křivky $E_0/_{\langle [m_A]P_A + [n_A]Q_A, [m_B]P_B + [n_B]Q_B \rangle}$.

Vyřešením výše zmíněného problému by útočník získal společný tajný klíč a mohl by dešifrovávat veškerou zašifrovanou komunikaci mezi oběma stranami (Alicí a Bobem).

Tabulka 2.2 přehledně zobrazuje operace prováděné oběma komunikujícími stranami a potřebné pro stanovení společného tajného klíče, které byly popsány v této podsekci.

2.6.2 Vygenerování parametrů

Pro pevně zvolené hodnoty l_A, e_A, l_B, e_B můžeme testovat náhodné hodnoty f tak, aby platilo, že $l_A^{e_A} \cdot l_B^{e_B} \cdot f + 1$ nebo $l_A^{e_A} \cdot l_B^{e_B} \cdot f - 1$ je prvočíslo.

Dle [8] lze získat supersingulární eliptickou křivku s danou kardinalitou $(p \mp 1)^2 = (l_A^{e_A} \cdot l_B^{e_B} \cdot f)^2$ nad $\mathbb{F}_{p^2}$ následujícím způsobem:

1. Nejprve náhodně vygenerujeme nějaký diskriminant $D < 0$. Hledáme tedy náhodné malé kvadratické reziduum q v $\mathbb{F}_p$. Pokud takové číslo najdeme, tak vypočteme číslo opačné $D = -q$. Pokud platí , že $D \not\equiv 0 \pmod 4$ a zároveň $D \not\equiv 1 \pmod 4$, pak D ještě navíc vynásobíme čtyřmi (tedy $D = 4D$). Výsledné číslo je hledaný diskriminant.

2. Poté je potřeba vypočíst polynom Hilbertovy třídy $H_D \in \mathbb{Z}[X]$. Způsobů na vypočtení je několik. Vycházela jsem z funkce *hilbert_class_polynomial()* z matematického softwaru SageMath [18]. Stejný algoritmus popsal Ben Lynn [15] a další tři algoritmy pro vypočtení polynomu Hilbertovy třídy jsou uvedeny v [14].

3. Dále vybereme jeden z kořenů H_D v $\mathbb{F}_{p^2}$. Označme ho r.

4. Vygenerujeme nějakou eliptickou křivku s j-invariantem hodnoty r. Tato křivka je supersingulární a může mít kardinalitu buď $(p \mp 1)^2 = (l_A^{e_A} \cdot l_B^{e_B} \cdot f)^2$, nebo $(p \pm 1)^2 = (l_A^{e_A} \cdot l_B^{e_B} \cdot f \pm 2)^2$.

5. Pokud má křivka kardinalitu $(p \mp 1)^2 = (l_A^{e_A} \cdot l_B^{e_B} \cdot f)^2$, tak jsme právě našli hledanou supersingulární eliptickou křivku.

Alice		**Bob**
	$p = l_A^{e_A} l_B^{e_B} f \pm 1$ E_0 nad $\mathbb{F}_{p^2}$	
Zvolí bázi $\{P_A, Q_A\}$		Zvolí bázi $\{P_B, Q_B\}$
	$- - - \longrightarrow$ $\{P_A, Q_A\}$	
	$\longleftarrow - - -$ $\{P_B, Q_B\}$	
Náhodně zvolí $m_A, n_A \in_R \mathbb{Z}/l_A^{e_A}\mathbb{Z}$		Náhodně zvolí $m_B, n_B \in_R \mathbb{Z}/l_B^{e_B}\mathbb{Z}$
Vypočte isogenium, výslednou křivku a obrazy báze Boba $\phi_A : E_0 \to E_A$ $E_A = E_0/_{\langle [m_A]P_A + [n_A]Q_A \rangle}$ $\phi_A(P_B), \phi_A(Q_B) \in E_A$		Vypočte isogenium, výslednou křivku a obrazy báze Alice $\phi_B : E_0 \to E_B$ $E_B = E_0/_{\langle [m_B]P_B + [n_B]Q_B \rangle}$ $\phi_B(P_A), \phi_B(Q_A) \in E_B$
	$- - - \longrightarrow$ $\phi_A(P_B), \phi_A(Q_B), E_A$	
	$\longleftarrow - - -$ $\phi_B(P_A), \phi_B(Q_A), E_B$	
Vypočte isogenium a výslednou křivku pomocí údajů od Boba $\phi'_A : E_B \to E_{AB}$		Vypočte isogenium a výslednou křivku pomocí údajů od Alice $\phi'_B : E_A \to E_{BA}$
	$E_{AB} = E_{BA}$ $j(E_{EB}) = j(E_{BA})$	

Tabulka 2.2: Schéma algoritmu výměny klíčů

6. Pokud má křivka kardinalitu $(p \pm 1)^2 = (l_A^{e_A} \cdot l_B^{e_B} \cdot f \pm 2)^2$, pak musíme najít kvadratický twist této eliptické křivky, abychom získali hledanou supersingulární eliptickou křivku požadované kardinality $(p \mp 1)^2 = (l_A^{e_A} \cdot l_B^{e_B} \cdot f)^2$.

2.6.3 Získání báze

Při hledání báze $E_0[l_A^{e_A}]$ můžeme postupovat následujícím způsobem:

1. Vygenerujeme bod P_A řádu $l_A^{e_A}$ — viz níže.
2. Vygenerujeme druhý bod Q_A řádu $l_A^{e_A}$ — viz níže.
3. Poté musíme zkontrolovat, zda jsou body P_A a Q_A nezávislé. To provedeme tak, že vypočteme Weilovo párování bodů P_A a Q_A v $E_0[l_A^{e_A}]$ a zkontrolujeme, zda je výsledek řádu $l_A^{e_A}$. To uděláme tak, že výsledek Weilova párování umocníme na $l_A^{e_A-1}$. Pokud tímto umocněním získáme bod 0, pak výsledek Weilova párování je řádu nižšího než $l_A^{e_A}$ a body P_A, Q_A nejsou nezávislé.
4. S velkou pravděpodobností jsou oba body nezávislé a našli jsme takto bázi. Pokud jsou body závislé (a tedy výsledek Weilova párování má řád nižší než $l_A^{e_A}$), pak se vrátíme do bodu číslo 2. a vygenerujeme nový bod Q_A.

Bod řádu $l_A^{e_A}$ na eliptické křivce E_0 můžeme nalézt takto:

1. Nejprve vybereme náhodný bod $P \in_R E_0(\mathbb{F}_{p^2})$.
2. Bod P vynásobíme číslem $(l_B^{e_B} \cdot f)$ a tím získáme bod P' řádu $l_A^{e_A}$ či dělícího $l_A^{e_A}$. Připomeňme, že křivka E_0 má řád $(p \mp 1)^2 = (l_A^{e_A} \cdot l_B^{e_B} \cdot f)^2$.
3. S velkou pravděpodobností bude mít bod P' požadovaný řád $l_A^{e_A}$, což ověříme tak, že $[l_A^{e_A-1}]P' \neq O$. Pokud má bod P' řád menší než (a tedy $[l_A^{e_A-1}]P' = O$), pak se vrátíme do bodu číslo 1. a generujeme nový bod P.

Podobně postupujeme i při generování báze $E_0[l_B^{e_B}]$.

2.6.4 Výpočet isogenia s daným jádrem

V této podkapitole popisuji, jak lze sestrojit isogenium $\phi_A : E_0 \to E_A$ stupně $l_A^{e_A}$ s jádrem $\langle [m_A]P_A + [n_A]Q_A \rangle$. Tedy $E_A = E_0/\langle [m_A]P_A + [n_A]Q_A \rangle$. Jeden ze způsobů, jak takové isogenium sestrojit, je popsán algoritmem 1 převzatým z [1].

Řád bodu R_0 je $l_A^{e_A}$, protože řády P_A i Q_A jsou $l_A^{e_A}$ a zároveň m_A a n_A jsou voleny tak, aby alespoň jeden z těchto dvou prvků nebyl dělitelný číslem

Algoritmus 1 Algoritmus sestrojení isogenia s jádrem $\langle [m_A]P_A + [n_A]Q_A \rangle$ založený na násobení

$R_0 \leftarrow [m_A]P_A + [n_A]Q_A$
for $0 \leq i < e_A$ **do**
  $P_i \leftarrow l_A^{e_A - i - 1} R_i$
  Výpočet $\phi_i : E_i \rightarrow E_i/\langle P_i \rangle$
  $E_{i+1} \leftarrow E_i/\langle P_i \rangle$
  $R_{i+1} \leftarrow \phi_i(R_i)$
end for

l_A. Požadované isogenium stupně $l_A^{e_A}$ tedy vznikne složením e_A isogenií stupně l_A. Pro hledanou eliptickou křivku E_A platí $E_A = E_{e_A}$ a pro isogenium ϕ_A můžeme psát $\phi_A = \phi_{e_A - 1} \circ \cdots \circ \phi_0$.

Pomocí stejného principu sestrojíme i ostatní isogenia stupně $l_A^{e_A}$ či $l_B^{e_B}$ potřebná k výměně klíčů.

Zbývá ještě popsat, jak sestrojit jednotlivá isogenia $\phi_0, \cdots, \phi_{e_A - 1}$ řádu l_A z bodů jádra isogenia. To můžeme provést pomocí **Véluových vzorců** (*Vélu's formulas*)[16, 17] následujícím způsobem:

Mějme dánu eliptickou křivku E_1 nad tělesem K a jádro isogenia zadané jako množinu bodů tvořící konečnou podgrupu $E_1(\overline{K})$. Potřebujeme zjistit, na jakou eliptickou křivku E_2 zobrazuje isogenium body z E_1 a jakým způsobem můžeme vypočíst obrazy bodů z E_1.

Vstup: Vstupní eliptická křivka (*domain*) E_1 v obecném Weierstrassově tvaru:

$$E_1 : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6,$$

a dále množina bodů F tvořící jádro nějakého isogenia (musí to tedy být konečná podgrupa $E_1(\overline{K})$).

Výstup: Koeficienty obecné Weierstrassovy rovnice výstupní křivky (*codomain*) E_2 separabilního isogenia s jádrem F. Dále také zobrazení umožňující zobrazit bod (x, y) na křivce E_1 na bod na křivce E_2.

Krok 1: Rozdělíme množinu bodů F:

1. Bod O z množiny vyřadíme. Tedy $F = F \setminus \{O\}$.

2. Nechť množina F_2 je tvořena všemi 2-torzními body z F a nechť množina $R = F \setminus F_2$.

3. Rozdělíme množinu R na dvě stejně velké množiny R_+ a R_- splňující toto: pokud platí $P \in R_+$, pak $\ominus P \in R_-$.

4. Nechť množina $S = R_+ \cup F_2$.

Krok 2: Pro každý bod $Q \in S$ definujeme následující veličiny:

$$\begin{aligned} Q &= (x_Q, y_Q) \\ g_Q^x &= 3x_Q^2 + 2a_2x_Q + a_4 - a_1y_Q \\ g_Q^y &= -2y_Q - a_1x_Q - a_3 \\ t_Q &= \begin{cases} g_Q^x & \text{pro} Q \in F_2 \\ 2g_Q^x - a_1g_Q^y & \text{pro} Q \notin F_2 \end{cases} \\ u_Q &= (g_Q^y)^2. \end{aligned}$$

Krok 3: Dále definujeme veličiny:

$$t = \sum_{Q \in S} t_Q, \quad w = \sum_{Q \in S} (u_Q + x_Q t_Q).$$

Krok 4: Nakonec vypočteme výstupní křivku E_2 takto:

$$y^2 + A_1xy + A_3y = x^3 + A_2x^2 + A_4x + A_6,$$

kde

$$\begin{aligned} &A_1 = a_1, \quad A_2 = a_2, \quad A_3 = a_3, \\ &A_4 = a_4 - 5v, \quad A_6 = a_6 - (a_1^2 + 4a_2)t - 7w. \end{aligned}$$

Krok 5: Vzorec pro výpočet obrazu (X, Y) bodu (x, y) je následující:

$$X = x + \sum_{Q \in S} \left[\frac{t_Q}{x - x_Q} + \frac{u_Q}{(x - x_Q)^2} \right]$$

$$Y = y - \sum_{Q \in S} \left[u_Q \frac{2y + a_1x + a_3}{(x - x_Q)^3} + t_Q \frac{a_1(x - x_Q) + y - y_Q}{(x - x_Q)^2} + \frac{a_1u_Q - g_Q^x g_Q^y}{(x - x_Q)^2} \right].$$

2.7 Stávající implementace

V současné době lze na Internetu nalézt pouze jednu implementaci výměny klíčů založené na isogeniích supersingulárních křivek [3]. Autory jsou Luca De Feo, David Jao, Jérôme Plût a další. Tato implementace se skládá z několika částí, které jsou provázány. Jedná se o část napsanou v programovacím jazyce C a část napsanou v jazyce podobném jazyku Python v matematickém systému SageMath [18]. Tyto dvě části navzájem propojuje kód napsaný v jazyce Cython. Dále je využíváno rozsáhlých knihoven, na kterých je matematický software SageMath postaven. Mezi tyto knihovny patří například knihovny GMP [27] a PARI [29], o kterých se dále zmiňuji v kapitole 3.

Generování veřejných dat provádí funkce *ss_isogeny_gen()*. Vstupními parametry této funkce jsou hodnoty l_A, l_B, e_A, e_B, f a hodnota $pm1$, která určuje, zda přičíst či odečíst 1, aby vzniklo prvočíslo. Tato funkce nejprve vypočte dané prvočíslo, poté vytvoří konečné pole, nalezne diskriminant, ze kterého vypočte polynom Hilbertovy třídy, a vytvoří supersingulární eliptickou

křivku. Pokud je potřeba, nalezne její kvadratický twist. Poté vygeneruje obě báze a zvolí optimální strategii výpočtu isogenia. Tato funkce vrací hodnoty $E_0, l_A, e_A, P_A, Q_A, l_B, e_B, P_B, Q_B$ a strategie $strA, strB$.

Samotnou výměnu klíčů provádí funkce *ss_isogeny_exchange()*. Její vstupní parametry jsou shodné s výstupními parametry funkce *ss_isogeny_gen()*. Tato funkce vygeneruje náhodná m_A, n_A, m_B, n_B. Potom vypočte isogenia ϕ_A, ϕ_B a eliptické křivky E_A, E_B (k tomu jsou využity strategie $strA, strB$). Zároveň se během výpočtu isogenia provádí i výpočet obrazů báze. Poté už se provede pouze výpočet isogenií ϕ'_A, ϕ'_B a křivek E_{AB}, E_{BA} a následná kontrola rovnosti j-invariantů křivek.

Kromě výše zmíněných dvou funkcí lze ještě volat funkci *ss_isogeny()*, která se postará o proběhnutí obou předchozích funkcí.

Funkce *ss_isogeny()*, *ss_isogeny_gen()* a *ss_isogeny_exchange()* jsou implementované v SageMath. Aritmetika nad tělesem $\mathbb{F}_{p^2}$ a některé algoritmy spojené s Montgomeryho eliptickou křivkou jsou napsány v jazyce C. Pro počítání aritmetiky modulo p je využívána knihovna GMP. Pro výpočet polynomu Hilbertovy třídy a pro nalezení jeho kořenů je použita knihovna PARI. Výpočet isogenia stupně $l_A^{e_A}$ je implementován pro $l_A = 2, 3$ v jazyce C a pro jiná l_A v jazyce Cython.

Oproti algoritmu uvedenému v sekci 2.6 je kód dostupný z [3] na mnohých místech optimalizovaný a mírně upravený. Popis vylepšení spolu s teorií týkající se kryptosystému založeného na isogeniích eliptických křivek byl publikován v časopisu [2]. Uvádím zde pouze některé z úprav.

- Aplikace obsahuje tabulku s předem vygenerovanými parametry l_A, l_B, e_A, e_B, f a informací, zda se má přičíst či odečíst 1, aby vzniklo prvočíslo. Prvočíslo p se tedy negeneruje náhodně, ale vybírá se z tabulky dle zadaných parametrů při volání funkce.

- Konečné těleso $\mathbb{F}_{p^2}$ je implementováno jako $\mathbb{F}_p[X]/(X^2+1)$, což vyžaduje, aby X^2+1 byl ireducibilní polynom nad tělesem $\mathbb{F}_{p^2}$. Pro prvočíslo p tedy musí platit, že -1 je kvadratickým nereziduem modulo p (to je ekvivalentní s $p \equiv 3 \pmod 4$ pro p prvočíslo [19]).

- Eliptické křivky jsou uloženy v Montgomeryho tvaru, což umožňuje mnohem rychlejší a efektivnější sestrojení isogenií.

- Dalším rozdílem je způsob sestrojení isogenia s jádrem $\langle [m_A]P_A + [n_A]Q_A \rangle$. Autoři využívají efektivnějšího způsobu sestrojení pomocí tzv. strategií. Více je možné se dočíst opět v [2].

Nevýhodou této implementace je její složitost. K úplnému pochopení je nutná znalost několika programovacích jazyků. Dalším faktorem, který brání snadnému porozumění algoritmu, je vícevrstvá struktura. Funkce napsané pro

SageMath volají funkce v jazyce Cython. Tyto zase volají funkce napsané v jazyce C. Celý kód se tímto stává nepřehledným a podstata algoritmu výměny klíčů zde zaniká.

Implementace rovněž není odladěná a stává se, že vyhodí nějakou výjimku (například je vygenerován diskriminant s hodnotou 0 a tudíž nelze vypočíst polynom Hilbertovy třídy).

2.8 Porovnání postkvantových a klasických algoritmů výměny klíčů

Postkvantové algoritmy výměny klíčů jsou algoritmy výměny klíčů, pro které není zatím znám žádný útok pomocí kvantového počítače. Zatím tedy neexistuje žádný kvantový algoritmus, který by je dokázal efektivně řešit, tj. v polynomiálním čase. Některé takové algoritmy a kryptosystémy nyní stručně popíši.

- **Výměna klíčů založená na isogeniích supersingulárních eliptických křivek** — viz sekce 2.6.
- **McEliece** kryptosystém byl vytvořen roku 1978 Robertem J. McEliecem [20]. Tento systém je založen na Goppa kódech, konkrétně na NP-těžkém problému dekódování lineárních kódů.
- **NTRU** je patentovaný kryptosystém veřejného klíče založený na celočíselných mřížích. První verze byla představena roku 1996 matematiky Jeffrey Hoffsteinem, Jillem Pipherem a Josephem H. Silvermanem.
- **Merklovo podpisové schéma** je schéma digitálního podpisu založené na tzv. hashovacích či Merklových stromech. Bylo vytvořeno Ralphem Merklem ve druhé polovině sedmdesátých let dvacátého století. Závisí na existenci bezpečnosti hashovacích funkcí. V roce 2015 studijní skupina pro postkvantovou kryptografii sponzorovaná Evropskou komisí doporučila používání podpisů založených na hashovacích funkcích pro dlouhodobou ochranu proti kvantovým počítačům [21].

Na rozdíl od postkvantových algoritmů, klasické algoritmy jsou dnes běžně používané v každodenní elektronické komunikaci. Jsou založené na jednom z následujících problémů:

Problém 1. [22] Nechť G je grupa, $b \in G$ a $y \in G$ je mocninou prvku b. Potom diskrétní logaritmus prvku y o základu b je každé číslo k takové, že $b^k = y$ a značíme ho $log_b y$. Problém nalezení k se nazývá **problém diskrétního logaritmu** (*discrete logarithm problem*), dále **PDL** (*DLP*).

Problém 2. [24, 5] Nechť E je eliptická křivka, $P \in E$ a $Q \in E$ je násobkem bodu P. Problém nalezení n splňujícího $Q = [n]P$ se nazývá **problém diskrétního logaritmu na eliptické křivce** (*elliptic curve discrete logarithm problem*), dále **ECPDL** (*ECDLP*).

Problém 3. [23] Nechť G je grupa, $n \in G$. Problém nalezení $n_1, \ldots, n_k \in G$ splňujících $n = n_1 \cdot \cdots \cdot n_k$ se nazývá **problém faktorizace** (*factorization problem*).

Roku 1994 představil Peter Shor algoritmus řešící faktorizaci složeného čísla N a algoritmus řešící diskrétní logaritmus [25]. Na kvantovém počítači tento algoritmus pracuje v polynomiálním čase [26], což je způsobeno efektivní kvantovou Fourierovou transformací. Tento objev odhalil, že s příchodem kvantových počítačů mohou být některé dnes běžně užívané kryptosystémy veřejného klíče prolomeny a stát se nepoužitelnými.

Nyní ve stručnosti uvedu přehled některých takových kryptosystémů, na které by měl objev kvantového počítače díky Shorovu algoritmu vliv.

- **RSA** je šifrovací systém veřejného klíče uvedený Rivestem, Shamirem a Adlemanem v roce 1970. Je založen na modulárním umocňování. Dvojice (e, n) tvoří veřejný klíč, kde e je exponent a n je modul tvaru $n = pq$. Dále platí, že p, q jsou prvočísla a $\gcd(e, \Phi(n)) = 1$ (Φ je Eulerova funkce). K dešifrování je potřeba znát soukromý klíč (d, n), kde d je inverze čísla e modulo $\Phi(n)$. Bez znalosti **faktorizace** čísla n nelze získat $\Phi(n)$, a tudíž ani d [23].

- **Diffie–Hellmanův protokol výměny klíčů** je protokol pro zřízení společného klíče založený na exponenciální šifře. Veřejnými prvky jsou čísla (m, a), kde m je prvočíslo a číslo a je generátorem multiplikativní grupy $\mathbb{Z}_m^\times$. Účastník A si zvolí m, a a náhodné k_1, $0 < k_1 < m$. Prostřednictvím komunikačního kanálu odešle účastníkovi B a, m a $y_1 = |a^{k_1}|_m$. Účastník B si zvolí náhodné k_2, $0 < k_2 < m$, a odešle účastníkovi A $y_2 = |a^{k_2}|_m$. Účastník A vypočte tajný klíč jako $K = |y_2^{k_1}|_m$ a účastník B jako $K = |y_1^{k_2}|_m$. Společný tajný klíč můžeme tedy zapsat jako $K = a^{k_1^{k_2}} = a^{k_2^{k_1}} = a^{k_1 k_2}$. Bez znalosti efektivního algoritmu pro řešení **diskrétního logaritmu** nelze k_1 a k_2 vypočíst z údajů m, a, y_1 a y_2, a tedy nelze získat sdílený tajný klíč K [22].

- **Kryptografie eliptických křivek** (*Elliptic Curve Cryptography, ECC*) je přístup ke kryptografii veřejného klíče založený na eliptických křivkách nad konečnými tělesy. Lze použít pro výměnu klíčů, šifrování, digitální podpis aj. Využívá problému **diskrétního logaritmu na eliptické křivce**. Pokud si jako soukromý klíč zvolíme číslo k a vypočteme $Q = [k]P$, pak i přes zveřejnění P, Q nemůže útočník bez znalosti efektivního algoritmu řešícího ECPDL číslo k najít (předpokládáme, že řád bodu P je velký)[24].

- **ElGamal** je algoritmus pro kryptografii s veřejným klíčem, který je založen na Diffie–Hellmanově výměně klíče, a tedy na **PDL**. Byl popsán roku 1985 Taherem ElGamalem [23].

2.9 Závěr analýzy

Uvedla jsem základní definice a věty potřebné k pochopení výměny klíčů založené na isogeniích supersingulárních eliptických křivek. Dále jsem popsala algoritmus výměny klíčů včetně příkladu. Poté jsem zanalyzovala jedinou dostupnou implementaci této výměny klíčů, u které jsem uvedla nejpodstatnější rozdíly oproti uvedenému algoritmu. Na závěr jsem popsala rozdíl mezi tzv. klasickými a postkvantovými algoritmy výměny klíčů.

Pro návrh a implementaci vzorové aplikace je nejdůležitější pochopení algoritmu v sekci 2.6, na kterém je celá aplikace postavena.

Kapitola 3

Návrh a realizace

Cílem praktické části mé bakalářské práce je implementovat vzorovou aplikaci výměny klíčů založené na isogeniích supersingulárních eliptických křivek s cílem názorně prezentovat základní principy z [1] a demonstrovat uvedený algoritmus. Z tohoto důvodu implementuji algoritmus uvedený v sekci 2.6 jen s drobnými úpravami.

První úprava se týká generování prvočísla p. Rovněž jako v [3] jsem se rozhodla použít předem vygenerovanou tabulku s parametry. Podle zadaných parametrů na příkazové řádce se vyberou z tabulky parametry l_A, l_B, e_A, e_B, f a ± 1, ze kterých se poté vypočte dané prvočíslo.

Druhá úprava se týká samotného tělesa $\mathbb{F}_{p^2}$, které jsem se rozhodla implementovat jako $\mathbb{F}_p[X]/(X^2+1)$, což, jak již bylo popsáno v sekci 2.7, vyžaduje, aby pro prvočíslo p navíc platilo $p \equiv 3 \pmod 4$.

Rozhodla jsem se implementovat aplikaci pomocí tříd či struktur, které usnadňují práci s jednotlivými objekty a zpřehledňují volání metod jednotlivých objektů. Jedná se především o třídu implementující konečné těleso, třídu implementující eliptickou křivku a třídu implementující bod na eliptické křivce. Toto umožňuje snazší porozumění samotnému algoritmu výměny klíčů.

3.1 Výběr jazyka a knihoven

Nejdůležitější požadavky kladené na programovací jazyk byly rychlost a především dostupnost vhodných matematických knihoven. V úvahu tedy připadaly kompilované programovací jazyky C a C++. Zvolen byl jazyk C++, a to především z důvodu toho, že umožňuje použití tříd a přetěžování operátorů.

Co se týče matematických knihoven, bylo vybíráno mezi knihovnami umožňujícími výpočty s libovolnou přesností. Toto nabízí například knihovny GMP [27], NTL [28] a PARI [29] knihovna. GMP knihovna byla z rozhodování vyřazena již na začátku, a to z důvodu toho, že neobsahuje žádné výpočty týkající se konečných těles či eliptických křivek. NTL knihovna sice implementuje výpočty nad konečnými tělesy, ale výpočty nad eliptickými křivkami rovněž

neobsahuje. Vybrala jsem tedy PARI knihovnu, která obsahuje velké spektrum funkcí týkajících se nejen konečných těles a eliptických křivek.

Knihovna PARI je napsána v jazyce C a většina funkcí, které obsahuje, používá rychlé a efektivní algoritmy. Vytvořena byla původně Henri Cohenem a jeho spolupracovníky z Université de Bordeaux ve Francii. Nyní je šířena pod licencí GPL [30] a je spravována Karimem Belabasem za pomoci mnoha dalších přispěvatelů. Kromě samotné knihovny je možné ze stránek PARI [29] získat i několik volitelných balíčků s předem vygenerovanými daty. Pro výpočty týkající se eliptických křivek nad konečnými tělesy s velkou charakteristikou je nutné mít nainstalovaný balíček *seadata*. Všechny matematické objekty (celá čísla, desetinná čísla, polynomy atd.) knihovny PARI jsou implementovány jako typ *GEN*.

3.2 Běh aplikace

Po spuštění vytvořené aplikace se nejprve ve funkci *int main(int argc, char *argv[])* inicializují struktury, které vyžaduje knihovna PARI, a poté se popořadě volají příslušné metody jednotlivých objektů implementující části algoritmu popsaného v sekci 2.6. Jednotlivé třídy a jejich metody jsou popsány v sekci 3.3. Zde jen pro úplnost uvádím pořadí, ve kterém se vytvářejí objekty a volají jejich metody.

1. Získání parametrů pomocí funkce *const Parameters * getParameters (const char *pars)*.

2. Výpočet prvočísla pomocí metody *GEN Parameters::evaluatePrime() const*

3. Vytvoření konečného tělesa pomocí konstruktoru *GFp2::GFp2 (GEN p_char)*

4. Vytvoření supersingulární eliptické křivky nad konečným tělesem pomocí *EllipticCurve GFp2::getSupersingularCurve (const GEN card)*

5. Alice i Bob: Získání nějaké báze pomocí konstruktoru *Basis (EllipticCurve *p_E, GEN cofactor, GEN factor, int p)*

6. VÝMĚNA BÁZÍ – v aplikaci reálně neprobíhá žádné posílání bází, dochází pouze k výpisu na standardní výstup, který slouží pouze k prezentaci algoritmu

7. Alice i Bob: Náhodné vygenerování m_A a n_A, respektive m_B a n_B, přímo ve funkci *int main(int argc, char *argv[])*

8. Alice i Bob: Výpočet isogenia ϕ_A, respektive ϕ_B, výstupní eliptické křivky E_A, respektive E_B, a výpočet obrazů báze pomocí funkce *EllipticCurve * isogeny (EllipticCurve *E1, EllipticCurvePoint &PA, EllipticCurvePoint &QA, GEN mA, GEN nA, int lA, int eA, EllipticCurvePoint *PB=NULL, EllipticCurvePoint *QB=NULL)*

9. VÝMĚNA OBRAZŮ BÁZE A VÝSTUPNÍ ELIPTICKÉ KŘIVKY – v aplikaci opět nedochází k žádnému posílání dat, dochází pouze k výpisu na standardní výstup

10. Alice i Bob: Výpočet isogenia ϕ'_A, respektive ϕ'_B, a výstupní eliptické křivky E_{AB}, respektive E_{BA}, pomocí funkce *EllipticCurve * isogeny (EllipticCurve *E1, EllipticCurvePoint &PA, EllipticCurvePoint &QA, GEN mA, GEN nA, int lA, int eA, EllipticCurvePoint *PB=NULL, EllipticCurvePoint *QB=NULL)*

11. Alice i Bob: Výpočet j-invariantu křivky E_{AB}, respektive E_{BA}, pomocí metody *GEN EllipticCurve::jInvariant ()*

3.3 Použité třídy, struktury a funkce

V této podsekci uvádím nejdůležitější třídy, struktury a funkce použité ve vytvořené aplikaci. Není cílem, aby byl výčet úplný.

3.3.1 Třída *GFp2*

Třída *GFp2* implementuje konečné těleso $\mathbb{F}_p[X]/(X^2+1)$. Obsahuje atributy ireducibilní polynom X^2+1 a charakteristiku tělesa p. Mezi její nejdůležitější metody patří:

- *GEN getRandomElement()* – vrací náhodný prvek z konečného tělesa $\mathbb{F}_p[X]/(X^2+1)$
- *GEN HilbertClassPolynomialRoot(GEN D)* – vytvoří pomocí funkce *HilbertClassPolynomial* polynom Hilbertovy třídy a poté vrátí některý z jeho kořenů v $\mathbb{F}_p[X]/(X^2+1)$
- *EllipticCurve getEllipticCurveFromJInvariant(GEN j)* – vrací eliptickou křivku nad tělesem $\mathbb{F}_p[X]/(X^2+1)$ s daným j-invariantem
- *EllipticCurve getSupersingularCurve(const GEN card)* – vrací supersingulární eliptickou křivku nad tělesem $\mathbb{F}_p[X]/(X^2+1)$ s danou kardinalitou. Nejprve získá pomocí metody *HilbertClassPolynomialRoot* nějaký kořen (ten je použit při generování křivky jako j-invariant), poté

vytvoří eliptickou křivku s daným j-invariantem pomocí metody *getEllipticCurveFromJInvariant* a nakonec, pokud daná křivka nemá požádovanou kardinalitu, nalezne kvadratický twist pomocí metody třídy *EllipticCurve quadraticTwist*.

3.3.2 Třída *EllipticCurve*

Třída *EllipticCurve* implementuje eliptickou křivku nad konečným tělesem $\mathbb{F}_p[X]/(X^2+1)$. Obsahuje ukazatel na toto těleso a koeficienty a_4, a_6 v zobecněné Weierstrassově rovnici. Ostatní parametry a_1, a_2, a_3 jsou vždy nulové. Mezi její nejdůležitější metody patří:

- *EllipticCurvePoint randomPoint()* – vrací náhodný bod na eliptické křivce
- *GEN jInvariant()* – vrací j-invariant eliptické křivky
- *void quadraticTwist()* – nalezne kvadratický twist eliptické křivky

3.3.3 Funkce *torsionPoint*

Funkce *EllipticCurvePoint torsionPoint(EllipticCurve *E, GEN cof, GEN fact_div_p)* vrací bod požadovaného řádu na eliptické křivce. Používá postup uvedený v podsekci 2.6.3.

3.3.4 Třída *EllipticCurvePoint*

Třída *EllipticCurvePoint* implementuje bod na eliptické křivce. Obsahuje ukazatel na eliptickou křivku, na které bod leží, a dále reprezentaci bodu v knihovně PARI (tzn. typu *GEN*). Mezi její nejdůležitější metody patří:

- *EllipticCurvePoint operator* (int n) const* a *EllipticCurvePoint operator* (GEN n) const* – oba tyto přetížené operátory implementují násobení bodu celým číslem. V prvním případě násobíme bod *integerem* a ve druhém typem *GEN*.
- *EllipticCurvePoint operator+ (const EllipticCurvePoint & src) const* – tento přetížený operátor implementuje součet dvou bodů na stejné eliptické křivce
- *friend GEN weilPairing(const EllipticCurvePoint &P, const EllipticCurvePoint &Q, GEN order)* – tato spřátelená funkce vypočítává Weilovo párování dvou bodů P a Q, kde oba body jsou řádu *order*.

3.3.5 Třída *Basis*

Třída *Basis* implementuje bázi popsanou v sekci 2.6. Obsahuje ukazatel na eliptickou křivku a dále pak body báze, které se generují v konstruktoru této třídy pomocí funkce *torsionPoint*, jak bylo popsáno v podsekci 2.6.3.

3.3.6 Funkce *HilbertClassPolynomial*

Funkce *GEN HilbertClassPolynomial(const GEN D)* vypočte ze záporného diskriminantu polynom Hilbertovy třídy.

3.3.7 Funkce *isogeny*

Funkce *EllipticCurve * isogeny(EllipticCurve *E1, EllipticCurvePoint &PA, EllipticCurvePoint &QA, GEN mA, GEN nA, int lA, int eA, EllipticCurvePoint *PB=NULL, EllipticCurvePoint *QB=NULL)* vypočte isogenium se vstupní eliptickou křivkou *E1* a jádrem $\langle [m_A]P_A + [n_A]Q_A \rangle$. Používá algoritmus uvedený v podsekci 2.6.4 a výstupní křivku vrací pomocí návratové hodnoty. Pokud jsou zadány i body PB a QB jako vstupně-výstupní parametry, pak tato funkce vypočte obrazy těchto bodů.

3.3.8 Struktura *iso*

Struktura *iso* implementuje isogenium s malým počtem bodů jádra za použití **Véluových vzorců** popsaných v podsekci 2.6.4. Obsahuje ukazatele na vstupní i výstupní eliptickou křivku, dále veličiny t a w a nakonec pole struktur *pointInfo* obsahující požadované informace o daných bodech z jádra (dle značení z 2.6.4 se jedná o body z množiny S). Všechny tyto atributy se vypočtou v konstruktoru. Nejdůležitější metodou je:

- *EllipticCurvePoint apply(EllipticCurvePoint &P)* – tato metoda aplikuje na bod P dané isogenium a vrací obraz bodu P rovněž dle Véluových vzorců

3.3.9 Struktura *pointInfo*

Struktura *pointInfo* obsahuje informace o bodu z množiny S potřebné k výpočtu obrazu bodu v isogeniu pomocí **Véluových vzorců**, jedná se o x_Q, y_Q, g_Q^x, g_Q^y, t_Q a u_Q.

3.3.10 Funkce *getParameters*

Funkce *const Parameters *getParameters(const char * pars)* vrací dle parametru na příkazové řádce strukturu *Parameters* s informacemi k vypočtení prvočísla.

3.3.11 Třída *Parameters*

Objekt třídy *Parameters* obsahuje informace potřebné k vypočtení prvočísla p. Jedná se o l_A, e_A, l_B, e_B, f a informaci určující, zda se má přičíst či odečíst číslo 1, aby vzniklo požadované prvočíslo dle vzorce uvedeného v 2.6.1. Její nejdůležitější metodou je:

- *GEN evaluatePrime() const* – tato metoda vypočte dané prvočíslo

3.4 Předpoklady

Před spuštěním je nezbytné mít nainstalovanou knihovnu PARI ve verzi 2.7.5 či novější. Starší verze této knihovny neobsahovaly některé funkce, které aplikace používá. Dále je potřeba mít nainstalovaný balíček *seadata*, který byl zmíněn v sekci 3.1. Pro automatické přeložení souborů a sestavení do výsledné aplikace a také pro automatické vygenerování dokumentace je nutné mít nainstalovaný nástroj GNU Make [31]. Dále je potřeba mít nainstalovaný kompilátor g++ [32] a pro vygenerování dokumentace nástroje Doxygen [33] a Graphviz [34].

Zde je přehled verzí, se kterými je ověřeno, že aplikace funguje a dokumentace se generuje správně.

- PARI 2.7.5 – některé starší verze PARI neobsahovaly všechny funkce, které aplikace používá
- GNU Make 3.81
- g++ 4.8.2
- Doxygen 1.8.6
- Graphviz 2.36.0

Na závěr této podsekce uvádím přehled akcí, které lze pomocí nástroje GNU Make spustit a které definuje soubor Makefile.

- *make* či *make all* – tyto příkazy jsou ekvivalentní, přeloží soubory, sestaví je do výsledné aplikace a vygenerují dokumentaci
- *make compile* – tento příkaz přeloží soubory a sestaví je do aplikace
- *make doc* – vygeneruje dokumentaci
- *make clean* – vymaže soubory vytvořené pomocí *make all*
- *make run* – spustí aplikaci bez parametrů na příkazové řádce

Přeložené soubory pomocí GNU Make jsou uloženy v adresáři *bin*, který se automaticky vytvoří. Výsledná aplikace se uloží do hlavního adresáře, jenž mimo jiné obsahuje i Makefile. Dokumentace se generuje do adresáře *doc*, který je rovněž vytvořen automaticky.

3.5 Spuštění

Aplikace se spustí z příkazové řádky buď zadáním příkazu *make run*, pokud jsme předem aplikaci přeložili a sestavili, či zadáním příkazu *./keyexchange* s jedním parametrem určujícím, jaké prvočíslo p se vygeneruje. Parametr se skládá obvykle ze tří částí oddělených spojovníkem. První část určuje parametr l_A, druhá část parametr l_B a třetí část teoretickou bezpečnost určenou v bitech.

Příklad. Pokud je na příkazové řádce zadán parametr *2-3-512*, pak se aplikace spustí s prvočísly $l_A = 2$ a $l_B = 3$. Teoretická bezpečnost bude 512 bitů.

Přehled všech parametrů, které je možné zadat, je uveden v tabulce 3.1. Dále je také možné spustit aplikaci bez parametru, což se rovněž děje při spouštění pomocí příkazu *make run*. V takovém případě se aplikace spustí stejně, jako by byl na příkazové řádce zadán parametr *2-3-40*.

Pro úplnost uvádím ukázku běhu programu pro parametr *2-3-8*.

```
Initializing PARI library.
Getting the parameters.
Evaluating the prime.
        p = 191
Constructing the finite field GF(p^2).
Constructing the elliptic curve.
        Y^2 = X^3 + (106*x + 17) * X + (108*x + 10)
Constructing Alice's basis.
        ( [ 66*x + 37 : 28*x + 10 : 1 ], [ 173*x + 58 : 160*x +
        146 : 1 ] )
Constructing Bob's basis.
        ( [ 65*x + 81 : 128*x + 126 : 1 ], [ 130*x + 189 :
        129*x + 33 : 1 ] )
SENDING THE BASES
Generating Alice's secret key.
        mA = 36, nA = 1
Generating Bob's secret key.
        mB = 2, nB = 1
Generating Alice's public data.
        phiA(PB) = [ 107*x + 93 : 167*x + 118 : 1 ]
        phiA(QB) = [ 132*x + 36 : 69*x + 166 : 1 ]
```

Parametr	l_A	l_B	e_A	e_B	f	± 1
2-3-8	2	3	6	1	1	-1
2-3-40	2	3	22	15	1	-1
2-3-128	2	3	63	41	11	-1
2-3-256	2	3	130	81	22	-1
2-3-512	2	3	258	161	186	-1
2-3-678	2	3	341	218	3	-1
2-3-768	2	3	386	242	2	-1
2-3-1024	2	3	514	323	353	-1
3-5-512	3	5	161	110	314	$+1$
3-5-512:-1	3	5	161	110	736	-1
5-7-32	5	7	9	7	16	-1
5-7-32:-1	5	7	9	7	18	$+1$
5-7-128	5	7	55	46	372	-1
5-7-512	5	7	110	91	284	-1
5-7-768	5	7	165	137	2968	-1
5-7-1024	5	7	220	182	538	$+1$
11-13-512:+1	11	13	74	69	1254	$+1$
11-13-512	11	13	74	69	384	-1
11-13-768	11	13	111	104	78	$+1$
11-13-1024	11	13	148	138	942	$+1$
17-19-512	17	19	62	60	120	-1
17-19-512:+1	17	19	62	60	210	$+1$
17-19-768	17	19	94	90	116	-1
17-19-1024	17	19	125	120	712	-1
23-29-512:-1	23	29	56	52	452	-1
23-29-512	23	29	56	52	286	$+1$
23-29-768	23	29	85	79	132	-1
23-29-1024	23	29	113	105	1004	-1
31-41-512	31	41	51	47	564	-1
31-41-768	31	41	77	72	166	$+1$
31-41-1024	31	41	103	95	448	-1

Tabulka 3.1: Všechny možné parametry, které lze zadat při spouštění aplikace

```
        EA: Y^2 = X^3 + (100*x + 161) * X + (104*x + 29)
Generating Bob's public data.
        phiB(PA) = [ 131*x + 188 : 55*x + 131 : 1 ]
        phiB(QA) = [ 161*x + 42 : 71*x + 120 : 1 ]
        EB: Y^2 = X^3 + (165*x + 141) * X + (176*x + 172)
SENDING THE PUBLIC DATA
Computing shared key on Alice's side.
        EAB: Y^2 = X^3 + (173*x + 120) * X + (52*x + 110)
        j(EAB) = 16
Computing shared key on Bob's side.
        EBA: Y^2 = X^3 + (173*x + 120) * X + (52*x + 110)
        j(EBA) = 16
```

Kapitola 4

Testování

Aplikace byla testována ve virtuálním stroji na stolním počítači Dell Optiplex 9020 s procesorem Intel Core i7-4790, 32 GB paměti RAM, SSD diskem a 64-bitovým operačním systémem Windows 7 Professional. Jako virtualizační program byl použit VMware Workstation Player [35]. Jako hostovaný systém byla použita 32-bitová verze Ubuntu 14.02. Virtuálnímu počítači bylo přiděleno 8 GB paměti a dvě jádra procesoru.

Verze aplikací, se kterými byla aplikace testována, jsou uvedeny v sekci 3.4.

Během testování jsem se soustředila na rychlost a správnost běhu aplikace. Správnost byla zajištěna kontrolou vstupních hodnot metod a funkcí. Nadále také kontrolou mezních hodnot, které se mohou vyskytovat. Aplikace po této stránce funguje dobře a vrací očekávané výsledky — výsledné eliptické křivky se shodují a jejich j-invarianty také.

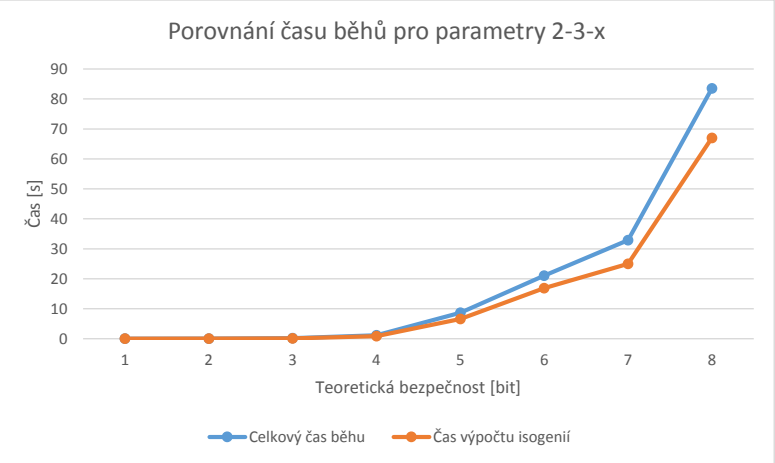
Co se týče rychlosti, měřila jsem pro každý běh aplikace pět různých časů. První čas udává celkový běh aplikace od inicializace struktur, které potřebuje PARI knihovna, až po jejich uvolnění. Další čtyři časy udávají celkovou dobu výpočtu isogenia řádu $l_A^{e_A}$. Během prvních dvou výpočtů isogenia se zároveň vypočítávají i obrazy báze, a tudíž jsou první dvě volání funkce vždy časově náročnější. Měření probíhalo pomocí funkcí knihovny PARI *void timer_start(pari_timer *T)* a *long timer_delay(pari_timer *T)*. Naměřené časy přehledně shrnuje tabulka 4.1 a vytvořené grafy 4.1 a 4.2. Je patrné, že doba výpočtu isogenií zabírá 50 až 90 % času běhu.

Graf 4.1 ukazuje závislost času běhu aplikace na zvolené teoretické bitové bezpečnosti. Je samozřejmé, že s rostoucí požadovanou bezpečností roste i doba běhu aplikace. Zároveň lze z tohoto grafu i z tabulky 4.1 usoudit, že s rostoucí požadovanou bezpečností zabírá pro parametry *2-3-x* výpočet isogenia větší část běhu aplikace.

Graf 4.2 ukazuje závislost běhu aplikace na zvolených parametrech l_A a l_B pro teoretickou 512-bitovou bezpečnost. Z grafu je patrné, že pro malé hodnoty l_A a l_B je čas běhu nejdelší, což může být dáno většími hodnotami

Parametr	Výpočet ϕ_A [ms]	Výpočet ϕ_B [ms]	Výpočet ϕ'_A [ms]	Výpočet ϕ'_B [ms]	Celkový čas běhu [ms]	Čas výpočtu isogenií [%]
2-3-8	4	0	0	0	8	50,0
2-3-40	4	4	0	4	24	50,0
2-3-128	40	36	36	32	208	69,2
2-3-256	260	192	245	172	1172	74,1
2-3-512	1 973	1 404	1 869	1 340	8 652	76,1
2-3-678	4 937	3 656	4 725	3 521	21 072	79,9
2-3-768	7 413	5 296	7 148	5 112	32 848	76,0
2-3-1024	19 829	14 080	19 372	13 709	83 512	80,2
3-5-512	1 564	1 176	1 493	1 068	6 400	82,8
3-5-512:-1	1 593	1 196	1 520	1 092	6 816	79,2
5-7-32	4	4	0	4	20	60,0
5-7-32:-1	4	4	4	0	24	50,0
5-7-128	148	144	132	116	748	72,2
5-7-512	1 148	1 036	1 052	913	5 632	73,7
5-7-768	4 305	3 812	4 045	3 496	21 516	72,8
5-7-1024	11 193	9 720	10 609	9 020	44 476	91,2
11-13-512:+1	977	965	805	777	4 300	82,0
11-13-512	941	937	781	757	5 144	66,4
11-13-768	3 344	3 297	2 901	2 792	13 840	89,1
11-13-1024	9 208	8 725	8 032	7 648	36 772	91,4
17-19-512	945	969	725	732	4 892	68,9
17-19-512:+1	1 004	1 048	777	781	4 124	87,5
17-19-768	3 453	3 448	2 805	2 761	21 172	58,9
17-19-1024	8 320	8 268	7 028	6 869	50 696	60,1
23-29-512:-1	1 037	1 092	748	761	5 612	64,8
23-29-512	1 032	1 101	756	765	5 600	65,3
23-29-768	3 192	3 320	2 481	2 464	17 832	64,2
23-29-1024	8 793	8 997	7 072	6 961	47 596	66,9
31-41-512	1 100	1 236	765	797	5 500	70,9
31-41-768	3 544	3 897	2 597	2 701	17 148	74,3
31-41-1024	8 729	9 208	6 717	6 724	45 980	68,2

Tabulka 4.1: Přehled časů běhů aplikace pro různé parametry



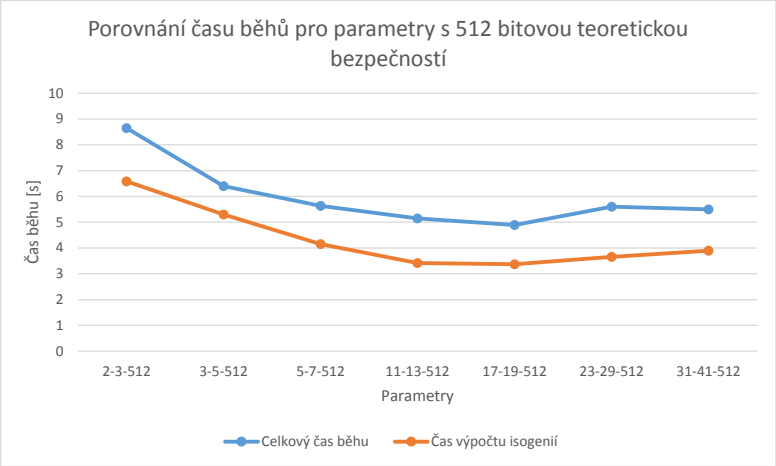
Obrázek 4.1: Graf časů běhů pro parametry 2-3-x

Parametry	Čas běhu aplikace [ms]	Čas běhu dostupné implementace [ms]
2-3-8	8	92
2-3-40	24	507
2-3-256	1 172	3 339
2-3-512	8 652	25 732
2-3-678	21 072	52 393
2-3-768	32 848	148 119
2-3-1024	83 512	290 415

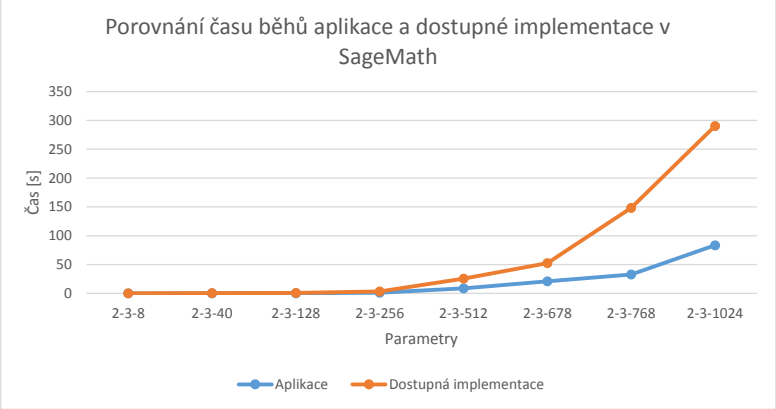
Tabulka 4.2: Přehled časů běhů aplikace v porovnání s dostupnou implementací v SageMath

e_A a e_B a tedy více iteracemi při výpočtu isogenia. Nejkratší čas byl dosažen pro $l_A = 17$ a $l_B = 19$. Pro velká l_A a l_B se již opět začíná čas zvyšovat — nejspíše v důsledku náročnosti operací nad danými tělesy.

Tabulka 4.2 obsahuje časy běhu aplikace v porovnání s časy běhu implementace dostupné z [3]. Stávající implementace byla spouštěna v SageMath verze 6.7 na stejném stroji a ve stejném prostředí jako vytvořená aplikace. Měření probíhalo pomocí funkce jazyka Python *timeit.default_timer()*. Graf 4.3 zobrazuje naměřené časy graficky. Z grafu i tabulky je patrné, že vytvořená aplikace v C++ je pro parametry *2-3-x* několikanásobně rychlejší než dostupná implementace v SageMath.



Obrázek 4.2: Graf časů běhů pro parametry s teoretickou bezpečností 512 bitů



Obrázek 4.3: Graf časů běhů vytvořené aplikace v porovnání s dostupnou implementací v SageMath

4.1 Odhalené problémy

Největším problémem, který se během vývoje aplikace objevil, byla skutečnost, že výpočet počtu bodů eliptické křivky nad konečným tělesem $\mathbb{F}_q$ je velice časově náročný pro velká q. Tento výpočet se prováděl při rozhodování, zda se má hledat kvadratický twist křivky, či nikoli.

Nalezená eliptická křivka E může mít jen dva různé řády $a = (p-1)^2$ a $b = (p+1)^2$ a řád bodu křivky dělí řád eliptické křivky. Dále platí $a \nmid b$. Rozhodla jsem se tedy použít následující postup. Nejprve náhodně zvolím bod P na eliptické křivce E. Poté vypočtu $P_1 = [a]P$ a $P_2 = [b]P$. Alespoň jeden z bodů P_1 a P_2 se musí rovnat bodu v nekonečnu. Důvodem je právě fakt, že řád bodů dělí řád křivky a tudíž platí $[\#E]P = O$ — viz Věta 2.2.4. Pokud $P_1 = O$ a zároveň $P_2 \neq O$, potom můžeme říci, že křivka E má řád a. Pokud $P_1 \neq O$ a zároveň $P_2 = O$, potom můžeme říci, že křivka E má řád b. Pokud se oba body P_1 a P_2 rovnají bodu v nekonečnu, pak musíme zvolit jiný bod P a celý postup zopakovat. Poslední zmíněný případ je jen velice málo pravděpodobný.

Ač se tato úprava zdá být triviální, tak vedla především u těles $\mathbb{F}_q$ s velkým q k výraznému urychlení běhu aplikace. Pro parametr *2-3-40* došlo ke dvacetinásobnému zrychlení ze 486 ms na 24 ms, pro parametr *2-3-128* došlo k 66-násobnému zrychlení ze zhruba 19 s na 288 ms a pro parametr *2-3-256* přetekl po více jak osmi minutách zásobník PARI knihovny. Po úpravě dokončí aplikace všechny výpočty pro parametr *2-3-256* po 1,28 s.

Dalším problémem bylo přetékání zásobníku PARI knihovny, na kterém jsou uloženy všechny výsledky, které programátor pomocí daných funkcí neodstraní. K přetékání docházelo pro parametry *2-3-512* a *2-3-1024*. Bylo možné zvolit dvě různá řešení. Buď pro zásobník knihovny vyhradit více místa, nebo odstraňovat v průběhu výpočtů co nejvíce nepotřebných dat. Rozhodla jsem se pro druhé řešení. V knihovně PARI existuje několik funkcí pro ošetřování zásobníku, například funkce *gerepile* či *gerepileupto*. Každá pracuje se zásobníkem trochu jinak a provádí něco jiného. V různých místech kódu jsem dle svého uvážení použila tu, která se mi jevila jako nejvhodnější.

Aplikace generuje velké množství dat náhodně, a tak se časy běhu pro stejné parametry od sebe mohou mírně lišit. Ošetřování zásobníku na výsledný čas nemělo žádný patrný vliv.

Závěr

Náplní mé bakalářské práce bylo prostudovat matematické základy potřebné k pochopení výměny klíčů založené na isogeniích supersingulárních eliptických křivek. Dále jsem prostudovala algoritmus této výměny klíčů a zanalyzovala jsem jedinou dostupnou implementaci na Internetu.

Navrhla jsem aplikaci realizující výměnu klíčů založenou na isogeniích supersingulárních eliptických křivek a poté jsem ji úspěšně implementovala v programovacím jazyce C++ s využitím knihovny PARI. Největší přínos této práce spatřuji v jednoduchém návrhu, který umožňuje názorně, přehledně a srozumitelně prezentovat algoritmus výměny klíčů. Dalším faktorem, který přispívá k přiblížení této aplikace a samotného algoritmu výměny klíčů co největšímu množství osob, je fakt, že jazyk C++ je dnes velice rozšířený.

V současné době je tato práce, pokud je mi známo, jediným textem v českém jazyce, který se zabývá výměnou klíčů založenou na isogeniích supersingulárních eliptických křivek. Rovněž mi není známa žádná publikace v českém jazyce, která by se zabývala isogenii či supersingulárními eliptickými křivkami v podobných souvislostech. Jedná se tedy nejspíše o unikátní práci obsahující veškerou podstatnou matematickou teorii spolu s analýzou algoritmu výměny klíčů zpracovanou do uceleného českého textu, který zpřístupňuje toto zajímavé téma českým čtenářům.

V budoucnu by bylo možné aplikaci rozšířit o implementaci kryptosystému veřejného klíče založeného na isogeniích supersingulárních eliptických křivek. Dále by bylo možné v aplikaci implementovat síťovou komunikaci a schéma výměny klíčů otestovat na dvou vzdálených zařízeních. V neposlední řadě by bylo možné funkce upravit tak, aby splňovaly bezpečnostní standardy, a výměnu klíčů založenou na isogeniích supersingulárních eliptických křivek zakomponovat do open source projektu OpenSSL.

Literatura

[1] Jao, David; De Feo, Luca. *Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies* [online]. 2011 [cit. 2015-12-11]. Dostupné z: `http://cacr.uwaterloo.ca/techreports/2011/cacr2011-32.pdf`.

[2] De Feo, Luca; Jao, David; Plût, Jérôme. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In: *Journal of Mathematical Cryptology*. 2014, vol. 8, no. 3, s. 209-247. ISSN 1862-2976

[3] De Feo, Luca; Jao, David; Plût, Jérôme a kol. *Quantum-Resistant Cryptosystems from Supersingular Elliptic Curve Isogenies* [online repozitář]. 2016 [cit. 2016-04-7]. Dostupné z: `https://github.com/defeo/ss-isogeny-software`.

[4] Silverman, Joseph H. *The Arithmetic of Elliptic Curves*. 2. vydání. New York: Springer-Verlag, 2009. 513 s. ISBN 978-0-387-09493-9.

[5] Kalvoda, Tomáš; Petr, Ivo. *Matematika pro kryptologii* [online]. 2016 [cit. 2016-04-10]. Dostupné z: `https://edux.fit.cvut.cz/courses/MI-MKY/_media/lectures/mi-mky-poznamky-v13.pdf`.

[6] Klouda, Karel. *Matematika pro informatiku, 5. přednáška* [online prezentace]. 2015 [cit. 2016-04-11]. Dostupné z: `https://edux.fit.cvut.cz/courses/MI-MPI/_media/lectures/mpi-prednaska-5-v2.pdf`.

[7] Stanovský, David. *Základy algebry*. Praha: Matfyzpress, 2010. 153 s. ISBN 978-80-7378-105-7

[8] Bröker, Reinier. Constructing supersingular elliptic curves. In: *Journal of Combinatorics and Number Theory*. 2009, vol. 1, no. 3, s. 269-273. ISSN 1942-5600

[9] Montgomery, Peter L. Speeding the Pollard and Elliptic Curve Methods of Factorization In: *Mathematics of Computation*. 1987, vol. 48, no. 177, s. 243-264. ISSN 0025-5718

[10] Bajko, Jaroslav. *Transfer eliptických křivek na torus*. Brno, 2011. 53 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství. doc. RNDr. Miroslav Kureš, Ph.D.

[11] Schoof, René. Counting points on elliptic curves over finite fields. In: *Journal de Théorie des Nombres de Bordeaux*. 1995, vol. 7, no. 1, s. 219-254. ISSN 1246-7405

[12] Cremona, John E.; Sutherland, Andrew V.. On a theorem of Mestre and Schoof. In: *Journal de Théorie des Nombres de Bordeaux*. 2010, vol. 22, no. 2, s. 353-358. ISSN 1246-7405

[13] Straka, Milan. *Kryptografie založená na kvadratických tělesech*. Praha, 2008. 54 s. Diplomová práce. Univerzita Karlova v Praze, Matematicko-fyzikální fakulta. RNDr. David Stanovský, Ph.D.

[14] Belding, Juliana; Bröker, Reinier; Enge, Andreas; Lauter, Kristin. Computing Hilbert Class Polynomials. In: *Lecture Notes in Computer Science 5011 z Algorithmic Number Theory Symposium VIII*. Německo: Springer-Verlag, 2008. s. 282-295. ISSN 0302-9743

[15] Lynn, Ben. Hilbert Class Polynomials. In: *Stanford.edu* [online]. Stanford University, 2015 [cit. 2016-05-02]. Dostupné z: `https://crypto.stanford.edu/pbc/notes/ep/hilbert.html`.

[16] Vélu, Jacques. Isogénies entre courbes elliptiques In: *Comptes rendus hebdomadaires des séances de l'Académie des sciences. Série A - Sciences mathématiques*. 1971, vol. 273, no. 4, s. 238-241.

[17] Shumow, Daniel. *Isogenies of Elliptic Curves: A Computational Approach*. Seattle, 2009. 84 s. Diplomová práce. University of Washington.

[18] SageMath. [cit. 2016-03-20] Dostupné z: `http://www.sagemath.org/`.

[19] Gauss, Carl Friedrich; Clarke, Arthur A. *Disquisitiones Arithemeticaes*. 2. přepracované vydání. New York: Springer-Verlag, 1986. 472 s. ISBN 0-387-96254-9.

[20] McEliece, Robert J. *A Public-Key Cryptosystem Based On Algebraic Coding Theory* [online]. 1978 [cit. 2016-04-22]. Dostupné z: `http://ipnpr.jpl.nasa.gov/progress_report2/42-44/44N.PDF`.

[21] Augot, Daniel; Batina, Lejla; Bernstein, Daniel J. a kol. *Initial recommendations of long-term secure post-quantum systems* [online]. 2015 [cit. 2016-04-27]. Dostupné z: http://pqcrypto.eu.org/docs/initial-recommendations.pdf.

[22] Lórencz, Róbert; Jureček, Martin. *Exponenciální šifra, zřízení společného klíče a problém diskrétního logaritmu* [online prezentace]. 2013 [cit. 2016-04-17]. Dostupné z: https://edux.fit.cvut.cz/courses/BI-BEZ/_media/bez_n2.pdf.

[23] Lórencz, Róbert; Kokeš, Josef. *RSA, kryptografie s veřejným klíčem, DSA, El-Gamalův algoritmus* [online prezentace]. 2013 [cit. 2016-04-17]. Dostupné z: https://edux.fit.cvut.cz/courses/BI-BEZ/_media/bez_n6.pdf.

[24] Lórencz, Róbert. *Základy kryptografie eliptických křivek a kvantové kryptografie* [online prezentace]. 2013 [cit. 2016-04-17]. Dostupné z: https://edux.fit.cvut.cz/courses/BI-BEZ/_media/bez_n8.pdf.

[25] Shor, Peter W. *Algorithms for Quantum Computation: Discrete Logarithms and Factoring* [online]. 1994 [cit. 2016-03-12]. Dostupné z: http://www.csee.wvu.edu/~xinl/library/papers/comp/shor_focs1994.pdf.

[26] Shor, Peter W. *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer* [online]. 1997 [cit. 2016-03-12]. Dostupné z: http://www.csee.wvu.edu/~xinl/library/papers/comp/shor_siam1997.pdf.

[27] GMP. The GNU Multiple Precision Arithmetic Library. [cit. 2016-04-15] Dostupné z: https://gmplib.org/.

[28] NTL: A Library for doing Number Theory. [cit. 2016-04-15] Dostupné z: http://www.shoup.net/ntl/.

[29] PARI. [cit. 2016-04-15] Dostupné z: http://pari.math.u-bordeaux.fr/.

[30] GNU General Public License. [cit. 2016-04-15] Dostupné z: http://www.gnu.org/licenses/gpl-3.0.en.html.

[31] GNU Make. [cit. 2016-04-20] Dostupné z: https://www.gnu.org/software/make/.

[32] GCC, the GNU Compiler Collection. [cit. 2016-04-15] Dostupné z: https://gcc.gnu.org/.

[33] Doxygen. [cit. 2016-04-20] Dostupné z: http://www.doxygen.org/.

[34] Graphviz. [cit. 2016-04-20] Dostupné z: http://www.graphviz.org/.

[35] VMware Workstation Player. [cit. 2016-05-03] Dostupné z: http://www.vmware.com/products/player/.

Příloha **A**

Seznam použitých zkratek

D-H Diffie–Hellmanova výměna klíčů

SSCDH Supersingular computational Diffie–Hellman problem

PDL Problém diskrétního logaritmu

DLP Discrete logarithm problem

ECPDL Problém diskrétního logaritmu na eliptické křivce

ECDLP Elliptic curve discrete logarithm problem

NTRU N-th degree truncated polynomial ring

RSA Rivest - Shamir - Adleman

ECC Elliptic curve cryptography

GMP The GNU Multiple Precision Arithmetic Library

NTL A Library for doing Number Theory

GPL GNU General Public License

SSD Solid-state drive

Příloha **B**

Obsah přiloženého DVD

- `readme.txt` ..... stručný popis obsahu DVD
- `application`
  - `Doxyfile` ..... soubor s nastavením pro Doxygen
  - `Makefile` ..... soubor s definicemi cílů pro GNU Make
  - `src` ..... zdrojové kódy implementace
- `pari`
  - `pari-2.7.5.tar.gz` ..... zdrojové soubory knihovny PARI 2.7.5
  - `seadata.tgz` ..... balíček Seadata pro PARI knihovnu
- `text`
  - `BP_Drhova_Klara_2016.pdf` ..... text práce ve formátu PDF
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX