VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2019 Bc. Daniel Herbrych

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

GENEROVÁNÍ ELIPTICKÝCH KŘIVEK PRO KRYPTOGRAFICKÝ PROTOKOL
ELLIPTIC CURVE GENERATOR FOR CRYPTOGRAPHIC PROTOCOL

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Daniel Herbrych**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Radek Fujdiak, Ph.D.**
SUPERVISOR

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

Student: Bc. Daniel Herbrych ***ID:*** 164730
Ročník: 2 ***Akademický rok:*** 2018/19

NÁZEV TÉMATU:

Generování eliptických křivek pro kryptografický protokol

POKYNY PRO VYPRACOVÁNÍ:

V rámci práce student zanalyzuje problematiku generování eliptických křivek pro kryptografické účely se zaměřením pro využití v protokolech využívající jednorázové křivky (např. eECDH). Z kvalitního teoretického i matematického základu následně vybranými metodami realizuje generátor, který bude schopný na základě vstupních parametrů generovat eliptické křivky pro kryptografické účely (velikost grupy i parametrů křivky, společně s typem křivky, bude vybráno na základě provedené podložené analýzy). V druhé fázi pak aplikuje vytvořený generátor do vybraného kryptografického protokolu, tedy prakticky jej zrealizuje a otestuje jeho efektivitu. Následně bude provedena optimalizace a finalizace (bezpečnost by měla být formálně ověřena).

DOPORUČENÁ LITERATURA:

[1] D. Hankerson, A. Menezes, S. Vanstone. Guide to Elliptic Curve Cryptography. Springer, 2003.

[2] A. Sutherland. „Elliptic Curves.“ MIT MATHEMATICS (18.783). 2017.

Termín zadání: 1.2.2019 ***Termín odevzdání:*** 16.5.2019

Vedoucí práce: Ing. Radek Fujdiak, Ph.D.
Konzultant:

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Tato práce má za cíl vytvořit generátor eliptických křivek pro kryptografický protokol. Je k tomu použita knihovna MIRACL a jazyk C++. Jedním ze stěžejních problémů je určení řádu grupy eliptické křivky. Pro toto se používá algoritmus SEA (Schoof Elkies Atkin), dle toho je tak metoda napříč zdroji zvána jako metoda počítání bodů na křivce, SEA metoda, případně i jinak. Druhou metodou pro generování je metoda komplexního násobení (complex multiplication, zkráceně CM). Obě jsou v generátoru k dispozici. Byla měřena závislost rychlosti základních operací nad eliptickou křivkou na velikosti křivky a také na jednotlivých parametrech křivky. Bylo implementováno hybridní schéma ECIES jako praktické ověření funkčnosti generátoru. Dále byly měřeny rychlosti ECIES šifrování a dešifrování v závislosti na různých parametrech, např. velikost křivky, jakou metodou byla křivka vygenerována, velikost zprávy a další.

KLÍČOVÁ SLOVA

Eliptické křivky, MIRACL, kryptografie, generování eliptických křivek, SEA algoritmus, CM metoda, ECIES

ABSTRACT

This thesis deals with creation of elliptic curves generator. MIRACL library and C++ language are used. One of important issues is to determine the order of the elliptic curve group. SEA algorithm (Schoof–Elkies–Atkin) is used for point counting on the elliptic curve. Method with this algorithm is called as counting points method, SEA method etc. Next method is CM method. Both methods are available in the generator. The measurements of dependency of basic operations speed on the group size and parameters were done. ECIES hybrid scheme was implemented. It is practical verification of proper functionality of the generator. Another benchmarks measured dependency of ECIES encryption and decryption on various parameters, e.g. size of the curve, generating method, message size etc.

KEYWORDS

Elliptic curves, MIRACL, cryptography, generating of elliptic curves, SEA algorithm, CM method, ECIES

HERBRYCH, Daniel. *Generování eliptických křivek pro kryptografický protokol*. Brno, 2019, 59 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Radek Fujdiak, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Generování eliptických křivek pro kryptografický protokol“ jsem vypracoval(a) samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Radku Fujdiakovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora(-ky)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic
http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno ....

podpis autora(-ky)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Tato práce se věnuje oblasti eliptických křivek v kryptografii. Eliptické křivky jsou v současné době méně používané jak např. RSA. Ať už je důvodem jejich pozdější nástup anebo třeba doznívající patenty, díky menším klíčům při zachování stejné úrovně zabezpečení mají menší nároky a například u omezených zařízení je to nesporná výhoda. Jejich nevýhodou je, že nejsou zcela prozkoumány, a to je také možná důvodem jejich omezeného nasazení. Stěžejním cílem této práce je vytvořit generátor eliptických křivek. Následuje implementace v kryptografickém protokolu. To obnáší vygenerování odpovídajících parametrů eliptické křivky, které jsou závislé na jejím typu. Nejčastěji jsou využívány eliptické křivky nad poli o tzv. charakteristice 2 (binární pole) anebo o charakteristice rovné prvočíslu většímu jak 3 (prvočíselné pole). Tento, ale i další základní pojmy, jsou vysvětleny v kapitole Teoretické základy. V této kapitole je úvod do kryptografie, stručné informace o symetrické a asymetrické kryptografii, dále je zmíněna jako nezbytný základ modulární aritmetika, a to včetně počítání s polynomy nad binárním polem. Pro práci s eliptickými křivkami je taktéž nutno znát základy algebraických struktur, kde jsou uvedeny definice základních pojmů z této oblasti algebry, tj. co jsou to tělesa, grupy, jejich vybrané vlastnosti atd.

V další kapitole je úvod do eliptických křivek, což zahrnuje rovnice eliptických křivek nad konkrétními konečnými poli (Weierstrassova forma), základní operace s body na eliptických křivkách, což je operace sčítání a násobení, dále aritmetiku křivek nad konečnými poli. Jsou uvedeny i další typy křivek, příkladem budiž křivky hypereliptické. Zde jsou dále obsaženy informace o jednotlivých kryptografických knihovnách, které s eliptickými křivkami pracují. Některé mají implementované křivky, které jsou doporučené, některé ale umožňují alespoň zčásti vytvářet křivky vlastní. Samozřejmě knihovny nejsou uvedeny všechny, je jich celá řada, vybrány jsou však především ty nejznámější. Každá knihovna používá určitý jazyk, někdy jsou ale poskytována rozhraní pro více programovacích jazyků.

V praktické části práce je už popsáno, jakým způsobem se křivky mají generovat, jaké algoritmy a dílčí kroky jsou ke generování křivek použity a jaké problémy se vyskytly. Jednou ze stěžejních částí je určení řádu křivky, neboli počet jejích bodů. Jsou zde uvedena možná řešení dílčích problémů, včetně jejich charakteristik, výhod a nevýhod. V závěru je pak shrnuto, čeho bylo dosaženo.

1 TEORETICKÉ ZÁKLADY

1.1 Úvod do kryptografie

Kryptografie je vědní obor, který se zabývá výzkumem kryptografických protokolů, algoritmů aj., které převádí otevřený text do podoby, kterou lze číst pouze s určitou znalostí. Děje se tak prostřednictvím tzv. šifrování, převod šifrovaného textu na otevřený text se nazývá dešifrování, viz Základní pojmy. Dále se zabývá například digitálními podpisy a dalšími způsoby, jak zajistit například autentičnost zprávy, integritu dat a další. Kryptografie je podčást kryptologie, která navíc sestává i z kryptoanalýzy. Kryptoanalýza naopak řeší, jak luštit zašifrovaný text, aniž by byla k dispozici potřebná znalost - tzv. klíč. Je to nauka, jak překonat kryptografické techniky. Kryptosystém (kryptografický systém) je obecně soubor kryptografických technik použitých k zabezpečení informací, nejčastěji je to šifrování.
Cílem kryptografie je zaručení důvěrnosti, integrity dat, autentizace, čerstvosti dat a nepopiratelnosti. Důvěrnost znamená udržení informací v tajnosti vůči těm, kteří nemají oprávnění informace mít. Způsob, jak dosáhnout důvěrnosti, není jen jeden, je to např. fyzická ochrana, matematické algoritmy, atd. Integrita dat znamená zajištění toho, aby data nebyla neoprávněně změněna. Manipulovat s daty lze třeba vložením, smazáním, nahrazením informace atd. Lze se tak spoléhat na to, že doručená zpráva nebyla porušena. Autentizace nebo také identifikace je prokázání toho, že dotyčná strana je opravdu ta, za koho se vydává. Pokud začnou komunikovat dvě strany, každá by se měla té druhé identifikovat. Čerstvost dat znamená, že data jsou aktuální a nejsou přenášena opakovaně. Nepopiratelnost pak poskytuje jistotu, že autor nemůže popřít, že je autor.

1.1.1 Základní pojmy

- Zpráva - nosičem informace můžou být např. úrovně elektrického napětí. Zpráva je pak posloupnost těchto úrovní napětí, ve které je zakódována nějaká informace. V kryptografii je to posloupnost celých čísel, ve které jsou obsaženy určité informace.

  Autentičnost zprávy pak znamená, že informace o původu jsou pravdivé. Důvěrnost zprávy znamená, že ke zprávě mají přístup pouze oprávněné osoby. K zajištění těchto dvou aspektů se používá více technik. Co se týče důvěrnosti, tak jedna z nich je skrytí zprávy. Věda řešící tuto oblast se jmenuje steganografie. Příkladem budiž využití nejméně významných bitů obrázků k přenosu zprávy. Další technika je ochrana přístupu, kde přístup ke zprávám mají jen oprávněné osoby. Posledním typem ochrany je transformace zpráv. Zde

se původní posloupnost (zpráva) transformuje na jinou posloupnost zvanou kryptogram. K tomu má přístup i potenciální útočník.
K zajištění autentičnosti je potřeba, aby původce zprávy přiložil tzv. pečeť.

- Šifrování - proces, během kterého se šifruje zpráva a výsledkem je kryptogram, který je bez znalosti klíče nečitelný.
- Dešifrování - proces, během kterého se dešifruje kryptogram a výsledkem je zpráva.
- Klíč - informace, která je potřebná na straně původce k šifrování a na straně adresáta k dešifrování. Můžou, ale nemusí být stejné.
- Šifra - šifrovací nebo dešifrovací algoritmus.
- Hešovací funkce - jednosměrná matematická funkce, je jednoduché vypočítat na základě vstupu výstup, ale naopak je to velice obtížné, takřka nemožné. Účelem je vytváření tzv. hešů. Na vstup funkce se přivede zpráva o libovolné délce a obrazem funkce je zmíněný heš, který má vždy konstantní délku (většinou 128 až 512 bitů) [1].

1.2 Symetrická a asymetrická kryptografie

Kryptografie se typicky dělí do dvou částí - symetrická a asymetrická. U symetrické kryptografie se používá jeden klíč jak pro šifrování, tak pro dešifrování. Před započetím komunikace je tedy třeba vyřešit distribuci klíče. Jedna z metod je třeba distribuce klíče kurýrem, anebo pomocí DH protokolu (Diffie-Hellman) pro výměnu klíčů.
Výhody symetrické kryptografie (anglicky symmetric-key cryptography) je rychlost, v krátkém čase lze zpracovat velké množství dat, dále klíče symetrických šifer jsou relativně krátké. Symetrické šifry lze použít např. pro generování pseudonáhodných čísel, u hešovacích funkcí, digitálních podpisů. Nevýhodou pak je, že stejný klíč musí sdílet obě strany. Z toho plyne, že pokud bude hodně účastníků, tak klíč bude muset mít každá dvojice z n účastníků. Vztah pro kombinaci bez opakování je [2]:

$$K(k,n) = \frac{n!}{(n-k)!\,k!}. \tag{1.1}$$

Po dosazení za $k = 2$:

$$K(2,n) = \frac{n!}{(n-2)!\,2!} = \frac{n(n-1)}{2} \tag{1.2}$$

klíčů. Pro asymetrickou kryptografii, anglicky public-key cryptography, je pak potřeba $2n$ klíčů, každý účastník má jeden veřejný a jeden soukromý klíč.
Asymetrické kryptosystémy se používají pro šifrování, podepisování a pro výměnu klíčů symetrických kryptosystémů. Asi nejznámější zástupce asymetrických kryptosystémů je algoritmus RSA, kde se využívá složitosti rozložit velké přirozené číslo

na součin prvočísel (tzv. problém faktorizace). Aby byl takový RSA kryptosystém bezpečný, je v současnosti potřeba, aby byl modulus alespoň 2048 b dlouhý. Výhodou asymetrické kryptografie je, že musí být jen jeden soukromý klíč držen v tajnosti, nicméně musí být zajištěna autenticita veřejných klíčů. Dále, dle situace, dvojice soukromý a veřejný klíč může být velice dlouho neměněná. Nevýhodou asymetrických kryptosystémů je, že jsou díky větším klíčům oproti symetrickým kryptosystémům pomalejší. Dále nebyla přímo dokázána bezpečnost, ta je totiž založená na předpokládané obtížnosti několika teoretických problémů.

1.3 Matematická notace

1.3.1 Modulární aritmetika

Modulární aritmetika, někdy také aritmetika zbytkových tříd, narozdíl od klasické aritmetiky, je definována na konečné množině $\mathbb{Z}_n$, jinými slovy pracuje s omezeným rozsahem čísel. Množina $\mathbb{Z}_n$ vznikne z množiny celých čísel $\mathbb{Z}$ tak, že čísla, která mají stejný zbytek po dělení číslem n, jsou totožná. Mějme celá čísla a, b a přirozené číslo n tak, že $a, b \in \mathbb{Z}, n \in \mathbb{N}$. Symbol $a \bmod n$ pak značí zbytek po dělení čísla a číslem n. Pokud platí $a \bmod n = b \bmod n$, pak říkáme, že číslo a je tzv. kongruentní s číslem b modulo n, neboli zbytky po dělení a a b jsou stejné. Znak kongruence jsou tři vodorovné čárky, píšeme tedy

$$a \equiv b \bmod n. \tag{1.3}$$

Tento zápis se čte: čísla a a b jsou kongruentní modulo n. Modulární aritmetika má někdy také označení anglicky clock arithmetic, hodinová aritmetika. Je to příklad z běžného života, kdy na ručičkových hodinách je např. 10 hodin, ale za 3 hodiny bude opět 1 hodina, v běžné aritmetice by 10+3 bylo 13. Zde, u hodin, je vše tedy modulo 12. Zde jsou základní vlastnosti modulární aritmetiky [13][15]:

$$a \pmod n + b(\mathrm{mod} n)] \bmod n \equiv (a + b) \bmod n, \tag{1.4}$$

$$a \pmod n - b \pmod n] \bmod n \equiv (a - b) \bmod n, \tag{1.5}$$

$$a \pmod n \cdot b \pmod n] \bmod n \equiv (a \cdot b) \bmod n. \tag{1.6}$$

Vlastnosti kongruence:

$$a \equiv b \bmod n, \tag{1.7}$$

$$a \equiv a \mod n, \forall a \in \mathbb{Z}, \tag{1.8}$$

$$a \equiv b \bmod n \rightarrow b \equiv a \mod n, \tag{1.9}$$

$$a \equiv b \bmod n \wedge b \equiv c \bmod n \rightarrow a \equiv c \bmod n, \tag{1.10}$$

$$a \equiv b \bmod n \wedge c \equiv d \bmod n \rightarrow a + c \equiv b + d \bmod n. \tag{1.11}$$

Mezi funkcemi v běžné aritmetice a modulární aritmetice je rozdíl. Například funkce 3^x je pro $x > 1$ exponenciální a s rostoucím x se funkční hodnota zvětšuje. Pokud ale funkce bude např. $3^x \bmod 5$, pak její funkční hodnoty budou nevyzpytatelnější. Třeba u příkladu $3^x \equiv 4 \bmod 5$ lze postupným zkoušením relativně snadno zjistit, že hledané x je rovno 2. Ale žádný takový snadný postup, jako je třeba logaritmus ke zjištění hodnoty x u exponenciální funkce, není. Pro větší čísla to ale rozhodně není efektivní postup a zjištění správného výsledku by trvalo dlouho. O takové funkci se pak řekne, že je jednosměrná. Znamená to, že lze snadno vypočítat funkční hodnotu pro nějaké x, ale dobrat se x zpětně je velice obtížné. Číslo x se nazývá diskrétní logaritmus. Této vlastnosti se využívá v kryptografii.

1.3.2 Aritmetika s polynomy nad $F(2^m)$

K eliptickým křivkám nad polem $F(2^m)$ (nebo ekvivalentně F_{2^m}) patří i aritmetika o m-bitových číslech. Tato čísla mohou být považována jako binární polynomy stupně $m - 1$. Binární řetězec $(a_{m-1} \cdots a_1 \ a_0)$ může být vyjádřen jako polynom $a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \cdots + a_2x^2 + a_1x + a_0$, kde $a_i \in 0, 1$. Například čtyřbitové číslo $(1011)_2$ bude reprezentováno polynomem $x^3 + x + 1$. Podobně jako je u modulární aritmetiky $\bmod p$, tak zde je ireducibilní polynom stupně m. Ireducibilní znamená, že tento polynom nejde rozložit na součin dalších dvou polynomů. Pokud je v nějaké operaci stupeň polynomu větší nebo roven m, pak výsledek je redukován na polynom o menším stupni jak m, a to pomocí ireducibilního polynomu.
Koeficienty tedy mohou být 0 nebo 1. Pokud by nastala situace, že má být koeficient větší, pak je redukován na 0 nebo 1 pomocí $\bmod 2$. Všechny operace níže jsou definovány nad polem F_{2^4}, ireducibilní polynom je $x^4 + x + 1$.

Sčítání

Mějme dva polynomy, $A = x^3 + x^2 + 1$, $B = x^2 + x$. Součet $A + B$ potom je x^3+2x^2+x+1. Nad koeficienty se provede operace $\bmod 2$, výsledek je tedy x^3+x+1. Binární reprezentace by byla $A = (1101)_2$ a $B = (0110)_2$, $A + B$ pak je $(1011)_2$, což je operace tzv. XOR (exclusive or, exkluzivní disjunkce), viz tabulka 1.1. Značí se operátorem $\oplus$. Píšeme tedy $A + B = A \oplus B$.

Odčítání

Sčítání a násobení jsou v poli nad F_{2^m} stejné. Mějme opět dva polynomy, $A = x^3 + x^2 + 1$, $B = x^2 + x$. Rozdíl $A - B$ pak je $x^3 - x + 1$. Nad koeficienty se opět provede operace $\bmod 2$, výsledek je tedy $x^3 + x + 1$. Opět je to operace XOR.

Tab. 1.1: Operace XOR

a	b	$a \oplus b$
1	1	0
1	0	1
0	1	1
0	0	0

Násobení

Mějme stejné polynomy A a B jako nahoře. Vynásobením polynomů $A \cdot B$ a aplikací mod 2 pak je mezivýsledek $x^5+x^3+x^2+x$. Protože $m = 4$, nejvyšší možný exponent může být 3. Výsledek se tedy redukuje pomocí ireducibilního polynomu. U dělení polynomu polynomem $(x^5+x^3+x^2+x) : x^4+x+1$ bude postup takový, že první člen x^5 prvního polynomu se vydělí prvním členem ireducibilního polynomu x^4, což je rovno x. Tímto x se násobí ireducibilní polynom a odečte se od prvního polynomu, to je x^3 a zároveň to je výsledek.
Pro názornost dělení polynomů je uveden další příklad, akorát není nad tělesem a tudíž koeficienty se nedělí modulo 2 jako v předchozím případě, v případě jiného tělesa obecně modulo p.
Příklad:

$$\Big(\quad 2x^3 \qquad\quad + x + 2\Big) \div \Big(2x + 1\Big) = x^2 - \tfrac{1}{2}x + \tfrac{3}{4} + \frac{\frac{5}{4}}{2x+1}.$$

$$\begin{array}{r}
-\,2x^3 - x^2 \\
\hline
\qquad -\,x^2 \;+ x \\
\qquad\quad x^2 + \frac{1}{2}x \\
\hline
\qquad\qquad \frac{3}{2}x + 2 \\
\qquad\quad -\,\frac{3}{2}x - \frac{3}{4} \\
\hline
\qquad\qquad\quad \frac{5}{4}
\end{array}$$

Dělení

Dělení je definováno jako násobení inverzním prvkem. Dělení $\frac{a}{b}$ mod $f(x)$ je definováno jako ab^{-1} mod $f(x)$, kde b^{-1} je inverzní prvek k b nad $f(x)$. Inverzní prvek $b^{-}1$ vzhledem k ireducibilnímu polynomu je definován tak, že platí $b \cdot b^{-1}$ mod $f(x) = 1$. Pro nalezení inverzního prvku lze použít Euklidův algoritmus.

1.3.3 Algebraické struktury

Nechť $\mathbb{A}$ je neprázdná množina, $\mathbb{O}$ pak neprázdná množina operací na množině $\mathbb{A}$. Uspořádanou dvojici $(\mathbb{A}, \mathbb{O})$ pak nazveme algebraickou strukturou. Množina $\mathbb{A}$ je

nosič této struktury. Jinými slovy, algebraické struktury jsou množiny, na kterých jsou zavedené nějaké operace. Operace na množině $\mathbb{A}$ je zobrazení

$$\alpha : \mathbb{A}^n \to \mathbb{A}, \tag{1.12}$$

které ke každé n-tici elementů množiny A přiřazuje jiný prvek z množiny $\mathbb{A}$. Číslo n se nazývá arita operace [14]:
1-ární operace se nazývá *unární*, čili $\alpha : \mathbb{A} \to \mathbb{A}$,
2-ární operace se nazývá *binární*, čili $\alpha : \mathbb{A} \times \mathbb{A} \to \mathbb{A}$,
3-ární operace se nazývá *ternární*, čili $\alpha : \mathbb{A} \times \mathbb{A} \times \mathbb{A} \to \mathbb{A}$.

Tělesa

Algebraické těleso $\mathbb{T}$ je jakákoliv množina se dvěma binárními operacemi, které splňuje jisté podmínky. Tato množina obsahuje prvky tělesa a může obsahovat cokoliv, takové těleso může obsahovat všechna přirozená čísla, sudá čísla anebo cokoliv jiného. Množina ale musí být neprázdná. Je potřeba dodat, co je to binární operace. Binární operace na množině $\mathbb{M}$ je zobrazení $f : \mathbb{M} \times \mathbb{M} \to \mathbb{M}$. Jinými slovy, se dvěma prvky množiny $\mathbb{M}$ se provede daná operace a výsledkem je opět prvek z množiny $\mathbb{M}$. Tyto binární operace jsou nazvány sčítání a násobení, nicméně tyto operace nemusí být klasické sčítání a násobení a je to právě kvůli tomu, že v množině mohou být jakékoliv objekty. Tyto operace ale nemohou být úplně libovolné, musí splňovat následující axiomy, aby daná množina byla tělesem [5][6]:

1. $a + b \in \mathbb{T}$ pro libovolné $a, b \in \mathbb{T}$,
2. $(a + b) + c = a + (b + c)$ pro libovolné $a, b, c \in \mathbb{T}$,
3. $a + b = b + a$ pro libovolné $a, b \in \mathbb{T}$,
4. $\exists\, 0 \in \mathbb{T}$ tak, že $a + 0 = a$ pro $\forall a \in \mathbb{T}$,
5. $\forall a \in \mathbb{T}\, \exists - a \in \mathbb{T}$ tak, že $a + (-a) = 0$,
6. $ab \in \mathbb{T}$ pro libovolné $a, b \in \mathbb{T}$,
7. $(ab)c = a(bc) \in \mathbb{T}$ pro libovolné $a, b, c \in \mathbb{T}$,
8. $ab = ba$ pro libovolné $a, b \in \mathbb{T}$,
9. $\exists 1 \in \mathbb{T}$ tak, že $1a = a$ pro $\forall a \in \mathbb{T}$,
10. $\forall a \in \mathbb{T}\, \exists\, a^{-1} \in \mathbb{T}, a \neq 0$ tak, že $a^{-1}a = 1$,
11. $a(b + c) = ab + ac$ pro libovolné $a, b, c \in \mathbb{T}$,
12. $0 \neq 1$.

Axiom 1 znamená, že množina $\mathbb{T}$ je tzv. uzavřená na sčítání, tj. pokud se sečtou obě čísla, výsledkem je opět číslo z množiny $\mathbb{T}$. Druhý axiom je asociativita sčítání, třetí pak komutativita sčítání. Čtvrtý je existence nulového prvku ke sčítání (nulový prvek není číslo nula), pátý pak je existence existence opačného prvku ke sčítání. Toto byly axiomy pro sčítání. Dalších pět axiomů je pro násobení, šestý pak

znamená, že množina je uzavřená vzhledem k násobení. Sedmý a osmý znamenají asociativitu a komutativitu násobení. Devátý je existence jednotkového prvku k násobení (jednotkový prvek není číslo jedna), desátý je existence inverzního prvku k násobení. Jedenáctý je axiom distributivity. Dvanáctý je axiom netriviality - nulový a jednotkový prvek nesmí být stejné. Dále říká, že těleso musí mít nejméně dva prvky. V každém tělese lze definovat operaci odečítání jako přičítání opačného prvku, tj. $a - b = a + (-b)$, a dělení jako násobením inverzním prvkem, tj. $a : b = ab^{-1}$, kde b nesmí být nula. Existuje-li kladné celé číslo v tělese $\mathbb{T}$ tak, že platí $n \cdot 1 = 0$, potom se nejmenší takové číslo nazývá *charakteristika tělesa*. Pokud takové n není, pak má těleso charakteristiku 0. Dále lze ukázat, že charakteristika tělesa může být buď 0, anebo prvočíslo. Pokud by charakteristika bylo číslo složené, pak by platilo $n = kl$ pro nějaká kladná celá čísla $k, l < n$. Pak tedy platí $n = kl = 0$, nule se ale může součin dvou prvků rovnat pouze tehdy, pokud je alespoň jeden z nich nulový. Z toho plyne, že buď $k = 0$ nebo $l = 0$, v každém případě je ale n větší než k i l. Navíc podle axiomu netriviality takové číslo musí být prvočíslo.
Tělesům, která jsou konečná, se říká Galoisova tělesa a značí se GF_q. V kryptografii se pak nejvíce používají GF_{p^m} neboli $GF(p^m)$, kde $q = p^m$, p je prvočíslo a m je přirozené číslo. Konkrétně to jsou konečná binární pole GF_{2^m} neboli $GF(2^m)$, dále pak GF_p neboli $GF(p)$, což jsou prvočíselná pole, kde p je prvočíslo různé od 2 a 3. V praxi se u $GF(p^m)$ většinou volí $m = 1$ (což je $GF(p)$) nebo $m = 2$. Je třeba ještě vysvětlit pojem *pole* (anglicky field), což je takové těleso, které má obě operace komutativní [5][6].

Grupy

Následuje výčet definic a vět ke grupám [8]:

Definice 1.3.1 *Grupoid $(G, \circ)$ je množina $\mathbb{G}$ spolu s binární operací na této množině. Binární operace na množině $\mathbb{G}$ je zobrazení $\mathbb{G} \times \mathbb{G} \to \mathbb{G}$ a značí se např. $\circ$.*

Operace může být komunikativní a asociativní, viz axiomy níže.

Definice 1.3.2 *Grupoidu $(\mathbb{G}, \circ)$ se říká komutativní, pokud je operace $\circ$ komutativní na $\mathbb{G}$. Dále se mu říká asociativní, pokud je operace $\circ$ asociativní na $\mathbb{G}$, neboli také tzv. pologrupa.*

Definice 1.3.3 *Grupoid nazveme grupa, pokud splňuje následující axiomy [6]:*

1. $\forall a, b \in \mathbb{G}, a \circ b \in \mathbb{G}$ *- uzavřenost,*
2. $\forall a, b, c \in \mathbb{G}, (a \circ b) \circ c = a \circ (b \circ c)$ *- asociativita,*
3. $\exists e \in \mathbb{G}, \forall a \in \mathbb{G}, a \circ e = e \circ a = a$ *- existence neutrálního prvku (nebo také jednotkového prvku),*

4. $\forall a \in \mathbb{G}, \exists i \in \mathbb{G}, a \circ i = i \circ a = e$ - *existence inverzního prvku, pokud platí navíc komutativita, pak je tato grupa komutativní, tzv. Abelova grupa,*
5. $\forall a, b \in \mathbb{G}, a \circ b = b \circ a$.

Definice 1.3.4 *V grupách ke každému prvku a existuje právě jeden inverzní prvek, označme ho* $a^{-}1$.

Věta 1.3.1 *Každý grupoid má nejvýše jeden neutrální prvek.*

Věta 1.3.2 *V libovolné pologrupě s neutrálním prvkem ke každému prvku existuje nejvýše jeden prvek inverzní.*

Definice 1.3.5 *Grupa se nazývá triviální, pokud má množina pouze jeden prvek. Tento prvek je zároveň neutrální.*

Pokud operace $\circ$ představuje sčítání, pak se tato grupa nazývá *aditivní*. Neutrální prvek se značí jako 0 (nulový prvek), inverzní prvek pak $-a$.
V případě, že operace $\circ$ představuje násobení, pak se tato grupa nazývá *multiplikativní*. Neutrální prvek se značí jako 1 (jednotkový prvek), inverzní prvek pak $a^{-}1$.
Počet prvků grupy se nazývá *řád grupy*, pokud má grupa konečný počet prvků. Pokud ne, pak je řád nekonečno. *Řád prvku a* je nejmenší přirozené číslo n tak, že $a^n = 1$. Pokud takové číslo není, řád tohoto prvku v grupě $\mathbb{G}$ je nekonečno. V každé konečné grupě má každý prvek konečný řád. Grupa je konečná právě tehdy, když počet prvků je konečný. Umocňování v grupě pak znamená opakované použití operace $\circ$ na $\mathbb{G}$.

Definice 1.3.6 *Exponent grupy je takové nejmenší přirozené číslo n, že pro každé* $a \in \mathbb{G}$ *platí* $a^n = e$*, neboli je to nejmenší společný násobek řádů všech prvků z grupy* $\mathbb{G}$ *[7].*

Definice 1.3.7 *Grupa se nazývá cyklická, pokud existuje takový prvek* $a \in \mathbb{G}$*, že pro každý prvek b existuje celé číslo i tak, že* $b = a^i$*. Prvek a je pak generátor grupy* $\mathbb{G}$ *[6].*

2 ELIPTICKÉ KŘIVKY V KRYPTOGRAFII

Kryptografie nad eliptickými křivkami (Elliptic Curve Cryptography, ECC) je založená na algebraických strukturách eliptických křivek nad konečnými tělesy. Dnes můžeme eliptické křivky nalézt v protokolech TLS nebo třeba SSH, bitcoin a další kryptoměny nevyjímaje. Další použití je například v Rakousku, kde občané mají ve svých občanských průkazech čip obsahující RSA nebo ECDSA (Elliptic Curve Digital Signature Algorithm) klíč. Dnes je stále používanější RSA, důvodů, proč to tak je, může být více. Např. že RSA je jednodušší, anebo že eliptické křivky jednoduše přišly až po RSA. Protože ECC poskytuje srovnatelnou úroveň zabezpečení s menším výpočetním výkonem a menšími nároky na energii, je výhodné ECC použít např. pro mobilní zařízení, obecně pak pro zařízení s omezenými zdroji. Zajímavostí je, že s použitím eliptických křivek v kryptografii přišli nezávisle na sobě Neal Koblitz z washingtonské univerzity a Victor S. Miller z IBM v tomtéž roce, a to 1985. Pojem eliptická křivka vznikl na základě toho, že dříve se počítaly pomocí kubických rovnic obvody elips. Eliptické křivky jsou podmnožinou kubických křivek.
ECC je analogie už existujících systémů s veřejným klíčem, kde modulární aritmetika je nahrazena aritmetikou, která je založená na operacích s body na eliptické křivce. Bezpečnost ECC stojí na problému diskrétního logaritmu nad eliptickými křivkami, což je v současnosti obtížněji řešitelné než klasický diskrétní logaritmus. Nejlepší algoritmy pro řešení mají exponenciální charakter, díky čemuž lze použít menší délku klíče. Eliptické křivky mají ale také určité nevýhody, například přetrvávající obavy ohledně prozkoumanosti křivek, doznívající patenty anebo chybějící masivnější nasazení. Není publikován žádný matematický důkaz o bezpečnosti ECC.

2.1 Úvod do teorie eliptických křivek

Eliptické křivky jsou speciální podtřídou kubických křivek, čili polynomů třetího stupně. Obecně rovinnou křivkou rozumíme množinu bodů vyhovující rovnici [4]:

$$F(x, y) = 0. \tag{2.1}$$

Eliptické křivky se používají nad konečnými tělesy, která lze popsat a každé těleso je určeno svým řádem, což je počet prvků. Těleso označíme F_q a $q = p^m$, kde q je počet prvků tělesa, p je prvočíslo a m přirozené číslo. Eliptická křivka může být definována jako množina [4]:

$$E = \{[x, y] \in F_q^2 \setminus \{[0, 0]\}, F(x, y) = 0\} \cup \{O\}, \tag{2.2}$$

kde $\{O\}$ je tzv. bod v nekonečnu. Dále tzv. Weierstrassova forma rovnice eliptické křivky má tvar [4]:

$$F(x, y) = y^2 + a_1xy + a_2y - x^3 - a_3x^2 - a_4x - a_5. \tag{2.3}$$

Tato rovnice se dále zjednodušuje pro jednotlivá tělesa a upravují se podmínky pro koeficienty tak, aby křivka nebyla singulární [5]:

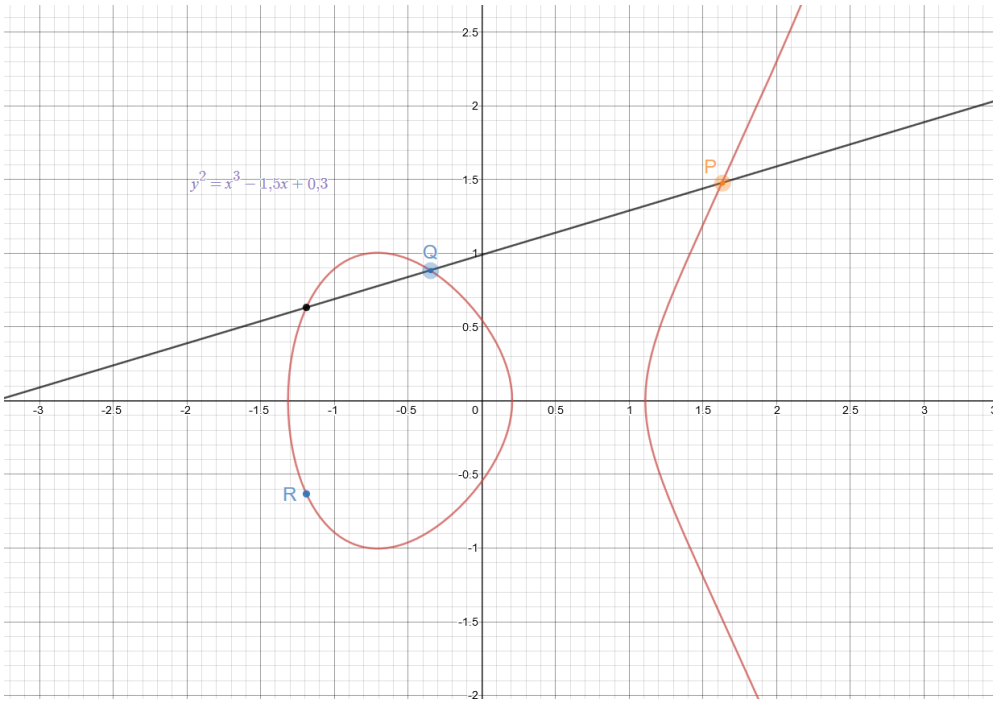
1. Pro $q = p^m$, kde $p > 3, m \geq 1$ a platí $4a^3+27b^2 \neq 0$. Eliptická křivka nad tímto tělesem (o charakteristice větší jak 3) pak má tvar $y^2 = x^3 + ax + b$, kde a, b jsou celá čísla modulo q.
2. Pro $q = 2^m$, kde $m \geq 1$ a $b \neq 0$. Eliptická křivka nad tímto tělesem (o charakteristice 2) pak má tvar $y^2 + xy = x^3 + ax^2 + b$, kde a, b jsou celá čísla modulo q. Toto je eliptická křivka nad binárním polem.
3. Pro $q = 3^m$, kde $m \geq 1$ a $b \neq 0$. Eliptická křivka nad tímto tělesem (o charakteristice 3) pak má tvar $y^2 = x^3 + ax^2 + b$, kde a, b jsou celá čísla modulo q.

2.1.1 Základní operace s body

Sčítání bodů

Nejprve je ukázáno grafické sčítání bodů na křivce v rovině. Na obrázku 2.1 je znázorněna eliptická křivka určená rovnicí $y^2 = x^3-1, 5x+0, 3$. Součet dvou různých bodů křivky dá opět bod na křivce. Jak lze vidět na obrázku, body $P = (x_p, y_p)$ a $Q = (x_q, y_q)$ jsou spojeny přímkou. Výsledkem sčítání je pak bod R, který je symetrický podle osy x (opačný) k bodu, který vznikl protnutím křivky sestrojenou přímkou. Při sčítání bodu a sebe samého, tj. $P + P$ bude přímka tečnou ke křivce v bodě P. Výsledkem je bod R. Pokud $y_p \neq 0$, pak tečna protne křivku v bodě opačném k bodu R.

Pokud budeme na křivce postupně sčítat body, tj. posloupnost $P, 2P, 3P \cdots$, tak jelikož má křivka konečný počet bodů, dojdeme do bodu, kdy se začne posloupnost opakovat. Během postupného přičítání narazíme v m-tém kroku na bod, kdy platí $mP = nP$, kde nP je nějaký předešlý bod. Z toho plyne $mP - nP = O, m > n$, tedy $(m-n)P = O$ a dostaneme tedy nějaké číslo $r = (m-n)$, pro které platí $rP = O$. Vždy se tedy postupným přičítáním dostaneme do bodu O a pak začíná opět posloupnost $P, 2P, 3P \cdots$. Nejmenší takové číslo r se nazývá *řád bodu P*. Řád bodu je důležitý parametr, v následujícím textu bude zmíněn například generátor grupy. Aby byla křivka považována za bezpečnou, musí generátor generovat dostatečný počet bodů křivky. S pomocí řádu křivky a řádu bodu se pak počítá tzv. kofaktor [4][6].



Obr. 2.1: Příklad sčítání bodů na křivce

Zdvojnásobení bodů

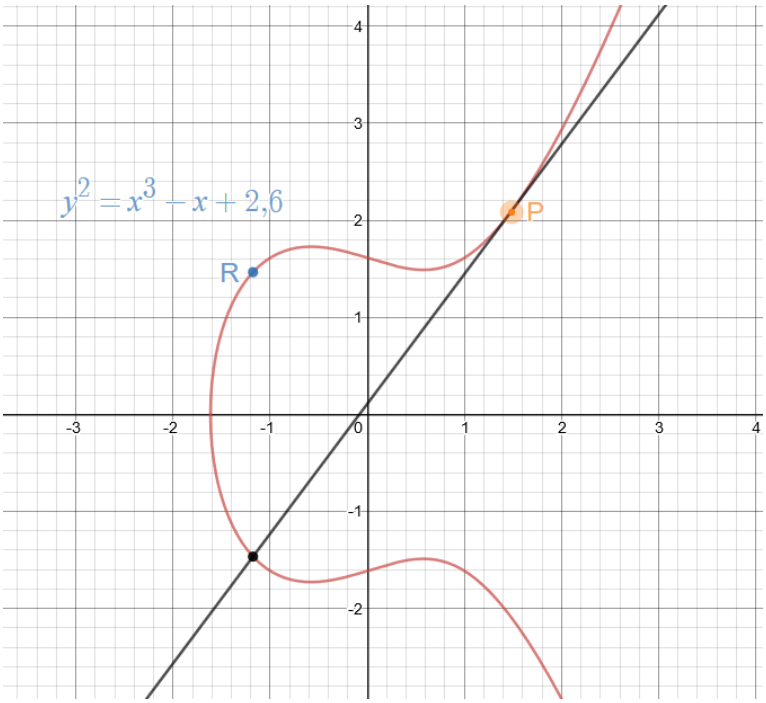
Zdvojnásobení (angl. point doubling) je přičtení bodu P k sobě samému, tj. výsledkem je $2P$. Geometricky se zdvojení provede tak, že se udělá tečna v bodě P, ta protne křivku v určitém bodě. K tomuto bodu se nalezne opět bod opačný R a ten je pak výsledkem. Pokud by souřadnice y byla nulová, pak tečna protne bod v nekonečnu O. Pak by platilo $2P = O$. Analyticky se pak zdvojnásobení provede pomocí vztahů z [4][6]. Znázornění je na obrázku 2.2.

Násobení bodů

Bod P na křivce je vynásoben skalárem k, výsledkem bude bod Q na též křivce, neboli $Q = kP$. Násobení se provádí dvěma základními operacemi, a to sčítáním bodů a zdvojnásobením bodu. Sčítání i zdvojnásobení už bylo vysvětleno. Jedna z metod pro násobení je metoda Double-and-Add [4][6].

2.1.2 Eliptické křivky nad prvočíselným polem

Eliptické křivky nad prvočíselným polem GF_p nejsou hladkými křivkami, a proto sčítání a zdvojnásobení bodů nelze interpretovat geometricky, ale jen algebraicky. Číslo p je prvočíslo a je větší jak 3. Souřadnice x a y náleží tělesu GF_p a platí



Obr. 2.2: Příklad zdvojnásobení bodu na křivce

následující rovnice [6]:

$$y^2 = x^3 + ax + b, \tag{2.4}$$

tj. platí $y^2 \equiv x^3 + ax + b \bmod p$. Koeficienty a, b jsou z tělesa F_p a musí platit podmínka $4a^3 + 27b^2 \not\equiv 0 \bmod p$. Takto definovaná množina pak tvoří grupu. V této grupě bude opačný bod k O bod $-O$, pro ostatní body na křivce $P = (x_p, y_p)$ bude opačný bod $-P = (x_p, y_p \bmod p)$. Dále bude platit $P + (-P) = O$ a $P + O = O$.

Sčítání bodů

Směrnice přímky, která prochází různými body P a Q, je dána vztahem (pokud nejsou opačné) [6]:

$$s = (y_q - y_p)/(x_q - x_p). \tag{2.5}$$

Z rovnice křivky lze odvodit bod R [5]:

$$x_r = s^2 - x_p - x_q, \tag{2.6}$$

$$y_r = s(x_p - x_r) - y_p. \tag{2.7}$$

Pokud $P = Q$, tato přímka se stane tečnou se směrnicí [5]:

$$s = (3x_p + a)/(2y_p). \tag{2.8}$$

Jak již bylo řečeno, křivka má dodefinovaný bod v nekonečnu. Pokud totiž sčítáme body opačné, pak jejich spojnice, která je rovnoběžná k ose y, protne křivku v nekonečnu, tj. $P + (-P) = O$. Dále pro každý bod P na křivce platí [5]:

$$P + O = P \tag{2.9}$$

$$O + O = O \tag{2.10}$$

$$-O = O. \tag{2.11}$$

Zdvojnásobení bodu

Prvně se spočítá tečna v bodě P [5]:

$$s = (3{x_p}^2 + a)/(2y_p), \tag{2.12}$$

kde a je jeden z parametrů eliptické křivky. Výsledný bod je pak určen vztahy [5]:

$$x = s^2 - 2x_p, \tag{2.13}$$

$$y = -y_p + (x_p - x). \tag{2.14}$$

Doménové parametry

Doménové parametry eliptických křivek nad prvočíselným polem F_p jsou p, a, b, G, n, h. Parametr p je prvočíslo, a, b jsou parametry definující křivku [6]:

$$y^2 \bmod p = x^3 + ax + b \bmod p. \tag{2.15}$$

Bod G je generátor (anglicky také base point), což je bod (x_G, y_G) na křivce vybraný pro další operace. Parametr n je řád generované podgrupy. Skalár pro násobení bodů je pak vybírán z intervalu $\langle 0; n-1 \rangle$. Dalším parametrem je h neboli kofaktor, kde $h = \#E(F_p)/n$. $\#E(F_p)$ je počet bodů na křivce. Většinou se volí parametry tak, aby kofaktor byl 1, 2, 4 či 8. Nejlépe, aby měl hodnotu 1, jinými slovy, aby generátor generoval všechny body eliptické křivky.

2.1.3 Eliptické křivky nad binárním polem

Binárním pole, zkráceně $GF(2^m)$ má charakteristiku 2. Platí, že $m \geq 1$ a $b \neq 0$. Existují dva typy těchto křivek, tzv. supersingulární a nonsupersingulární. Supersingulární křivky mají j-invariant rovný 0. Mají rovnici [4]:

$$y^2 + cy = x^3 + ax + b, \tag{2.16}$$

kde $a, b, c \in GF(2^m)$ a $c \neq 0$. U tohoto typu křivek se snadněji určí řád křivky (počet bodů na křivce), což je jinak obtížná úloha využívající náročné algoritmy. Kvůli

útoku, který navrhli Menezes, Okamoto a Vanstone (tzv. MOV attack), se v současných kryptosystémech nepoužívají [9].
Druhý typ - obyčejné křivky, nonsupersingulární, má j-invariant různý od 0. Eliptická křivka pak nad tímto tělesem má tvar [4]:

$$y^2 + xy = x^3 + ax^2 + b, \tag{2.17}$$

kde a, b jsou celá čísla z tělesa $GF(2^m)$ modulo 2^m, $b \neq 0$. Všechny operace jsou počítány pomocí polynomiální aritmetiky.

Sčítání bodů

Mějme body P, Q a zároveň platí $P \neq \pm Q$, pak souřadnice nového bodu R, který je daný součtem bodů P, Q, jsou [5]:

$$s = \frac{y_P + y_Q}{x_P + x_Q}, \tag{2.18}$$

$$x_R = s^2 + s + x_P + x_Q + a, \tag{2.19}$$

$$y_R = y_P + x_R + s(x_P + x_R). \tag{2.20}$$

Zdvojnásobení bodu

Mějme bod P, pak $2P = R$ a pokud $x_P \neq 0$, bude platit [5]:

$$s = x_P + \frac{y_P}{x_P}, \tag{2.21}$$

$$x_R = s^2 + s + a, \tag{2.22}$$

$$y_R = x_P^2 + (s+1)x_R. \tag{2.23}$$

Doménové parametry

Doménové parametry eliptických křivek nad polem F_{2^m} jsou $m, f(x), a, b, G, n, h$. Parametr m je celé číslo, prvky konečného tělesa F_{2^m} jsou celá čísla o maximální délce m bitů. Dále $f(x)$ je ireducibilní polynom řádu m. Čísla a, b jsou parametry definující křivku [4]:

$$y^2 + xy = x^3 + ax^2 + b. \tag{2.24}$$

Bod G je generátor (x_G, y_G), což je bod na křivce vybraný pro další operace. Parametr n je řád generované podgrupy. Skalár pro násobení bodů je pak vybírán z intervalu $\langle 0; n-1 \rangle$. Dále h je kofaktor, kde $h = \#E(F_{2^m})/n$. $\#E(F_{2^m})$ je počet bodů na křivce.

2.1.4 Koblitzovy křivky

Koblitzovy křivky popisují buď binární anomální křivky nad $GF(2^m)$ s rovnicí [4]:

$$y^2 + xy = x^3 + ax^2 + 1, \tag{2.25}$$

kde $a, b \in 0, 1$, anebo křivky nad konečným polem F_p [4]:

$$y^2 = x^3 + ax + b. \tag{2.26}$$

Křivka je anomální právě tehdy, když platí $\#E(F_q) = q$, kde $\#E$ je řád křivky, q je počet prvků tělesa F_q. Toto je zobecnění Koblitzových křivek, ale stejné principy týkajících se efektivity výpočtů platí u obou forem.
Jedna z Koblitzových křivek, konkrétně *secp256k1*, je použita u měny Bitcoin. Její parametry jsou $a = 0, b = 7$. Tato i ostatní Koblitzovy křivky, které jsou standardizované v SECG, jsou nad konečnými poli. Výhodou těchto křivek je velmi efektivní zdvojnásobování bodu na křivce. Dále jsou nonsupersingulární, neboli obyčejné, což je chrání před útokem typu MOV. Řád grupy má jedno číslo z prvočíselného rozkladu velké, takže jsou tyto křivky chráněny také před výpočtem diskrétního logaritmu algoritmem baby-step/giant-step i Pollard-Rho algoritmem (algoritmus pro řešení diskrétního logaritmu nad eliptickými křivkami). Zdvojnásobování bodu je velice efektivní. Další výhodou je, že jdou snadno nalézt a také jsou nenáročné na implementaci. Jednou z charakteristik Koblitzových křivek je, že řád křivky lze vypočítat snadno [10].

2.1.5 Hypereliptické křivky

Hypereliptické křivky jsou speciální třídou algebraických křivek a může na ně být nahlíženo jako na zobecnění eliptických křivek. Každá hypereliptická křivka má rod $g \geq 1$. Hypereliptická křivka s rodem 1 je eliptická křivka.

Definice 2.1.1 *Mějme hypereliptickou křivku C o rodu $g \geq 1$ nad polem F, $\bar{F}$ je pak algebraický uzávěr, to je algebraicky uzavřené nadtěleso. Její rovnice pak má formu [12]:*

$$C : y^2 + h(x)y = f(x) \in F[x, y], \tag{2.27}$$

kde $h(x) \in F[x]$ je polynom stupně maximálně g, $f(x) \in F[x]$ je polynom stupně maximálně $2g + 1$ a nejsou zde žádná řešení $(x, y) \in \bar{F} \times \bar{F}$ vyhovující rovnici $y^2 + h(x)y = f(x)$ a zároveň parciálním derivacím $2y + h(x) = 0$ a $h'(x)y - f(x) = 0$.

Bod singularity na křivce C je řešení $(x, y) \in \bar{F} \times \bar{F}$, které zároveň vyhovuje rovnici $y^2 + h(x)y = f(x)$ a zároveň parciálním derivacím $2y + h(x) = 0$ a $h'(x)y - f(x) = 0$. Z definice tedy plyne, že hypereliptická křivka nemá žádné body singularity.

V HECC (hypereliptic curve cryptography) se v současnosti experimentuje s rodem $g = 2$. HECC je rychlejší jak ECC, co se týče určitých operací, v jiných je zase pomalejší [11][12].

2.2 Knihovny pracující s ECC

2.2.1 TinyECC

TinyECC je knihovna, která byla primárně navržena pro ECC v bezdrátových senzorových sítích a která poskytuje určitou flexibilitu konfigurace a integrace různých operací do bezdrátových senzorových sítí. Co se týče bezpečnosti, TinyECC poskytuje schémata, kde byla prokázána a dobře prostudována bezpečnost, tj. ECDSA, ECDH a ECIES. TinyECC už obsahuje parametry křivek doporučených SECG (Standards for Efficient Cryptography Group). Jsou to křivky *secp160k1*, *secp160r1* a *secp160r2*. Vylepšení efektivity: První je typ konečných polí, nad kterými se provádí základní operace. Podporována jsou zde prvočíselná pole F_p, kde p je velké prvočíslo, protože pole typu F_{2^m}, kde m je celé číslo, nejsou dostatečně podporována mikroprocesory u senzorů. Za druhé, jde zde použít spousta optimalizací pro základní operace, které lze vypnout anebo zapnout. Většina kódu je napsána v jazyce nesC kvůli přenositelnosti, některé kritické operace jako násobení velkých čísel v assembleru, ale jen pro Atmega128l(MICAz), MSP430(TelosB/Tmote Sky) a XScale(Imote2). Operace jako modulární sčítání a násobení jsou převzány z jiné knihovny. Při nasazení TinyECC v senzorových sítích se používá jazyk Java a je potřeba mít operační systém TinyOS, což je operační systém pro bezdrátová zařízení s nízkým výkonem [17][18].

2.2.2 OpenSSL

OpenSSL poskytuje dvě hlavní knihovny - libssl a libcrypto. Libcrypto má základní kryptografii a používá ji libssl. Nicméně libcrypto naopak může být použita bez libssl. Je napsána v jazyce C. OpenSSL podporuje RSA, DSA, z ECC pak ECDSA a ECDH. Ve většině případů by mělo být použito vysokoúrovňové rozhraní, tzv. EVP rozhraní. To nabízí soubor funkcí pro šifrování či dešifrování, podepisování a ověřování, generování hešů a další. K dispozici je API pro jazyk C. Dále jsou tu nízkoúrovňová rozhraní pro práci s jednotlivými algoritmy. Sice nejsou doporučeny nezkušeným uživatelům, ale poskytují větší kontrolu a možnosti, které s použitím vysokoúrovňových rozhraní nejsou možné, ale například pro použití pouze ECDSA a ECDH stačí rozhraní vysokoúrovňové. Díky low-level rozhraní lze provádět základní operace s body, testovat, zda bod leží na křivce a další [16].

2.2.3 WolfSSL

WolfSSL, dříve CyaSSL, je knihovna určená pro embedded zařízení, zařízení s omezenými zdroji a RTOS (real-time operating system). Je až dvacetkrát menší jak OpenSSL a jejími výhodami jsou tedy velikost a také rychlost. Je napsána v ANSI C. WolfSSL obsahuje SSL/TLS knihovnu a má také kryptografické jádro zvané wolfCrypt. Hlavní jazyk je C, použít lze ale i jazyk Java skrze JNI (Java Native Interface) wrapper a C# wrapper. Dále lze použít Python, PHP a Perl (skrze swig rozhraní).
WolfSSL zastřešuje i jiné produkty - další knihovny, např. pro SSH, ty ale nejsou pro ECC důležité. Velikost WolfSSL je 20 až 100 kB, runtime paměť je 1 až 36 kB. WolfCrypt podporuje generování klíčů a co se týče ECC algoritmů, tak to jsou ECDH-ECDSA, ECDHE-ECDSA, ECDH-RSA, ECDHE-RSA. Nejsou podporovány pouze tyto algoritmy, ale jsou zde i funkce umožňující operace s body na křivce [19][20].

2.2.4 RelicToolKit

Relic je kryptografická knihovna s důrazem kladeným na efektivitu a flexibilitu. Má implementovanou aritmetiku s prvočíselnými a binárními poli a eliptické křivky nad těmito poli. Podporovány jsou taktéž kryptografické protokoly jako třeba RSA, ECDSA, ECMQV, ECSS, ECIES a další. Přímo v modulu "Elliptic curve cryptography" je sada funkcí umožňující práci s body. Používaný je zde jazyk C [21].

2.2.5 borZoi

BorZoi je knihovna přímo pro ECC, která implementuje následující algoritmy za použití eliptických křivek definovaných nad konečnými poli o charakteristice 2, čili binárnámi: ECDH, ECDSA, ECIES. AES, hešovací algoritmus SHA-1, funkce pro odvození klíče KDF2 (IEEE P1363A) a technologie HMAC (IEEE P1363) jsou podporovány též. Používaný jazyk je zde C++. Lze ale použít i knihovnu jBorZoi, která je určená pro jazyk Java. Jsou zde low-level třídy reprezentující polynomy tvaru

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0,$$

kde koeficienty a_i jsou modulo 2. Všechny běžné operace s polynomy jako sčítání, odčítání, násobení a dělení jsou podporovány. Další třídou je konečné binární pole, kde prvky jsou polynomy tvaru

$$b_{m-1} x^{m-1} + b_{m-2} x^{m-2} + \cdots + b_0,$$

kde všechny operace jsou modulo ireducibilní polynom

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0.$$

Ireducibilní polynom je takový polynom, který nelze rozložit na součin dalších dvou polynomů. Dále je zde třída pro práci s body a třída reprezentující eliptickou křivku

$$E : y^2 + xy = x^3 + ax^2 + b.$$

Lze zde vytvořit vlastní křivky rozdílné od těch už definovaných. Je doporučeno zkontrolovat si dle IEEE P1363, zda uživatelem definovaná křivka není slabá - taková, že může snížit úroveň zabezpečení [22].

2.2.6 Crypto++

Crypto++ je volně dostupná C++ knihovna podporující celou řadu algoritmů, např. AES, blokové a proudové šifry, hešovací funkce, algoritmy asymetrické kryptografie atd. Z ECC pak hlavně ECDSA, ECGDSA,ECIES, ECDH a ECMQV. Jsou zde i další funkce, třeba jako generátor pseudonáhodných čísel, aritmetika konečných polí, a to jak binárních, tak prvočíselných, nebo také operace s polynomy. Dále je zde generování prvočísel a další algoritmy, ikdyž ne přímo pro kryptografii. Jsou zde přímo dvě třídy reprezentující eliptické křivky nad prvočíselnými a binárními poli. Crypto++ má křivky, které schválili ANSI, Brainpool a NIST, neposkytuje přímo funkcionalitu na generování křivek, ale lze toho docílit pomocí nástroje Elliptic Curve Builder (ECB). Crypto++ je komplexní a rozšířená knihovna, která je také, narozdíl od některých jiných knihoven, stále aktualizovaná. Co se týče ECB builderu, ten nelze v této práci použít, protože je to jen grafický nástroj s omezenými možnostmi [23].

2.2.7 LibTomCrypt

LibTomCrypt je také rozšířená, stále aktualizovaná knihovna. Je napsána v ISO C, potřeba je sada překladačů GCC. Opět jsou zde podporovány různé blokové i proudové šifry, hešovací funkce, autentizace, generování pseudonáhodných čísel, asymetrická kryptografie - RSA, DSA, samozřejmě ECC (ECDSA a ECDH) a další. Jsou zde připraveny křivky nad prvočíselným polem a to podle NIST, křivka musí být tvaru

$$y^2 = x^3 - 3x^2 + b.$$

Lze však i specifikovat vlastní křivky [24].

2.2.8 MIRACL

MIRACL je knihovna napsaná v jazyce C a bývá používána často konkrétně pro ECC. Přímo je uvedeno, že narozdíl od spousty jiných kryptografických knihoven umožňuje pracovat s prostředími s omezenými zdroji, např. mobilní aplikace, embedded atd. MIRACL je používána ve spoustě organizací, z těch větších jsou uvedeny třeba Hitachi, Intel, Panasonic a další. Je k dispozici C++ rozhraní. Z ECC jsou zde podporovány křivky nad prvočíselným i binárním polem. Obecně je zde mnoho matematických low-level operací. Co se týče ECC, tak jsou implementovány funkce pro práci s body, různé algoritmy jako SEA, Schoofův algoritmus atd. Jak již bylo řečeno, použít lze jazyky C nebo C++ [25].

2.2.9 Bouncy Castle

Tato knihovna je vyvinuta Legion of the Bouncy Castle. Používaný jazyk je Java, ovšem je dostupné i API pro C#, které podporuje většinu toho, co API pro Javu. Java API podporuje Java ME a JCA/JCE (Java Cryptography Architecture). JCA je framework od Oracle, už obsahuje nějaké funkce, ale eliptické křivky bohužel přímo v JCA nejsou. Java ME je platforma Javy, konkrétně je to část Java SE, která nabízí API pro vyvíjení aplikací pro omezená zařízení. Tato knihovna je zde i pro Android, akorát se jmenuje Spongy Castle. Jsou podporovány ECDH, ECDSA, ECIES, ECMQV, ElGamal pro ECC, ale i další funkce, ikdyž ne nutně pro ECC. Dále jsou zde balíčky pro matematické operace, jedním z nich je balíček pro matematiku eliptických křivek. Jsou zde eliptické křivky nad binárními i prvočíselnými poli, rozhraní pro násobení bodů, třída reprezentující konečná pole, zvolení vlastních doménových parametrů atd. Jsou zde i už hotové implementace křivek, a to *curve25519*, *SM2-P256V1* a dále většina křivek nad prvočíselnými poli podle SEC specifikace. Bouncy Castle je rozsáhlá, má mnoho funkcionalit a taktéž je stále udržovaná. Původem je z Austrálie, takže je volně použitelná. K dispozici je pouze Javadoc (automaticky generovaná dokumentace) [26].

2.2.10 FlexiProvider

FlexiProvider je kryptografická knihovna pro jazyk Java. Může být přidána do aplikací postavených na JCA/JCE. *CoreProvider* obsahuje klasické algoritmy asymetrické kryptografie - třeba RSA, dále blokové šifry, AES, 3-DES, MD5, SHA1, HMAC a navíc vlastní generátor pseudonáhodných čísel. *ECProvider* je pak záležitostí eliptických křivek. Obsahuje algoritmy ECDSA, ECDH a ECIES. Dále jsou zde desítky připravených standardizovaných křivek, ať už nad prvočíselnými, tak nad binárnámi poli. Lze ale i vytvářet křivky s vlastními parametry [27].

2.2.11 ECCelerate

ECCelerate je knihovna v jazyce Java vyvinutá institutem IAIK (Institute for Applied Information Processing and Communications) na univerzitě Graz University of Technology v Německu. Je to knihovna přímo pro ECC a je potřeba Java 1.6 nebo vyšší (aktuální je Java 12 ke květnu 2019). Musí být přidána ale i kryptografická knihovna IAIK JCE, aby ECCelerate fungovala. Jsou zde podporovány klasicky ECDH, ECIES, ECDSA a ECDH. Navíc je implementována i aritmetika eliptických křivek i konečných polí, a to jak prvočíselných, tak binárních. Jsou zde metody pro práci s body [28][29].

2.3 Generování parametrů

Hledání vhodných nových eliptických křivek, resp. doménových parametrů křivky, provádějí různé autority, popřípadě výzkumníci. Nezávislost na třetích stranách ovšem může být ku prospěchu, a to konkrétně v případě eliptických křivek. Kdykoliv může být nalezen nový útok či slabina některých křivek, a proto má hledání nových křivek smysl. Kryptosystémy nad eliptickými křivkami mohou mít více podob. Článek od pana Lenstry [34] je rozlišuje na tzv. shared systems a non-shared systems. V prvním zmíněném typu účastníci komunikace sdílejí konečné pole F_p, eliptickou křivku E, generátor vzniklé grupy G a jeho řád n. V systémech non-shared pak si pak každý účastník volí parametry sám. V obou případech si účastník volí soukromý klíč - náhodné číslo $k \leq q$ a spočítá $G_A = kG$. V systémech typu shared je veřejným klíčem strany a přímo bod G_A, kde čtveřice (F_p, E, G, n) už byla definována třetí stranou. V non-shared systémech se veřejný klíč skládá ze všech předchozích elementů, čili pětice (F_p, E, G, n, G_A), kde všechny parametry jsou unikátní pro stranu A. Oba typy systémů mají své výhody a nevýhody. Výhodou shared systémů je, že veřejným klíčem strany a je pouze bod G_A, a ne celá pětice. Další výhodou je, že čtveřice (F_p, E, G, n, G_A) byla předpočítána a nalezena někým jiným, takže žádný účastník nebude muset vykonávat složité výpočty, které by stály čas a energii. K získání klíčů prostě jen vygenerují náhodné číslo z intervalu a pak ho vynásobí generátorem G. Naproti tomu v non-shared systémech může být získání klíčového páru komplikovanější, a to kvůli složitým výpočtům ke zjištění, zda existuje odpovídající generátor G s dostatečně velkým prvočíselným řádem.
Na straně druhé, non-shared systémy můžou mít navrch, pokud není efektivita a rychlost na prvním místě. U shared systémů je možné, že v případě nalezení tzv. index calculus útoku na eliptické křivky, by útok na veřejný klíč jedné strany ovlivnil bezpečnost také dalších účastníků. Nicméně nutno podotknout, že je pravděpodobné, že tento typ útoku nikdy nebude fungovat [35]. Tento problém by se non-

shared systémů netýkal, pokud by konečná tělesa byla různá, ale eliptické křivky obou stran by byly totožné. V článku [34] je navržen non-shared systém, který sice negeneruje nové křivky jako tato práce, ale každý každý účastník si generuje vlastní grupu. Pole, se kterými se zde pracuje, jsou pouze prvočíselná. Bohužel je zde pouze osm eliptických křivek, které ve výsledku mohou být použity. Generování parametrů probíhá tak, že se prvně zvolí prvočíslo p požadované velikosti tak, že platí $p \equiv 3 \pmod 4$. Tím se zajistí, že eliptická křivka nebude supersingulární. Potom se toto číslo testuje, zda splňuje nějakou z matematických podmínek, které jsou v článku [34] v tabulce 1 na straně 3 uvedeny. Pokud jsou dané podmínky splněny, další parametry jsou pak přímo uvedeny anebo jsou lehce vypočitatelné, např. rovnice křivky je přímo uvedená a řád grupy se vypočítá jednoduchým vztahem. Tento článek je tedy jednou z možností jak implementovat non-shared systém. Nespornou výhodou je oproti klasickým non-shared systémům to, že se nepočítají body na křivce, což je jeden z kroků při hledání nových křivek. Ty se dají generovat jak CM metodou (complex multiplication), tak metodou se SEA algoritmem, přičemž CM metoda nenabízí takovou škálu možností jako metoda pomocí SEA algoritmu, kde jsou křivky opravdu náhodné. Obě tyto metody budou zmíněny dále, protože jsou použity v této práci. Z non-shared systémů je převzata ta myšlenka, že není vždy nutné důvěřovat třetím stranám a tak bude použit vlastní generátor eliptických křivek.

2.4 ECIES

Asi nejrozšířenější schéma pro šifrování a dešifrování založené na eliptických křivkách je hybridní schéma ECIES - Elliptic Curve Integrated Encryption Scheme. Prapůvodcem je schéma DLAES (Discrete Logarithm Augmented Encryption Scheme) z roku 1997 představené pány Mihir Bellare a Philip Rogaway [31]. DLAES bylo následně v roce 1998 vylepšeno a přejmenováno na DHAES (Diffie-Hellman Augmented Encryption Scheme), ale kvůli záměnám s AES (Advanced Encryption Standard) bylo opět přejmenováno, a to na DHIES (Diffie-Hellman Integrated Encryption Scheme). DHIES představuje vylepšenou verzi ElGamalova schématu. Je nutno podotknout, že ECIES nemá jednotný standard a jednotlivé verze mezi sebou nemusí být kompatibilní.

Nyní následuje popis toho, jak ECIES funguje. Předpokládejme, že Alice chce poslat Bobovi zprávu m. Alice nechť má soukromý klíč u a veřejný klíč U. Bob bude podobně mít soukromý klíč v a veřejný klíč V. Vztah mezi soukromým u a veřejným klíčem U je $U = u \cdot G$, kde G je generátor. Soukromý klíč je prvek pole a veřejný klíč je bod na eliptické křivce.

Alice bude postupovat následovně:

1. Alice si vytvoří dočasný klíčový pár tak, že $U = u \cdot G$. Pár je generován pseudonáhodně.
2. Po vytvoření klíčového páru Alice použije funkci pro ustanovení klíčů k vytvoření sdíleného tajemství, které je vypočítáno vynásobením soukromého klíče Alice veřejným klíčem Boba, čili $u \cdot V$.
3. Alice použije funkci pro odvození klíče KDF (Key Derivation Function), vstupem je vypočítané sdílené tajemství a případně další parametry. Výstupem pak je symetrický klíč k_{ENC} pro šifrování a MAC klíč k_{MAC} pro podpis.
4. Alice zašifruje zprávu m symetrickou šifrou ENC za pomocí klíče k_{ENC}, zašifrovanou zprávu označme c.
5. Alice použije jednu z MAC funkcí (Message Authentication Code) a ze zašifrované zprávy c vytvoří podpis - tag.
6. Alice odešle kryptogram $(U||tag||c)$ Bobovi.

Bob musí pro dešifrování udělat následující:

1. Po obdržení kryptogramu $(U||tag||c)$ musí Bob rozdělit trojici na jednotlivé prvky a získat tím U, tag a c.
2. Bob vynásobí získaný veřejný klíč Alice U svým soukromým klíčem v a dostane sdílené tajemství $v \cdot U$. Tento výsledek je totožný s $u \cdot V$.
3. Díky tomuto sdílenému tajemství a případně dalším volitelným parametrům, stejných jako Alice, vytvoří Bob stejné klíče k_{ENC} a k_{MAC} pomocí KDF.
4. Bob vypočítá $tag*$ ze zašifrované zprávy c pomocí klíče k_{MAC}. Následně Bob porovná tag z přijatého kryptogramu s vypočítaným podpisem $tag*$. V případě, že porovnané podpisy nejsou shodné, kryptogram je Bobem zamítnut.
5. V případě, že podpisy z předchozího kroku byly shodné, Bob pokračuje dešifrováním zašifrované zprávy c symetrickým algoritmem ENC a symetrickým klíčem k_{ENC}. Výsledkem dešifrování je původní zpráva m.

2.4.1 Volitelné parametry

V předchozím textu byly zmíněny volitelné parametry. Tyto parametry musí být stejné na obou stranách, tj. jak u příjemce, tak u odesílatele.

Komprese bodu

Pokud spolu mají dvě strany komunikovat, musí být jasné, zda se použije komprese bodu. Jeden z doménových parametrů, generátor G, je bod na křivce a tento bod má souřadnice $[G_x, G_y]$. Jelikož je známa rovnice eliptické křivky, lze do ní dosadit souřadnici G_x a tím dostat ypsilonovou souřadnici G_y. Bod G se tedy reprezentuje jedním ze dvou způsobů, první reprezentací je celý bod $[G_x, G_y]$, druhou pak je pouze

jednou souřadnicí G_x. V případě, že je komprese použita, musí se dopočítat druhá souřadnice a tento úkon stojí prostředky. Pokud je tedy vyžadována maximální rychlost, případně je dané zařízení omezené atd., komprese bodu by se použít neměla. Nepoužití komprese indikuje v hlavičce bajt $0x04$ a za ním následuje celý binárně reprezentovaný bod G. V případě použití komprese je indikací bajt $0x02$ anebo $0x03$, za kterýmž následuje pouze první souřadnice bodu G.

Další parametry

Napříč standardy jsou různé odlišnosti, následuje několik z nich. Například mezi originální DHIES verzí a verzí ve standardu ANSI X9.63 je ten, že DHIES používá k vytvoření klíčů ze sdíleného tajemství jako KDF pouze hešovací funkci, zatímco ANSI X9.63 používá KDF. DHIES interpretuje výsledné bity z KDF tak, že zleva začíná MAC klíč a až poté šifrovací klíč. v ANSI X9.63 je to přesně naopak. DHIES povoluje specifickou symetrickou šifru, blokovou i proudovou. ANSI X9.63 naproti tomu povoluje pouze XOR funkci.
Právě díky těmto rozdílům a dalším, kterých je mezi standardy spousta, nelze prakticky implementovat ECIES tak, aby bylo plně kompatibilní se všemi ostatními standardy [32].

3 PRAKTICKÁ ČÁST

Pro realizaci praktické části byl použit jazyk C++, vývojové prostředí pak bylo Microsoft Visual Studio 2017. Generátor eliptických křivek byl vytvořen pro kryptografickou knihovnu MIRACL. MIRACL je open source SDK (source development kit) speciálně pro kryptografii nad eliptickými křivkami. Tato knihovna má implementovaný SEA algoritmus, který je potřeba pro počítání bodů na křivce. Podporuje ale pouze křivky nad prvočíselným polem. MIRACL je standardně knihovna pro jazyk C, ovšem je zde i rozhraní, díky kterému lze použít C++. Knihovna dostala začátkem roku 2019 také přehlednější manuál. S pomocí této knihovny pak bylo implementováno hybridní schéma ECIES, kde je použit vytvořený generátor křivek. Tato implementace schématu ECIES slouží jako ověření funkčnosti generátoru. V praktické části se generátor použil i při měření závislosti různých křivek a rychlosti násobení, šifrování atd. Generátor ve spojení s ECIES protokolem byl vytvořen pro Windows, kde problematický není samotný generátor, ale Windows Sockets 2 sloužící pro komunikaci. Generování jako takové funguje není pouze pro Windows a většina pozdějších měření byla uskutečněna na školním Blade serveru a OS Ubuntu 18.04.

3.1 Generování křivek a popis generátoru

Samotné generování křivek zahrnuje několik kroků. Způsobů, jak křivky generovat, je více. U křivek nad prvočíselným polem F_p je cílem generovat doménové parametry p, a, b, G, n, h, které již byly zmíněny. Dále jsou uvedeny dvě metody, pomocí kterých je dosaženo generování křivek. První z nich je metoda počítání bodů na křivce, také zvaná SEA metoda, protože používá SEA algoritmus. Druhý způsob je metoda komplexního násobení - complex multiplication. Obě metody jsou umístěny ve zvláštním souboru pojmenovaném *generator.cpp*. Je nutné používat MIRACL knihovnu. V souboru *generator.cpp* jsou pak funkce pojmenované *complexMultiplication* a *randomFpCurve*. První zmíněná má jako vstupní parametry požadovanou velikost křivky, kofaktor, diskriminant, od kterého se má začít vyhledávat, ukazatel na pole datového typu *Big*, což je datový typ MIRACLu pro velká čísla. Volitelně lze na vstupu zadat i cestu k souboru, kam se posléze parametry křivky zapíší. Generuje křivky nad prvočíselným polem a mohou mít velikost od 100 do 2048 bitů. Druhou funkcí je *randomFpCurve*, která generuje taktéž křivky nad prvočíselným polem. Vstupní parametr je zde opět velikost křivky a volitelně lze zadat cestu k výstupnímu souboru. Stejně tak lze volitelně zadat jeden nebo i oba parametry a a b z Weierstrassovy rovnice. Funkce vrací ukazatel na pole, kde jsou uložené parametry vygenerované křivky. Zde je spodní hranicí velikost křivky 64 bitů, horní hranice

je přibližně 600 bitů. Důvodem je to, že SEA algoritmus knihovny MIRACL používá kolekci modulárních polynomů. Každý takový polynom je asociován s nějakým malým prvočíslem. Tyto polynomy je potřeba vygenerovat spuštěním programu *mueller*. Toto je potřeba udělat pouze jednou a je generován soubor *mueller.raw*. Čím více polynomů je vygenerováno, tím větší je pak možno použít modulus a tím pádem mohou být generovány větší křivky. Na školním Blade serveru generování těchto modulárních polynomů běželo cca týden a větší křivku než 600 bitů se nepovedlo vygenerovat. Navíc soubor *mueller.raw* má přes 700 MB. S touto skutečností je tedy třeba počítat, pokud je požadováno generovat větší křivky.

3.1.1 Metoda počítání bodů na křivce

Generování probíhá následující způsobem. Prvně se náhodně pomocí knihovny MIRACL vygeneruje prvočíslo p. Parametrem je požadovaná velikost prvočísla p v bitech. Pro úroveň zabezpečení k bitů se volí prvočíslo o velikosti $2k$. Poté se náhodně volí eliptické křivky E nad polem F_p tak dlouho, dokud počet bodů $\#E(F_p)$ není prvočíslo nebo alespoň nemá dostatečně velký faktor po prvočíselném rozkladu. Chování počtu bodů $\#E(F_p)$ při volení různých eliptických křivek je náhodné a jeho výpočet má polynomiální složitost pomocí SEA algoritmu. Během hledání vhodné křivky lze požadovat, aby křivka byla z anglického pojmu tzv. twist-secure, neboli neplatí pouze to, že počet bodů, pro který platí $\#E(F_p) = p + 1 - t$, kde $|t| \leq 2\sqrt{p}$, musí být téměř prvočíslo, ale platí navíc i to, že $p + 1 + t$ je prvočíslo nebo téměř prvočíslo. Číslo $p + 1 + t$ je totiž počet bodů grupy tzv. kvadratického twistu $\widetilde{E}$, kde

$$\widetilde{E} : y^2 = x^3 + r^2ax + r^3b$$

a kde $r \neq 0$ a zároveň r je jakékoliv nečtvercové číslo v F_p. Generování twist-secure křivek ovšem zpomalí celý proces. Pouze pro příklad, viz měření v článku [33], generování křivky s úrovní zabezpečení 128 bitů touto metodou, aniž by křivka byla twist-secure, trvalo průměrně 2 minuty. Na straně druhé pak byly twist-secure křivky a ty trvalo vygenerovat průměrně 83 minut. Rozdíl je tedy značný, a to i přesto, že procesor byl Intel Core i7-3820QM 2,7GHz. Ani volba tzv. Montgomery-friendly prvočísel, která mají speciální výpočetní vlastnost $p \equiv \pm 1 \bmod 2^{32}$ nebo 2^{64}, neměla téměř žádný vliv. Tedy po zvolení p, a, b se počítá řád křivky, a to za pomocí SEA algoritmu (Schoof Elkies Atkin). Vypočítané číslo se faktorizuje, neboli rozloží na prvočísla, a určí jeho největší faktor r. Optimalizace v tomto kroku je ta, že není potřeba v nějakých případech řád faktorizovat úplně. Pokud po odstranění několika malých dělitelů stále zbytek není prvočíslo, kofaktor by stejně už byl příliš velký a začne se s novou křivkou.

Pouze pro příklad mějme číslo 128. Dělíme ho 2, dostaneme 64, což není prvočíslo. Dělíme znovu, máme 32, opět to není prvočíslo. Po dalším vydělení máme 16, není

prvočíslo. Teď už je známo, že číslo 128 bylo děleno 8 a zbytek stále nebyl prvočíslo, v tomto bodě se výpočet ukončí a začíná se s novou křivkou.
V dalším kroku se vypočítá kofaktor

$$h = \frac{\#E(F_p)}{r}.$$

Pokud je kofaktor moc velký, začne se opět s novou křivkou. Dále se zvolí náhodný bod P na křivce a spočítá se

$$G = hP.$$

Pokud G je bod v nekonečnu, začne se znovu s výběrem jiného bodu. Pokud není, G je tedy generátorem o řádu r. V tomto bodě jsou už vygenerovány všechny parametry, ale konkrétně určení řádu eliptické křivky (počítání bodů) je ta nejnáročnější část celého procesu.

Určení řádu křivky

Určení řádu křivky, neboli výpočet počtu bodů na křivce, je problém, který lze řešit vícero cestami. O problému byl diskutováno na webu www.stackexchange.com v sekci kryptografie, kde bylo navrhnuto použít SEA algoritmus (Schoof Elkies Atkin), což je tradiční způsob. Více možností bohužel zmíněno nebylo, leda připomínky, že při aplikaci celé řady kritérií pro bezpečné křivky budou výsledkem opět známé křivky, např. Curve25519. Avšak křivky se dají použít i tzv. on the fly, což znamená, že budou po použití zahozeny. Nemusela by tedy být aplikována tak přísná kritéria, protože křivky nebudou dopředu potenciálním útočníkům známy a nemohou tím pádem být vystaveny dlouhodobé kryptoanalýze. Co se týče SEA algoritmu, tak ten dostane na vstupu křivku E nad konečným polem F_q. Výstupem je řád této křivky. Tento, ale i další algoritmy, jsou už implementovány v některých matematických knihovnách. SEA algoritmus má původ ve Schoofově algoritmu z roku 1985, který byl posléze vylepšen pány Elkies a Atkins a došlo ke značnému zrychlení. Jinou, nejpřímočařejší, nejjednodušší a taktéž nejméně efektivní možností je tzv. naivní algoritmus, který spočítá v postupném procházení x hodnot a testuje se, zda vyhovuje Weierstrassově formě rovnice eliptické křivky. Tento algoritmus má složitost $O(q)$, protože se musí projít všechny prvky.
Dalším možným algoritmem je tzv. Baby-step giant-step algoritmus. Prvně je ale potřeba vědět, co je Hasseho teorém. Ten říká, že řád eliptické křivky nad tělesem F_q leží v intervalu $(q+1-2\sqrt{q}; q+1+2\sqrt{q})$. Tento algoritmus je založený na tom, že se tentokrát hodnoty x volí náhodně a opět se testuje, zda jsme našli druhou mocninu čísla v F_q. Pokud ano, vypočítají se kořeny, abychom dostali bod $P = (x, y)$. Pokud je nalezeno jediné číslo k v tomto intervalu takové, že platí $kP = O$, pak je toto číslo

k hledaným řádem. Pokud se stane, že je takových čísel více, volí se jiný náhodný bod.

3.1.2 Metoda complex multiplication

Dalším algoritmem je tzv. CM metoda (complex multiplication), která funguje jinak než metoda s počítáním bodů na křivce. Zde je vstupem požadovaná velikost křivky a výstupem už samotná křivka. Některé implementace také umožňují požadovat přímo už nějaký řád křivky, to ale není tento případ. Tato metoda je sofistikovanější, ale je také rychlejší a taktéž často používaná. V závislosti na použití generátoru může být tato metoda vhodnější kvůli své rychlosti. Rychlejší generování křivek má ale za cenu ztrátu pestrosti či rozmanitosti křivek oproti metodě se SEA algoritmem, ten totiž hledá křivky náhodné (náhodnost je v rámci pseudonáhodného generátoru).

3.2 Použití generátoru

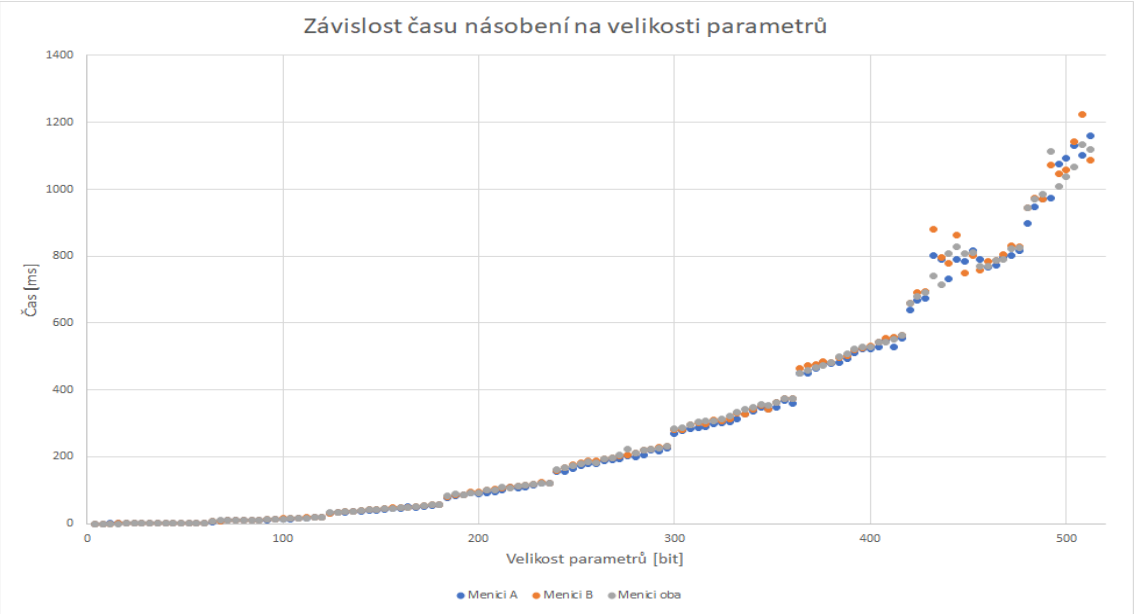
Generátor křivek je použit pro splnění více cílů. Prvním je měření efektivity křivek, hledání křivek, kde jsou elementární operace rychlejší než u křivek jiných. Druhým cílem je měření šifrování a dešifrování ECIES v závislosti na různých parametrech či faktorech. Posledním cílem je použití generátoru v kryptografickém protokolu, implementováno bylo konkrétně již zmíněné schéma ECIES - Elliptic Curve Integrated Encryption Scheme.

3.2.1 Měření efektivnosti křivek

V rámci této práce byla provedena měření rychlosti násobení bodu skalárem v závislosti na velikosti použitých parametrů v bitech. Měření byla provedena na počítači s procesorem Intel Core i5-5200U 2.20 GHz. Knihovna byla tedy použita MIRACL a ta jako algoritmus pro skalární násobení používá metodu *wNAF*. Měření byla provedena od nejmenších velikostí parametrů až po velikost 512 bitů. Prvně se měřila rychlost u křivek nad prvočíselným polem daných rovnicí

$$y^2 \equiv x^3 + ax + b \bmod p$$

s měnící se velikostí parametru a. V druhém měření se měnila velikost parametru b. V posledním, třetím měření, se měnily parametry oba, čili a i b. Tím, že se parametry měnily, je myšleno to, že k m bitové křivce je generován alespoň jeden parametr o velikosti m bitů. Vždy k těmto parametrům bylo generováno prvočíslo odpovídající velikosti p. K měření času (OS Windows 10) byla použita funkce *QueryPerformanceCounter* z Windows API. Postup měření je následující. Prvně se inicializuje křivka



Obr. 3.1: Měření závislosti rychlosti násobení na velikosti křivky

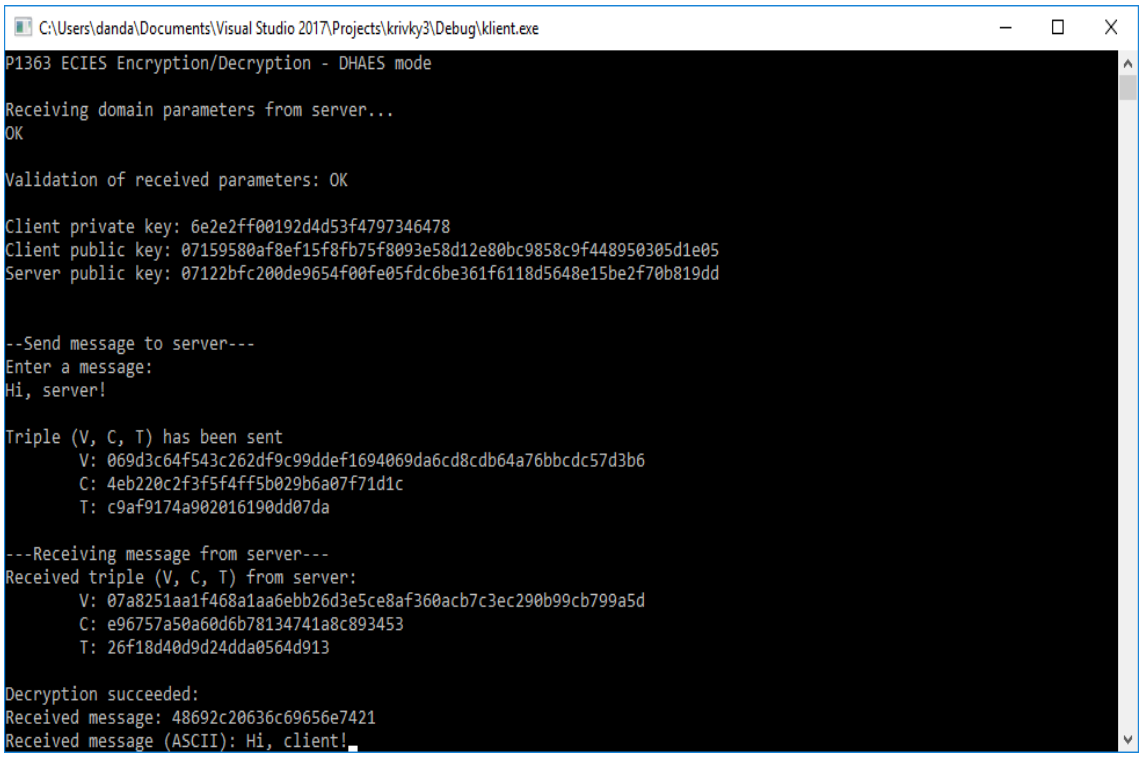
s požadovanou velikostí parametrů. Potom se nalezne nějaký bod na křivce, který bude násoben skalárem. V tomto bodě započne měření času a proběhne *for* cyklus s 1000 opakováními, kdy se nalezený bod násobí skalárem. Nyní se ukončí měření času a vypočítá se průměrný čas jedné operace násobení. Tato měření jsou provedena pro křivky až do 512b se skokem 4 bity, tj. $4, 8, \cdots 512$. Trvání výpočtů byla relativně zdlouhavá a náročná, proto nebylo použito více jak 1000 opakování. Součtem všech průměrných hodnot všech tří měření a převedením milisekund na hodiny vyšlo číslo cca 31 hodin, což přibližně odpovídá skutečné době trvání měření. Kvůli určitým chybám a opakovaným měřením se toto ještě prodloužilo. Proto bylo 1000 opakování ponecháno jako kompromis mezi přesností a potřebným časem pro výpočet. Na obrázku 3.1 výše lze vidět graf s provedenými měřeními. Ať už se měnil jeden, druhý, nebo oba parametry, závislost rychlosti byla vždy velice podobná až na několik případů. V případě, kdy se měnil parametr b, lze naleznout křivky, kde násobení trvalo déle než ostatní křivky.

3.2.2 ECIES protokol

Generátor eliptických křivek byl použit ve spojení se schématem ECIES. Výsledkem není ECIES v pravém slova smyslu, protože neexistuje odpovídající standard. Standardizovány jsou pouze části a to dle IEEE P1363. Rozdíl tedy je ten, že namísto jedné vybrané eliptické křivky se používají křivky náhodně generované a jsou použity právě jednou, po použití jsou zahozeny. Tato vlastnost minimalizuje dobu, po kterou mohou být parametry křivky vystaveny kryptoanalýze. V práci pánů Miele a Lenstra

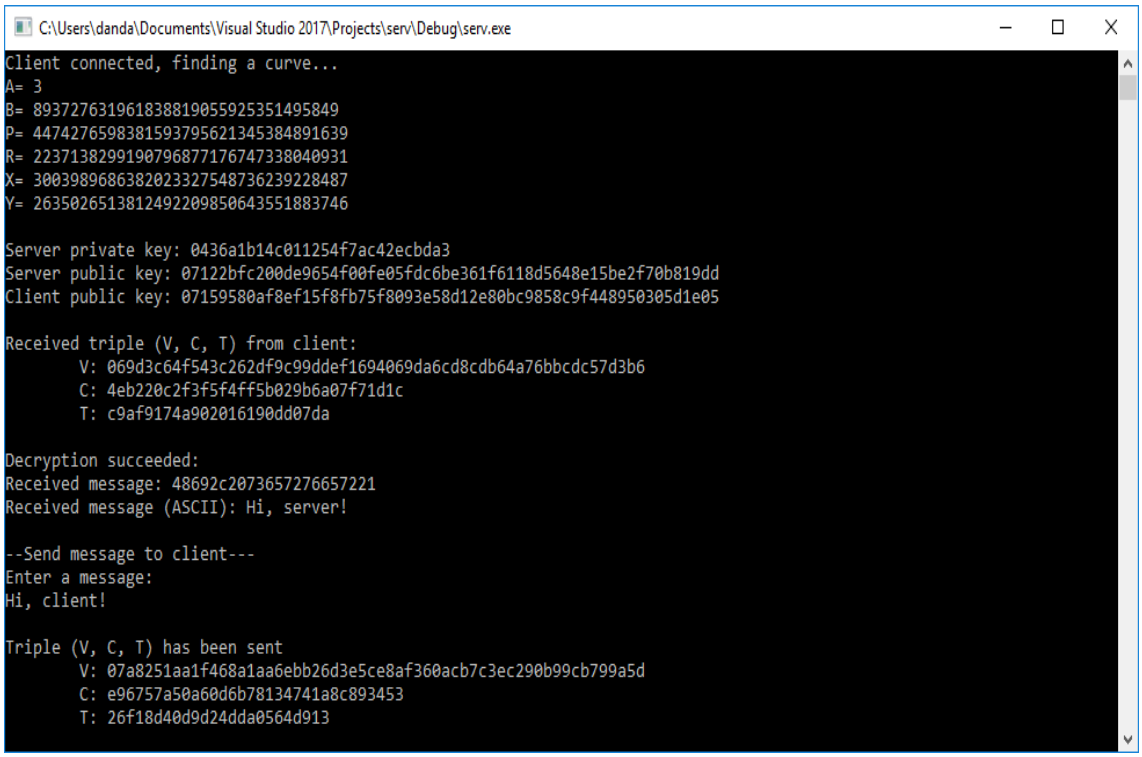
[33], která se zabývá tímto jednorázovým použitím, se nazývají doménové parametry jako short-lived či ephemeral, v češtině efemérní neboli výstižněji - pomíjející, prchavý. Výhodou těchto efemérních parametrů tedy je, že před použitím nejsou dlouhodobě známy. Podobná publikace, u které je stejně jako u této práce cílem nezapojovat do procesu a nedůvěřovat zbytečně dalším stranám, je od pana Lenstry [34] a byla zmíněna dříve. Zde si jednotlivé strany volily pouze prvočíslo p splňující daná kritéria a tím pádem i grupu. V práci [33] se autoři rozhodovali mezi náchylností na útoky zatím neznámé (což jsou prakticky všechny kryptosystémy) a náchylností na útoky vzniklé v důsledku sdílení parametrů. Z těchto dvou možností zvolili tu první. Schéma ECIES bylo implementováno v rámci modelu klient-server. Veškeré výpočty a generování nových křivek obstarává server. Screenshoty níže jsou ukázkou komunikace klienta a serveru. První obrázek je klient. Komunikace začíná tak, že se klient připojí k serveru a ten mu obratem zašle čerstvě vygenerované doménové parametry. Klient si prvně inicializuje náhodný generátor, v tomto případě byl použit pseudonáhodný generátor knihovny MIRACL. Funkce pro inicializaci generátoru přebírá hodnotu neurčené délky, kterou společně s časem dne (32 bitů) zkombinuje a použije jako semeno generátoru. Obecně lze použít i hardwarový generátor náhodných čísel, který je založen na nějaký fyzikálních vlastnostech, příkladem budiž okolní teplota, atmosférický šum, atd. Klient následně zvaliduje přijaté doménové parametry. Toto by se mělo dělat obecně, pokud jsou doménové parametry získány od někoho jiného. Validací je myšleno ověření toho, že parametry křivky náleží prvkům konečného pole, že křivka není anomální, že generátor grupy je validní (že tento bod opravdu leží na křivce) a že jeho řád prvočíselný. Otestuje se také, že generátor není bod v nekonečnu. Pokud není, přesto se ještě vynásobí svým řádem, čili spočítá se rG a výsledný bod by měl být bod v nekonečnu. Pokud tomu tak není, parametry nebudou přijaty jako validní. Dále se taktéž ověří, že pole je prvočíselné a také to, že platí $4a^3 + 27b^2 \neq 0 \bmod q$. Nakonec se ještě ověří tzv. MOV podmínky kvůli představenému MOV útoku, a to pány Menezes, Okamoto a Vanstone. Ti prezentovali redukci diskrétního logaritmu nad eliptickými křivkami na klasický diskrétní logaritmus, který lze řešit v subexponenciálním čase. Kontrola MOV podmínek je ochranou proti tomuto útoku. Podmínka je splněna, pokud neexistuje žádné $k \in \{1, \ldots, B\}$ tak, že $\#E(F_p)|q^k - 1$. Hodnota B je kladné celé číslo zvané MOV threshold, které bývá alespoň $B \geq 20$.

Dalším krokem na straně klienta je vygenerování klíčového páru, čili veřejného a soukromého klíče. Klient přijme veřejný klíč serveru VK_s. Pak je pro demonstraci zaslána serveru zpráva „Hi, server!“, která je načtena z konzole a vypsána. Nyní se volá funkce pro šifrování, kde vstupem jsou náhodný generátor, veřejný klíč protistrany (zde VK_s), zpráva k zašifrování a délka MAC kódu v bajtech. Tato funkce si vnitřně vygeneruje jednorázový veřejný klíč V (někde také zváno jako session key). Dále



Obr. 3.2: Ukázka klienta

se použije primitivum ECPSVDP-DH v kombinaci s KDF2. Tato kombinace je dle [36] typická, obecně procesu ustanovení tajného klíče je docíleno kombinací SVDP primitiva (secret value derivation primitive) a funkce pro odvození klíče (key derivation function - KDF). Funkce pro odvození klíče mapuje sdílené tajemství (výsledek SVDP primitiva) na jeden nebo více sdílených klíčů. Následně se výsledek funkce KDF2 rozdělí na dva stejně velké klíče - první je AES šifrovací klíč a druhý je MAC klíč. KDF2 funkce používá jako hešovací funkci SHA256. Dále se AES šifrou zašifruje vstupní zpráva m. Následuje výpočet autentizačního kódu zprávy MAC (tag). Výstupem celého procesu je jednorázový veřejný klíč V, šifrový text C a MAC kód T. Tato trojice V, C, T se odešle. Dešifrování probíhá analogickým způsobem. Po přijetí zprávy - trojice V, C, T - se dešifrovací funkci předají jako vstupní parametry tato trojice a příjemcův (v tomto případě klientův) soukromý klíč. Za pomoci primitiva ECPSVDP-DH v kombinaci s KDF2 se určí společný klíč a z něj dále klíč pro dešifrování AES šifrou a MAC klíč. Šifrový text C se dešifruje. Příjemce následně vypočítá tag T' a porovná ho s přijatým tagem T. Pokud jsou rozdílné, přijatý kryptogram je zamítnut. V případě screenshotu klienta, viz 3.2, je ukázaná přijatá trojice V, C, T od serveru. Po dešifrování a ověření autenticity lze vidět původní zprávu odeslanou serverem: „Hi, client!“. Co se týče serveru, ten má postup přijímání a odesílání zpráv stejný, stejně tak šifrování a dešifrování zpráv je stejné. Server v případě připojení klienta vygeneruje novou křivku nad prvočíselným polem,



Obr. 3.3: Ukázka serveru

a doménové parametry následně odešle klientovi. Pro generování křivky lze použít metodu se SEA algoritmem nebo CM metodu. Stejně jako u screenshotu klienta, viz 3.3, i zde je ukázána přijatá a odeslaná zpráva, včetně jejich kryptogramů.

3.2.3 ECIES - měření rychlosti šifrování a dešifrování zprávy

Bylo provedeno několik variant měření, kde byla porovnávána rychlost šifrování a dešifrování zprávy v závislosti na určitých parametrech. Kromě posledního měření, kde byl měřen vliv velikosti zprávy na rychlost šifrování a dešifrování, byla vždy velikost posílané zprávy deset bajtů. Tato měření byla provedena na školním HP Blade serveru s procesorem Intel Xeon E7-4820 (2,0 GHz/8-core/18 MB cache). Prvně bylo testováno, zda budou nalezeny křivky, kde bude ECIES šifrování či dešifrování rychlejší než u křivek ostatních. Pak následuje měření, kde se zjišťovala závislost rychlosti ECIES šifrování a dešifrování na velikosti křivky. Potom bylo provedeno porovnání ECIES a implementovaného hybridního RSA-AES schématu. Předposlední částí je měření rychlosti šifrování a dešifrování v závislosti na velikosti zprávy. Poslední pak je porovnání křivek generovaných SEA a CM metodami.

Náhodné křivky a rychlost šifrování/dešifrování

V prvním měření bylo vygenerováno sto náhodných křivek vždy o určité velikosti, a to 64 až 512 bitů (celkem 2000 křivek). Bylo zjišťováno, zda se mezi křivkami najdou takové křivky, u kterých bude šifrování či dešifrování znatelně rychlejší než u ostatních. Ke generování křivek byla použita SEA metoda, aby byla zajištěna

Tab. 3.1: ECIES šifrování

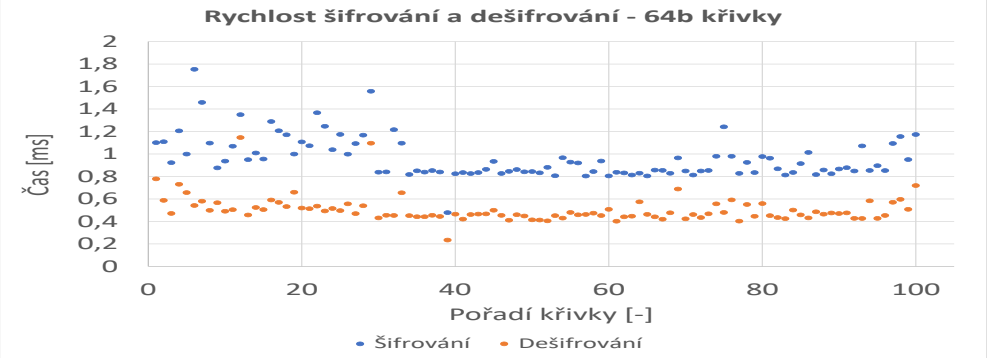
	Šifrování					
	min	max	avg	med	dif-min	dif-max
64b	0,478422	1,75331	0,962895	0,888441	0,503142	0,820874
80b	0,954492	2,26832	1,17611	1,14508	0,188433	0,928663
96b	1,07274	2,23329	1,370477	1,32778	0,217251	0,629571
112b	1,24218	2,25656	1,50577	1,458715	0,175053	0,498609
128b	1,2853	2,92356	1,484891	1,44212	0,134415	0,968872
144b	2,41714	2,79627	2,58116	2,575035	0,063545	0,083339
160b	2,54495	4,99651	2,819809	2,728155	0,097474	0,771932
192b	3,01852	3,56887	3,296158	3,29465	0,084231	0,082736
224b	4,62035	12,1798	6,80849	5,87322	0,321384	0,788914
240b	5,35284	8,74538	6,379963	6,225505	0,160992	0,370757
256b	5,63089	12,0776	6,930752	6,451045	0,18755	0,74261
288b	9,30124	16,356	10,60299	10,22155	0,122772	0,542583
304b	9,40463	14,1847	10,40116	10,29265	0,09581	0,363761
320b	9,6588	13,8281	10,84798	10,69045	0,109622	0,274717
352b	14,5049	20,286	15,98735	15,6413	0,092726	0,268878
384b	15,4284	20,4646	17,03375	16,797	0,094245	0,201415
416b	22,5812	30,6781	24,82415	24,35375	0,090353	0,235817
448b	23,4401	29,0201	25,66951	25,52985	0,08685	0,130528
480b	33,4488	47,635	37,26442	36,1077	0,102393	0,278297
512b	34,3112	44,7224	37,41743	36,95755	0,083016	0,195229

Tab. 3.2: ECIES dešifrování

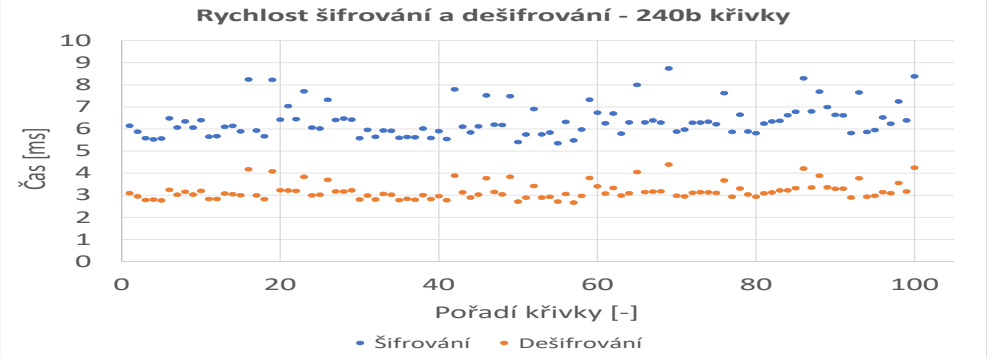
	Dešifrování					
	min	max	avg	med	dif-min	dif-max
64b	0,234687	1,14638	0,50606	0,47026	0,536247	1,265304
80b	0,475859	1,04807	0,568853	0,562992	0,163476	0,842428
96b	0,512436	1,34423	0,658177	0,63557	0,221431	1,042355
112b	0,580489	1,24777	0,741954	0,717623	0,217621	0,681736
128b	0,620516	1,17935	0,720328	0,701376	0,138564	0,637241
144b	1,14847	1,34979	1,2701	1,269905	0,095764	0,062743
160b	1,23773	1,74644	1,378055	1,346065	0,101829	0,267322
192b	1,46519	1,74547	1,614587	1,61567	0,092529	0,081063
224b	2,27435	6,12782	3,396401	2,938725	0,330365	0,80421
240b	2,66419	4,39025	3,181411	3,08973	0,162576	0,37997
256b	2,76902	6,90178	3,500055	3,223035	0,208864	0,971906
288b	4,54966	7,00161	5,24246	5,106845	0,132152	0,335558
304b	4,58393	7,11338	5,183864	5,11896	0,115731	0,372216
320b	4,78405	6,9694	5,422996	5,34068	0,117822	0,285157
352b	7,15107	10,1323	7,979122	7,78454	0,103777	0,269851
384b	7,63165	10,6238	8,516657	8,36655	0,103915	0,247414
416b	11,1005	15,241	12,37774	12,12615	0,103189	0,231323
448b	11,5576	14,1048	12,78442	12,7301	0,095962	0,103281
480b	16,4803	23,4595	18,55179	17,96785	0,11166	0,264541
512b	16,9794	22,1349	18,6577	18,48405	0,089952	0,186368

jejich opravdová náhodnost. Výsledky měření jsou uvedeny v tabulkách 3.1 a 3.2, kde každé velikosti křivky náleží rychlost šifrování a dešifrování, a to průměrná, minimální, maximální, mediánová a také rozdíl minima či maxima v procentech oproti průměru. Z tabulek 3.1 a 3.2 lze vidět, že mezi minimálními a maximálními hodnotami jsou značné rozdíly. Rozdíly oproti vypočteným průměrům jsou v desítkách procent, například u velikosti křivek 128 bitů činil nárust u některé náhodně zvolené křivky téměř 97 %. Naopak u křivek o velikosti 64 bitů byla nalezena taková

křivka, kde šifrování trvalo téměř poloviční dobu. Čím větší velikost křivek byla, tím se takto radikálně rychlosti šifrování lišily méně. Z tohoto měření tedy plyne závěr ten, že některé křivky jsou efektivnější než jiné, a takové by pak mohly být použity tam, kde je stěžejní rychlost. Co se týče dešifrování, stejně jako u šifrování byly mezi minimy a maximy markantní rozdíly. Pro příklad jsou přiloženy dva grafy ilustrující, jak hodně se můžou rychlosti mezi jednotlivými křivkami lišit. Je vidět, že několik málo křivek se značně liší oproti průměru. Na obrázcích 3.4 a 3.5 lze vidět vždy všech sto křivek k dané velikosti křivky (64 a 240 bitů). Například u grafu pro 64 bitové křivky je vidět jedna křivka, která měla znatelně rychlejší šifrování i dešifrování než ostatní. Stejně tak je vidět několik křivek, kterým tyto operace trvaly o poznání déle. Na grafu druhém, tedy s křivkami o velikosti 240 bitů, lze vidět i několik křivek, které byly naopak horší, a to jak v šifrování, tak dešifrování.



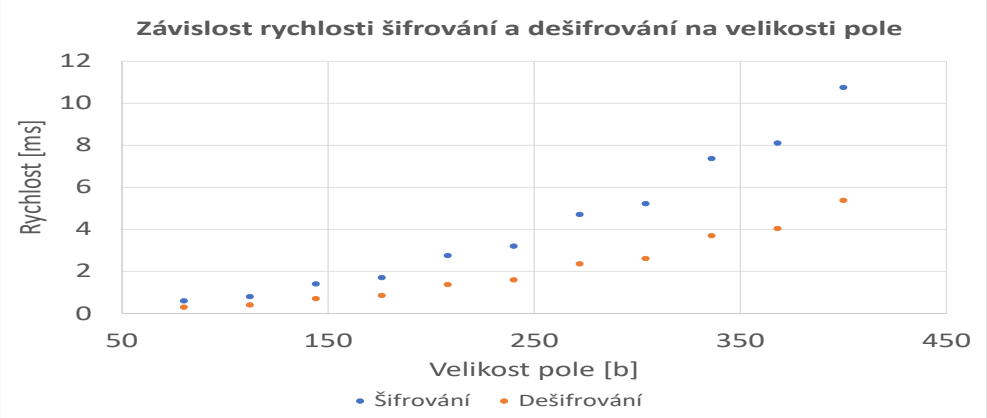
Obr. 3.4: Rychlost šifrování a dešifrování - 64b



Obr. 3.5: Rychlost šifrování a dešifrování - 240b

Velikost křivky a rychlost šifrování/dešifrování

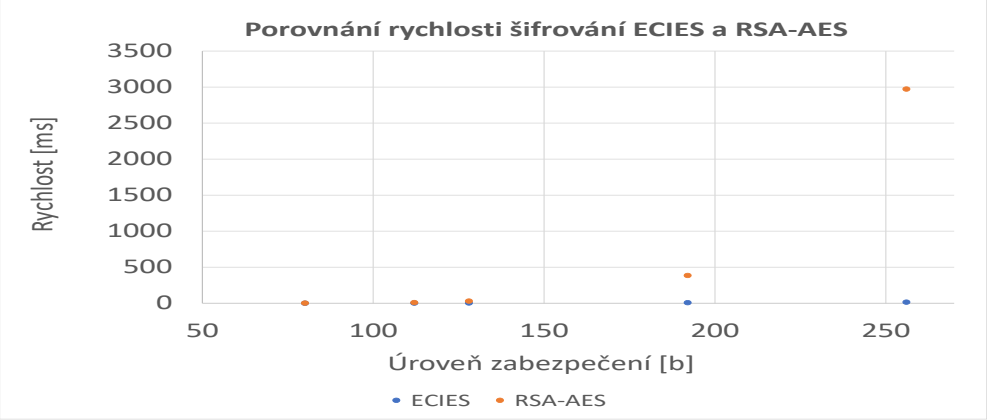
V tomto měření byla vždy na začátku vygenerována nějaká křivka, tj. určily se parametry a a b Weierstrassovy rovnice. Pak byl v cyklu volen modulus p o velikostech od 80b až do 496b s krokem 32b. Ke generování křivek byla opět použita SEA metoda, která je zde nutná, protože CM metodě nelze předat jako vstup parametry a a b Weierstrassovy rovnice. Měření tedy probíhalo tak, že se zvolily náhodně parametry a a b a uložily. Potom se v cyklu generovaly křivky o daných velikostech, kde se vždy jako parametr generující funkci předaly i parametry a a b. Tento proces se opakoval stokrát, čili výsledkem je vždy sto křivek o dané velikosti. Opět je přiložen graf jakožto výsledek měření, viz obrázek 3.6. Parametry a a b zůstávaly v rámci jedné křivky stejné, aby se vyloučila možnost vlivu vhodně zvolených parametrů a byl sledován opravdu pouze vliv velikosti křivky. Dle grafu lze usuzovat, že by rychlost šifrování a dešifrování mohla být vyjádřena exponenciální funkcí, případně i kvadratickou. Co se týče poměru šifrování a dešifrování, tak dešifrování vychází vždy téměř poloviční. Je to proto, že odesílatel musí generovat jednorázový klíčový pár, kdežto příjemce při dešifrování použije svůj soukromý klíč. Odesílatel při šifrování použije dvakrát skalární násobení, ale příjemce pouze jednou. Proto, obecně řečeno, by dešifrování mělo trvat zhruba poloviční dobu. Záleží samozřejmě na konkrétní implementaci, ale v tomto případě relativně přesně platí teoretický poměr obou hodnot.



Obr. 3.6: Závislost rychlosti šifrování a dešifrování na velikosti pole

Porovnání s RSA

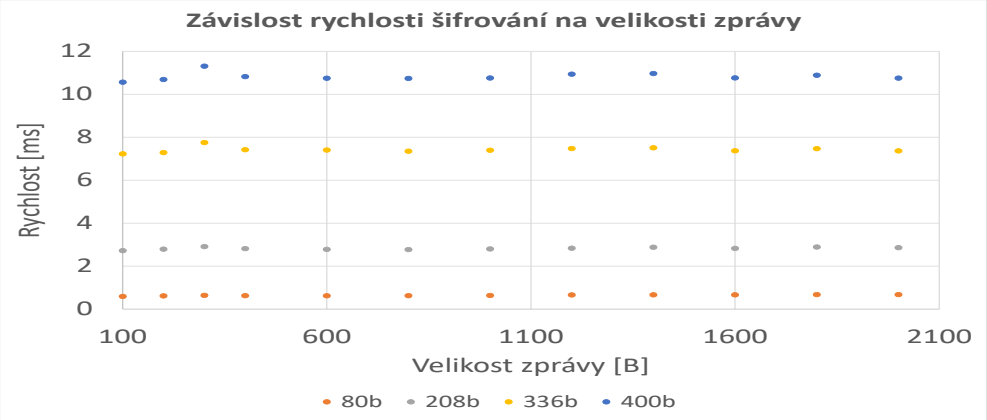
V dalším měření byly porovnávány rychlosti šifrování a dešifrování s RSA. Bylo implementováno odpovídající RSA-AES schéma a to po vzoru z [38]. Čili prvně se vygeneroval AES klíč o velikosti 16 bajtů. Potom se zpráva zašifrovala pomocí AES-128 v módu CBC. Následně se pomocí příjemcova veřejného RSA klíče zašifroval AES klíč. Příjemcovým RSA klíčem se podepsala zpráva a tento podpis se uložil. Výsledný kryptogram pak tedy je zašifrovaný AES klíč, zašifrovaná zpráva a podpis. Dešifrování RSA-AES pak probíhalo tak, že se získal AES klíč pomocí příjemcova soukromého RSA klíče. Získaným AES klíčem pak tedy lze rozšifrovat už samotnou zprávu. Odesílatelovým veřejným RSA klíčem se zkontroluje validnost podpisu.
V tomto měření byly testovány odpovídající úrovně zabezpečení. Potřebné velikosti modulusu pro RSA pak byly převzaty z [37], strana 64. Jak lze vidět na výsledném grafu ukazujícím šifrování, viz obrázek 3.7, RSA-AES šifrování bylo znatelně pomalejší, a to přibližně od úrovně zabezpečení 192 bitů. Co se týče dešifrování, tak graf ukázán není, protože šifrování a dešifrování u RSA-AES trvalo prakticky stejně dlouho. U ECIES pak dešifrování trvalo dobu poloviční, stejně jako u předchozích měření. Největší rozdíl mezi jednotlivými schématy byl u úrovně zabezpečení 256 bitů, kde šifrování RSA trvalo průměrně 2973 milisekund, zatímco u ECIES pouhých 16 milisekund. Opět zde bylo použito tisíc opakování a pak byl vypočítán průměr. Co se týče rychlosti, ECIES je lepší alternativa k RSA-AES, a to hlavně díky neustále rostoucím požadavkům na úroveň zabezpečení.



Obr. 3.7: Porovnání rychlosti šifrování ECIES a RSA v závislosti na velikosti úrovně zabezpečení

Závislost šifrování na velikosti zprávy

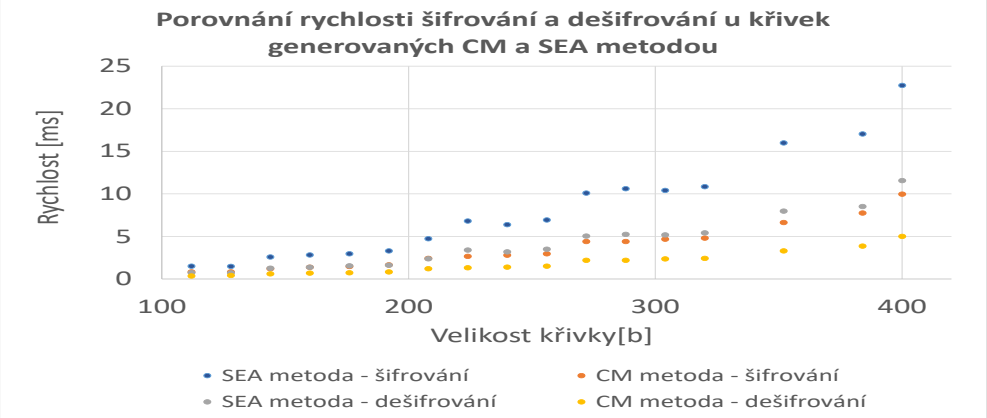
Zde byla měřena závislost šifrování a dešifrování na velikosti zprávy. Rozmezí velikostí zpráv bylo od 100 do 2000 bajtů. V grafu 3.8 je ukázáno pouze šifrování čtyřech velikostí křivek. Je vidět, že hodnoty se víceméně drží ve vodorovné přímce, hodnoty se téměř nemění. Dešifrování z tabulky graficky znázorněno nebylo, protože graf vypadá víceméně stejně, akorát s menšími hodnotami. Co už mělo efekt, bylo zvětšení zprávy na 20 000 bajtů, kde trvání trvalo o cca 10 % déle.



Obr. 3.8: Rychlost šifrování a dešifrování v závislosti na velikosti zprávy

Šifrování a dešifrování u křivek generovaných SEA a CM metodou

V tomto měření byly porovnávány křivky náhodné a křivky generované CM metodou. Velikost křivek byla od 112 bitů do 400 bitů. Výsledek je ukázán v grafu 3.9. Je vidět, že křivky generované CM metodou jsou znatelně rychlejší, šifrování i dešifrování trvá zhruba poloviční dobu. Není překvapivé, že jsou rychlejší. Překvapivé ale je, že jsou rychlejší o tolik. Křivky generované CM metodou jsou strukturovanější, jsou generovány dle určitého algoritmu. Například CM metoda může hledat (a tak je to i v implementaci knihovny MIRACL) prvně parametr a Weierstrassovy rovnice tak, že $a = -3$ nebo $a = 1$. Ve výsledku tedy závisí na požadavcích a použití. Pokud je požadována rychlost, pak CM křivky jsou vhodnější. Pokud by někdo nedůvěřoval CM metodě kvůli tomu, že generuje strukturovanější křivky a teoreticky může být někdy napadnuta, SEA metoda se jeví jako vhodnější, ovšem i na generování křivek je časově náročnější.



Obr. 3.9: Porovnání rychlosti šifrování a dešifrování u křivek generovaných SEA a CM metodou

3.3 Použité algoritmy

Algoritmus 1 Generovani krivek nad $\mathbb{F}_p$

INPUT: modulus p
OUTPUT: usporadana sestice T, kde $T = (p, a, b, G, r, h)$

1: a = nahodne cislo
2: b = nahodne cislo
3: **if** diskriminant je 0 **then** marker
4: $radKrivky$ = SEA(a,b,p)
5: **if** $radKrivky$ je p **then** marker
6: **if** kontrolaMOV není OK **then** marker
7: r = nejvetsi cislo prvociselneho rozkladu
8: kofaktor $h = \#E(\mathbb{F}_p)/r$
9: **if** h je prilis velke **then** marker
10: nalezeni generatoru G
11: **return** T

Algoritmus 2 Nalezeni generatoru grupy

INPUT: usporadana petice $S = (p, a, b, r, h)$
OUTPUT: pokud rad krivky $\#E = hr$, pak generator radu r, jinak false

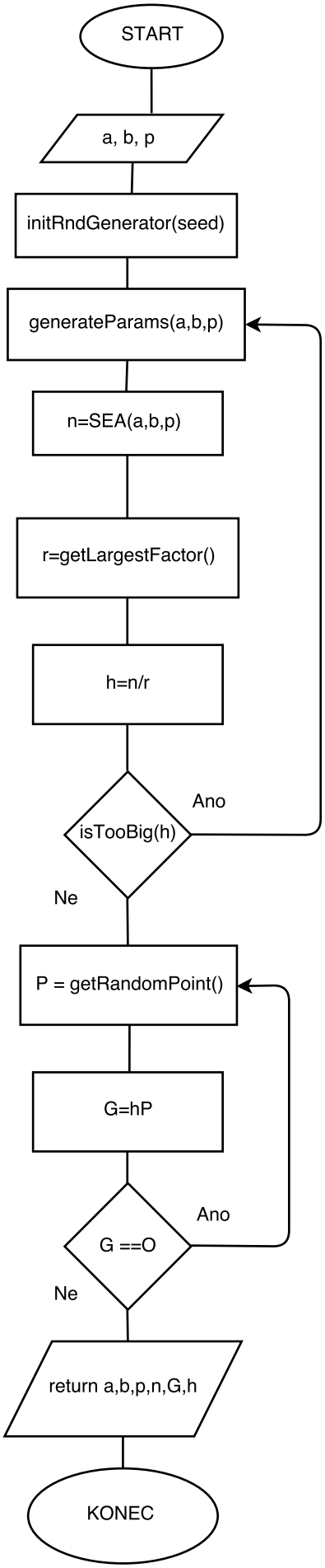
1: generovani nahodneho bodu P
2: bod $G = hP$
3: **if** G je bod v nekonecnu **then** marker
4: bod $Q = rP$
5: **if** Q neni bod v nekonecnu **then**
6: **return** false
7: **else**
8: **return** G

Algoritmus 3 Vygenerovani nahodneho bodu

INPUT: krivka E
OUTPUT: bod P

1: k = getOrderOfTheGroup(E)
2: length = bitLength(k)
3: k = getRndBigNum(length) //nahodne cislo o length bitech
4: P = k*getGenerator () //nahodny nasobek generatoru
5: return P

Výše jsou ukázány algoritmy, co byly použity v generátoru. Prvním je Algoritmus 1, což je kostra celého generování. Následuje Algoritmus 2, kde je ukázáno, jak se hledá generátor grupy. Poslední je Algoritmus 3 ukazující generování náhodného bodu na křivce. Nakonec následuje diagram znázorňující průběh generování nové křivky.


START
a, b, p
initRndGenerator(seed)
generateParams(a,b,p)
n=SEA(a,b,p)
r=getLargestFactor()
h=n/r
isTooBig(h)
Ano
Ne
P = getRandomPoint()
G=hP
G ==O
Ano
Ne
return a,b,p,n,G,h
KONEC

4 VÝSLEDKY

V praktické části bylo uvedeno, jakým způsobem se můžou nové křivky generovat. Byly popsány základní kroky a to slovně, pseudokódem a také vývojovým diagramem. Jsou přiloženy pseudokódy celého procesu, ale také SEA algoritmu a generování náhodného bodu na křivce. Výsledkem této práce je generátor eliptických křivek s pomocí kryptografické knihovny MIRACL. Generovat křivky lze za pomocí metody se SEA algoritmem anebo s pomocí CM metody. Obě metody mají své výhody a nevýhody. Metoda se SEA algoritmem dlouho počítá řád grupy, díky čemuž není vhodná pro situace, kdy je požadováno generovat křivky co nejrychleji. Na druhou stranu, tato metoda generuje už z podstaty jejího fungování náhodné křivky bez nějakého vzoru a křivky jsou tedy rozmanitější. Náhodnost je daná použitým náhodným generátorem, ovšem stačí jakýkoliv pseudonáhodný generátor. Dosažení co největší náhodnosti není v tomto případě esenciální. Druhou metodou je CM metoda, která hledá křivky na základě velikosti požadované křivky. Tato metoda generuje více strukturované křivky, zato ale velice rychle. V situacích, kdy je potřeba křivky generovat rychle, se tato metoda jeví jako jednoznačně vhodnější. Zároveň ale může být nalezen útok (stejně jako prakticky na cokoli jiného) na tyto strukturovanější křivky, což může být teoreticky slabina. CM metoda je ale normálně používaná a bylo představeno už více jejích verzí, popřípadě modifikací. Bylo provedeno měření závislosti rychlosti vykonávaných základních operací na velikosti eliptické křivky. Dále se měřilo, jaký efekt na výkon mají měnící se jednotlivé parametry eliptické křivky. Bylo zjištěno, že pokud se mění jednotlivé parametry, pak náročnost základních operací byla stejná. Co se týče dopadu velikosti eliptické křivky, ten je stěžejní. Čím větší je eliptická křivka, tím náročnější jsou základní operace. Toto chování bylo očekáváno. Další měření se týkala implementovaného protokolu ECIES. Prvně se měřilo šifrování a dešifrování u náhodných křivky o daných velikostech, tedy generovaných SEA metodou. Zjišťovalo se, zda se najdou takové křivky, které budou výrazně rychlejší či pomalejší než ostatní. Takové křivky nalezeny byly a byly popsány rozdíly v jejich rychlostech. Rychlejší či pomalejší křivky tedy existují. Další měření zkoumalo vliv velikosti křivky při zachování stejných parametrů a a b Weierstrassovy rovnice. Parametry zůstávaly stejné, aby se vyřadil jejich případný efekt v důsledku vhodného zvolení. Vliv se zdá být kvadratický či exponenciální. Dále se porovnávalo šifrování a dešifrování ECIES s šifrováním a dešifrováním RSA-AES. ECIES od přibližně úrovně zabezpečení o 192 bitech začalo být rychlejší než RSA-AES. RSA-AES je implementované hybridní schéma odpovídající ECIES. ECIES se tedy vyplatí zejména při větších úrovních zabezpečení. Předposlední měření zkoumalo závislost šifrování a dešifrování na velikosti zprávy. Protože dešifrování má prakticky stejný graf, akorát menší hodnoty, nebylo v grafu znázorněno. Měřeny

byly zprávy od sta do dvou tisíc bajtů. Prakticky žádný nárust v době šifrování se nekonal, a to u žádné z proměřených velikostí křivek. Až u velikosti zprávy dvacet tisíc bajtů byl už viditelnější nárust v době šifrování a dešifrování. Nárust doby trvání šifrování v tomto případě oproti době trvání šifrování u zpráv o velikosti dva tisíce bajtů byl asi deset procent. Poslední měření porovnávalo šifrování a dešifrování u křivek generovaných CM a SEA metodou. Křivky generované CM metodou z měření vyšly překvapivě znatelně lépe, což je způsobeno pravděpodobně sofistikovanějším algoritmem, ovšem za cenu strukturovanějších křivek.

Vytvořený generátor křivek byl použit v kryptografickém protokolu, hlavně jako ověření jeho funkčnosti. Bylo implementováno hybridní schéma ECIES, které bylo popsáno v teoretické části práce. Jazyk byl použit C++, pro který poskytuje MIRACL knihovna rozhraní. Jako metoda pro generování je zvolena CM metoda z důvodu rychlosti, ovšem SEA metoda může být použita též. Funkčnost protokolu, potažmo generátoru, demonstrují screenshoty, kde je ukázána komunikace mezi klientem a serverem. Klient po připojení k serveru obdrží informace o čerstvě vygenerované křivce, která bude použita pro další komunikaci. Vypsány do konzole jsou informace jako veřejné a soukromé klíče, zpráva v čitelné podobě, zašifrovaná zpráva atd.

5 ZÁVĚR

V teoretické části byla probrána témata týkající symetrické a asymetrické kryptografie, následně základů problematiky eliptických křivek, a to modulární aritmetika, vybrané části algebraických struktur, konkrétně tělesa a grupy. Dále je už úvod do problematiky eliptických křivek, rovnice křivek nad různými tělesy a jejich podmínky, operace s body na křivce a na křivce nad tělesy. Zmíněny byly i další typy křivek a to např. hypereliptické, co jsou to singulární křivky, Koblitzovy křivky a tak dále. V následující části jsou už popsány kryptografické knihovny pracující alespoň do určité části s eliptickými křivkami, ze kterých pak byla vybrána knihovna MIRACL jakožto ta, díky které byl realizován generátor eliptických křivek. Některé knihovny totiž neumožňují ani základní operace s body a další funkce, které jsou v tomto případě esenciální. Použitá zařízení měla procesory Intel Core i5-5200U 2,20 GHz, resp. Xeon E7-4820 (2,0 GHz/8-core/18 MB cache). S výkonnějšími zařízeními se samozřejmě křivky generují rychleji. Dále bylo teoreticky popsáno hybridní schéma ECIES, které následně bylo implementováno kvůli ověření funkčnosti generátoru.

V praktické části je popsáno měření efektivity různých eliptických křivek a následně je toto měření interpretováno. Měřily se ať už základní operace, tak šifrování a dešifrování za různých podmínek a vliv dalších faktorů, jako je velikost pole, velikost zprávy, zvolená metoda generování křivky atd. Dále se také ECIES porovnávalo s RSA-AES. Kromě měření bylo implementováno již zmíněné schéma ECIES, které sloužilo jako praktické ověření funkčnosti generátoru křivek. Dvě metody mohou být použity během generování - CM metoda a SEA metoda. CM metoda generuje křivky strukturovanější oproti SEA metodě, ale velice rychle. Někdo by mohl pochybovat o CM metodě v tom smyslu, že není tak bezpečná jako SEA metoda, protože by mohl být teoreticky nalezen útok, přestože to není pravděpodobné. V tom případě, pokud by nebyla vyžadována rychlost, je vhodnější SEA metoda. V ECIES protokolu je jako výchozí CM metoda kvůli rychlosti generování křivky. Jsou přiloženy různé algoritmy a pseudokódy, příkladem budiž SEA algoritmus, hledání generátoru grupy, atd.

LITERATURA

[1] BURDA, K. *Bezpečnost informačních systémů.* První vydání. Brno: VUT v Brně, 2013. 152 stran. ISBN 978-80-214-4890-2.

[2] KUBESA, M. *Základy diskrétní matematiky.* Západočeská univerzita v Plzni, Technická univerzita Ostrava. 2011. 136 stran.

[3] MENEZES, A.; van OORSCHOT, P.; VANSTONE, S. *Handbook of applied cryptography* Vyd. 1. Boca Raton: CRC Press, 1997, 780 s. ISBN 08-493-8523-7. Dostupné z URL: <http://cacr.uwaterloo.ca/hac/>.

[4] HANKERSON, D.; MENEZES, A.; VANSTONE, S. *Guide to Elliptic Curve Cryptography* Springer-Verlag New York, Inc., 2003, 332 s. ISBN 0-387-95273-X. Dostupné z URL: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.394.3037&rep=rep1&type=pdf>.

[5] OCHODKOVÁ, E.: *Přínos teorie eliptických křivek k řešení moderních kryptografických systémů.* 2015. Katedra informatiky, FEI, VŠB. 12s.

[6] OCHODKOVÁ, E.: *Matematické základy kryptografických algoritmů.* 2012. VŠB – Technická univerzita Ostrava, Západočeská univerzita v Plzni. 186s.

[7] KUREŠ, M.: *Algebraické a číselně teoretické pozadí eliptické kryptografie s vybranými algoritmy, zejména pro určení řádu eliptické křivky* [online]. VUT v Brně, FSI. 2014. Dostupné z URL: <http://kvaternion.fme.vutbr.cz/2014/kvat6_separaty/kv14_2_kures.pdf>.

[8] KUČERA, R.: *Algebra 1* [online]. Masarykova univerzita. 2015. Dostupné z URL: <https://is.muni.cz/el/1431/jaro2016/M2150/um/Algebra2015grupy.pdf>.

[9] MENEZES, A.; OKAMOTO, T.; VANSTONE, S. *Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field* [online]. 1993. Dostupné z URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=259647>.

[10] BJOERNSEN, K.; *Koblitz Curves and its practical uses in Bitcoin security* [online]. Dostupné z URL:

<https://koclab.cs.ucsb.edu/teaching/ecc/project/2015Projects/Bjoernsen.pdf>.

[11] MENEZES, A.; WU, Y.; ZUCCHERATO, R.: *An elementary introduction to hyperelliptic curves* [online]. 1996. Dostupné z URL: <https://www.math.uwaterloo.ca/~ajmeneze/publications/hyperelliptic.pdf>.

[12] VÁCHA, P.: *Počítání bodů na eliptických a hypereliptických křivkách* [online]. 2013. Katedra algebry. Matematicko-fyzikální fakulta. Univerzita Karlova v Praze. Dostupné z URL: <https://is.cuni.cz/webapps/zzp/download/120138215>.

[13] *Modulární aritmetika* [online]. Institut biostatistiky a analýz Masarykovy univerzity. Dostupné z URL: <http://portal.matematickabiologie.cz/index.php?pg=zaklady-informatiky-pro-biology--teoreticke-zaklady-informatiky\tolerance9999\emergencystretch3em\hfuzz.5\p@\vfuzz\hfuzz--teorie-cisel--modularni-aritmetika>.

[14] MARVAN, M.: *Učební texty k přednášce algebraické struktury* [online]. Matematický ústav. Slezská univerzita v Opavě. Dostupné z URL: <https://www.slu.cz/math/cz/knihovna/ucebni-texty/algebraicke-struktury/1-algebraicke-struktury.pdf>.

[15] *Modulární aritmetika* [online]. Dostupné z URL: <http://fimatek.chciweb.eu/mod.html>.

[16] RISTIĆ, I.: *OpenSSL Cookbook* [online]. 2015, poslední aktualizace březen 2016. Dostupné z URL: <https://www.feistyduck.com/library/openssl-cookbook/online/>.

[17] LIU, A.; NING, P.: *TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks* [online]. 2017. Dostupné z URL: <https://discovery.csc.ncsu.edu/pubs/ipsn08-TinyECC-IEEE.pdf>.

[18] *TinyOS: An open-source OS for the networked sensor regime.* [online]. Dostupné z URL: <http://tinyos.stanford.edu/tinyos-wiki/index.php/TinyOS_Overview>.

[19] *WOLFSSL MANUAL* [online]. 2017. Dostupné z URL: <https://www.wolfssl.com/documentation/wolfSSL-Manual.pdf>.

[20] *wolfCrypt API Reference* [online]. 2016. Dostupné z URL: <https://www.wolfssl.com/documentation/wolfCrypt-API-Reference.pdf>.

[21] Aranha, D. F.; Gouv, C. P. L. *RELIC is an Efficient LIbrary for Cryptography* [online]. 2017. Dostupné z URL: <https://github.com/relic-toolkit/relic>.

[22] *borZoi Manual* [online]. 2003. Dostupné z URL: <http://www.csshl.net/sites/default/files/downloadable/crypto/ecc/borZoi.pdf>.

[23] *Crypto++ Library 5.6.5 API Reference* [online]. 2017. Dostupné z URL: <https://www.cryptopp.com/docs/ref/index.html>.

[24] *LibTomCrypt Developer Manual* [online]. 2017. Dostupné z URL: <https://github.com/libtom/libtomcrypt/releases/download/v1.18.0/crypt-1.18.0.pdf>.

[25] *M.I.R.A.C.L. Users Manual* [online]. Shamus Software Ltd. 2005. Dostupné z URL: <http://read.pudn.com/downloads113/ebook/474678/miracl-manual.pdf>.

[26] *The Legion of the Bouncy Castle* [online]. Dostupné z URL: <http://www.bouncycastle.org/documentation.html>.

[27] *FlexiProvider Documentation* [online]. Dostupné z URL: <https://www.flexiprovider.de/javadoc/flexiprovider/docs/index.html>.

[28] *IAIK ECCelerate Provider API Documentation* [online]. Graz University of Technology. Dostupné z URL: <http://javadoc.iaik.tugraz.at/ECCelerate/current/index.html>.

[29] *A Guide to the IAIK ECCelerate Library* [online]. Graz University of Technology. 2016. Dostupné z URL: <http://jce.iaik.tugraz.at/sic/Media/Files/PDF_Manuals/ECCelerate_Tutorial>.

[30] GALIN, B.: *Schoof-Elkies-Atkin Algorithm* [online]. 2007. Department of Mathematics, Stanford University. Dostupné z URL: <http://www.bens.ws/papers/SchoofElkiesAtkinAlgorithm.pdf>.

[31] BELLARE, Mihir; ROGAWAY, Phillip: *Minimizing the use of random oracles in authenticated encryption schemes* [online]. International Conference on Information and Communications Security. Springer, Berlin, Heidelberg, 1997. Dostupné z URL: <https://link.springer.com/chapter/10.1007/BFb0028457>.

[32] MARTÍNEZ, V. Gayoso, et al. A Comparison of the Standardized Versions of ECIES. Information Assurance and Security (IAS), 2010 Sixth International Conference on. IEEE, 2010. Dostupné z URL: <https://ieeexplore.ieee.org/abstract/document/5604194/>.

[33] MIELE, A; LENSTRA, A. K. Efficient ephemeral elliptic curve cryptographic keys. International Information Security Conference. Springer, Cham, 2015. p. 524-547. Dostupné z URL: <https://eprint.iacr.org/2015/647.pdf>.

[34] LENSTRA, A. K. Efficient identity based parameter selection for elliptic curve cryptosystems. Australasian Conference on Information Security and Privacy. Springer, Berlin, Heidelberg, 1999. p. 294-302. Dostupné z URL: <https://infoscience.epfl.ch/record/149483/files/EPFL-CONF-149483.pdf>.

[35] SILVERMAN, J. H.; SUZUKI, J. Elliptic curve discrete logarithms and the index calculus. International Conference on the Theory and Application of Cryptology and Information Security. Springer, Berlin, Heidelberg, 1998. p. 110-125. Dostupné z URL: <https://link.springer.com/content/pdf/10.1007/3-540-49649-1_10.pdf>.

[36] DAMGARD, I. B. Lectures on data security: modern cryptology in theory and practice. New York: Springer, 1999. ISBN 3-540-65757-6.

[37] BARKER, Elaine, et al. Recommendation for key management part 1: General (revision 3). NIST special publication, 2012, 800.57: 1-147.

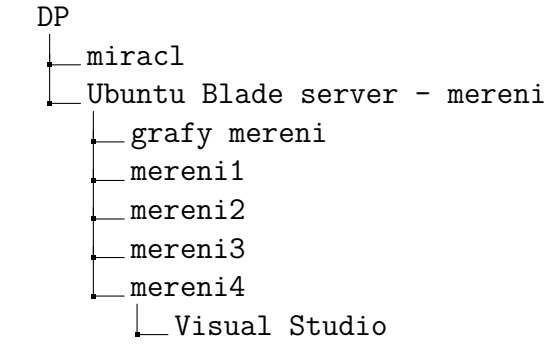
[38] GAYOSO MARTÍNEZ, V.; HERNÁNDEZ ENCINAS, L.; QUEIRUGA DIOS, A. Security and practical considerations when implementing the Elliptic Curve Integrated Encryption Scheme. Cryptologia, 2015, 39.3: 244-269. Dostupné z URL: <https://www.researchgate.net/publication/277941706_Security_and_Practical_Considerations_When_Implementing_the_Elliptic_Curve_Integrated_Encryption_Scheme>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

$\rightarrow$	Zobrazení
$\wedge$	A zároveň
$\equiv$	Kongruence, např. $a \equiv b \, mod \, n$
$\oplus$	Operace XOR, např. $1 \oplus 0$
$\in$	Náleží, je prvkem, např. $2 \in N$
$\exists$	Existuje, pro nějaké
$\forall$	Pro všechna, pro každé
API	Application Programming Interface
DSA	Digital Signature Algorithm
ECC	Elliptic Curve Cryptography
ECDGSA	Elliptic Curve German Digital Signature Algorithm
ECDH	Elliptic Curve Diffie Hellman
ECDSA	The Elliptic Curve Digital Signature Algorithm
ECIES	Elliptic Curve Integrated Encryption Scheme
ECMQV	Elliptic Curve Menezes-Qu-Vanstone
ECSS	Elliptic Curve Schnorr Signature
F_q	Konečné pole
F_p	Prvočíselné pole
F_{2^m}	Binární pole
GCC	GNU Compiler Collection
HECC	Hyperelliptic Curve Cryptography
HMAC	Hash-based Message Authentication Code
JNI	Java Native Interface
KDF	Key Derivation Function
MAC	Message Authentication Code
mod	Modulo, např. x mod 5
NIST	National Institute of Standards and Technology
RSA	Algoritmus RSA (autoři Rivest, Shamir, Adleman)
SEA	Schoof–Elkies–Atkin algoritmus
SECG	Standards for Efficient Cryptography Group
SSH	Secure Shell
SVDP	Secret Value Derivation Primitive
TLS	Protokol Transport Layer Security
wNAF	w-ary Non-Adjacent Form
$\mathbb{Z}$	Množina celých čísel
$\mathbb{Z}_n$	Množina zbytkových tříd

A OBSAH PŘILOŽENÉHO CD

Je zde soubor *readme.txt* popisující všechny přílohy, především pak popisuje demonstrace vytvořeného generátoru a návod, jak ukázky lze spustit, případně i jak kód kompilovat a linkovat. Níže je adresářová struktura.



Ve složce *miracl* jsou všechny soubory knihovny MIRACL a navíc soubory náležící k této práci. Mimo jiné jsou zde spustitelné soubory (a jejich kódy) *demonstraceCM.exe* a *demonstraceSEA.exe* jednoduše ukazující generování křivek CM metodou, resp. SEA metodou. Dále je zde demonstrace protokolu ECIES - soubory *server.exe* a *klient.exe*. Ve složce *Ubuntu Blade server - mereni* jsou pak data jednotlivých měření (roztříděná do složek) a související grafy uvedené v této práci (a některé i navíc). Ve složce *Visual Studio* pak jsou řešení z Visual Studia obsahující projekty představující klienta a server. V adresáři *DP* jsou soubory *klient.ecs* a *server.ecs* představující křivku použitou během průběhu ECIES protokolu. Dále je zde zmíněný soubor *readme.txt*. Nakonec je tu text samotné práce ve formátu PDF.