VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2018 Bc. Josef Brychta

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

SROVNÁNÍ KRYPTOGRAFICKÝCH PRIMITIV VYUŽÍVAJÍCÍCH ELIPTICKÝCH KŘIVEK NA RŮZNÝCH HARDWAROVÝCH PLATFORMÁCH

COMPARISON OF CRYPTOGRAPHIC PRIMITIVES USED IN ELLIPTIC CURVE CRYPTOGRAPNY ON DIFFERENT HARDWARE PLATFORMS

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — **Bc. Josef Brychta**

AUTHOR

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

SUPERVISOR

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

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

Student: Bc. Josef Brychta ***ID:*** 158524
Ročník: 2 ***Akademický rok:*** 2017/18

NÁZEV TÉMATU:

Srovnání kryptografických primitiv využívajících eliptických křivek na různých hardwarových platformách

POKYNY PRO VYPRACOVÁNÍ:

Úkolem diplomové práce je na vybraných hardwarových platformách (min. třech) implementovat variaci kryptografických knihoven obsahujících primitiva pro eliptické křivky. Následně bude nutno tyto knihovny patřičně upravit a vytvořit vlastní měřicí schémata tak, aby mohlo dojít k porovnání jednotlivých implementací. Hlavním úkolem tak bude nejen implementace knihoven, ale hlavně návrh a realizace testovacích scénářů společně s vytvořením měřicích metod pro různé knihovny (kryptografická primitiva i základní algebraické operace) a hardwarové platformy. Ve výsledku bude provedena řada experimentálních testů zaměřena na různé křivky i jejich parametry tak, aby výsledky práce zahrnovaly komplexně problematiku eliptických křivek v kryptografii. Hlavními parametry zde budou energetická náročnost, časová náročnosti, výpočetní náročnost a paměťové náročnosti a další.

DOPORUČENÁ LITERATURA:

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

[2] H. Cohen, G. Frey. Handbook of Elliptic and Hyperelliptic Curve Cryptography, Taylor and Francis, 2006.

Termín zadání: 5.2.2018 ***Termín odevzdání:*** 21.5.2018

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

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

ABSTRAKT

Diplomová práce se zabývala implementací variací kryptografických knihoven obsahujících primitiva pro eliptické křivky. Vytvořením vlastních měřicích schémat tak, aby mohlo dojít k porovnání jednotlivých implementací. Hlavním úkolem tak byla nejen implementace knihoven, ale hlavně návrh a realizace testovacích scénářů společně s vytvořením měřicích metod pro různé knihovny a hardwarové platformy. Ve výsledku byla provedena řada experimentálních testů zaměřena na různé křivky i jejich parametry tak, aby výsledky práce zahrnovaly komplexně problematiku eliptických křivek v kryptografii. Hlavními parametry zde byly energetická, časová a paměťová náročnost.

KLÍČOVÁ SLOVA

Kryptografická primitiva, kryptografie, eliptické křivky, kryptografické knihovny, Raspberry Pi, SECP, SECT, BRAINPOOL, GOST, FRP, Double and add, wNAF, Montgomery ladder, OpenSSL, libecc, TinyECC, WolfSSL, AvrCryptoLib, WiseLib, Crypto++, LibTomCrypt, MIRACL.

ABSTRACT

This master thesis deals with the implementation of variants of cryptographic libraries containing primitives for elliptic curves. By creating custom metering charts to compare each implementation. The main task was not only the implementation of libraries but also the design and implementation of test scenarios together with the creation of measurement methods for different libraries and hardware platforms. As a result, a number of experimental tests were conducted on different curves and their parameters so that the results of the work included complex problems of elliptic curves in cryptography. The main parameters were power, time and memory consumption.

KEYWORDS

Cryptographic primitives, cryptography, elliptic curves, cryptographic libraries, Raspberry Pi, SECP, SECT, BRAINPOOL, GOST, FRP, Double and add, wNAF, Montgomery ladder, OpenSSL, libecc, TinyECC, WolfSSL, AvrCryptoLib, WiseLib, Crypto++, LibTomCrypt, MIRACL.

BRYCHTA, Josef. *Srovnání kryptografických primitiv využívajících eliptických křivek na různých hardwarových platformách*. Brno, 2018, 83 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 „Srovnání kryptografických primitiv využívajících eliptických křivek na různých hardwarových platformách“ 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, vstřícnost, konzultace, trpělivost, starostlivost a podnětné návrhy k práci. Jeho ochota a nadhled mne dokázaly vždy motivovat. Dále bych chtěl poděkovat jeho kolegovi panu Ing. Pavlu Maškovi, Ph.D. za vypůjčený hardware a přátelskou atmosféru u nich v kanceláři. V neposlední řadě bych chtěl poděkovat rodině, přítelkyni a přátelům za trpělivost a porozumění při psaní této diplomové práce.

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

Kryptografie je v dnešní době používána prakticky ve všech oblastech, které jakkoliv souvisí s počítači či sítěmi. Od internetového bankovnictví, platebních karet, šifrování souborů, disků nebo jen zabezpečený přístup na webové stránky. Pro usnadnění práce vývojářům byly navrženy různé kryptografické knihovny, které mají za cíl implementovat požadované kryptografické funkcionality. Těchto kryptografických knihoven je spousta a každá je vhodná na jinou platformu. Motivací této diplomové práce bylo usnadnění výběru kryptografických knihoven vývojářům otestováním variace knihoven a experimentálním proměřením kryptografických primitiv na eliptických křivkách a tím zjistit jejich reálnou efektivitu nezávisle na hardwarových platformách.

Diplomová práce byla rozdělena do pěti kapitol. Úvod do kryptografie a kryptografie eliptických křivek. Rešerše knihoven zabývajících se kryptografií eliptických křivek. Vybrané eliptické křivky, knihovny a hardwarové platformy. Příprava měření kryptografických primitiv na eliptických křivkách. V kapitole úvod do kryptografie a kryptografie eliptických křivek se práce zabývala v první řadě úvodem do kryptografie. Základními primitivy používanými v kryptografii, kryptografií nad eliptickými křivkami, aritmetikou grupy, aritmetikou na eliptických křivkách, primitivy na eliptických křivkách využitých v praktické části práce. Algoritmy multiplication využité v praktické části práce, diskrétními body na eliptické křivce a kryptografickou funkcí založenou na eliptických křivkách. V kapitole rešerše knihoven zabývajících se kryptografií eliptických křivek byly kriticky rozebrány kryptografické knihovny OpenSSL, libecc, TinyECC, WolfSSL, AvrCryptoLib, WiseLib, Crypto++, LibTomCrypt a MIRACL. V kapitole vybrané eliptické křivky, knihovny a hardwarové platformy byl popsán a vysvětlen výběr eliptických křivek použitých v praktické čísti práce. Kapitola se zabývá parametrizací eliptických křivek standardů NIST, který se dělí na SECP a SECT křivky. Dále křivek standardů BRAINPOOL, GOST a FRP. Vybranými knihovnami a jejich implementací do hardwarových zařízení. Popisem vybraných hardwarových zařízení raspberry Pi Zero/1/2/3 a instalací těchto zařízení. V kapitole příprava měření kryptografických primitiv na eliptických křivkách se práce zabývá způsoby měření kryptografických knihoven, měřicími metodami, popisem použitých algoritmů. Měřením časové a paměťové náročnosti kryptografických primitiv a dalších. V poslední kapitole experimentální měření kryptografických primitiv na eliptických křivkách se práce zabývá výsledky, analýzou a porovnáním měřených veličin na eliptických křivkách.

1 ÚVOD DO KRYPTOGRAFIE A KRYPTOGRAFIE ELIPTICKÝCH KŘIVEK

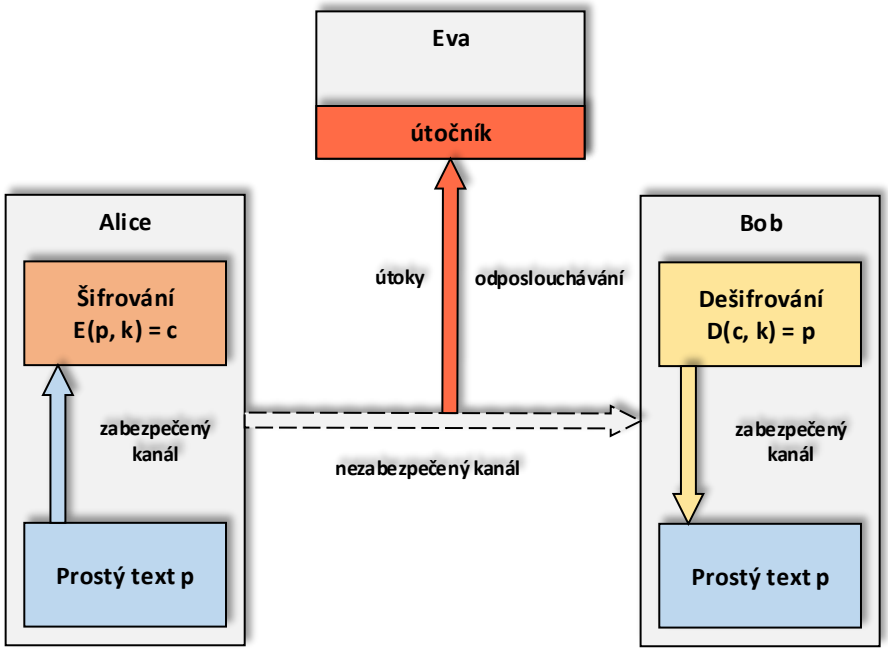
Lidé každý den sdílejí své informace. Většina těchto informací je důvěrná. Jen málo z tohoto sdílení je však zajištěno úměrně jejich významu a citlivosti proti zneužití. V této části bude rozebrána problematika kryptografie obecně a kryptografie eliptických křivek.

1.1 Úvod do kryptografie

Na začátek je nutno uvést, že kryptografie je součástí kryptologie. Ta se skládá ze dvou oblastí [1]:

- Kryptografie – je oblast zabývající se vytvářením kryptografických systémů.
- Kryptoanalýza – je oblast zabývající se útoky na kryptografické systémy, je to vlastně opak kryptografie.

Kryptografie je oblast počítačové vědy a matematiky, která se zaměřuje na bezpečnostní techniky komunikace mezi subjekty (v našem případě Alice a Bob), zatímco další strana (v našem případě Eva) je útočník nacházející se mezi nimi(Obr. 1.1). Kryptografie je založena na metodách jako je šifrování, dešifrování, podepisování, generování pseudonáhodných čísel atd.



Obr. 1.1: Základní myšlenka zabezpečené komunikace mezi dvěma subjekty [1].

Čtyři základní cíle kryptografie jsou [1]:

- Důvěrnost – definuje soubor pravidel, která omezují přístup nebo přidávají omezení určitých informací.
- Integrita dat – zabezpečuje konzistenci a přesnost dat během celého životního cyklu.
- Autentizace – potvrzuje pravdivost atributu na základě, který je některými údajně pravdivý.
- Nepopiratelnost – zajišťuje neschopnost autora popřít informaci.

Kryptosystém, který zahrnuje mj. i podpisová schémata, řešení autorizace a autentizace, integrity aj. se skládá ze dvou typů kryptografických šifer – symetrická a asymetrická. Symetrická šifra je založena na tom, že oba subjekty se domluví na šifrovacím klíči (tajném klíči), pomocí kterého oba subjekty šifrují i dešifrují zprávy. Naopak asymetrická šifra používá k šifrovaní i dešifrování klíče rozdílné – veřejný a privátní klíč.

V následujícím odstavci je pro názornost jednoduše popsána komunikace mezi dvěma subjekty (Alicí a Bobem) využívající symetrickou a posléze asymetrickou šifru:

- Symetrická:
  1) Alice a Bob se dohodnou na symetrickém kryptosystému.
  2) Alice a Bob se dohodnou na společném tajném klíči (tajném klíči).
  3) Alice zašifruje zprávu pomocí tajného klíče.
  4) Alice pošle šifrovanou zprávu Bobovi.
  5) Bob dešifruje šifrovanou zprávu tajným klíčem.
- Asymetrická:
  1) Alice a Bob se dohodnou na asymetrickém kryptosystému s veřejným klíčem.
  2) Bob posílá Alici svůj veřejný klíč.
  3) Alice zašifruje zprávu pomocí Bobova veřejného klíče.
  4) Alice pošle šifrovanou zprávu Bobovi.
  5) Bob dešifruje zprávu pomocí svého soukromého klíče.

Stěžejní problematika je výměna tajných klíčů. Klíče si musí subjekty mezi sebou vyměňovat bezpečně. Eva, která odposlouchává šifrovanou komunikaci mezi dvěma subjekty (Alicí a Bobem), se snaží přijít na sdílené tajemství (soukromý klíč). Pokud se Evě podaří odposlechnout šifrovanou zprávu, tak v případě symetrické šifry stačí uhodnout šifrovací klíč a může dešifrovat veškerou tajnou komunikaci [1, 2].

1.2 Základní primitiva používaná v kryptografii

V této kapitole budou definována základní primitiva používaná v kryptografii a při testování kryptografických knihoven. Nejprve bude nutné zmínit aritmetiku velkých čísel, konečná pole a modulární algebru.

1.2.1 Aritmetika velkých čísel

V kryptografických systémech dnešní doby jsou používána čísla o velikosti např. 128 b až 512 b či více. Tato čísla odpovídají velikosti klíčů a dalším parametrům kryptografických algoritmů [3]. S takto velkými čísly není možné na dnešních výpočetních zařízeních pracovat. Je třeba jim vytvořit speciální reprezentace, např. tzv. poziční číselná soustava [3]. Např. pokud máme číslo x = 54321 s bází b = 10, vyjádření poziční číselnou soustavou bude:

$$x_{10} = 5 \cdot 10^4 + 4 \cdot 10^3 + 3 \cdot 10^2 + 2 \cdot 10^1 + 1 \cdot 10^0. \tag{1.1}$$

1.2.2 Konečná pole

Uvažujme například množinu reálných čísel $\mathbb{R}$. Pokud v této množině definujeme aritmetické operace sčítání (+) a násobení (.), vznikne celočíselné pole. U takového pole jsou výsledky zasaženy velkou zaokrouhlovací chybou a výpočetní operace jsou pomalé. Proto nejsou taková pole vhodná v kryptografických systémech. Používají se tzv. konečná pole, konečná struktura nebo Galosovo pole GF (z angličtiny "Galoi's field"). Tato označení definují pole s konečným počtem prvků. Platí pro ně speciální axiomy z algebraických struktur (grupoid, pologrupa, monoid). Uvažujme konečné pole $\mathbb{F}$ s operacemi sčítání a násobení. Pro takto definované pole tedy platí Tab. 1.1:

Tab. 1.1: Vztahy pro definované konečné pole [1].

Název	Definice
Asociativní zákon	$(a+b)+c = a+(b+c)$
	$(a \cdot b) \cdot c = a \cdot (b \cdot c)$
Komutativní zákon	$a+b = b+a$
	$a \cdot b = b \cdot a$
Distributivní zákon	$a \cdot (b+c) = a \cdot b + a \cdot c$
Existence opačného prvku	$\forall a \in \mathbb{F}\ existuje\ -a : a + (a) = 0$
Existence neutrálního prvku	$a,\ 0,\ 1 \in \mathbb{F} : a + 0 = a,\ a \cdot 1 = a$
Uzavřenost pole vůči sčítání a odčítání	$a + b = c \Rightarrow c \in \mathbb{F}$
	$a \cdot b = c \Rightarrow c \in \mathbb{F}$

1.2.3 Modulární algebra

V předchozí kapitole byly definovány vztahy pro konečná pole. Nyní je třeba definovat algebraické operace [3]:

- Sčítání (+): Nechť p je prvočíslo definující řád konečné pole $\mathbb{F}_p$. Mějme dvě čísla $a,\ b \in \mathbb{F}_p$, potom $a + b = c \in \mathbb{F}_p$, kde $c \in \{0, ..., p-1\}$, tuto hodnotu budeme dále označovat jako zbytek r. Poté tedy můžeme definovat tzv. sčítání modulo p jako $a + b \equiv r\ (mod\ p)$.
- Odčítání: Nechť $\mathbb{F}_p$ o řádu p a dvě čísla $a,\ b,\ r \in \mathbb{F}_p$. Definujme opačný prvek $-a$ k prvku a, pro který platí $a + (-a) = 0$, tedy odčítání dvou prvků zavedeme jako součet $b - a = b + (-a)$. Následně můžeme definovat odčítání podobně jako sčítání modulo p, tedy jako $b + (-a) \equiv r\ (mod\ p)$.
- Násobení (.): Nechť $\mathbb{F}_p$ o řádu p a dvě čísla $a,\ b \in \mathbb{F}_p$, potom $a \cdot b = r \in \mathbb{F}_p$, kde $r \in \{0, ..., p-1\}$. Poté tedy můžeme definovat tzv. násobení modulo p jako $a \cdot b \equiv r\ (mod\ p)$.
- Dělení: Nechť $\mathbb{F}_p$ o řádu p a dvě čísla $a,\ b,\ r \in \mathbb{F}_p$. Definujme inverzní prvek a^{-1} k prvku a, pro který platí $a \cdot a^{-1} = 1$, tedy dělení dvou prvků zavedeme jako násobení $b/a = b \cdot a^{-1}$. Následně můžeme definovat dělení podobně jako násobení modulo p, tedy jako $b \cdot a^{-1} \equiv r\ (mod\ p)$.

1.3 Kryptografie nad eliptickými křivkami

Kryptografie nad eliptickou křivkou z anglického Elliptic curve cryptography – ECC je jedna z metod kryptografie veřejných klíčů v moderních kryptografických systémech založena na algebraických strukturách eliptických křivek nad konečnými tělesy. Bezpečnost této metody závisí na obtížnosti problému algoritmu. O tomto problému bude psáno v pozdějších kapitolách. Nejdříve bude představena problematika eliptických křivek [4, 5]. Kryptografické systémy s veřejným klíčem tohoto typu jsou založeny na kryptografické funkci. Funkci, pro kterou výstup náležící určitému vstupu, lze spočítat relativně snadno, ale vstup náležící určitému výstupu je nemožné spočítat. Požadovaná kryptografická funkce je vytvořena opakovaným užitím určité operace na jeden prvek ze skupiny prvků – grupy (o grupách bude řečeno později). Vstupem do jednosměrné funkce je číslo, udávající kolikrát má být tato určitá operace provedena. Toto je výpočetně nemožné určit od někoho, kdo zná jen prvek, na který se operace vztahuje, a výsledný výstup, ale nezná odpovídající vstup [6]. V nejranější podobě kryptografických systémů s veřejným klíčem zahrnovala zvolená skupina prvků jen množinu celých čísel [4]:

$$1...p-1, \tag{1.2}$$

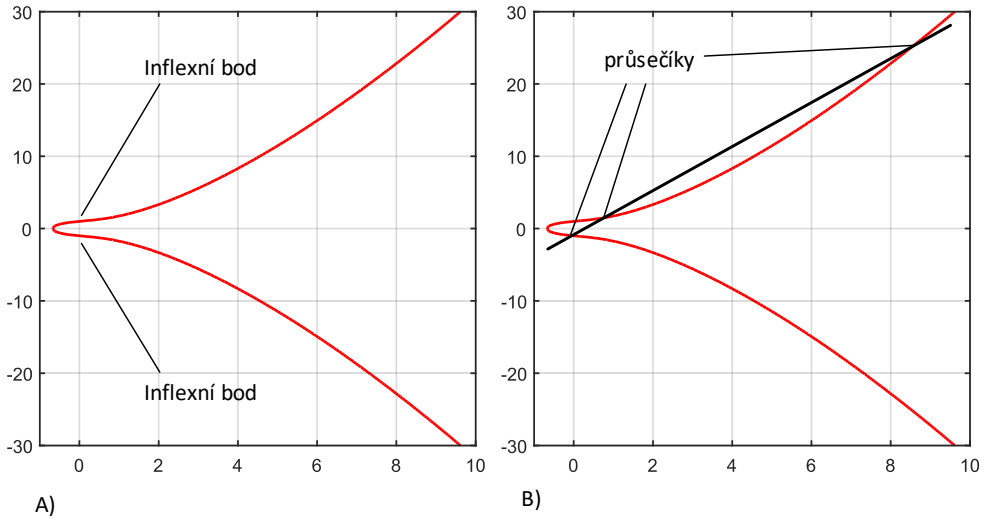
kde p bylo velké prvočíslo a skupina operací byla násobení dvou celých čísel modulo p. Avšak v kryptografických systémech s eliptickou křivkou jsou zvolená skupina prvků body na eliptické křivce a skupina operací jsou další dva body. V následujících kapitolách to bude vysvětleno graficky [4, 5].

1.3.1 Eliptické křivky

Nejznámější interpretace pro aplikaci eliptických křivek je rovinná křivka, která se skládá z bodů vyhovujících rovnici [6]:

$$y^2 = x^3 + ax + b. \tag{1.3}$$

Prvním faktem vyplývajícím z rovnice je, že výraz y^2 znamená, že křivka je symetrická kolem osy x. To znamená, že pro každou kladnou hodnotu y existuje její záporná hodnota se stejnou hodnotou na souřadnici x. Druhým faktem je, že výraz x^3 znamená, že pro určité hodnoty a a b existuje přesně jeden inflexní bod na křivce na každé straně osy x. Inflexní body jsou umístěny v místě, kde křivka prochází osou y. Na Obr. 1.2, část A je znázorněna eliptická křivka pro konkrétní body (a=3, b=1). Symetrie kolem osy x je zde jasně vidět. Na obrázku jsou také vyznačeny body inflexe. Inflexní bod je takový bod grafu funkce, ve kterém dochází k přechodu mezi konvexní a konkávní částí grafu. V inflexním bodě se mění zakřivení grafu funkce a tečna grafu v tomto bodě graf protíná [7]. Nezbytným důsledkem pro křivky těchto dvou vlastností je, že přímka, která protíná křivku ve dvou bodech, také protíná křivku v bodě třetím, jak je vidět na Obr. 1.2, část B.



Obr. 1.2: A) Příklad eliptické křivky pro a=3, b=1 s vyznačenými body inflexe, B) Průsečíky přímky na eliptické křivce.

Existuje zde ale výjimka, a to pro vertikální přímky, které protnou křivku pouze ve dvou či jednom bodě (v případě, že linka je vertikální tečkou) [7].

1.3.2 Formy eliptických křivek v kryptografii

V této kapitole bude pro snadnější pochopení psáno o nejznámější tzv. Weierstrassově formě eliptických křivek. Je nutno zmínit, že existují i ostatní formy. Mezi nejpoužívanější formy eliptických křivek tedy patří např. [1, 2]:

- Weierstrassova,
- Montgomeryho,
- Hessianova,
- Edwardova.

1.3.3 Grupa v souvislosti s eliptickou křivkou

Křivka nad reálnými čísly nemá v tomto okamžiku žádný další účel, takže budeme dále uvažovat jen body s celočíselnými čísly v rozsahu:

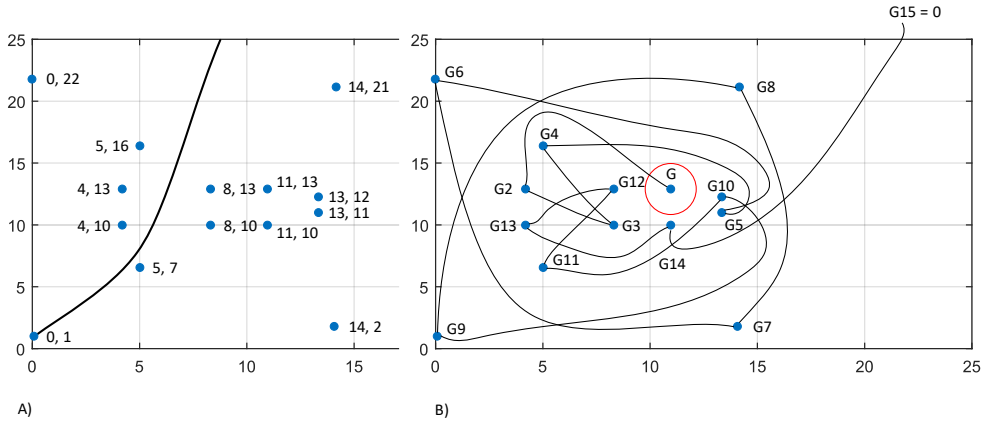
$$\{0...p-1\}. \tag{1.4}$$

Až na výjimku tvoří tyto body konečnou grupu. Stejně jako pole, grupa obsahuje sadu prvků. Ale na rozdíl od pole, které definuje čtyři matematické operace, grupa definuje pouze jednu matematickou operaci (operaci grupy) [4, 5]. Při této operaci je skupina uzavřena (tj. když jsou dvě grupy prvků kombinovány operací skupiny, výsledek je jiný prvek grupy). Aby skupina bodů vytvořila grupu, musí mít definovaný identity element – známý jako fiktivní bod v nekonečnu, označený 0. Nesmíme opomenout, že body jsou ve dvojicích uspořádány symetricky kolem horizontální osy:

$$y = p/2. \tag{1.5}$$

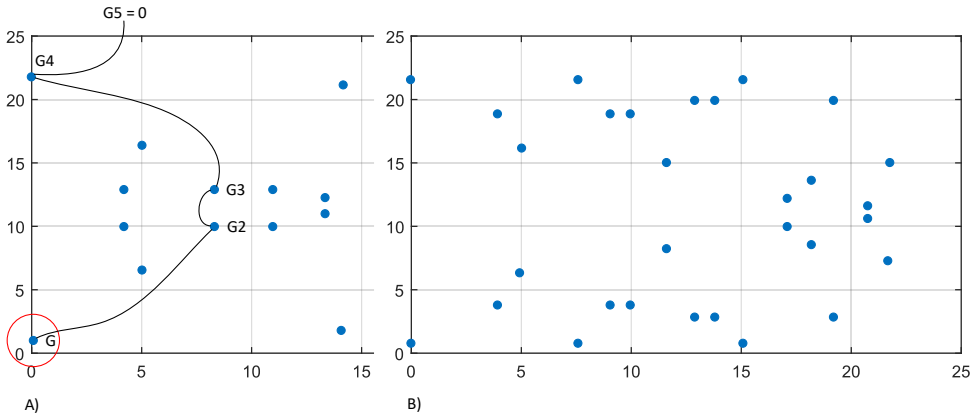
Každý prvek dvojice je aditivum (tj. když jsou přidány dohromady, výsledek je 0). Když přičteme k bodu p jeho inverzní aditivum (-p), dostaneme vertikální přímku, která prochází tzv. identity elementem. Tím, že prochází tímto identity elementem, dokončíme grupu s pořadím (tj. s počtem prvků grupy) q. V následujícím příkladu má q hodnotu 15: $P + (-P) = O$, $P + O = P$. Protože souřadnice bodů jsou prvky pole, lze stále provést addition křížových bodů. Použijeme modifikovaný vzorec pro výpočet souřadnic bodů tak, že sečteme dva body: $s = (3x_1^2 + a) \cdot (2y_1^{-1})^{-1}\ mod\ p$, $x_2 = s^2 - 2x_1\ mod\ p$, $y_2 = -y_1 + s \cdot (x_1 - x_2)\ mod\ p$. Bez ohledu na to, které dva body jsou zvoleny jako vstup, je výsledek vždycky další bod [4, 5]. Lze si všimnout, že se nevyskytuje v těchto rovnicích b a jen se promítne

v doubling bodů. Některé body se nazývají generátory, protože generují násobky každého bodu ve skupině (viz Obr. 1.3, část A, B).



Obr. 1.3: A) Souřadnice bodů, B) Body na eliptické křivce jsou násobky generátorů.

Body ve skupině jsou generovány postupným aplikováním skupinových operací (addition bodu) do bodu nazvaného generátor (nebo také základní bod). Například pokud zvolíme bod (11, 13), jeho násobky generují 15 bodů. Nad 15 se cyklus opakuje tak, že: $G16 = G$, $G17 = G2$. Ne všechny body jsou však generátory (viz Obr. 1.4, část A). Pokud si vybereme bod (0, 1), například jeho násobky generují pět bodů, takže pořadí podskupiny vytvořené bodem (0, 1) je pouze 5. I když je P velké, není zaručeno, že bude určitá podskupina velká také. Pokud je pořadí skupiny prvočíslo, pak je každý bod násobkem každého jiného bodu kromě bodu nekonečna. Například na Obr. 1.4, část B.

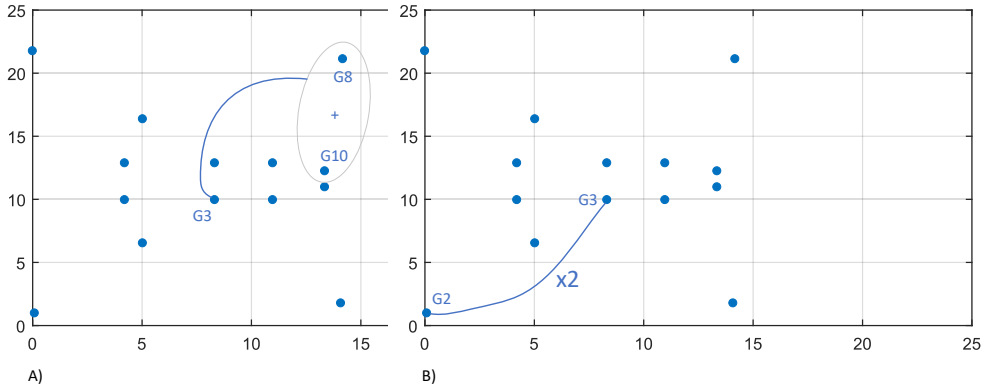


Obr. 1.4: A) Ne všechny body na eliptické křivce jsou generátory, B) Body na křivce $y^2 = x^3 + 5x + 1 \ mod\ 22$.

Tato křivka má 15 bodů a bod v nekonečnu, takže celkem 31, což je prvočíslo. Proto jsou všechny body s výjimkou bodu nekonečna generátory. Generování skupiny v pořadí 31 a také menších podskupin.

1.4 Aritmetika grupy

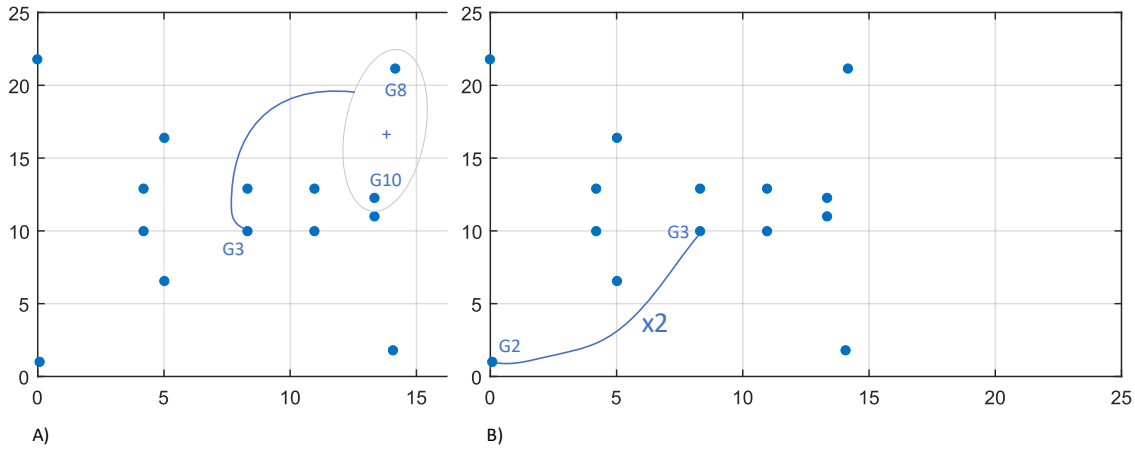
Pokud jsou body vyjádřeny jako násobky generátorů, mohou být addition modulo násobků q. Pro názornost byly použity minulé příklady: $G11+G3 = G14$, znázorněn je na Obr. 1.5, část A a příklad: $G13 + G12 = G10$, znázorněn je na Obr. 1.5, část B.



Obr. 1.5: A) Addition bodů G11 + G3 = G14 na eliptické křivce, B) Addition bodů G13 + G12 = G10 na eliptické křivce.

Z těchto příkladů můžeme vidět, že addition bodů zobrazuje distributivní vlastnictví: $Gi + Gj = G(i + j)$. Pořadí skupiny q je určeno parametry křivky a poli modulo, to je skupina, která určuje zabezpečení kryptografických schémat založených na eliptické křivce. V tomto jednoduchém příkladu by bylo možné držet všechny body v paměti počítače a přímo provádět operace s těmito body. Ale v kryptografických aplikacích v reálném světě je to nemožné. Místo toho jsou body vypočteny, jak je potřeba [4, 6, 14]. Protože existují q-kvadratické způsoby výběru dvojice bodů a pouze q možných výsledků, mnoho kombinací produkuje stejný výsledek.

Znázornění: $G8 + G10 = G3$, $G6 + G12 = G3$, $Gi + Gj = G(i + j)$ je na Obr. 1.6, část A. Bod lze také zdvojnásobit (dále doubling): $2 \cdot G9 = G9 + G9 = G3$. Znázornění je na Obr. 1.6, část B:



Obr. 1.6: A) Addition bodu G8 + G10 = G3 na eliptickou křivku, B) Doubling bodu 2 . G9 = G3 na eliptické křivce.

1.5 Celočíselné násobení eliptických křivek

Primitivní operace použitá ve všech kryptografických aplikacích založených na eliptických křivkách s veřejným klíčem je násobení bodu celým číslem. To zahrnuje opakované použití operací doubling a addition, přičemž počet operací je určen počtem bitů v binárním zobrazení celočíselného multiplikátoru. Každý bit multiplikátoru je zpracováván samostatně a postupně.
Výsledek zpracování každého bitového násobitele je výsledkem a doubling výsledku zpracování předchozího bitu a volitelné addition základního bodu v závislosti na hodnotě aktuálního bitu multiplikátoru [4, 6]. Což představuje y v binární formě s $x+1$:

$$y = y_0 + y_1 \cdot 2 + y_2 \cdot 2^2 + ... + y_x \cdot 2^x, \tag{1.6}$$

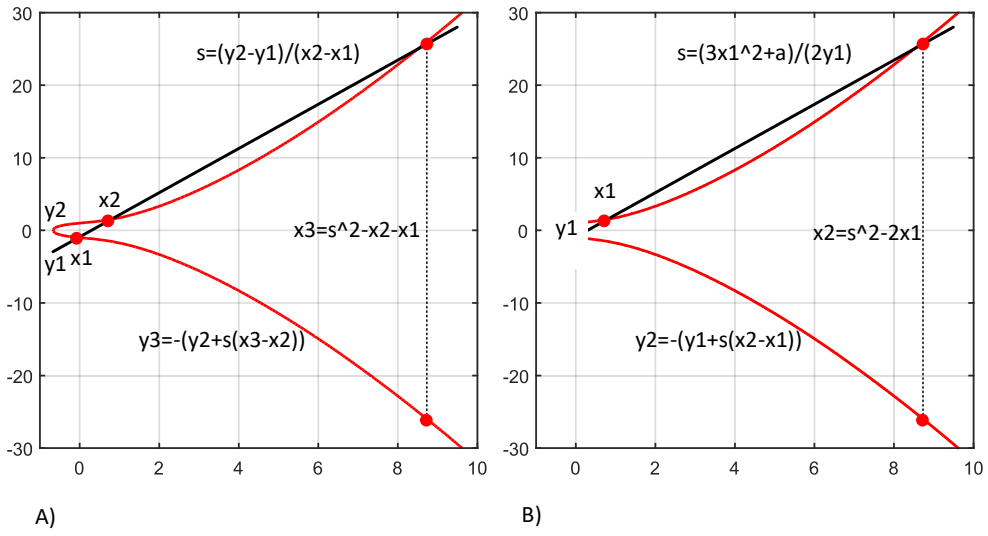
kde: $[y_0..y^x]$ jsou bity (0, 1). Algoritmus pro výpočet P = Gy je:

```
P:=0
for i from x to 0 do
P:= 2P if yi == 1 then P:= P + G
return P
```

Pokud například n bylo 256-bitové číslo, pak by tento algoritmus vyžadoval maximálně 512 operací, zatímco nativní přístup přidání G k sobě n-krát by vyžadoval přinejmenším nemožných $2^{255} - 1$ operací. Protože příkaz *if* závisí na hodnotě jednoho bitu od n je nejistota výpočtu konečného výsledku 2^m. Za m dostatečné velikosti je prakticky nemožné zjistit hodnotu n potřebnou k násobení G k získání P. Pokus o výpočet n z P a G je známo jako problém diskrétního algoritmu při výpočtu s eliptickou křivkou [6, 8].

1.6 Aritmetika na eliptických křivkách

Na základě faktu tří průsečíků při protnutí křivky přímkou získáváme možnosti, jak za použití aritmetických zákonů nalézt třetí bod při znalosti dalších dvou. Tomuto se říká z angličtiny addition, ale nemá souvislost s matematickou operací stejného názvu [8]. Addition dvou bodů zahrnuje nalezení bodu třetího. Je to průsečík mezi křivkou procházející dvěma body na přímce a protnutím v ose x. Toto vede k získání dalšího bodu na křivce, jak je možno vidět na Obr. 1.7, část A – zrcadlením bodu podle osy x. Bod může být přidán opakovaně, tj. vynásobením číslem, aniž by došlo k opakujícímu se cyklu. Na Obr. 1.7 je označen sklon přímky – jako s a body x_3, y_3 jako souřadnice bodů, které jsou součtem dvou bodů x_1, y_1 a x_2, x_2. Výpočet s a y_3 je elementárním výpočtem při použití geometrických trojúhelníků, zatímco výpočet x_3 se počítá nalezením kořenů kubické rovnice [8, 9]. Aplikováním addition na sebe sama se říká doubling bodu, to znamená nalezení druhého bodu, kde tečna v prvním bodě protíná křivku a pak odráží tento bod v ose x, jak je naznačeno na Obr. 1.7, část B [9]. Jak je uvedeno výše, výpočet s a y_2 je elementárním výsledkem za použití geometrie trojúhelníků a výpočet x_2 je čítán výpočtem kořenů kubické rovnice.



Obr. 1.7: A) Addition s body na eliptické křivce B) Doubling s body na eliptické křivce.

1.7 Primitiva na eliptických křivkách využita v praktické části práce

Uvažujme eliptickou křivku E ve Weierstrassově formě:

$$Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6 \tag{1.7}$$

nad polem K. Nechť $P = (x_1,\ y_1)$ je bod na $E(K)$ [10].

1.7.1 Generování náhodného bodu na eliptické křivce

Standardní metodou pro generování náhodného bodu na eliptické křivce je volba náhodné x-souřadnice a vyřešení kvadratické rovnice pro y. (Pokud neexistuje žádné řešení, je vybrána nová x-souřadnice.) Pro liché charakteristiky to lze provést jednou, jestliže je možné najít kvadratické kořeny prvků. V konečném poli primárního pořadí mohou být kvadratické kořeny získány následujícím způsobem. V cyklické skupině lichého řádu t je snadné vypočítat kvadratické kořeny: jednoduše exponenciovat $(t + 1)/2$. Obecně platí, že pro liché prvočíslo $\mathbb{F}_q^*$ je izomorfní $\mathbb{Z}_{2^s}^+ \times \mathbb{Z}_t^+$, což vede k následujícím postupu pro výpočet $b = \sqrt{a} \in \mathbb{F}_q^*$.

V konkrétních případech existují jednodušší způsoby generování náhodných bodů. Například jestliže p je prvočíslo s $p = 2 \bmod 3$ pak kořenová krychle $x \in \mathbb{F}_q$ je prostě $x^{(2p-1)/3}$. Pokud tedy $a = 0$ v rovnici eliptické křivky, pak je jednodušší, abychom nejprve vypočítali y a až pak x.

1.7.2 Negace bodu na eliptické křivce

Pro výpočet $-P$ musíme najít přímku skrz P a O a najít třetí průsečík. Přímka skrz P a O je svislá přímka přes P, takže musíme najít průsečíky $X = x_1$ a křivky. Jinými slovy, musíme vyřešit:

$$Y^2 + (a_1x_1 + a_3)Y(x_1^3 + a_2x_1^2 + a_4x_1 + a_6) = 0. \tag{1.8}$$

Protože víme, že jedno řešení je $Y = y_1$, a také víme, že druhé musí být $-a_1x_1 - a_3 - y_1$. To znamená, $P = (x_1, a_1x_1a_3y_1)$. Jelikož platí, že pokud $K \neq 2$, můžeme křivku přetransformovat tak, že $a_1 = a_3 = 0$, takže abychom našli inverzní P, jednoduše negujeme její souřadnici y [10].

1.7.3 Doubling bodu na eliptické křivce

Abychom našli $P+P = 2P$ (jehož souřadnice určíme ($x_3,\ y_3$)), potřebujeme rovnici tangenty u P. Gradient tečny v bodě $(X,\ Y)$ je dán $(dE/dX)/(dE/dY) = (3X^2 +$

$2a_2X - a_1Y + a_4)/(2Y + a_1X + a_3)$. Takže nastavení:

$$\lambda = \frac{3x_1^2 + 2a_2x_1 - a_1y_1 + a_4}{2y_1 + a_1x_1 + a_3}, \tag{1.9}$$

že tečna u P je daná rovnicí $Y = \lambda X - \lambda x_1 + y_1$. Nahrazením této rovnice za E a vyvrácení koeficientu X^2 dává součet kořenů: $\lambda^2 + \lambda a_1 - a_2$, což znamená, že souřadnice x třetího průsečíku musí být:

$$x_3 = \lambda^2 + \lambda a_1 - a_2 - 2x_1 \tag{1.10}$$

a odpovídající souřadnice y lze nalézt z rovnice tangenty u P, což znamená, že souřadnice x třetího průsečíku musí být:

$$y' = \lambda x_3 - \lambda x_1 + y_1. \tag{1.11}$$

Nakonec musíme vyloučit $(x_3,\ y')$, který je shora [10]:

$$y_3 = -a_1x_3 - a_3 - \lambda x_3 + \lambda x_1 - y_1. \tag{1.12}$$

1.7.4 Addition bodů na eliptické křivce

Předpokládejme, že máme druhý bod $Q = (x2,\ y2)$ jiný než P. Chceme najít $P+Q$, jehož souřadnice budeme naznačovat $(x3,\ y3)$. Pokud $P = -Q$, pak $P + Q = O$. V opačném případě gradient přímky určený P *a* Q je:

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \tag{1.13}$$

a rovnice přímky mezi P a Q je $Y = \lambda X - \lambda x_1 + y_1$. Substitucí do této křivky dostaneme rovnici:

$$(\lambda X - \lambda x_1 + y_1)^2 + (a_1X + a_3)(\lambda X - \lambda x_1 + y_1) = X^3 + a_2X^2 + a_4X + a_6. \tag{1.14}$$

Součet kořenů, tj. negace koeficientu $X2$ je $\lambda^2 + a_1\lambda - a_2$, proto

$$x_3 = \lambda^2 + a_1\lambda - a_2 - x_1 - x_2 \tag{1.15}$$

a odpovídající y souřadnice může být nalezena substitucí do rovnice přímky $P - Q$:

$$y' = \lambda x_3 - \lambda x_1 + y_1. \tag{1.16}$$

Stejně jako předtím musíme vynechat tento třetí bod (x_3, y'), který dává výše [10] :

$$y_3 = -a_1x_3 - a_3 - \lambda x_3 + \lambda x_1 - y_1. \tag{1.17}$$

1.8 Algoritmy multiplication využité v praktické části práce

Zde jsou popsány metody násobení využity v praktické části práce.

1.8.1 Double and add

Nejjednodušší metodou je metoda Double and add. Algoritmus funguje následujícím způsobem. Pro výpočet dP začněte binární reprezentací d [4]:

$d = d_0 + 2d_1 + 2^2d_2 + \cdots + 2^md_m$, kde $d_0 .. d_m \in \{0,\ 1\}$.

Iterační algoritmus, stoupající index:

```
N <- P
Q <- 0
for i from 0 to m do
   if di = 1 then
       Q <- point_add(Q, N)
   N <- point_double(N)
return Q.
```

Iterační algoritmus, klesající index:

```
Q <- 0
for i from m down to 0 do
   Q <- point_double(Q)
   if di = 1 then
       Q <- point_add(Q, P)
return Q.
```

1.8.2 wNAF

Hodnota NAF multiplikátoru d musí být vypočtena nejprve pomocí následujícího algoritmu [4]:

```
i <- 0
  while (d > 0) do
      if (d mod 2) = 1 then
          di <- d mods 2w
          d <- d - di
      else
          di = 0
      d <- d/2
      i <- i + 1
  return (di-1, di-2,..., d0).
```

Kde je funkce *mods* definována jako [4]:

```
if (d mod 2w) >= 2w-1
    return (d mod 2w) - 2w
else
    return d mod 2w.
```

NAF nyní potřebuje provést násobení. Tento algoritmus vyžaduje předběžné výpočty bodů $\{1, 3, 5, \ldots, 2^{w-1} - 1\}P$ a jejich negace, kde P je bod, který má být vynásoben. Následující algoritmus provede násobení dP [4]:

```
Q <- 0
for j <- i - 1 downto 0 do
    Q <- point_double(Q)
    if (dj != 0)
        Q <- point_add(Q, djG)
return Q.
```

1.8.3 Montgomery ladder

Přístup Montgomery ladder vypočítá bodové násobení ve stanoveném čase. Algoritmus používá stejnou reprezentaci jako z double-and-add [4].

```
R0 <- 0
R1 <- P
for i from m downto 0 do
   if di = 0 then
      R1 <- point_add(R0, R1)
      R0 <- point_double(R0)
   else
      R0 <- point_add(R0, R1)
      R1 <- point_double(R1)
return R0.
```

1.9 Diskrétní body na eliptické křivce

Matematické výpočty s body, jejichž souřadnicemi jsou reálná čísla, se nedají dost dobře využít v kryptografických aplikacích, protože počítačové systémy nepracují předvídatelně s reálnými čísly. Co je potřeba, je velká konečná množina diskrétních hodnot stejně jako prvků v „poli“. Pole je sada prvků, které jsou „uzavřeny“ pod operacemi sčítání, odčítání, násobení a dělení, které vyžadujeme pro výpočty s body. Uzavřené znamená, že po použití kterékoliv výše zmíněné operace na jakýkoliv prvek pole je výsledkem také prvek pole. Zvolíme-li pole, zachováme přidanou vlastnost, kterou mají skutečné body na eliptické křivce, ale prvky pole jsou diskrétní a konečné, a proto je možné počítačem manipulovat s opakovatelnými výsledky [6, 11]. Matematické identity (tj. rovnice) jsou zachovány, když prvky pole a operace jsou

nahrazeny pro reálná čísla a odpovídají skutečné operaci. Tímto způsobem je možno vypočítat třetí průsečík přímky, pro který souřadnice ostatních dvou průsečíků jsou pole, a výsledek je také souřadnicemi pole. Z tohoto důvodu je výsledek addition diskrétních bodů předvídatelný a přesný, což je po kryptografickém primitivu vyžadováno [11, 12]. Celočíselné modulo, primárně p, tvoří vhodné pole. Polynomy s binárními koeficienty modulo a neredukovatelný polynom tvoří také vhodné pole. Sada bodů dědí vlastnosti eliptické křivky stejně jako přímky protínající se na třech místech. Takže pro addition dvou bodů stačí provést požadované geometrické výpočty pro doplněk bodu v sadě celých čísel modulo p [12]. Operace modulo p jsou podobné, ale ne zcela stejné, s ekvivalentem a známými operacemi nad celými čísly. Přidání je například stejné, kromě případu, kdy je
výsledek větší nebo roven p, p je odečteno, takže výsledek je v rozsahu:

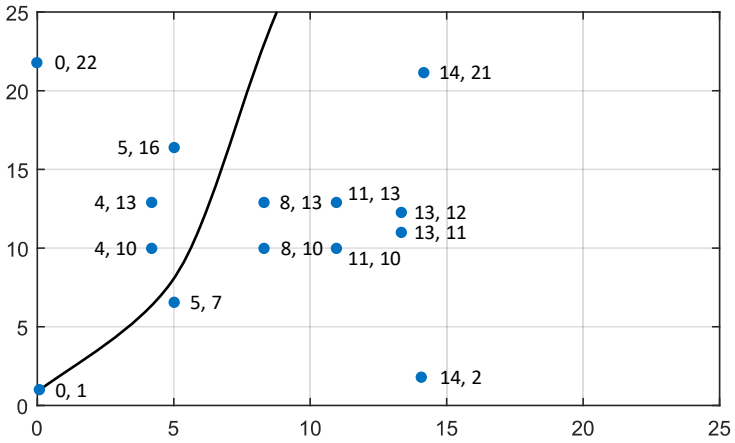
$$\{0...p-1\}. \tag{1.18}$$

Podobně pro odečítání, je-li výsledek odčítání menší než 0, pak p je přidáno, takže výsledek opět spadá do rozsahu (1.18). V případě násobení se dvě čísla vynásobí a pak se vynásobí i zbytek po dělení p [6, 11, 12]. Opět platí, že výsledek spadá do rozsahu (Rov. 1.18). Dělení je trochu komplikovanější. Každé číslo v rozsahu (Rov. 1.18) má tzv. multiplikativní inverzi. Pokud je číslo vynásobeno jeho multiplikativní inverzí, výsledkem je 1 modulo p. Například inverzní rovnicí: $13\ mod\ 35 = 27$, protože: $13 \cdot 27 = 351$, což je: $10 \cdot 35 + 1$. Takže namísto přímého rozdělení jednoho čísla druhým musí být první číslo vynásobeno multiplikativní inverzí druhého čísla: $12/13\ mod\ 35 = 12 \cdot 16\ mod\ 35 = 8\ mod\ 35$. To znamená, že v oblasti celých čísel modulo 23, 12 děleno 13 je výsledek 8. Stejně jako v reálných číslech není definováno dělení nulou. Obr. 1.8 zobrazuje všechny body, které splňují rovnici eliptické křivky:

$$y^2 = x^3 + 3x + y. \tag{1.19}$$

Se souřadnicemi nad celými čísly modulo p, kde p=23. Pro představu, když x=11: $y^2 = (1334 \cdot 11 + 1)\ mod\ 23 = 8$, $y = \sqrt{8} = 10\ nebo\ 13$. Pro výsledky znázorněné rovnicí od $10 \cdot 10\ mod\ 23 = 8, 13 \cdot 13\ mod\ 23 = 8$ do $:13 \cdot 13\ mod\ 23 = 8$. Průběžná čára na Obr. 1.8 je původní křivka přes tuto oblast roviny x-y. Lze si všimnout, že s výjimkou bodu (0, 1) žádný z bodů není shodný s řešením rovnice s reálnými čísly v této oblasti, ani v jiné libovolné oblasti křivky. Skutečností také je, že ne každá hodnota x má řešení; pro hodnoty x není pravá strana rovnice křivky dokonalý čtverec. Dále si lze všimnout, že body tvoří páry uspořádané symetricky kolem vodorovné čáry:

$$y = p/2, \tag{1.20}$$



Obr. 1.8: Diskrétní body na eliptické křivce.

takže jich musí být sudý počet. Je to proto, že pro každé x, pro které má rovnice řešení, na křivce existují dvě řešení: x, x a přijaté modulo p:

$$-y = p - y. \tag{1.21}$$

V následujících kapitolách budou použita velká písmena pro body na křivce a normální malá písmena pro označení celých čísel.

1.10 Příklad eliptické křivky v kryptografii

Křivka, která byla použita pro účely názorné prezentace (a=3, b=1), je samozřejmě moc triviální pro použití v reálných kryptografických aplikacích. Pro srovnání, zde jsou reálné parametry křivky: P-256, křivka specifikována NIST [14, 15].

$a = -3$,

$b = 5ac633d8\ aa3a92e7\ b2ebbd55\ 759886bc\ 652d06b0\ cc54b0f6\ 3bce4c3e\ 28d2604b$,

$p = 115792089210356248762697446949407573430086143415290314195533631308867097853951$.

Souřadnice x a y a základního bodu jsou:

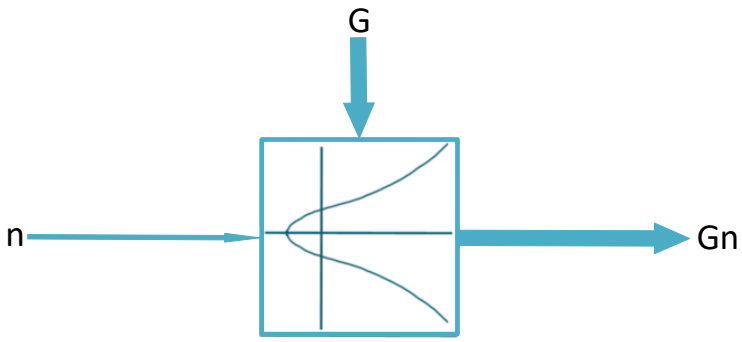
$xg = 6b18d1f2e12c4257f8bce7e577047d812deb43a0f5a13945d897c296$,

$yg = 4fe343e2fe2a7f9b8ee6eb4a2bce43576b325ececbb7406838bf51f5$,

p je vyjádřeno jako číslo v desítkové soustavě, zatímco a a b jsou vyjádřeny v šestnáctkové soustavě. Aby bylo zajištěno, že vědci záměrně nepoužili křivku s těžko odhalitelnou slabinou, byla zveřejněna v technické dokumentaci kryptografická funkce použitá při výpočtu parametrů křivky – parametr b a vstup do funkce [15].

1.11 Kryptografická funkce založená na eliptických křivkách

Z výše uvedeného je možno vidět, jak body na eliptické křivce se souřadnicemi přes pole prvočíselného modula p mohou tvořit grupu a jak může být grupa použita pro rekonstrukci kryptografické funkce vhodné pro použití v kryptografických aplikacích. Graficky je možné znázornit kryptografickou funkci jako na Obr. 1.9. Vstup G a výstup Gn představují body křivky a vstup n představuje celá čísla.



Obr. 1.9: Kryptografická funkce založená na eliptických křivkách [16].

Zatímco je snadné vypočítat Gn dané n a G, je výpočetně nemožné vypočítat n dané G a Gn. Toto jsou důležité aspekty, které se uplatňují v praxi při používání kryptografie s eliptickou křivkou a budování silných mechanizmů [17].

2 REŠERŠE KNIHOVEN ZABÝVAJÍCÍCH SE KRYPTOGRAFIÍ ELIPTICKÝCH KŘIVEK

2.1 OpenSSL

Knihovna OpenSSL EC (Open Secure Sockets Layer Eliptic Curves) se zabývá kryptografií s eliptickou křivkou. Tato knihovna poskytuje rozsáhlou sadu funkcí pro provádění operací na eliptických křivkách s konečnými poli. Spadá pod projekt nazývající se OpenSSL, ten je zaměřen na implementaci protokolu SSL (Secure Sockets Layer). SSL je protokol, resp. vrstva mezi transportní a aplikační vrstvou, která poskytuje zabezpečení komunikace šifrováním a autentizaci komunikujících stran. Následovníkem SSL je protokol TLS (Transport Layer Security). Dále se práce bude zabývat již pouze knihovnou OpenSSL EC. OpenSSL EC poskytuje podporu pro kryptografii eliptických křivek. Je základem pro implementaci OpenSSL ECDSA a ECDH. Primárním zaměřením je tedy implementace těchto algoritmů na webový server. Dovoluje však i primitivní operace s eliptickými křivkami. Kód z balíku OpenSSL je možno použít pod licencí OpenSSL Licence, kdy za těchto podmínek je možné knihovnu použít či šířit dál v modifikované formě či bez ní. Celá licence v originálním znění je k dispozici na webových stránkách: [31]. Tato knihovna poskytuje rozsáhlou řadu funkcí pro provádění operací na eliptických křivkách s konečnými poli. Implementace knihovny OpenSSL EC umožňuje [32]:

- Podpora eliptických křivek: 2^m:
  - Standardizované křivky NIST (SECP, SECT), BRAINPOOL
  - SECP: 112R1/R2, 128R1/R2, 160K1/R1/R2, 224K1/R1, 256K1/R1, 521R1.
  - SECT: 113R1/R2, 131R1/R2, 163K1/R1, 163R2, 193R1/R2, 233K1/R1, 239K1, 283K1/R1, 409K1/R1, 571K1/R1.
  - BRAINPOOL: P160T1/R1, P192T1/R1, P224T1, P224R1, P256T1/R1, P320T1/R1, P384T1/R1, P512T1/R1.
- Metody pro eliptické křivky nad konečnými poli a poli 2^m:
  - Multiplication, addition, negace, doubling.
- Operace grupy:
  - Vytvořit novou nebo zkopírovat obsah stávající grupy,
  - uvolnit nebo smazat, nastavit generátor, order, kofactor,
  - nastavit jméno, nastavení a vrácení parametrů eliptické křivky p, a, b, ctx, *pro křivky* ($y^2 = x^3 + a \cdot x + b$),
  - nastavení a vrácení parametrů eliptické křivky 2^m p, a, b, ctx, *pro křivky* ($y^2 = x^3 + a \cdot x + b$),

  - zkontrolovat validitu, spočítat diskriminant, nalézt všechny křivky.
- Operace s bodem na eliptické křivce:
  - Vytvořit nový, kopírovat, uvolnit, smazat bod,
  - nastavit tzv. identity element (bod nekonečna),
  - podpora souřadnic typu jacobian, affine, x9.62 zkomprimované,
  - kóduje a dekóduje bod z a k oktetovému řetězci.
- Aritmetika s bodem na eliptické křivce:
  - Addition dvou bodů, doubling bodu, negace bodu, multiplication bodů,
  - zkontrolovat, zda je neutrálním prvkem skupiny,
  - zkontrolovat, zda je bod na křivce, porovnat dva body,
  - vypočítat $r = generator \cdot n\ sum_{i=0}^{num-1} p[i] \cdot m[i]$,
  - vypočítat r = generátor $\cdot\ n + q \cdot m$,
  - násobky generátoru pro rychlejší násobení bodu.
- Práce s klíčem:
  - Vytvořit, uvolnit, zkopírovat nebo smazat,
  - vytvořit nový pomocí pojmenované křivky jako podkladové,
  - zvýšit interní referenční počet klíčů,
  - vrátit, nastavit soukromý klíč objektu,
  - vytvořit nový privátní klíč,
  - ověřit platnost soukromého nebo veřejného klíče,
  - nastavit veřejný klíč od afinních kořenů.
- Kódování a dekódování:
  - Umožňuje kódovat a dekódovat většinu parametrů,
  - kódovat a dekódovat znění klíče.

2.2 libecc

Tento software implementuje kryptografickou knihovnu založenou na eliptických křivkách (ECC). API podporuje podpisové algoritmy uvedené v normě ISO 14888-3: 2016 s následujícími specifickými křivkami a hashovými funkcemi:

- Podpisy: ECDSA, ECKCDSA, ECGDSA, ECRDSA, EC , SD, ECFSDSA.
- Hash funkce: Hash funkce SHA-2 a SHA-3 (224, 256, 384, 512).

Pokročilé využití této knihovny zahrnuje také možnost implementace protokolů Diffie – Hellman založených na eliptických křivkách, stejně jako jakýkoli algoritmus na vrcholových polích založených na eliptických křivkách (nebo primární pole celých čísel). Ve srovnání s jinými kryptografickými knihovnami poskytujícími takové funkce jsou rozlišujícími body:

- Zaměření na čitelnost a auditovatelnost kódu. Kód je čistý C99, bez dynamického přidělení a obsahuje pre / post-asserts v kódu. Proto je tato knihovna

dobrým kandidátem pro vložené cíle (měla by být snadno přenosná přes různé platformy).

- Čistá separace vrstev pro všechny potřebné matematické abstrakce a operace. V každé vrstvě se používala silná písma (C99) matematických objektů.
- Knihovna NENÍ navržena k tomu, aby trhala rekordy ve výkonu, přestože odvádí dobrou práci. Podobně s využitelností paměti při práci s knihovnou (pokud jde o použití ROM a RAM) není nejmenší dosažitelná (i když bylo vynaloženo určité úsilí na omezení a běžné platformy.
- Jádro knihovny libecc nemá žádnou externí závislost (ani standardní knihovnu libc), která by byla přenositelná. Další informace naleznete v části o přenositelnosti.

Implementace knihovny libecc umožňuje [33]:

Podpora eliptických křivek: 2^m:
  - Standardizované křivky NIST (SECP), BRAINPOOL, GOST, FRP.
  - SECP: 192R1, 224R1, 256R1, 384R1, 521R1.
  - BRAINPOOL: P224R1, P256R1, P384R1, P512R1.
  - GOST: 256,512.
  - FRP: 256V1.
- Metody pro eliptické křivky nad konečnými poli:
  - Montgomeryho metoda addition, doubling, multiplication,
  - skalární metoda addition, doubling, multiplication.
- Operace s bodem na eliptické křivce:
  - Vytvořit nový, kopírovat, uvolnit, smazat bod,
  - nastavit tzv. identity element (bod nekonečna),
  - podpora souřadnic typu jacobian, affine, x 9.62 zkomprimované,
  - kóduje a dekóduje bod z a k oktetovému řetězci.
- Aritmetika s bodem na eliptické křivce:
  - Spočítat adition dvou bodů, doubling bodu,
  - multiplication bodů, vypočítat inverzní bod,
  - zkontrolovat, zda je neutrálním prvkem skupiny,
  - zkontrolovat, zda je bod na křivce, porovnat dva body,
  - násobky generátoru pro rychlejší násobení bodu,
  - vypočítat $r = generator \cdot n\ sum_{i=0}^{num-1} p[i] \cdot m[i]$,
  - vypočítat r = generátor $\cdot\ n + q \cdot m$.

2.3 TinyECC

TinyECC (Tiny Eliptic Curves) je kryptografická knihovna umožňující mj. šifrování veřejných klíčů za pomocí eliptických křivek. Poskytuje schéma digitálního podpisu ECDSA (Elliptic Curve Digital Signature Algorithm), protokol výměny klíčů ECDH (Elliptic Curve Diffie-Hellman) a schéma šifrování veřejných klíčů ECIES (Elliptic Curve Integrated Encryption Scheme). TinyECC používá řadu optimalizačních přepínačů, které mohou na základě potřeb vývojáře zapnout nebo vypnout specifické optimalizace. TinyECC 2.0 je primárně určeno pro senzorové platformy se systémem TinyOS 2.x. Aktuální verze je implementována v nesC (The National Electrical Safety Code) s dalšími optimalizačními platformami pro oblíbené platformy senzorů. TinyECC 2.0 podporuje standardy SECG (Standards for Efficient Cryptography Group) doporučené parametry pro 128 bitové, 160 bitové a 192 bitové eliptické křivky. Implementuje 3 výše zmíněné algoritmy eliptických křivek [18]. Další funkce knihovny jsou vypsány v následujícím seznamu:

- Barrettova redukce – je alternativní metodou pro modulární redukci [27]. Převádí redukční modulo libovolného celého čísla na dva násobky a několik redukcí modulo celých čísel formy 2^n.
- Projektivní souřadnicové systémy – pro ATmega128 neexistuje operace dělení, takže inverzní operace je výrazně náročnější než násobení. Je efektivnější uvádět eliptické křivky v projekčních souřadnicích místo afinních. V TinyECC je použito vážené projektivní reprezentace (Jacobian representation) [19] pro zrychlení přidání bodů, zdvojnásobení bodů a násobení skalárních bodů.
- Optimalizace křivek – pro všechny křivky standardu NIST (National Institute of Standards and Technology) a většinu křivek standardu SECG byly pro pseudo – Mersenne primitiva zvolena podkladová pole p, která umožňují optimalizovanou modulární redukci [20]. Byl implementován tento optimalizovaný modulární redukční algoritmus pro urychlení modulárního násobení a modulového čtverce.
- Metoda posuvného okna – byla zavedena pro urychlení násobení skalárních bodů. Tradiční metoda, jak provádět násobení skalárních bodů, je binární metoda. Binární metoda skenuje bity skalárního n zleva doprava, jeden po druhém. V každém kroku se provede zdvojnásobení bodu. V závislosti na hodnotách skenovaných bitů se provede následné přidání bodu. Metoda posuvného okna [21] skenuje k hodnoty. Zdvojení bodů se provádí k časům v každém kroku, v závislosti na naskenované k hodnoty bitů se provádí následným přidáním bodu. Musíme předběžně vypočítat všechny přidané body, což je výsledek možných hodnot k bitů vynásobením základního bodu. Metoda posuvného okna může urychlit násobení skalárního bodu snížením celkového počtu bodo-

vých přírůstků, ale je zapotřebí více paměti.

- Inline sestavení a hybridní násobení – operace s přirozeným číslem v TinyECC jsou založeny na implementaci v RSAREF (RSA reference) 2.0 [24]. Operace s přirozeným číslem v systému RSAREF 2.0 jsou nezávislé na platformě, ale nejsou efektivní. Byl použit inline kód [22, 28, 28] pro urychlení kritických operací, jako je násobení a násobení pro MICAz, TelosB / Tmote Sky a Imote2 motes. Zejména hybridní násobení v [20] je implementováno v inline sestavě, což může účinně urychlit TinyECC.
- Optimalizace pro modulární doplnění a modulární odečítání [23], abychom urychlili modulární sčítání a modulární odečítání.
- Shamirův trik – abychom zkrátili dobu ověření podpisu [23].

2.4 WolfSSL

Knihovna WolfSSL (dříve CyaSSL) vytvořená roku 2004 Larrym Stefonicem a Toddem Ouskem. Rozhodli se vytvořit knihovnu SSL s otevřeným kódem a dvojí licencí, jelikož na trhu nic v té době nebylo. OpenSSL bylo v té době dostupné, avšak trh vyžadoval alternativu přenositelnou, menší, rychlejší a dostupnou pod jasnou komerční licencí pro komerční vývojáře. WolfSSL je kompatibilní s OpenSSL, MySQL, OpenWRT, Mongoose, cURL a Ubuntu. V současnosti ji používá 2 miliardy připojení. WolfSSL je knihovna založená na jazyce C zaměřená na embedded a RTOS prostředí, především díky malé velikosti, rychlosti, přenositelnosti a funkcím [34]. Použití knihovny WolfSSL ECC je svobodné, lze ji svobodně šířit nebo měnit za podmínek GNU (General Public License), kterou zveřejnila nadace pro svobodný software. Celé znění této licence je dostupné z webu [46]. Funkce WolfSSL s ECC [35]:

- Typ eliptických křivek:
  - Umožňuje využít podporu eliptické křivky standardu SECP R2, R3,
  - umožňuje využít podporu eliptické křivky Brainpool, Koblitz.
- Velikost eliptických křivek:
  - 112, 128, 160, 192, 224, 239, 256, 320, 512, 521 bit klíč.
- Nastavení eliptických křivek:
  - Velikost v bitech, ID, jméno, prime, A, B, order, Gx, Gy, Oid, oid sum, cofactor.
- Práce s body na eliptické křivce:
  - Addition dvou bodů, doubling bodu, negace bodu, multiplication bodů,
  - namapovat bod z jakobského souřadnicového systému do affiního souřadnicového systému,
  - alokovat nový bod, uvolnit, zkopírovat, porovnat, zkontrolovat validitu.
- Práce s eliptickou křivkou:

– Vytvořit novou, zkontrolovat validitu, podpis.

2.5 AvrCryptoLib

Knihovna AvrCryptoLib je sada implementací různých kryptografických primitiv. Kvůli individuálním omezením mikrokontrolerů (velmi málo místa, RAM a Flash pohybující se od několika bajtů po několik KiB) neboť normální neoptimalizované implementace nejsou využitelné. Proto se snaží poskytovat speciální implementace, které respektují extrémně omezené zdroje aplikací mikrokontrolerů. Avšak nemá přímo algoritmy pro práci s eliptickými křivkami jako takovými. Proto nebude v této práci zahrnuta. Tato knihovna podporuje širokou škálu šifer a hašovacích funkcí. Zde je seznam podporovaných [36, 37]:

- Blokové šifry:
  – AES, XTEA, CAST5, Camellia, Skipjack, Noekeon, RC5, RC6, SEED, DES, TDES (EDE-DES, 3DES), SHABEA, Serpent, Present.
- Proudové šifry:
  – ARCFOUR (RC4), Trivium, Grain, MUGI.
- Hašovací funkce:
  – Blake, BlueMidnightWish, Grøstl, MD5, SHA-256, SHA-1, SHA-3 (Keccak), SHABAL, Skein, Twister, Whirlpool.
- Ostatní:
  – HMAC-SHA256 (RFC 2104), PRNG.

Je naprogramována převážně v jazyce C a assembleru (ATmega, gnu-avr-as). Její licence je otevřená až na jeden patent u šifrovacího algoritmu MD5 (US patent 5,724,428).

2.6 WiseLib

WiseLib je knihovna algoritmů primárně určená pro síťová zařízení. Obsahuje různé třídy algoritmů (například lokalizační nebo směrovací), které lze implementovat do platforem (iSense, Contiki nebo simulátor síťového provozu Shawn). Knihovna je psána v jazyce C++ a používá šablony stejným způsobem jako Boost a CGAL. To umožňuje napsat obecný a nezávislý kód, který je velmi efektivně kompilován pro širokou škálu platforem. Knihovna WiseLib je šířena volně pod licencí GNU. Knihovna podporuje tyto funkce s eliptickou křivkou [38]:

- Operace s bodem:
  – Nastavit P_0 na x a y na 0, nastavit P_0=P_1, ověřit, zda x a y z P_0 je nula,
  – ověřit, zda jsou body P_1 a P_2 stejné, převést bod na oktet a obráceně,

  - ověřit, zda je bod na eliptické křivce.
- • Funkce eliptické křivky:
  - Skalární násobení na eliptické křivce,
  - vytvořit soukromý klíč pomocí náhodného seedu,
  - vygenerovat veřejný klíč vynásobením soukromého klíče bodem G,
  - inicializovat 128 bitovou eliptickou křivku přes F_p,
  - inicializovat 160 bitovou eliptickou křivku přes F_p,
  - inicializovat 192 bitovou eliptickou křivku přes F_p.

2.7 Crypto ++

Knihovna Crypto ++ je volně šiřitelná knihovna napsaná v jazyce C ++, která se zabývá kryptografií. Knihovna obsahuje následující funkce [39]:

- Ověřené šifrovací schémata (GCM, CCM, EAX, OCB).
- Vysokorychlostní proudové šifry (ChaCha (ChaCha8 / 12/20), Panama, Sosemanuk, Salsa20, XSalsa20).
- AES, AES kandidáty (AES (Rijndael), RC6, MARS, Twofish, had, CAST-256).
- Blokové šifry (ARIA , IDEA, Triple DES (DES-EDE2 a DES-EDE3), Camellia, SEED, Kalyna, RC5, Blowfish, TEA, Trifish, Skipjack, SHACAL-2, XTEA).
- Módy blokových šifer pro blokování (ECB, CBC, CBC kódování krádeží (CTS), CFB, OFB, režim čítače (CTR)).
- Autentizační kodéry (VMAC, HMAC, GMAC (GCM), CMAC, CBC-MAC, DMAC, Two-Track MAC, BLAKE2 (BLAKE2b, BLAKE2s) , Poly1305, SipHash).
- Hašovací funkce (BLAKE2 (BLAKE2b, BLAKE2s), Keccack (F1600), SHA-1 , SHA-2, SHA-3, Tiger, WHIRLPOOL, RIPEMD-128, RIPEMD-256, RIPEMD-160).
- Kryptografie s veřejným klíčem (RSA, DSA , Determinsitic DSA, ElGamal, Nyberg-Rueppel (NR), Rabin-Williams (RW), Německý digitální podpis ECGDSA, LUC, LUCELG,).
- Padding systémy s veřejným klíčem (PKCS 1 v2.0, OAEP, PSS, PSSR, IEEE P1363 EMSA2 a EMSA5).
- Šifrovací standardy (Diffie-Hellman (DH), Unified Diffie-Hellman (DH2), Menezes-Qu-Vanstone (MQV), Hashed MQV (HHQQV), LUCDIF, XTR-DH).
- ECC kryptografie (ECDSA, Deterministický ECDSA, ECGDSA, ECNR, ECIES, ECDH, ECMQV).
- Starší podporované algoritmy (MD2, MD4, MD5, Panama Hash, DES, ARC4, SEAL 3.0, WAKE-OFB, DESX (DES-XEX3), RC2, SAFER, 3-WAY, GOST,

SHARK, CAST-128).

- Knihovna obsahuje také další funkce jako jsou generátory pseudonáhodných čísel, benchmarky a testování validace.

Tato knihovna nepodporuje práci s kryptografickými primitivy. Z tohoto důvodu nebude knihovna implementována.

2.8 LibTomCrypt

Projekt LibTomCrypt jsou otevřené knihovny napsané v jazyce C pod WTFPL (Do What the Fuck You Want To Public License). Knihovny podporují různá kryptografická a algebraická primitiva, které umožňují vývojářům mnohem efektivněji pracovat v oblasti kryptografie. V současné době se projekty skládají ze tří významných knihoven LibTomCrypt, LibTomMath
a TomsFastMath.

LibTomCrypt je komplexní, modulární a přenosná kryptografická sada nástrojů, která poskytuje řadu známých publikovaných blokových šifer, hašovacích funkcí, řetězcových algoritmů, generátorů pseudonáhodných čísel, kryptografii s veřejným klíčem a řadu dalších funkcí. LibTomCrypt byla navržena tak, aby byla velmi jednoduchá. Má modulární a standardní rozhraní API, které umožňuje přidávat nebo odstraňovat nové šifry, haš a PRNG bez změny na koncové aplikaci. Obsahuje snadno použitelné funkce a kompletní uživatelskou příručku, která obsahuje mnoho příkladů zdrojových úseků. LibTomCrypt je zdarma pro všechny účely pod veřejnou doménou. To zahrnuje komerční využití, přerozdělení i větvení. Obsahuje tyto implementace [40, 41]:

- Blokové šifry:
  - AES, Anubis, Blowfish, CAST5, DES, 3DES, 3DES, KASUMI, Khazad, Multi2, Noekeon, RC2, RC5, RC6, K64, SK64, K128, SK128, SAFER +, SEED, Skipjack, Twofish, XTEA.
- Proudové šifry:
  - ChaCha, RC4, SOBER-128.
- Řetězové algoritmy:
  - CBC, CFB, CTR, ECB, F8, LRW, OFB, XTS.
- Hašovací funkce:
  - Blake2b (160/256/384/512), Blake2s (128/160/224/256), MD2/4/5, RIPE-MD (128/160/256/320), SHA-1, SHA-2 (224/256/384/512/512-224/512-256), SHA-3 (224/256/384/512), SHA-3-SHAKE, TIGER-192, WP.
- Autentizační algoritmy:
  - Blake2b (160/256/384/512), Blake2s (128/160/224/256), MD2, MD4,

MD5, RI-PE-MD (128/160/256/320), SHA-1, SHA-2 (224/256/384/512/512-224/512-256), SHA-3 (224/256/384/512), SHA-3-SHAKE, TIGER-192, WP.

- Autentizační algoritmy:

  – Blake2b MAC, Blake2s MAC, CMAC, F9 MAC, HMAC, PMAC Authentication, Pelican MAC, Poly1305 MAC, XCBC MAC.
- Autentizační módy:

  – CCM Mode (NIST spec), ChaCha20-Poly1305 (IETF spec RFC7539), EAX Mode, GCM Mode (IEEE spec), OCB Mode v1, OCB Mode v3 (IETF spec RFC7253).
- Pseudonáhodné generátory čísel:

  – ChaCha20, Fortuna, RC4, SOBER-128, Yarrow, podpora pro /dev/random, /dev/urandom a Win32 CSP RNG.
- Algoritmy s veřejným klíčem:

  – RSA, ECC (EC-DSA X9.62 signatures, X9.63 EC-DH), DSA, DH.
- Ostatní standardy:

  PKCS 1, PKCS 5 a další.

2.9 MIRACL

MIRACL (Multiprecision Integer and Rational Arithmetic Cryptographic Library) – je softwarová knihovna napsaná v jazyce C, open source SDK pro kryptografii eliptických křivek.

Knihovna MIRACL se skládá z více než 100 implementací, které pokrývají všechny aspekty aritmetiky. Obsahuje dva nové datové typy – velký pro velká celá čísla a blikající pro velká racionální čísla. Velké celočíselné rutiny jsou založeny na algoritmu Knutha. Plovoucí lomítko, které pracuje se zaoblenými frakcemi, původně navrhli D. Matula a P. Kornerup. Všechny implementace byly důkladně optimalizovány pro rychlost a efektivitu a současně zůstaly standardní, přenosné, napsané v jazyce C. Nicméně jsou k dispozici volitelná alternativní jazyková řešení rychlého sestavení pro určité kritické rutiny, zejména pro populární řadu procesorů Intel 80x86. K dispozici je také rozhraní C++ [42].

Knihovna neumožňuje moc detailní nastavení kryptografických primitiv. Podporuje pouze následující v omezených možnostech [43]:

- Nastavit parametry křivky standardu NIST P256.
- Inicializovat doménovou strukturu EC GF (p).
- Vypočítat veřejný a privátní klíč EC GF (p).
- Validovat veřejný klíč EC GF (p).
- Nastavení primitiv standardu P1363 EC GF (p).

3 VYBRANÉ ELIPTICKÉ KŘIVKY, KNIHOVNY A HARDWAROVÉ PLATFORMY

V této kapitole jsou popsány eliptické křivky vybrané v praktické části práce. Popsána hardwarová zařízení pro implementace kryptografických knihoven k následnému testování kryptografických primitiv. Byla také popsána instalace a implementace knihoven na jednotlivá zařízení.

3.1 Vybrané eliptické křivky

V praktické části práce bylo měřeno na těchto vybraných eliptických křivkách. Eliptické křivky byly zvoleny dle doporučených standardů společností NIST [44] , BRAINPOOL [45], GOST a FRP. Všechny křivky a jejich použité parametry jsou dostupné viz příloha A.

3.1.1 Standard NIST

Eliptické křivky od společnosti NIST se dělí na eliptické křivky standardů SECP nad $\mathbb{F}_p$ a SECT nad $\mathbb{F}_{2^m}$. Tyto se dále dělí na k-křivky a r-křivky. K-křivky jsou přidružené Koblitzovy křivky. Např. doménové parametry eliptické křivky nad $\mathbb{F}_{2^m}$ asociované jako Koblitzova křivka SECT233K1 jsou specifikovány $T = (m,\ f(x),\ a,\ b,\ G,\ n,\ h)$, kde $m = 233$ a reprezentace $\mathbb{F}_{2^{233}}$ je definována jako: $f(x) = x^{233} + x^{74} + 1$ [44].

3.1.2 Standard BRAINPOOL

Další měřené křivky a jejich doporučená parametrizace byly doporučeny společností BRAINPOOL, které se dělí na BRAINPOOL_T (twisted) a BRAINPOOL_R (random). Bod p specifikuje základní pole. Bod A a B jsou koeficienty rovnice $y^2 = x^3 + A * x + B$. mod$p$ definuje eliptickou křivku. $G = (x,\ y)$ je základní bod, tj. Bod E v primárním pořadí, přičemž x a y jsou jeho souřadnice x a y. q je primární pořadí skupiny vytvořené G. h je kofaktor G v E, tj. $E(GF(p))/q$. Pro tzv. twisted křivku BRAINPOOL_T křivku udáváme také koeficient Z, který definuje izomorfismus F. Křivky musí být nad $\mathbb{GF}_p$ -izomorfní ke křivce $E' : y^2 = x^3 + A' * x + B' \bmod p\ s\ A' = -3 \bmod p$. Tato vlastnost dovoluje použití aritmetických výhod křivek s $A = -3$. Pro $p = 3 \bmod 4$, přibližně polovina tříd izomorfizmu eliptických křivek nad $\mathbb{GF}_p$ obsahuje křivku $E'\ s\ A' = -3 \bmod p$. Přesněji, pokud je křivka $E : y^2 = x^3 + A * x + B \bmod p\ s\ -3 = A * u^4$ řešitelná v $\mathbb{GF}_p$ a $u = Z$ je řešení této rovnice, pak je požadavek splněn pomocí kvadratického

zkroucení $E' : y2 = x^3 + Z^4 * A * x + Z^6 * B \bmod p$ a $\mathbb{GF}_p$ -isomorfismus udává $F(x, y) := (x * Z^2, y * Z^3)$ [45].

3.1.3 Přehled použitých eliptických křivek

Seznam použitých křivek v praktické části práce (viz Tab. 3.1). Parametrizace všech křivek se nachází v příloze A. Křivky GOST256, GOST512 a FRP256V1 jsou interní křivky knihovny OpenSSL a nemohly být implementovány do ostatních knihoven.

Tab. 3.1: Přehled použitých eliptických křivek v praktické části práce

SECP	SECT	BRAINPOOL	GOST	FRP256V1
SECP112R1	SECT113R1	n/a	n/a	n/a
SECP112R2	SECT113R2	n/a	n/a	n/a
SECP128R1	SECT131R1	n/a	n/a	n/a
SECP128R2	SECT131R2	n/a	n/a	n/a
SECP160K1	SECT163K1	BRAINPOOLP160T1	n/a	n/a
SECP160R1	SECT163R1	BRAINPOOLP160R1	n/a	n/a
SECP160R2	SECT163R2	n/a	n/a	n/a
SECP192K1	n/a	BRAINPOOLP192T1	n/a	n/a
n/a	SECT193R1	BRAINPOOLP192R1	n/a	n/a
n/a	SECT193R2	n/a	n/a	n/a
SECP224K1	SECT233K1	BRAINPOOLP224T1	n/a	n/a
SECP224R1	SECT233R1	BRAINPOOLP224R1	n/a	n/a
n/a	SECT239K1	n/a	n/a	n/a
SECP256K1	n/a	BRAINPOOLP256T1	GOST256	FRP256V1
n/a	n/a	BRAINPOOLP256R1	n/a	n/a
n/a	SECT283K1	n/a	n/a	n/a
n/a	SECT283R1	n/a	n/a	n/a
n/a	n/a	BRAINPOOLP320T1	n/a	n/a
n/a	n/a	BRAINPOOLP320R1	n/a	n/a
n/a	n/a	BRAINPOOLP384T1	n/a	n/a
SECP384R1	n/a	BRAINPOOLP384R1	n/a	n/a
n/a	SECT409K1	n/a	n/a	n/a
n/a	SECT409R1	n/a	n/a	n/a
n/a	n/a	BRAINPOOLP512T1	GOST512	n/a
SECP521R1	n/a	BRAINPOOLP512R1	n/a	n/a
n/a	SECT571K1	n/a	n/a	n/a
n/a	SECT571R1	n/a	n/a	n/a

3.2 Vybrané knihovny a jejich implementace

Tato kapitola se zabývá výběrem a implementací kryptografických knihoven využitých v praktické části práce.

3.2.1 Implementace knihovny OpenSSL

OpenSSL používá ke konfiguraci knihovny vlastní systém sestav. Konfigurace umožní knihovně nastavit rekurzivní *makefile*. Jakmile je nakonfigurován, je třeba vytvořit knihovnu pomocí příkazu:

```
make.
```

Pro vytvoření knihovny OpenSSL je třeba použít kompilátor C. Po konfiguraci a sestavení knihovny by se měl vždy provést test, který zajistí, že knihovna je správně sestavená.

Pomocí příkazu:

```
./configure
```

je možno nakonfigurovat proces kompilace a instalace pomocí možností a přepínačů. *Configure* správně zpracovává triplet host-arch-compiler a *config* ne. *config* se pokusí odhadnout triplet. Je možno použít přednastavený konfiguraci na operační systémy Ubuntu, Debian:

```
./config
Operating system: x86_64-whatever-linux2
Configuring for linux-x86_64
Configuring for linux-x86_64
    no-ec_nistp_64_gcc_128 [default]
    OPENSSL_NO_EC_NISTP_64_GCC_128 (skip dir)
    no-gmp          [default]
    OPENSSL_NO_GMP (skip dir)
    no-jpake        [experimental]
    OPENSSL_NO_JPAKE (skip dir)
    no-krb5         [krb5-flavor not specified]
    OPENSSL_NO_KRB5
    ...
```

Po konfiguraci knihovny je třeba spustit *make*. A zkompilovat například pomocí:

```
./config <options ...>  -- openssldir=/usr/local/ssl
make
make test
sudo make install
```

Na závěr je třeba ověřit testem správnost sestavení OpenSSL, který rozpoznává jak *OPENSSL_init_ssl*, tak i *SSL_library_init*:

```
if test "$with_openssl" = yes ; then
  dnl Order matters!
  if test "$PORTNAME" != "win32"; then
     AC_CHECK_LIB(crypto, CRYPTO_new_ex_data, [],
     [AC_MSG_ERROR
     ([library 'crypto' is required for OpenSSL])])
     FOUND_SSL_LIB="no"
     AC_CHECK_LIB(ssl, OPENSSL_init_ssl,
     [FOUND_SSL_LIB="yes"])
     AC_CHECK_LIB(ssl, SSL_library_init,
     [FOUND_SSL_LIB="yes"])
     AS_IF([test "x$FOUND_SSL_LIB" = xno],
     [AC_MSG_ERROR
     ([library 'ssl' is required for OpenSSL])])
  else
     AC_SEARCH_LIBS
     (CRYPTO_new_ex_data, eay32 crypto, [],
     [AC_MSG_ERROR
     ([library 'eay32' or 'crypto' is required for OpenSSL])])
     FOUND_SSL_LIB="no"
     AC_SEARCH_LIBS
     (OPENSSL_init_ssl, ssleay32 ssl, [FOUND_SSL_LIB="yes"])
     AC_SEARCH_LIBS
     (SSL_library_init, ssleay32 ssl, [FOUND_SSL_LIB="yes"])
     AS_IF([test "x$FOUND_SSL_LIB" = xno],
     [AC_MSG_ERROR
     ])
  fi
fi
```

3.2.2 Implementace knihovny libecc

V hlavní složce se nachází soubor *makefile*. Ten spustíme příkazem:

```
make
```

Tímto se ve vlastním adresáři knihovny zkompilují různé sestavení – tři statické knihovny, z nichž každá obsahuje (na základě) předchozí:

- libarith.a: tato knihovna obsahuje aritmetické vrstvy přirozených čísel a konečných polí přes prvočísla.
- libec.a: tato knihovna je založena na libarith.a a obsahuje implementaci eliptických křivek (abstrakce bodů, bodová addition / doubling a skalární multiplication).
- libsign.a: tato knihovna je založena na libec.a a obsahuje všechny podporované podpisové algoritmy ISO 14888-3 přes některé staticky definované křivky

a hashové funkce.

- Dva binární soubory založené na statické knihovně libsign.a: ec_self_tests: autotesty pro podpis / ověřovací algoritmus ISO 14888-3 se známými a náhodnými zkušebními vektory, stejně jako testy výkonu. Spuštění testů bez argumentu provede tři testy (známé a fixní zkušební vektory, náhodný podpis, ověřovací kontroly a měření výkonu). Každý z testů lze spustit i samostatně.

Pro známé testovací vektory:

```
$ ./build/ec_self_tests vectors
  ======= Known test vectors test ===================
  [+] ECDSA-SHA224/secp224r1 selftests: known test vectors
  sig/verif ok
  [+] ECDSA-SHA256/secp256r1 selftests: known test vectors
  sig/verif ok
  [+] ECDSA-SHA512/secp256r1 selftests: known test vectors
  sig/verif ok
  ...
```

Pro podpis a ověření pravosti (s náhodnými klíči a náhodnými daty):

```
$ ./build/ec_self_tests rand
  ======= Random sig/verif test ====================
  [+]  ECDSA-SHA224/FRP256V1 randtests: random import/export
  with sig(0)/verif(0) ok
  [+] ECDSA-SHA224/SECP224R1 randtests: random import/export
  with sig(0)/verif(0) ok
  ...
```

Pro měření výkonu:

```
$ ./build/ec_self_tests perf
  ======= Performance test =====================
  [+]  ECDSA-SHA224/FRP256V1 perf: 462 sign/s and 243 verif/s
  [+] ECDSA-SHA224/SECP224R1 perf: 533 sign/s and 276 verif/s
  ...
```

3.3 Vybrané hardwarové platformy a jejich instalace

Zde budou představeny hardwarové platformy vybrané k měření, jejich instalace a implementace knihoven, následně měření kryptografických primitiv těchto knihoven. Vybraná zařízení byla od společnosti Raspberry, konkrétně modely Pi Zero, 1, 2 a 3. Vybrána byla z důvodu dobrého poměru cena vs výkon a výkon vs kompaktnost. Narozdíl od mikropočítačů od společnosti Arduino, Raspberry umožňuje nainstalovat jakýkoliv operační systém (Windows, Linux). Použitý operační systém na zařízeních Raspberry Pi byl nativní, dobře podporovaný OS Raspbian, což je obdoba OS Debian.

3.3.1 Raspberry Pi Zero

Raspberry Pi Zero W disponuje čipsetem BCM2835 přetaktovaným na 1 GHz, 512 MB RAM (Obr. 3.6, část A). Přesné hardwarové parametry jsou vypsány v Tab. 3.2 [47].

Tab. 3.2: Parametry zařízení Raspberry Pi Zero W [47].

Architektura	ARMv6Z (32-bit)
SoC	Broadcom BCM2835
CPU	ARM1176JZF-S, 1 GHz jednojádrový
GPU	Broadcom VideoCore IV @ 250 MHz (BCM2837: 3D part of GPU @ 300 MHz, video part of GPU @ 400 MHz)
	OpenGL ES 2.0 (BCM2835, BCM2836: 24 GFLOPS / BCM2837: 28.8 GFLOPS)
	MPEG-2 and VC-1 (with license), 1080p30 H.264/MPEG-4 AVC high-profile decoder and encoder (BCM2837: 1080p60)
SDRAM	512 MB (sdílená s GPU)
Interní paměť	MicroSDHC
Jmenovitý výkon	100 mA (0.5 W) v průměru na volnoběh
	maximálně 350 mA (1.75 W) při zatížení (s připojeným monitorem, klávesnicí a myší)

3.3.2 Raspberry Pi 1

Raspberry Pi 1 model B 512 MB RAM (Obr. 3.6, část B) je vylepšená varianta předchozího modelu A. Přesné hardwarové parametry jsou vypsány v Tab. 3.3 [48].

Tab. 3.3: Parametry zařízení Raspberry Pi 1, model B 512 MB RAM [48].

Architektura	ARMv6Z (32-bit)
SoC	Broadcom BCM2835 (CPU, GPU, DSP, SDRAM, jeden USB port)
CPU	ARM1176JZF-S, 700 MHz jednojádrový
GPU	Broadcom VideoCore IV @ 250 MHz, OpenGL ES 2.0 (24 GFLOPS), MPEG-2 a VC-1 (slicencí), 1080p30 H.264
	/MPEG-4 AVC dekodér a kodér s vysokým profilem
SDRAM	512 MB (sdílená s GPU)
Interní paměť	Slot na SD / MMC / SDIO kartu
Jmenovitý výkon	700 mA (3,5 W)

3.3.3 Raspberry Pi 2

Jedná se o druhou generaci Raspberry Pi s novým 900 MHz 4-jádrovým procesorem, 1 GB RAM (Obr. 3.6, část C). Přesné hardwarové parametry jsou vypsány v Tab. 3.4.

Tab. 3.4: Parametry zařízení Raspberry Pi 2, Model B 1 GB [49].

SoC	Broadcom BCM2836 (CPU, GPU, DSP, SDRAM, jeden USB port)
CPU	ARM Cortex-A7, 900 MHz čtyřjádrový
GPU	Broadcom VideoCore IV @ 400 MHz / 300 MHz, OpenGL ES 2.0 (24 GFLOPS), MPEG-2 a VC-1 (s licencí),
	1080p30 H.264/MPEG-4 AVC dekodér a kodér s vysokým profilem
SDRAM	1 GB (sdílená s GPU)
Interní paměť	MicroSDHC slot
Jmenovitý výkon	800 mA (4,0 W)

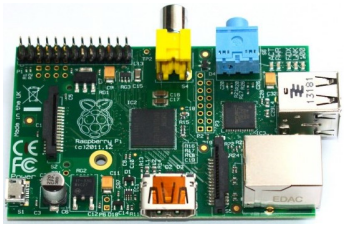
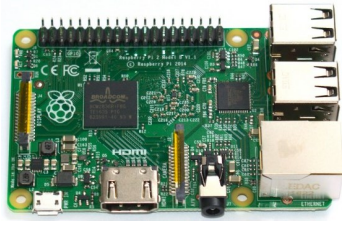
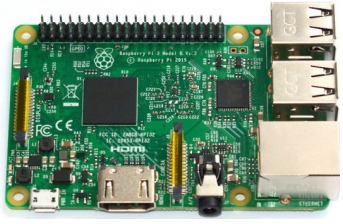
3.3.4 Raspberry Pi 3

Jedná se o třetí generaci Raspberry Pi s 1.2GHz 64-bit quad-core procesorem ARMv8, 1 GB RAM (Obr. 3.6, část D). Přesné hardwarové parametry jsou vypsány v Tab. 3.5 [50].

Tab. 3.5: Parametry zařízení Raspberry Pi 3, Model B 64-bit 1 GB RAM [50].

Architektura	ARMv8-A (64/32-bit)
SoC	Broadcom BCM2837
CPU	ARM Cortex-A53, 1,2 GHz 64-bit čtyřjádrový
GPU	Broadcom VideoCore IV @ 250 MHz (BCM2837: 3D part of GPU @ 300 MHz, video part of GPU @ 400 MHz)
	OpenGL ES 2.0 (BCM2835, BCM2836: 24 GFLOPS / BCM2837: 28.8 GFLOPS)
	MPEG-2 and VC-1 (with license), 1080p30 H.264/MPEG-4 AVC high-profile decoder and encoder (BCM2837: 1080p60)
SDRAM	1 GB (sdílená s GPU)
Interní paměť	MicroSDHC, USB Boot Mode
Jmenovitý výkon	300 mA (1,5 W) v průměru na volnoběh
	1,34A (6,7 W) při maximálním zatížení (připojený monitor, klávesnice, myš a WiFi)

Tab. 3.6: Zařízení Raspberry Pi X, na kterých bylo testováno [51, 52, 53, 54].

A) 
B) 
C) 
D) 

3.3.5 Instalace hardwarových platforem

Proces instalace OS Raspbian na zařízení Raspberry Pi 3 trval přibližně 30 minut. Pro účely diplomové práce byla použita distribuce OS Raspbian, což je upravená verze linuxové distribuce Debian. Tato distribuce je optimalizována pro hardware Raspberry. Dle vývojáře je na této distribuci předinstalováno 35 tisíc linuxových balíčků. Nabízí širokou podporu ze strany vývojářů. OS Raspbian je stále v aktivním vývoji, spolupracuje se všemi verzemi Raspberry Pi, o kterých je v této práci řeč. Na zařízení Raspberry Pi 2 a Raspberry Pi 3 je možno nainstalovat také jiné distribuce operačních systémů. Výčet podporovaných operačních systémů:

- Ubuntu MATE,
- Ubuntu Core,
- Windows 10 IoT Core,
- OSMC,
- LibreELEC,
- PiNet,
- RISC OS,
- Oracle Weather Station.

Na Raspberry Pi první generace a Raspberry Pi Zero není možné nainstalovat všechny výše zmíněné operační systémy. Z tohoto a dalších důvodů byla vybrána jednotná distribuce a to již zmíněný OS Raspbian. V první fázi je třeba naformátovat microSD paměťovou kartu, například na formát FAT32. Nakopírovat soubory rozbaleného archívu na paměťovou kartu. Po vložení paměťové karty do zařízení a připojení adaptéru nabootuje zařízení do instalátoru NOOBS (New Out Of the Box Software). NOOBS je oficiální instalátor od společnosti Raspberry. Obsahuje poslední verzi OS Raspian a několik dalších distribucí OS. Volitelně je zde také možno přidat instalátory jiných operačních systémů. Balíček NOOBS je možné stáhnout z webu výrobce [58]. Instalace OS Raspbian je velice jednoduchá a intuitivní. Zeptá se jen na možnosti zformátování paměťové karty a na pár dalších volitelných nastavení systému. Vše je samozřejmě možno nakonfigurovat i po dokončení instalace. Následně již instalátor kopíruje systémové soubory OS Raspbian na kartu, což je závislé na použité kartě a rychlosti hardware. Na zařízení Raspberry Pi 3 s použitou kartou Samsung microSDHC 32GB UHS-I trvalo kopírování souborů kolem 20 minut. Na Raspberry Pi 2 trvala instalace téměř srovnatelně dlouho. Na Raspberry Pi 1 a Zero trvala instalace znatelně déle, jelikož mají pomalejší hardware. Zde se doba instalace blížila jedné hodině na jednom zařízení. Po dokončeni instalace zařízení restartuje a nabootuje do nově nainstalovaného prostředí. Po prvním spuštění je třeba nastavit standardní parametry nově nainstalovaného operačního systému.

4 PŘÍPRAVA MĚŘENÍ PRIMITIV NA ELIPTICKÝCH KŘIVKÁCH

Měření bylo prováděno vždy na vzorku 1000 hodnot daného měření. Bylo zvoleno 1000 vzorků jako dobrý kompromis mezi dobou měření a dobrou přesností výpočtů. Při polovičním vzorku hodnoty ještě občas kolísaly, naopak při dvojnásobném vzorku hodnoty zůstávaly stejné jako při daném vzorku. Při tomto počtu vzorků se časy měnily při opakovaném měření stejného úkonu až na zanedbatelném řádu za desetinnou čárkou. Nutno dodat, že hodnoty v této diplomové práci jsou měřeny v milisekundách [ms] a byly průměrovány.

Nabízí se tři možnosti zpracování získaných hodnot pro měření kryptografických primitiv na eliptických křivkách. Aritmetický průměr, modus a medián. V tomto případě byl zvolen aritmetický průměr. Zajímavé výsledky by mohly být také pomocí mediánu či modu. Když se zamyslíme nad hlavními rozdíly u vzorku 1000 hodnot zjistíme, že výsledný rozdíl není zásadní. Zajímavé srovnání náhodných 1000 hodnot z rozsahu jedna až pět můžeme vidět v Tab. 4.1. Jsou zde srovnány aritmetický průměr, modus a medián z těchto hodnot. Soubor s těmito výpočty je přiložen na CD pod názvem prum_mod_med.xlsx Průměr byl také zvolen po úvaze, zda měřit medián či modus, a jelikož v kryptografických algoritmech nemůžeme počítat s ideální hodnotou – byl zvolen z tohoto důvodu průměr, který ale může vykazovat statistickou chybu. S tou bylo počítáno a do grafů byla zahrnuta.

Tab. 4.1: Srovnání hodnot z průměru, modusu a mediánu.

průměr	modus	medián
2,98	3	3

4.1 Způsoby měření efektivity kryptografických knihoven

Tato kapitola se zabývá popisem metod, pomocí kterých jsou realizována měření efektivity kryptografických knihoven – počítání kryptografických primitiv s jednotlivými knihovnami. Měřící metody byly prezentovány na soutěži EEICT 2018 [55]. Následující parametry měření jsou realizovány pomocí knihovny psutil (process and system utilities) a napsány v jazyce Python:

- Vytížení procesoru,
- paměťová náročnost,
- vytížení grafického adaptéru,

- vytížení disku.

Dále jsou realizována tato měření:

- Časová náročnost,
- energetická náročnost,
- počet strojových cyklů.

Následující podkapitoly jsou rozděleny na měřící metody realizované knihovnou psutil a další měřící metody. V jednotlivých podkapitolách je úvod do problematiky dané knihovny, sestrojeného komplexního algoritmu a realizovaných metod.

4.1.1 Měřící metody realizované knihovnou psutil

Zde jsou popsány měřící metody pomocí knihovny psutil. V první části kapitoly je popsána knihovna psutil a její možnosti, následně popis realizovaných měřících metod pomocí knihovny psutil a jejich možnosti. Psutil je multiplatformní knihovna pro získávání informací o běžících procesech a využití systému (CPU, RAM, GPU, disky, sítě, čidla) naprogramována v jazyce Python. Je zaměřená primárně na sledování systému, profilování a omezení procesních prostředků a řízení běžících procesů. Implementuje mnoho funkcí nabízených nástroji příkazového řádku systému UNIX, jako jsou: ps, top, lsof, netstat, ifconfig, who, df, kill, free, nice, ionice, iostat, pidof, tty, taskset, pmap. Psutil v současné době podporuje následující platformy:

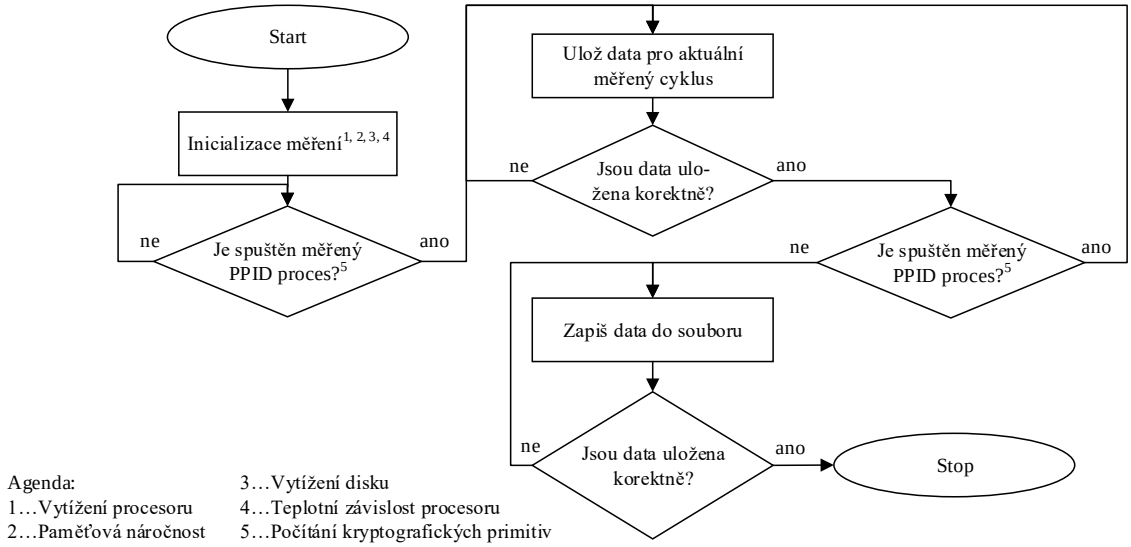
- Linux,
- Windows,
- OSX,
- FreeBSD, OpenBSD, NetBSD,
- Sun Solaris,
- AIX.

Podporuje 32bitové i 64bitové architektury s verzemi Pythonu od 2.6 do 3.6 (uživatelé Pythonu 2.4 a 2.5 mohou používat verzi 2.1.3). Podporuje také implementaci PyPy [59]. Knihovna psutil byla vytvořena a je spravována uživatelem Giampaolo Rodola. Je distribuována zdarma či za dobrovolný příspěvek autorovi [60].

4.1.2 Popis komplexního měřícího algoritmu

Měřící algoritmus obsahuje následující měřící metody: vytížení procesoru, paměťová náročnost, vytížení grafického adaptéru, vytížení disku – ty jsou realizovány pomocí knihovny psutil, která je popsána výše a jsou napsána v jazyce Python. Po načtení kryptografických knihoven a spuštění výpočetních operací těchto knihoven se spustí jednotlivá měření. Měření probíhá v předem nastaveném intervalu v průběhu běhu výpočetních operací. Po skončení běhu výpočetních operací se ukončí také měření.

Výsledky jednotlivých měření se zapíší do souboru. Vývojový diagram těchto měření je znázorněn na Obr. 4.1.



Obr. 4.1: Diagram měřících metod pomocí knihovny psutil.

V dalších podkapitolách jsou rozepsány možnosti jednotlivých metod pomocí knihovny psutil.

4.1.3 Vytížení procesoru

Měření vytížení procesoru dokáže měřit tyto parametry:

- Čas strávený vázanými (prioritizovanými) procesy prováděnými v uživatelském režimu,
- čas strávený čekáním na dokončení I/O operací,
- čas strávený pro obsluhu hardwarových, softwarových přerušení,
- čas strávený spuštěním virtuálního procesoru,
- čas strávený využitím v režimu uživatel, systém, nečinný,
- frekvenci aktuální, průměrnou, minimální, maximální vázanou na dané jádro či procesor.

4.1.4 Paměťová náročnost

Měření paměťové náročnosti dokáže měřit tyto parametry:

- Celková fyzická, swap paměť,
- použitou, aktivní, neaktivní, buffer, mezipaměť,
- procentuálně jednotlivá vytížení,

- počet bajtů, které systém swapoval z disku,
- počet bajtů, které systém swapoval na disk.

4.1.5 Vytížení grafického adaptéru

Zde se bude měřit pouze přidělená velikost paměti ze sdílené a její procentuální využití. To se dá nastavit v operačním systému Raspbian v nastavení systému.

4.1.6 Vytížení disku

Měření vytížení disku dokáže měřit tyto parametry:

- Rozdělení disku na sektory a jejich využití,
- využití daného oddílu,
- read_count: počet čtení,
- write_count: počet zápisů,
- read_bytes: počet přečtených bajtů,
- write_bytes: počet zapsaných bajtů,
- čas strávený čtení z disku,
- čas strávený zápisem na disk,
- čas strávený vlastními I/O operacemi,
- počet sloučených čtení,
- počet sloučených zápisů.

4.2 Další měřící metody

Zde jsou popsány další metody realizované v rámci měření efektivity kryptografických knihoven.

4.2.1 Energetická náročnost

Zde připadá v úvahu měření za pomocí speciální měřícího přístroje citlivého na velmi malé proudy. Další variantou je měření pomocí přídavného kitu. V diplomové práci je realizována metoda pomocí hardwarového multimetru.

4.2.2 Počet hodinových cyklů

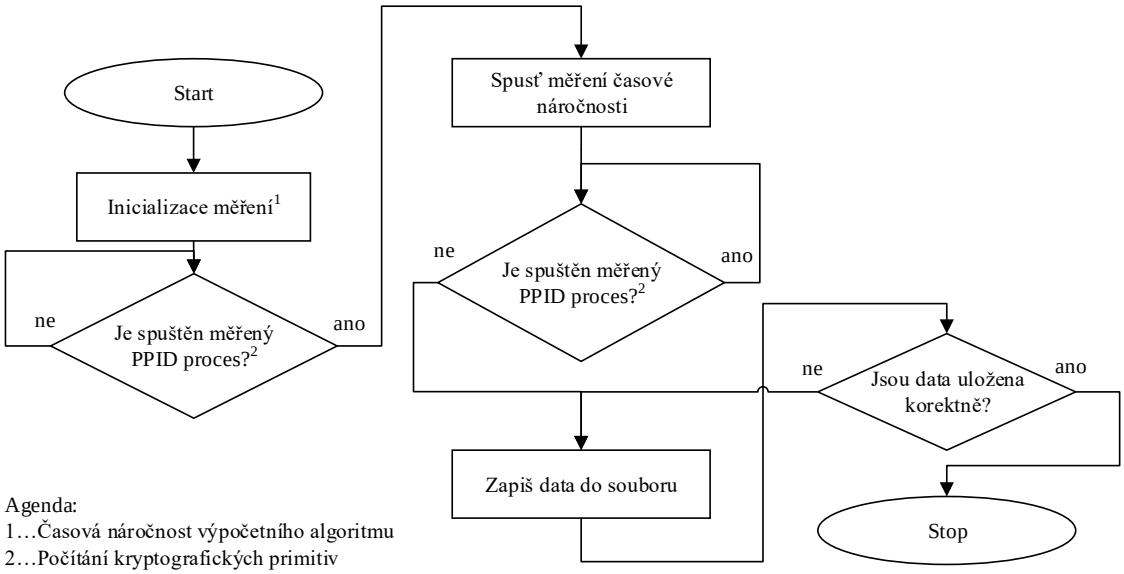
Počet hodinových cyklů se vypočítá jako převrácená hodnota času:

$$CC = \frac{1}{f} \cdot t, \tag{4.1}$$

kde:
CC...hodinový cyklus,
f...frekvence procesoru zařízení,
t...čas měření kryptografického primitiva.

4.2.3 Časová náročnost

Měření časové náročnosti je realizováno v jazyce Python. Po načtení kryptografických knihoven a spuštění výpočetních operací těchto knihoven se spustí měření. Měření se ukončí s ukončením výpočetních operací. Výsledky časové náročnosti se zapíší do souboru. Vývojový diagram tohoto měření je znázorněn na Obr. 4.2.



Obr. 4.2: Diagram měřící metody časová náročnost.

5 EXPERIMENTÁLNÍ MĚŘENÍ PRIMITIV NA ELIPTICKÝCH KŘIVKÁCH

Tako kapitola se zabývá výsledky měření a jejich analýzou. Výsledky jsou měřeny pro následující kryptografická primitiva: generování náhodného bodu, addition bodů, doubling bodu, inverze bodu, multipling v knihovně OpenSSL, libecc a v pythonu. Algoritmy využívají metod skalárních, Montgomery a u násobení metodu skalární, wNAF a Montgomeryho žebřík. Měřená zařízení: Raspberry Pi Zero/1/2/3.

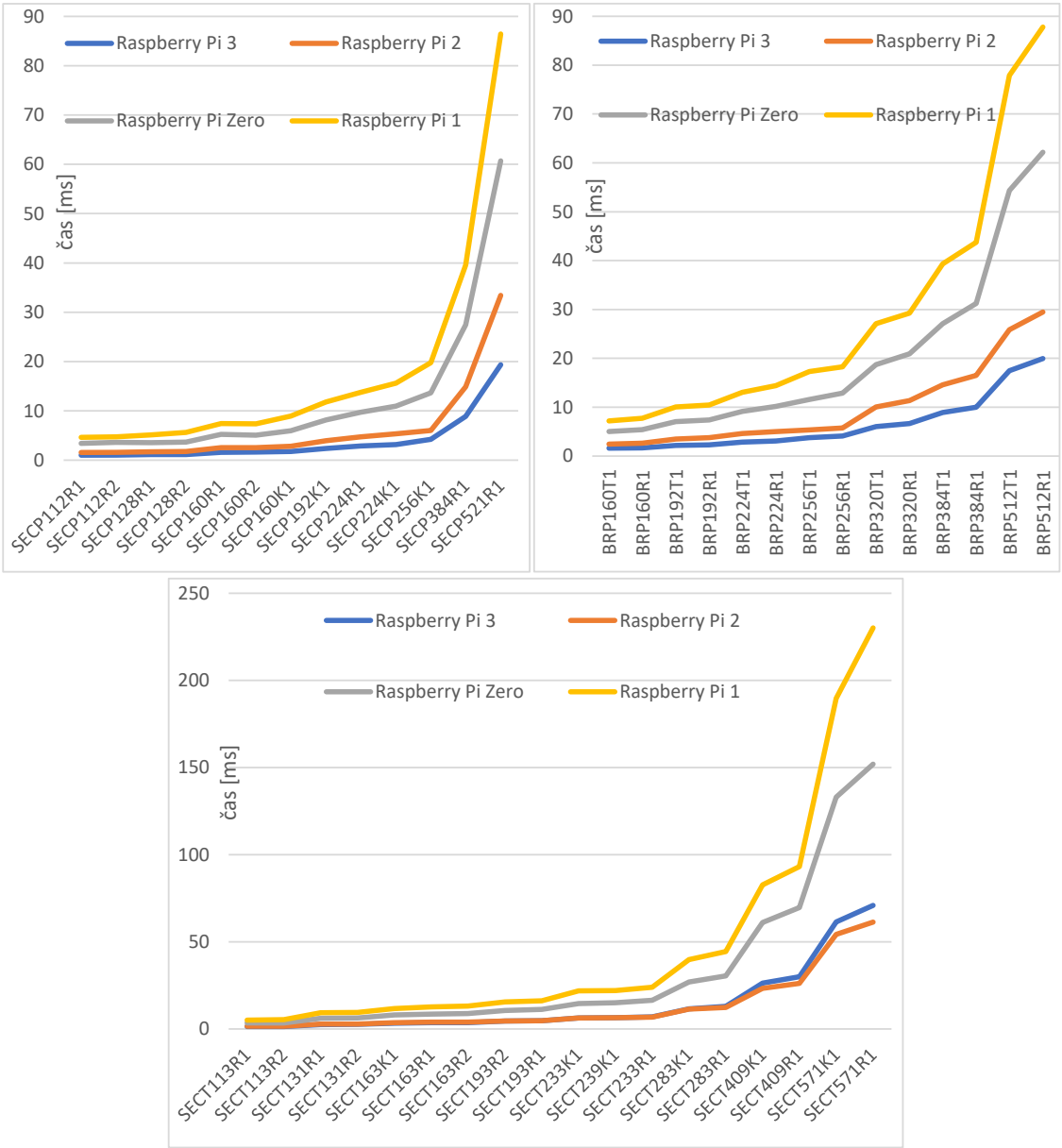
5.1 Měření v knihovně OpenSSL

Diskuze výsledků k následujícím kapitolám 5.1.1 až 5.1.5: Jednotlivé podkapitoly obsahují diskuzi k časovým výsledkům, nyní bude globálně shrnut za knihovnu OpenSSL časový narůst jednotlivých operací od nejméně náročné po nejnáročnější. Nejméně náročnou měřenou operací byla negace bodu na eliptické křivce. Maximální hodnota na nejpomalejším zařízení (Raspberry Pi 1) a nejnáročnější křivce (SECT571R1) dosahovala 0,4905 ms. Druhou nejméně náročnou měřenou operací bylo addition se svými 7,7463 ms. Nutno podotknout, že addition bylo na dvou ostatních zařízeních (Raspberry Pi Zero/2) pomalejší než doubling, avšak na Raspberry Pi 1 je druhou nejméně náročnou operací operací. Třetí nejpomalejší operací bylo tedy doubling se svými 7,7534 ms. Zde je rozdíl jen nepatrný mezi operacemi addition a doubling. Další operací bylo generování bodu se svými 230,1236 ms. A poslední a tedy nejnáročnější operací bylo multiplication se svými 850,9717 ms. Nejméně náročnou operací naopak na nejrychlejším zařízení (Raspberry Pi 3) a stejné (nejnáročnější) křivce byla také operace negace bodu s časem 0,0445 ms. Druhou nejméně náročnou operací stejně jako v případě Raspberry Pi 3 bylo addition s časem 2,0852 ms. Třetí nejpomalejší operací bylo doubling se svými 2,0853 ms. Další, stejně jako na Raspberry Pi 3, bylo generování bodu s časem 70,9011 ms. A nejnáročnější operací bylo multiplication s časem 260,0244 ms. Výsledky mají stejné pořadí jako na nejpomalejším zařízení.

Je zde tedy vidět jistý vztah výkonnosti procesoru v závislosti na křivce. Na ostatních zařízeních se pořadí výsledků liší pouze mezi zařízeními Raspberry Pi Zero/2 a to konkrétně v případě addition a doubling, kde si vyměnili pořadí o desetiny milisekund. Výsledky jsou také ovlivněny metodami výpočtu a použitým algoritmem. OpenSSL používá pro addition, doubling, inverzi klasické skalární metody. Pro multiplication používá metodu wNAF. Metody jsou popsány v kapitolách 1.7 a 1.8. Originální grafy a tabulky jsou přiloženy v příloze a na CD pod názvem OpenSSL_mereni.

5.1.1 Generování náhodného bodu

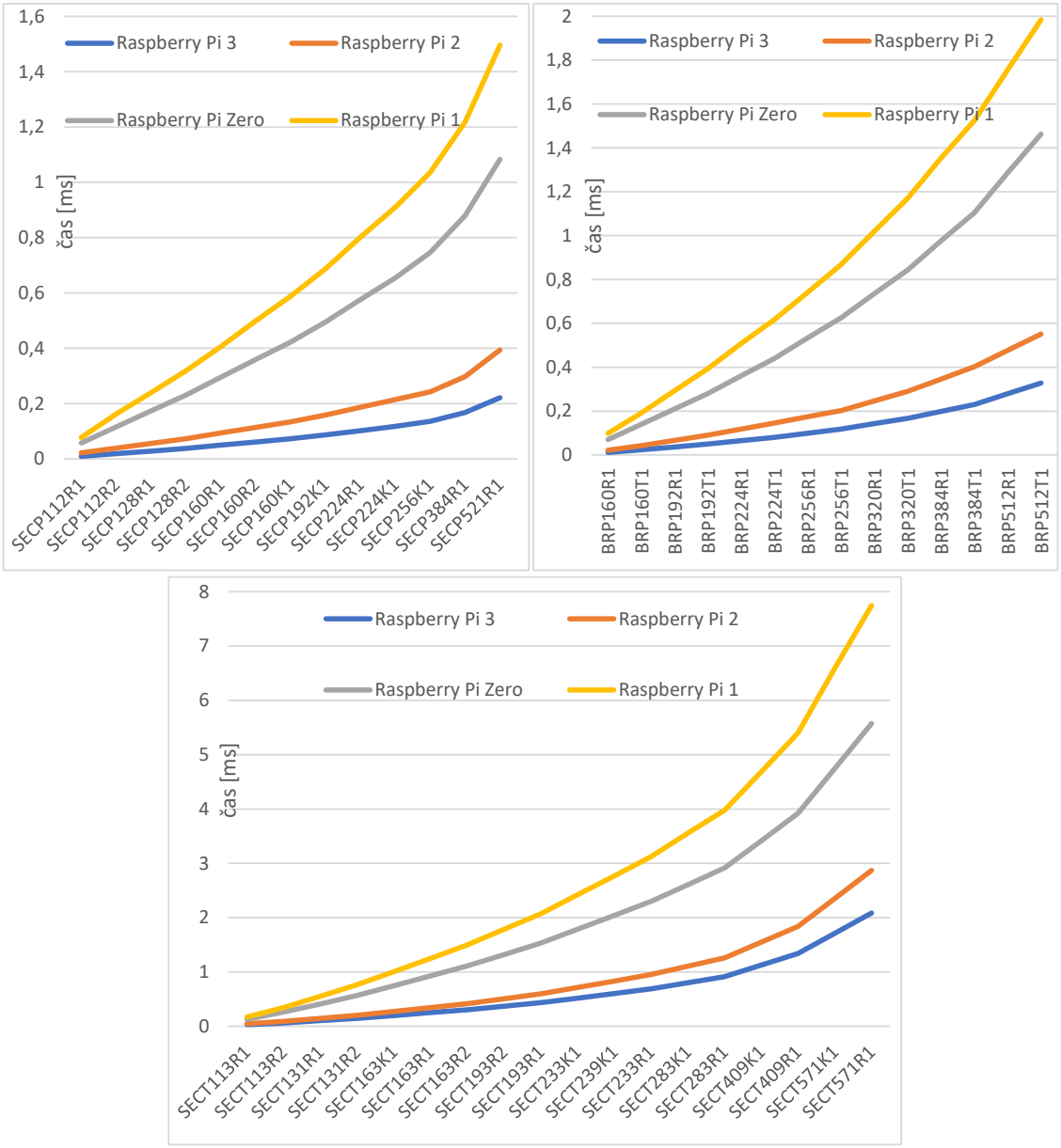
Výsledky měření generování náhodného bodu na eliptických křivkách standardů SECP, SECT a BRAINPOOL (v grafu jen BRP) (Obr. 5.1). Z grafů lze vyčíst, že křivky standardů SECP a BRAINPOOL mají čas generování náhodného bodu do cca 90 ms. Křivky standardu SECT mají čas generování náhodného bodu až do cca 250 ms. Tudíž jsou až 2,5x pomalejší než křivky SECP a BRAINPOOL při operaci generování náhodného bodu.



Obr. 5.1: Měření generování náhodného bodu v knihovně OpenSSL.

5.1.2 Addition dvou bodů

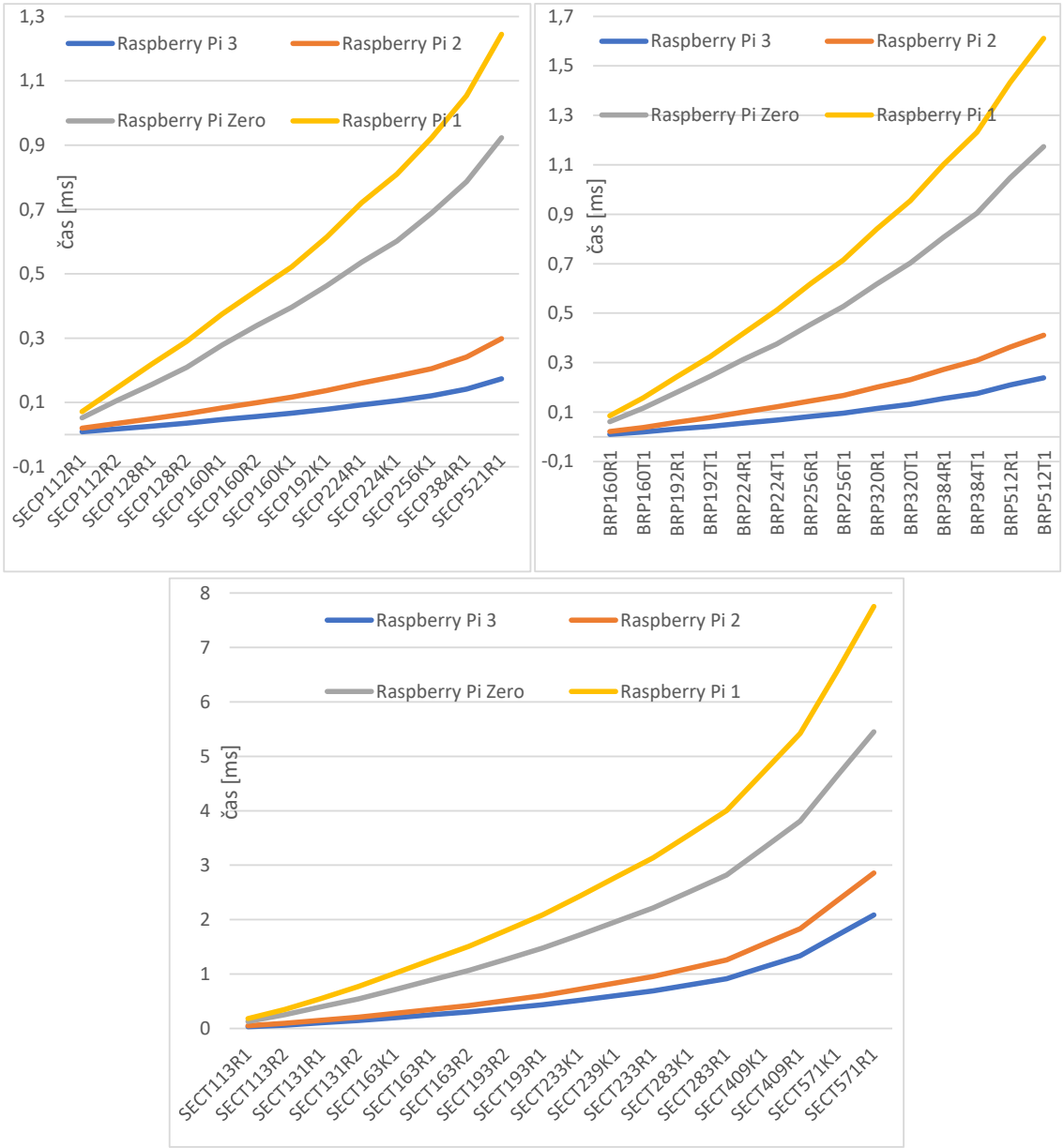
Výsledky měření addition dvou bodů na eliptických křivkách standardů SECP, SECT a BRAINPOOL (v grafu jen BRP) (Obr. 5.2). Z grafů lze vyčíst, že křivky standardů SECP mají čas addition dvou bodů do cca 1,6 ms, BRAINPOOL mají čas addition dvou bodů do cca 2 ms. Křivky standardu SECT mají čas addition dvou bodů až do cca 8 ms. Tudíž jsou až 4x pomalejší než křivky SECP a BRAINPOOL při operaci addition dvou bodů.



Obr. 5.2: Měření addition dvou bodů v knihovně OpenSSL.

5.1.3 Doubling bodu

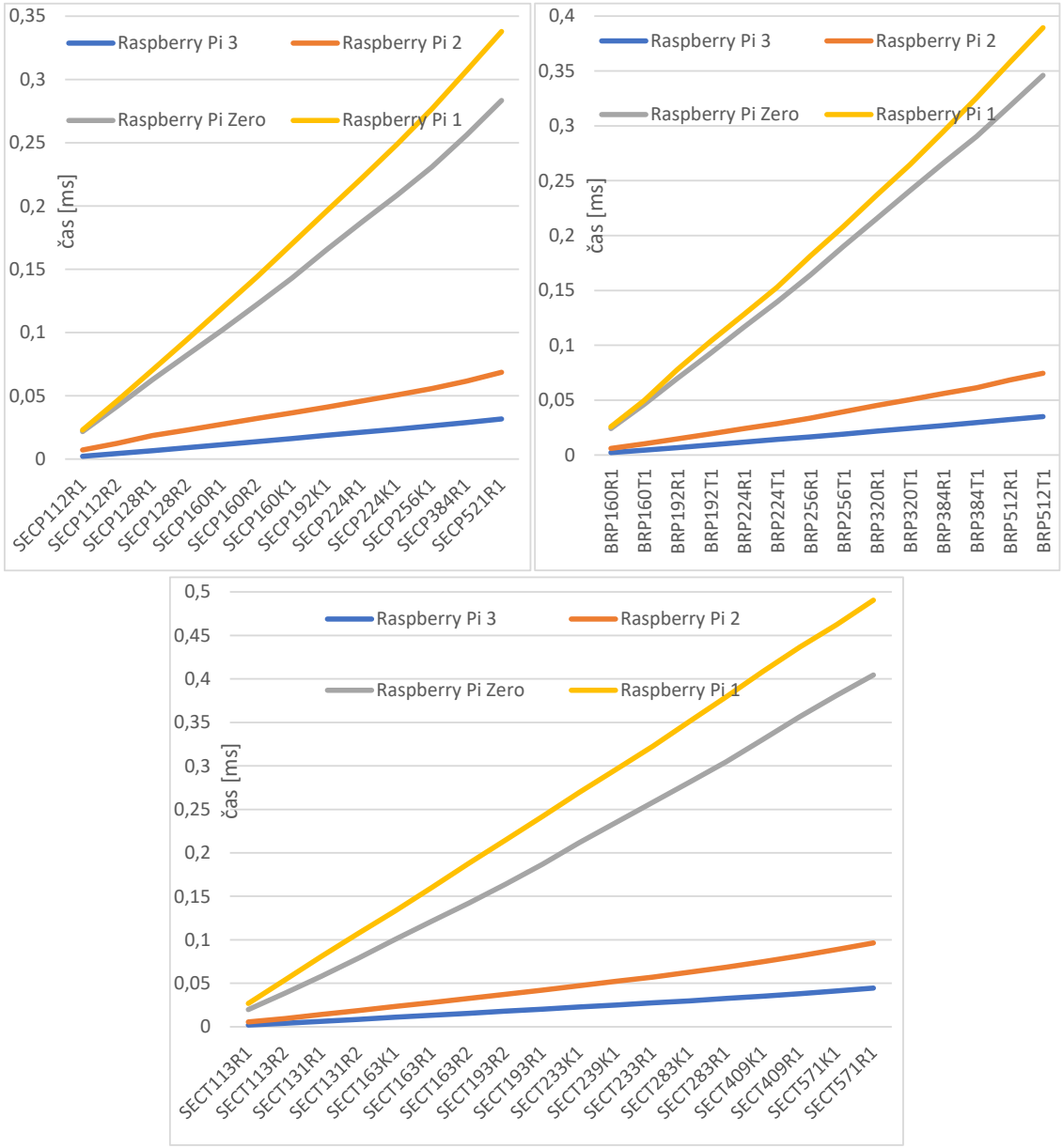
Výsledky měření doubling bodu na eliptických křivkách standardů SECP, SECT a BRAINPOOL (v grafu jen BRP) (Obr. 5.3). Z grafů lze vyčíst, že křivky standardů SECP mají čas doubling bodu do cca 1,3 ms, BRAINPOOL mají čas doubling bodu do cca 1,7 ms. Křivky standardu SECT mají čas doubling bodu až do cca 8 ms. Tudíž jsou až 4,5x pomalejší než křivky SECP a BRAINPOOL při operaci doubling bodu.



Obr. 5.3: Měření doubling bodu v knihovně OpenSSL.

5.1.4 Negace bodu

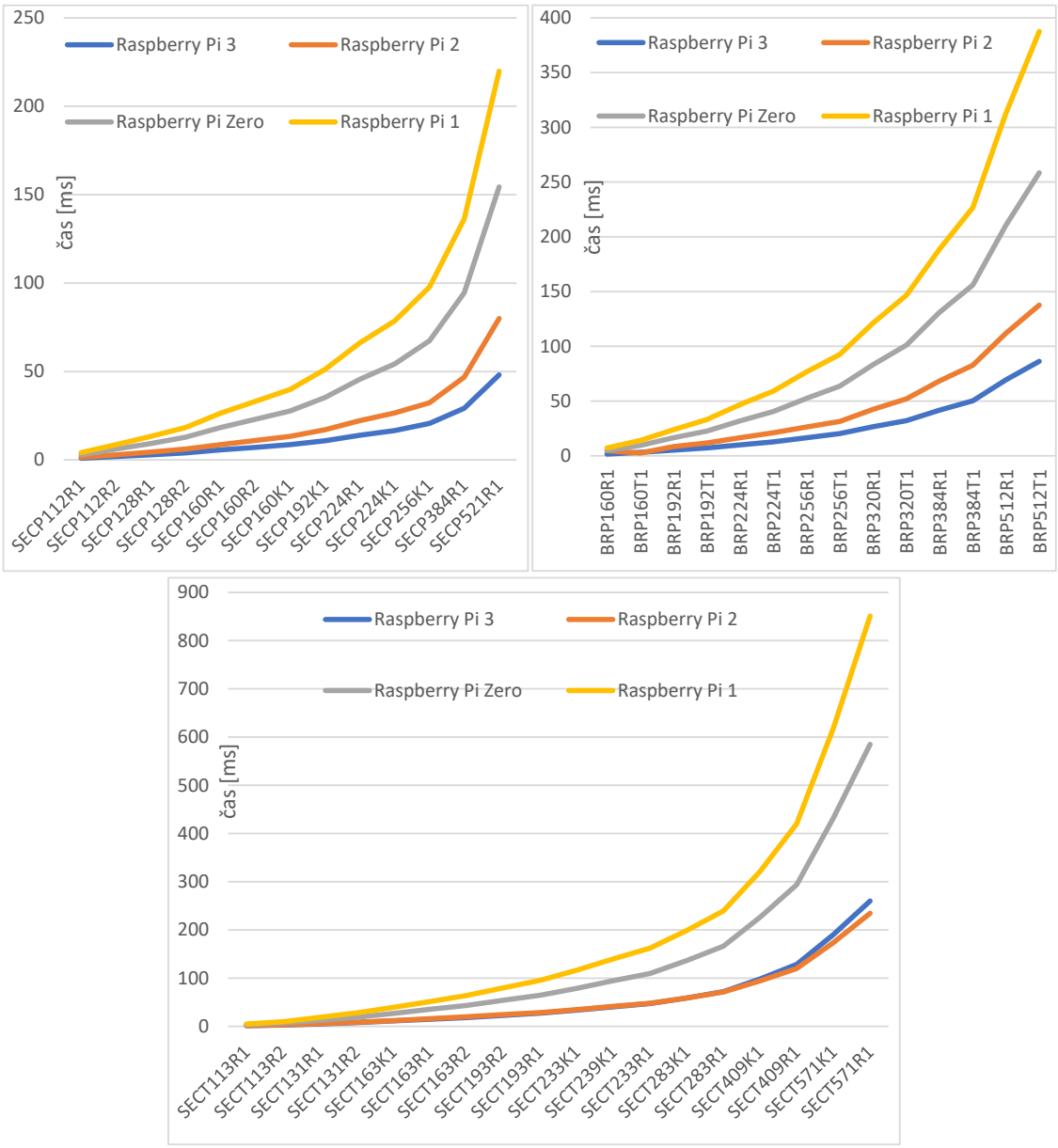
Výsledky měření negace bodu na eliptických křivkách standardů SECP, SECT a BRAINPOOL (v grafu jen BRP) (Obr. 5.4). Z grafů lze vyčíst, že křivky standardů SECP a BRAINPOOL mají čas negace bodu do cca 0,4 ms. Křivky standardu SECT mají čas negace bodu až do cca 0,5 ms. Zde není tak násobný rozdíl mezi standardy SECP, SECT a BRAINPOOL při operaci negace bodu. Výsledky jsou zde nejvíce lineární.



Obr. 5.4: Měření negace bodu v knihovně OpenSSL.

5.1.5 Multiplikace dvou bodů

Výsledky měření multiplikace dvou bodů na eliptických křivkách standardů SECP, SECT a BRAINPOOL (v grafu jen BRP) (Obr. 5.5). Z grafů lze vyčíst, že křivky standardů SECP mají čas multiplikace dvou bodů do cca 250 ms, BRAINPOOL mají čas multiplikace dvou bodů do cca 400 ms. Křivky standardu SECT mají čas multiplikace dvou bodů až do cca 900 ms. Tudíž jsou až 3,5x pomalejší než křivky SECP a 2,3x pomalejší než křivky BRAINPOOL při operaci multiplikace dvou bodů.



Obr. 5.5: Měření multiplikace dvou bodů v knihovně OpenSSL.

5.2 Měření v knihovně libecc

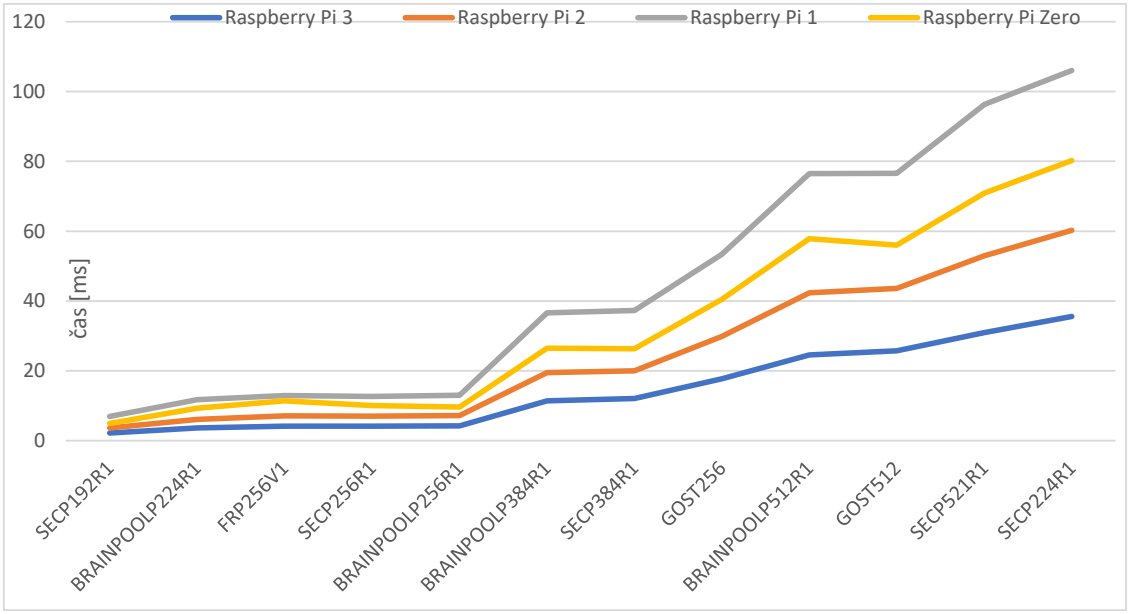
Diskuze výsledků k následujícím kapitolám 5.2.1 až 5.2.4: Jednotlivé podkapitoly obsahují diskuzi k časovým výsledkům, nyní bude globálně shrnut za knihovnu libecc časový narůst jednotlivých operací od nejméně náročné po nejnáročnější. Nejméně náročnou měřenou operací byla operace doubling dvou bodů metodou Montgomery na eliptické křivce. Maximální hodnota na nejpomalejším zařízení (RaspBerry Pi 1) a nejnáročnější z měřených křivek v rámci knihovny libecc (SECP521R1) dosahovala 0,348 ms. Druhou nejméně náročnou měřenou operací bylo addition metodou Montgomery se svými 0,416 ms. Třetí nejpomalejší operací bylo doubling metodou skalární se svými 0,512 ms. Další operací bylo addition bodu metodou skalární se svými 0,597 ms. Výrazně náročnější metodou je zde generování náhodného bodu na eliptické křivce s časem 76,518 ms. A nejnáročnějšími metodami je zde multipling metodou Montgomery a metodou skalární se svými 390,125 ms a 662,321 ms. Nejméně náročnou operací naopak na nejrychlejším zařízení (Raspberry Pi 3) a stejné křivce byl doubling dvou bodů metodou Montgomery na eliptické křivce s časem 0,122. Druhou nejméně náročnou měřenou operací na Raspberry Pi 3 bylo addition metodou Montgomery se svými 0,128 ms. Třetí nejpomalejší operací bylo doubling metodou skalární se svými 0,197 ms. Další operací bylo addition bodu metodou skalární se svými 0,227 ms. Výrazně náročnější metodou, stejně jako u nejpomalejšího zařízení je generování náhodného bodu na eliptické křivce s časem 30,995 ms. A nejnáročnějšími metodami je zde multipling metodou Montgomery a metodou skalární se svými 158,933 ms a 289,114 ms. Mezi naměřenými hodnotami a měřenými křivkami si lze všimnout určité závislosti. V knihovně libecc mají všechna zařízení pevně dané pořadí výkonnosti operace v závislosti na měřené eliptické křivce.

Je zde tedy, jako v předchozím měření v knihovně OpenSSL, vztah výkonnosti zařízení v závislosti na měřené křivce. Srovnání s předchozí měřenou knihovnou OpenSSL je následující. Hodnoty generování bodu na eliptické křivce jsou pomalejší než v knihovně OpenSSL. Hodnoty skalárních metod addition a doubling jsou také pomalejší. Násobení wNAF, v knihovně OpenSSL, je ještě rychlejší než Montgomeryho metoda u knihovny libecc. libecc tedy podává kvalitní výsledky u operací addition a doubling a Montgomeryho metody v porovnání se skalárními v knihovně OpenSSL, ale naopak v případě násobení a generování bodu má knihovna OpenSSL navrch. Implementace Montgomeryho metod zde výrazně urychluje časy až na polovinu oproti druhé implementované metodě, tedy obyčejné skalární metodě. Výsledky jsou také ovlivněny metodami výpočtu a použitým algoritmem. OpenSSL používá pro addition a doubling klasické skalární metody. Pro multiplication používá metodu skalární a Montgomery ladder. Metody jsou popsány v kapitolách 1.7 a 1.8. Originální grafy a tabulky jsou přiloženy v příloze a na CD pod názvem libecc_mereni.

5.2.1 Generování náhodného bodu

Výsledky měření generování náhodného bodu na eliptických křivkách standardů SECP a BRAINPOOL (v grafu jen BRP), GOST a FRP (Obr. 5.6). Z grafů lze vyčíst, že křivky standardů SECP jsou nepatrně rychlejší při velikosti tělesa 192bit až 256bit jsou nejrychlejšími křivkami SECP, BRP a FRP naopak GOST je 4,5x pomalejší (cca 4 ms vs 18 ms). Naopak od 256bit nahoru křivka SECP pomalejší než ostatní. Nejrychlejšími křivkami při největším tělesu jsou BRAINPOOL512R1 a GOST512. Generování náhodného bodu je zde oproti knihovně libecc mírně pomalejší u větších křivek (256bit a výš). U malých křivek (192bit až 256 bit) jsou, až na malé rozdíly, časově skoro stejné. Můžeme zde vidět strmější narůst než u knihovny OpenSSL. Závislost výkonnosti zařízení je zde učebnicově ukázková. U knihovny OpenSSL nenabývá takto rovnoměrně na čase v závislosti na zařízení a křivce. Operace generování bodů se používá téměř u každého kryptografického protokolu. Je tedy potřeba, aby tyto operace byly rychlé a efektivní na dané křivce.

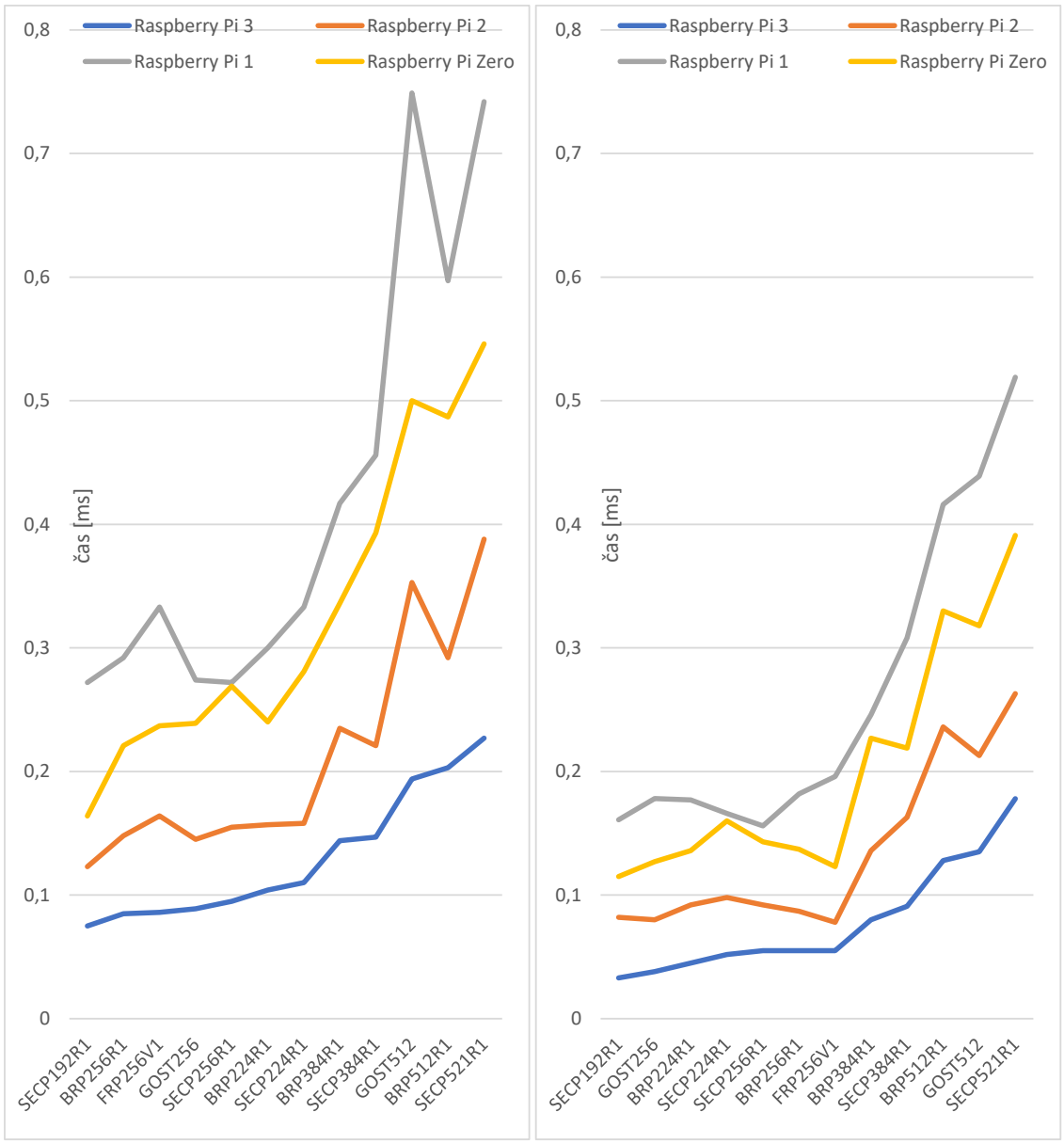
U knihovny libecc ještě stojí za zmínku chyba knihovny v implementaci jedné křivky. Po konzultaci s vedoucím práce bylo dospěno k závěru, že křivka SECP224R1 je zde nejpomalejší bezdůvodně. Přitom v ostatních knihovnách vychází až řádově časově rychleji. Nastavení parametrů daných křivek je standardizované a neměly by se tedy lišit až v takovém velkém časovém horizontu při stejné algoritmické metodě generování náhodného bodu. U ostatních operací křivka vychází již standardně.



Obr. 5.6: Měření generování náhodného bodu v knihovně libecc metodou skalární (vlevo) a Montgomery (vpravo).

5.2.2 Addition dvou bodů

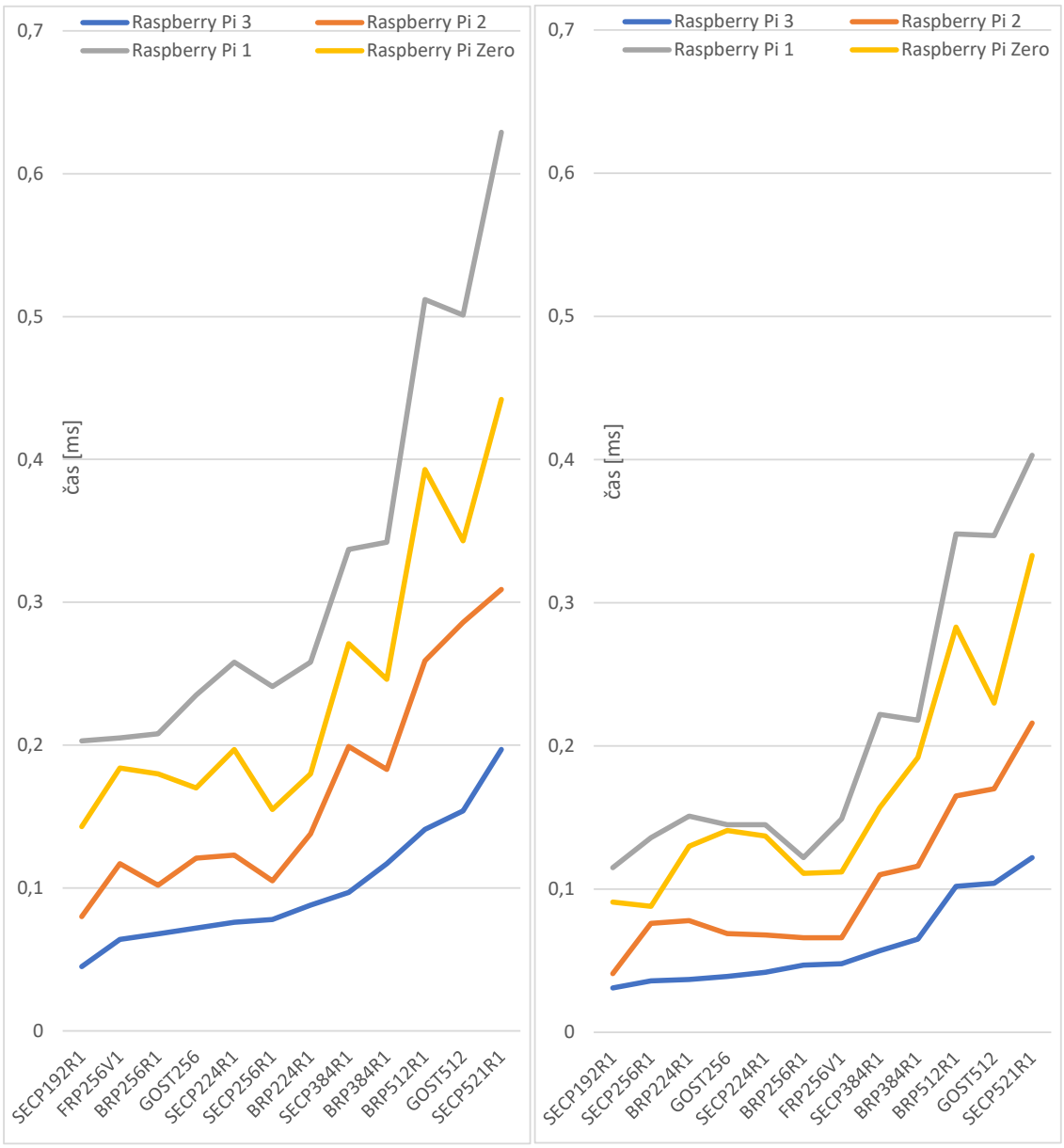
Výsledky měření addition dvou bodů na eliptických křivkách standardů SECP a BRAINPOOL (v grafu jen BRP), GOST a FRP (Obr. 5.7). Z grafů lze vyčíst, že se zde ztratil rozdíl mezi časy při addition dvou bodů. Zde porovnáme skalární metodu vs Montgomery. Montgomery je na pomalejším zařízení až o 1/3 rychlejší. U nejrychlejšího zařízení je rozdíl menší. Nejpomalejší i nejrychlejší křivkou je ta samá, ale pořadí křivek se v závislosti na metodě změnilo.



Obr. 5.7: Měření addition dvou bodů v knihovně libecc metodou skalární (vlevo) a Montgomery (vpravo).

5.2.3 Doubling bodu

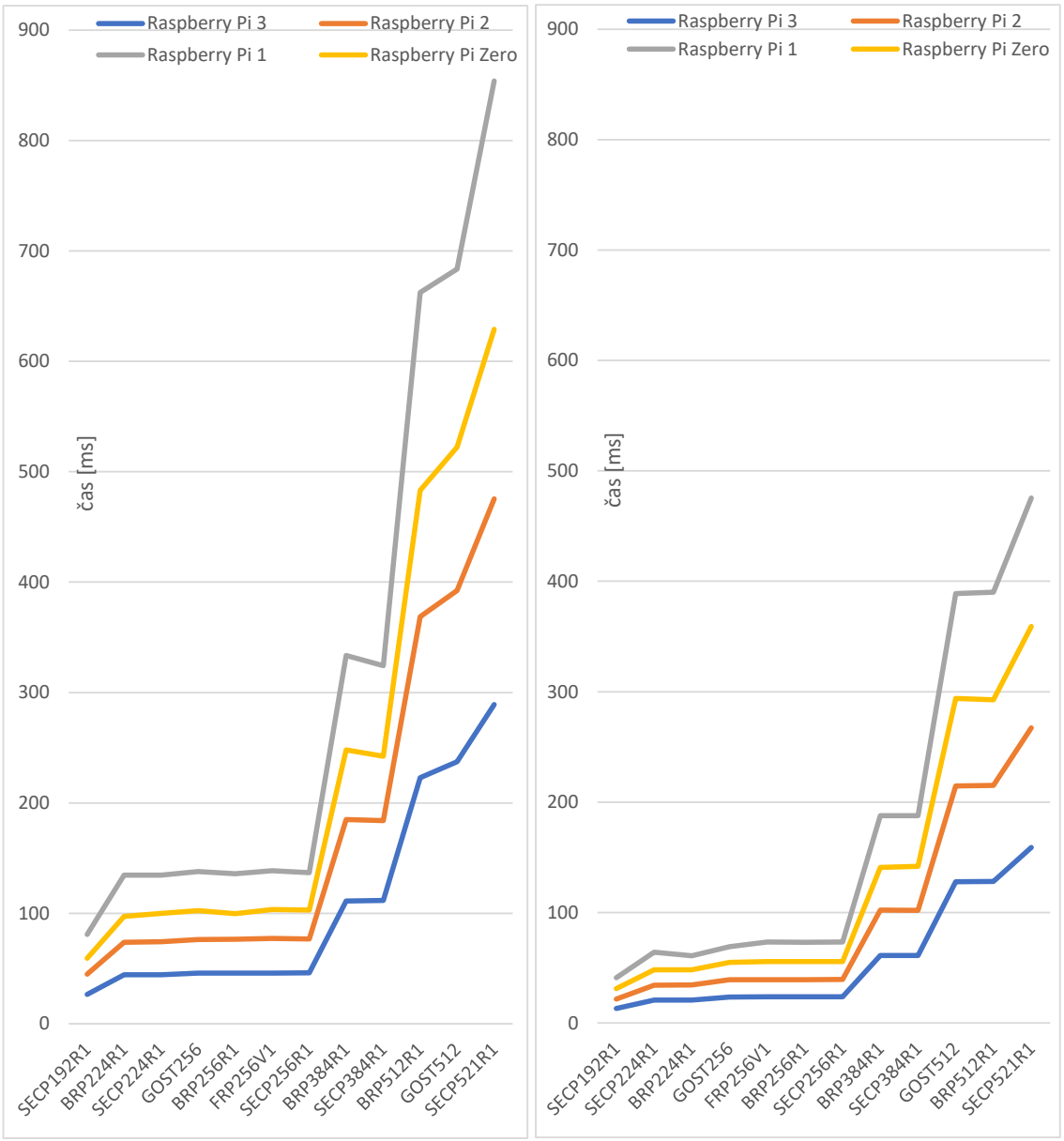
Výsledky měření doubling bodu na eliptických křivkách standardů SECP a BRAINPOOL (v grafu jen BRP), GOST a FRP (Obr. 5.8). Z grafů lze vyčíst, že pořadí křivek zůstalo stejné v závislosti na metodě. Doubling je nepatrně náročnější metoda než addition. Skalární doubling vs Montgomery je cca v poměru 1,8 násobku nezávisle na zařízení.



Obr. 5.8: Měření doubling bodu v knihovně libecc metodou skalární (vlevo) a Montgomery (vpravo).

5.2.4 Multiplikace dvou bodů

Výsledky měření multiplikace dvou bodů na eliptických křivkách standardů SECP a BRAINPOOL (v grafu jen BRP), GOST a FRP (Obr. 5.9). Z grafů lze vyčíst, že rozdíl mezi skalární metodou a Montgomery je zde až cca dvojnásobný. Pořadí křivek v závislosti na metodě zůstalo opět stejné. Průběhy jsou zde už velice podobné a mají definovaný charakter. Násobení je zde nejnáročnější metoda. Nejpomalejší křivka (SECP521R1) na nejpomalejším zařízení potřebuje 853,945 ms.

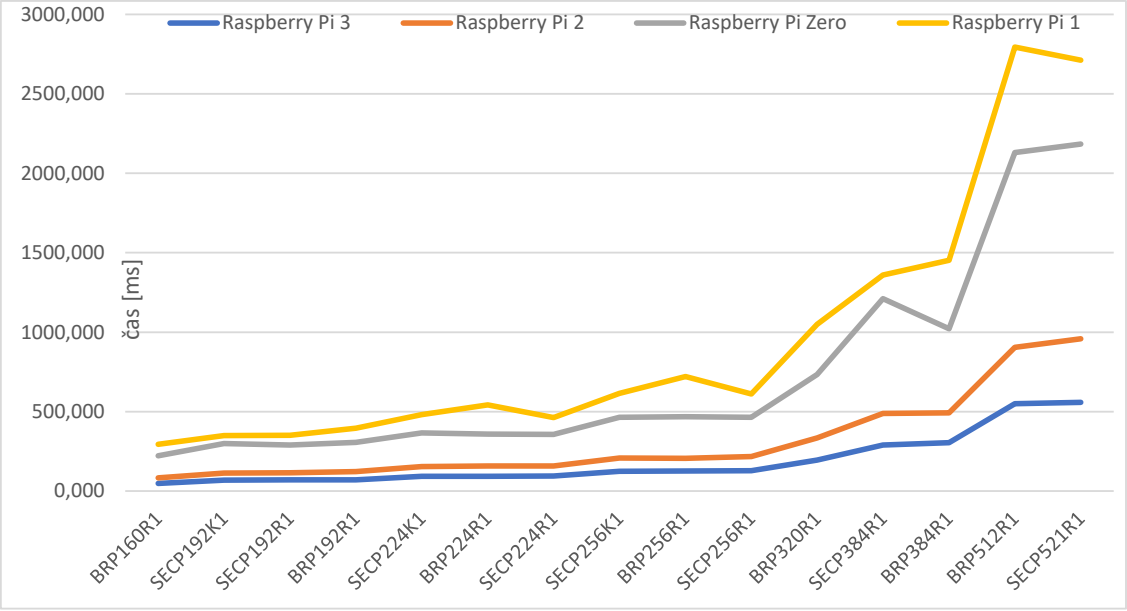


Obr. 5.9: Měření multiplikace dvou bodů v knihovně libecc metodou skalární (vlevo) a Montgomery (vpravo).

5.3 Měření v pythonu

V pythonu byly měřeny pouze výsledky multiplication metodou skalární, jelikož python vykazoval velkou časovou náročnost měření. Je zdaleka nejpomalejší metodou na operace kryptografických primitiv na eliptických křivkách. Vykazuje až několikanásobně pomalejší časy měření než v předchozích dvou knihovnách OpenSSL a libeec. Toto se ukázkově projevilo i při měření proudového zatížení na metodách multiplication napříč knihovnami, kde proudová osa vykazuje znatelná okna proudového zatížení při časech multiplikace. Ostatní knihovny jsou napsané v jazyce C a nevykazují tak znatelné výkyvy na proudové ose. Výsledky proudového zatížení budou v jedné z následujících kapitol. Nadruhou stranu python má výhodu multiplatformnosti. Je lehce přenositelný v rámci různých operačních systémů, které podporují python, což je v dnešní době většina. Tudíž lze bez jakéhokoliv doinstalovávání přenést skript a spustit například na Windows či macOS.

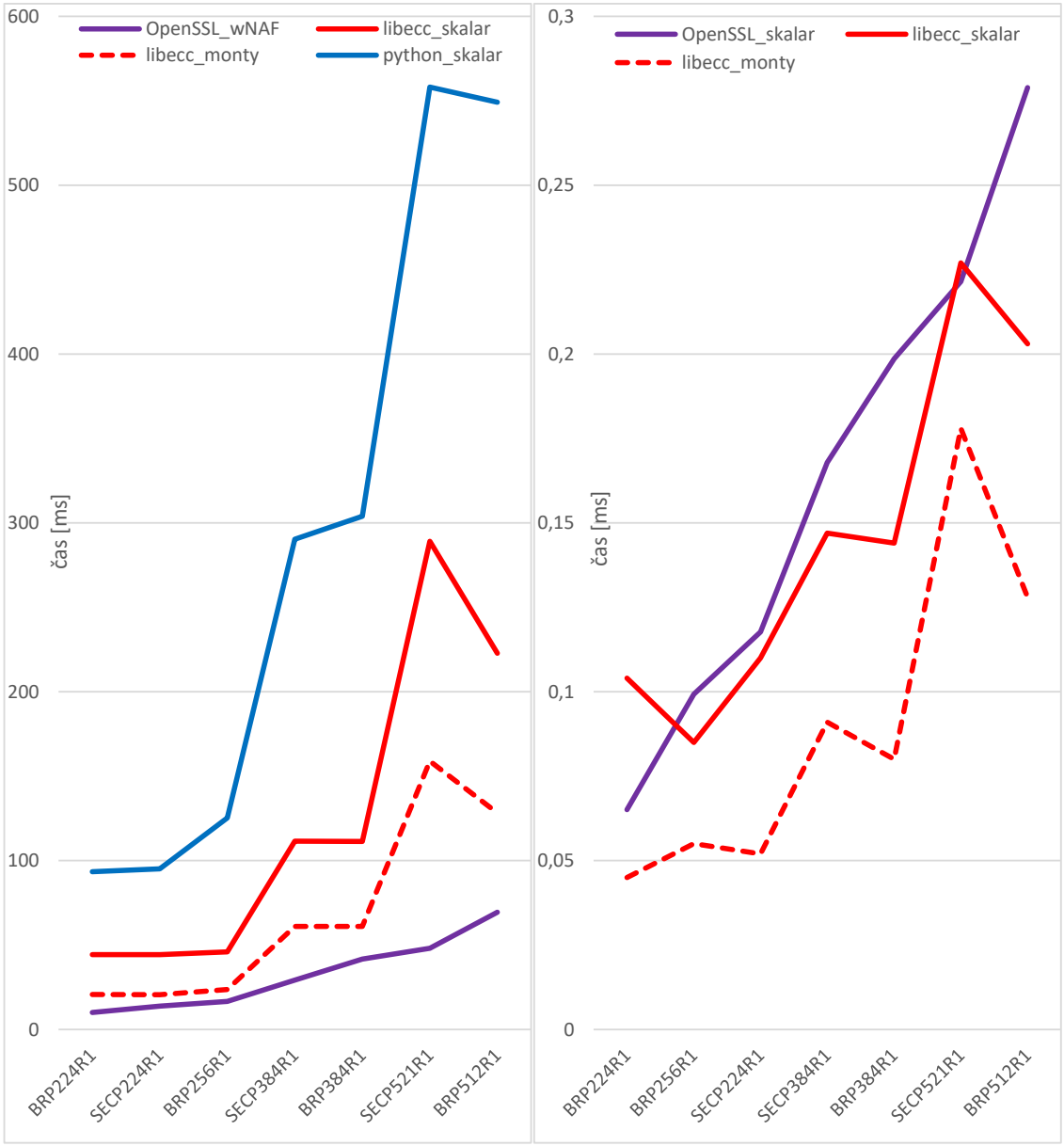
Výsledky měření multiplikace dvou bodů na eliptických křivkách standardů SECP a BRAINPOOL (v grafu jen BRP)(Obr. 5.10). Z grafu je vidět, že se zde opakuje stejný jev jako dříve a to, že křivky standardu SECP jsou rychlejší pří menším tělese (192b až 224b) a pomalejší při větším tělese (224b až 521b). Můžeme si zde všimnout také zajímavého úkazu, že na nejpomalejším zařízení se nejpomalejší křivka liší od ostatních zařízení (BRP512R1 vs SECP521R1). Originální grafy a tabulky jsou přiloženy v příloze a na CD pod názvem python_mereni.



Obr. 5.10: Měření multiplikace dvou bodů v pythonu metodou skalární.

5.4 Srovnání výsledků napříč knihovnami

Tato kapitola se zabývá srovnáním vybraných operací (multipling a addition) na stejných eliptických křivkách BRAINPOOL (v grafu jen BRP) 224R1, 256R1, 384R1, 512R1 a křivkách SECP 224R1, 384R1, 521R1. V knihovnách OpenSSL, libecc a python. OpenSSL měřilo metodou wNAF, libecc metodami skalární a Montgomeryho (v grafu monty) a python metodou skalární. Zarovnání hodnot (od nejrychlejší) je dle zařízení Raspberry Pi 3 (jako v předcházejících kapitolách). Výsledky měření jsou na následujícím grafu (Obr. 5.11).



Obr. 5.11: Srovnání výsledků napříč knihovnami.

5.5 Měření paměťové náročnosti multiplication

Toto měření probíhalo v knihovnách OpenSSL, libecc a python. Operace multiplikace na 7 křivkách standardu BRAINPOOL 224R1, 256R1, 384R1, 512R1 a standardu SECP 224R1, 384R1, 521R1. V následující tabulce jsou výsledky (Obr. 5.12).

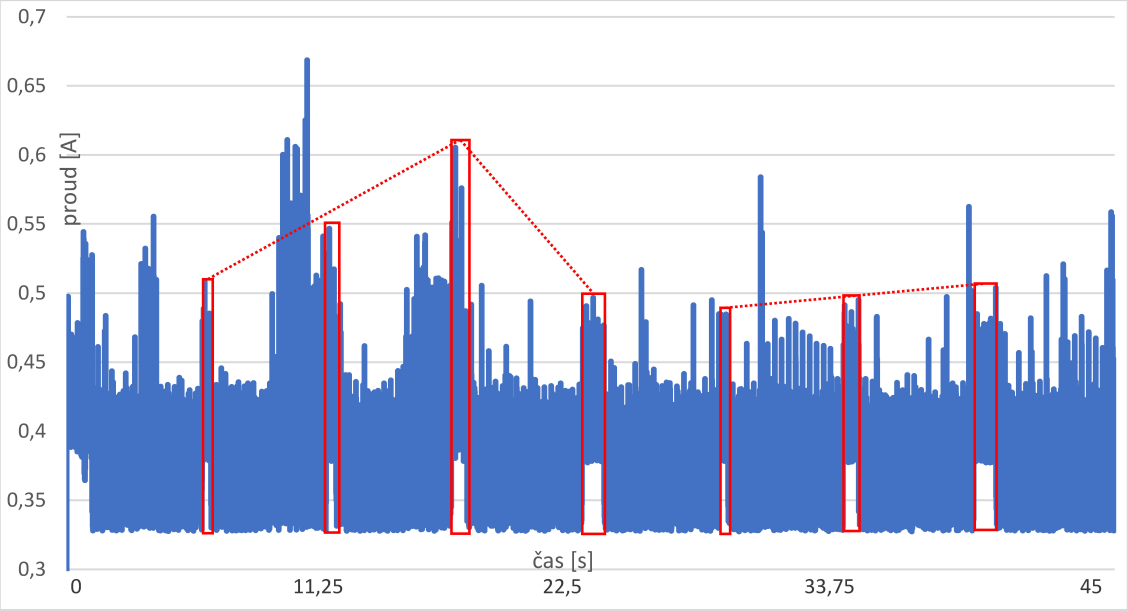
		paměťová náročnost [kB]	
knihovna	křivka	Raspberry Pi 3	Raspberry Pi 2
OpenSSL	BRP224R1	≈≤1000	≈≤1000
	BRP256R1		
	BRP384R1		
	BRP512R1		
	SECP224R1		
	SECP384R1		
	SECP521R1		
libecc	BRP224R1	≈≤1300	≈≤1300
	BRP256R1		
	BRP384R1		
	BRP512R1		
	SECP224R1		
	SECP384R1		
	SECP521R1		
python	BRP224R1	≈≤3000	≈≤3000
	BRP256R1		
	BRP384R1		
	BRP512R1		
	SECP224R1		
	SECP384R1		
	SECP521R1		

Obr. 5.12: Paměťové srovnání multiplication napříč knihovnami.

Z naměřených hodnot lze vyvodit, že nemají vliv parametry křivky ani velikost tělesa měřené křivky, ale pouze použitá kryptografická knihovna. OpenSSL obecně dává dobré výsledky (viz předcházející kapitoly) při menším paměťovém vytížení. Knihovna libecc není tak dobře optimalizována, avšak paměťové vytížení je jen nepatrně vyšší než u knihovny OpenSSL. Knihovna libecc dává také velmi dobré výsledky při relativně dobrém paměťovém zatížení. Měření v pythonu je nejpomalejší a má největší paměťové vytížení. Její výhoda je multiplatformnost, jelikož python lze zkompilovat bez jakéhokoliv instalování do jiných operačních systémů. Byla testována také na PC s Intel Core i5 a 8GB RAM a knihovna využívala nezávisle na křivce kolem 500 kB paměti – tedy 6x méně než na zařízeních Raspberry Pi.

5.6 Proudové zatížení v pythonu

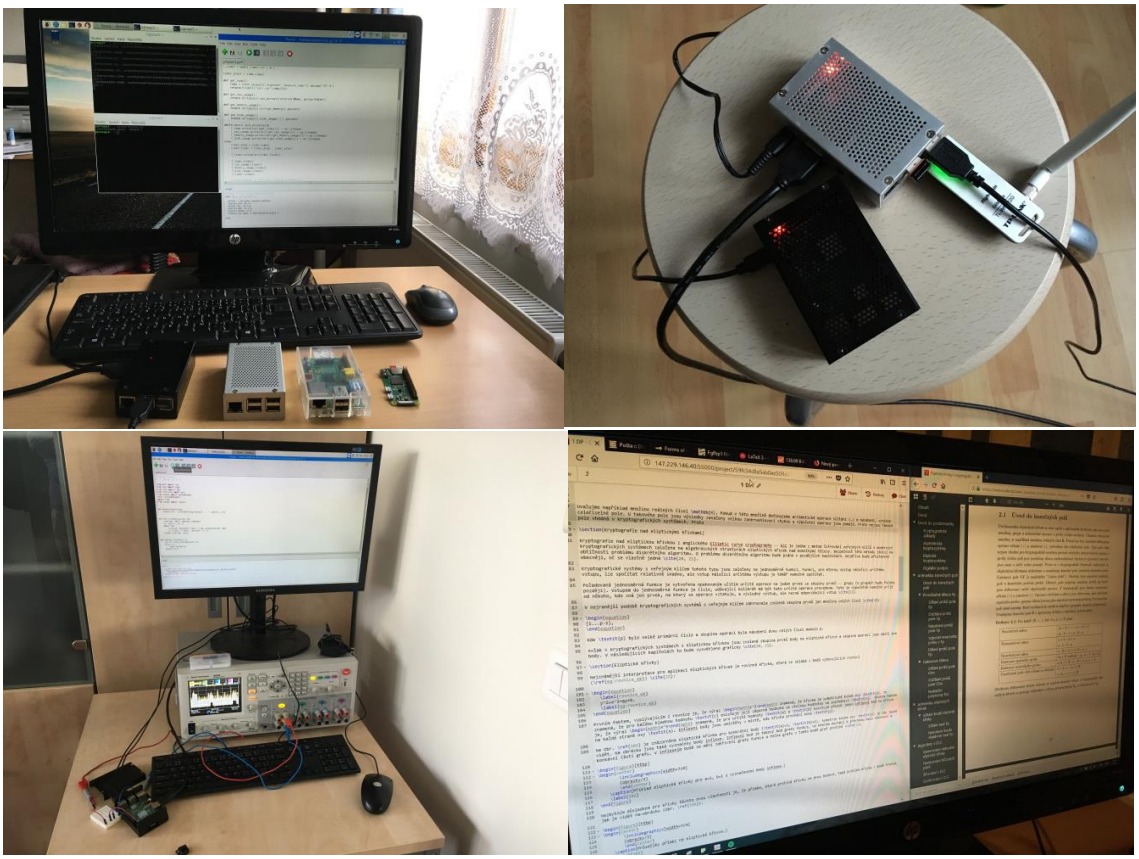
Křivky byly vybrány stejné jako u předcházejících kapitol z důvodu objektivního srovnání. Měření probíhalo následovně. Algoritmus byl upraven pro multiplikaci generátoru dané křivky a vždy vynásoben stejnou hodnotou. Ta byla vzata s nejmenší křivky (SECP224R1) – její souřadnice y křivky brainpoolP192r1, tedy 192bit číslo. Mezi měřeními byly z důvodu přehlednosti 5 s čekací intervaly. Jak je vidět na grafu Obr. 5.13, okna, kdy probíhalo samotné měření multiplikace, jsou vyznačena červeným obdélníkem. Mezi obdélníky jsou čekací intervaly. Je zde vidět určitá časová i proudová závislost u všech křivek s výjimkou čtvrté měřené křivky brainpool512r1. U této křivky je vidět časové náročnost větší než u předcházejících křivek standardu brainpool, avšak proudový pokles. Dále následuje měření třech křivek standardu SECP. Opět od nejmenší po největší. Zde je vidět již rovnoměrný narůst jak na časové ose, tak na proudové. Spojnice peaků, které pravděpodobně vykazovala operace multiplication, jsou spojeny červenou přerušovanou čarou. Nutno podotknout, že zobrazené peaky v oknech nemusí být pouze počítání multiplication, ale mohou obsahovat i vytížení jiných systémových procesů. Ačkoliv byl algoritmus optimalizován na co nejnižší úroveň zatížení systému např. operacemi jako je for, if, které byly z důvodů měření úplně odstraněny, můžeme vidět náhodné peaky i v čekacím intervalu. Byly naměřeny všechny knihovny, avšak měření proudového zatížení v knihovnách OpenSSL a libecc nevykazovalo takto přehledně tzv. okna. Z tohoto důvodu nebyly zahrnuty do této kapitoly, ale výsledky naměřených hodnot jsou v přílohách na CD.



Obr. 5.13: Proudové zatížení v pythonu.

5.7 Foto z měření

Tato kapitola obsahuje fotografie ze zrealizovaného měření (Obr. 5.14). Fotografie vlevo nahoře zobrazuje operační systém Raspbian, na kterém probíhala všechna měření na jednotlivých hardwarových zařízeních. Na fotce můžeme vidět zleva Raspberry Pi 3, 2, 1, Zero. Na fotografii je vidět vývojové prostředí ThonnyPython, na kterém byla programována měření v prostředí python a dvě otevřené konzole s probíhajícím měřením na eliptických křivkách. Na fotografii vpravo nahoře jsou vidět připojená zařízení Raspberry Pi 3 (černé) a 2 (stříbrné). K raspberry Pi 2 je připojen Wi-Fi dongle, který umožňuje komunikovat přes Wi-Fi. Na fotografii vlevo dole je znázorněno měření proudového vytížení na přístroji Agilent Technologies s označením N6705B a k němu připojené zařízení Raspberry Pi 3. Na monitoru měřící algoritmus z měření multiplication v pythonu na eliptických křivkách. Poslední fotografie vpravo dole zobrazuje samotné psaní práce v prostředí ShareLaTex a v pravé části fotografie zkompilované výsledné PDF.



Obr. 5.14: Foto z měření.

6 ZÁVĚR

V diplomové práci bylo úkolem implementovat na vybraných hardwarových platformách variaci kryptografických knihoven obsahujících primitiva pro eliptické křivky. Následně bylo nutno tyto knihovny patřičně upravit a vytvořit vlastní měřicí schémata tak, aby mohlo dojít k porovnání jednotlivých implementací. Hlavním úkolem tak bylo nejen implementovat knihovny, ale hlavně návrh a realizace testovacích scénářů společně s vytvořením měřicích metod pro různé knihovny a hardwarové platformy. Ve výsledku byla provedena řada experimentálních testů zaměřena na různé křivky i jejich parametry tak, aby výsledky práce zahrnovaly komplexně problematiku eliptických křivek v kryptografii. Hlavními parametry zde byly energetická, časová a paměťová náročnost.

První třetina teoretické části diplomové práce se zabývala úvodem do kryptografie a kryptografie eliptických křivek. Základními primitivy používanými v kryptografii, kryptografií nad eliptickými křivkami, aritmetikou grupy, aritmetikou na eliptických křivkách, primitivy na eliptických křivkách využitých v praktické části práce. Algoritmy multiplication využité v praktické části práce, diskrétními body na eliptické křivce a kryptografickou funkcí založenou na eliptických křivkách.

Druhá třetina teoretické části diplomové práce se zabývala vybranými eliptickými křivkami, knihovnami a hardwarovými platformami. Byl popsán a vysvětlen výběr eliptických křivek použitých v praktické čísti práce. Tato část se také zabývá parametrizací eliptických křivek standardů NIST, který se dělí na SECP a SECT křivky. Dále křivek standardů BRAINPOOL, GOST a FRP. Vybranými knihovnami a jejich implementací do hardwarových zařízení. Popisem vybraných hardwarových zařízení raspberry Pi Zero/1/2/3 a instalací těchto zařízení.

Třetí třetina teoretické části diplomové práce se zabývala přípravou měření kryptografických primitiv na eliptických křivkách. Práce se zde zabývala způsoby měření kryptografických knihoven, měřicími metodami, popisem použitých algoritmů. Měřením časové a paměťové náročnosti kryptografických primitiv a dalších.

První třetina praktické části diplomové práce se zabývala kritickou rešerší kryptografických knihoven, ve které byly popsány vlastnosti a využití daných knihoven. Jednalo se o kryptografické knihovny OpenSSL, libecc, TinyECC,WolfSSL, AvrCryptoLib, WiseLib, Crypto++, LibTom-Crypt a MIRACL. Praktické měření bylo realizováno na třech knihovnách – OpenSSL, libecc a měření v pythonu. Měření v pythonu bylo realizováno dostupnými metodami v tomto programovacím prostředí, nikoliv speciální knihovnou. Obsahuje zajímavé porovnání optimalizovaných knihoven napsaných v jazyku C a python.

Druhá třetina praktické části diplomové práce se zabývala sestrojením multiplatformního měřícího algoritmu pro měření kryptografických primitiv na eliptických

křivkách v kryptografických knihovnách. Algoritmus je sestrojen v jazyku python a je multiplatformní, jelikož python je přenositelný z platformy na platformu. Byl sestrojen algoritmus na měření využití systémových prostředků – procesor, paměti RAM a FLASH. Dále byl navržen algoritmus pro měření času a teploty hardwaru. Využito bylo po konzultaci s vedoucím diplomové práce pouze časové, paměťové (RAM) a proudové měření. Proudové měření bylo realizováno fyzickým připojením na zařízení od společnosti Agilent Technologies s označením N6705B. Dále bylo realizováno měření procesorových cyklů, přepočtením časové závislosti a parametrů procesoru na procesorové cykly.

Třetí třetina praktické části diplomové práce obsahuje výsledné hodnoty a grafy z časového a proudového měření. V tabulce byly zpracovány hodnoty z paměťového měření. Ostatní hodnoty a grafy se nachází v přílohách na CD. Ke každé kapitole věnované jednomu typu kryptografického primitiva jsou grafy a jejich slovní diskuze. Na závěr práce obsahuje průřez naměřenými hodnotami z vybraných kryptografických primitiv (multiplication, addition) na vybraných křivkách a ve všech měřených knihovnách a zařízeních (Raspberry Pi Zero/1/2/3). Autor práce považuje zadání za splněné, jelikož byla zrealizována a změřena všechna možná kryptografická primitiva, která se standardně na eliptických křivkách používají v kryptografických protokolech jako je například ECDH, ECIES, ECDSA, ECMQV, ECQV. Byla zrealizována měření paměťová, časová i energetická, jak požaduje zadání. Bylo zrealizováno celkem 14 typů měření na různých knihovnách a 48 eliptických křivkách (včetně křivek GOST a FRP, které nejsou parametricky popsány v příloze diplomové práce, jelikož jsou implementovány pouze v knihovně OpenSSL). Tato měření byla zrealizována na hardwarových platformách Raspberry Pi Zero, 1, 2 a 3. V tištěné příloze diplomové práce jsou uvedeny parametry eliptických křivek SECT, SECP a BRAINPOOL použitých v kryptografických měřících algoritmech. Dále seznam příloh na CD.

LITERATURA

[1] MAO, W. *Modern Cryptography: Theory and Practice.*. 2004, ISBN 1-306-6943-1. [cit. 2. 10. 2017].

[2] SCHNEIER, B. *Applied Cryptography.*. 1994, ISBN SBN 0-471-59756-2. [cit. 2. 10. 2017].

[3] FUJDIAK, R. *Analýza a optimalizace datové komunikace pro telemetrické systémy v energetice).* 2017 [cit. 8. 12. 2017]. Dostupné z URL: <https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=157492>.

[4] HANKERSON, D; MENEZES, A; VANSTONE, S. *Guide to Elliptic Curve Cryptography*. 2004, ISBN 978-0387952734 [cit. 2. 10. 2017].

[5] Rouse, M. *Elliptical curve cryptography (ECC)* [online]. 2005, poslední aktualizace 2005 [cit. 2. 10. 2017]. Dostupné z URL: <http://searchsecurity.techtarget.com/definition/elliptical-curve-cryptography>.

[6] ANOOP, M. *ECC* [online]. 2015, poslední aktualizace 2015 [cit. 2. 10. 2017]. Dostupné z URL: <https://www.johannes-bauer.com/compsci/ecc>.

[7] KHAMSI, A. *Concavity and Points of Inflection* [online]. 1999, poslední aktualizace 2017 [cit. 2. 10. 2017]. Dostupné z URL: <http://www.sosmath.com/calculus/diff/der15/der15.html>.

[8] GNU. *Elliptic curve arithmetic* [online]. 2017, poslední aktualizace 2017 [cit. 2. 10. 2017]. Dostupné z URL: <https://rosettacode.org/wiki/Elliptic_curve_arithmetic>.

[9] GARCIA, J. *Elliptic Curve Crypto ,Point Doubling* [online]. 2017, poslední aktualizace 2017 [cit. 2. 10. 2017]. Dostupné z URL: <https://hackernoon.com/elliptic-curve-crypto-point-doubling-b98508d40a88>.

[10] LYNN, J. *Crypto.stanford.edu* [online]. 2018, poslední aktualizace 2018 [cit. 17. 3. 2017]. Dostupné z URL: <https://crypto.stanford.edu/pbc/notes/elliptic/explicit.html>.

[11] CORBELLINI, A. *Elliptic Curve Cryptography: finite fields and discrete logarithms* [online]. 2015, poslední aktualizace 2015 [cit. 2. 10. 2017]. Dostupné z URL: <http://andrea.corbellini.name/2015/05/23/elliptic-curve-cryptography-finite-fields-and-discrete-logarithms>.

[12] JOHNSTON, O. *A Discrete Logaritm Attack On Elliptic Curves* [online]. 2010, poslední aktualizace 2010 [cit. 2. 10. 2017]. Dostupné z URL: <https://eprint.iacr.org/2010/575.pdf>.

[13] JOHNSTON, O. *Elliptic Curve Cryptosystems* [online]. 2010, poslední aktualizace 2010 [cit. 2. 10. 2017]. Dostupné z URL: <https://eprint.iacr.org/2010/575.pdf>.

[14] BOS, W; HALDERMAN, A; HENINGER, N. *Elliptic Curve Cryptography in Practice* [online]. 2013, poslední aktualizace 2013 [cit. 2. 10. 2017]. Dostupné z URL: <https://eprint.iacr.org/2013/734.pdf>.

[15] TREMEL, E. *Real-World Performance of Cryptographic Accumulators* [online]. 2013, poslední aktualizace 2013 [cit. 2. 10. 2017]. Dostupné z URL: <https://cs.brown.edu/research/pubs/theses/ugrad/2013/tremel.pdf>.

[16] ZYGO T. *ECDH* [online]. 2017, poslední aktualizace 2017 [cit. 2. 10. 2017]. Dostupné z URL: <https://zygoteiot.files.wordpress.com/2017/03/ecdh.png?w=700>.

[17] ZYGOTE. *A PRIMER TO ELLIPTIC CURVE DIFFIE-HELLMAN* [online]. 2017, poslední aktualizace 2017 [cit. 2. 10. 2017]. Dostupné z URL: <https://zygoteiot.wordpress.com/2017/03/16/a-primer-to-elliptic-curve-diffie-hellman>.

[18] CYBER, DL. *TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks (Version 2.0)* [online]. 2010, poslední aktualizace 2011 [cit. 9. 10. 2017]. Dostupné z URL: <http://discovery.csc.ncsu.edu/software/TinyECC>.

[19] BLAKE, I; SERROUSI, G; a SMART, N. *Elliptic Curves in Cryptography*. 1999, Cambridge University Press London Mathematical Society Lecture Note Series 265 [cit. 9. 10. 2017].

[20] GURA, N; PATEL, A; a WANDER, A. *Comparing elliptic curve cryptography and RSA on 8-bit CPUs*. 2004, Workshop on Cryptographic Hardware and Embedded Systems [cit. 9. 10. 2017].

[21] CETIN, K. *High-Speed RSA Implementation*. 1994, RSA Laboratories Technical Report TR-201, Version 2.0 [cit. 9. 10. 2017].

[22] ATMEL. *8-bit AVR Instruction Set* [online]. 2016, poslední aktualizace 2016 [cit. 9. 10. 2017]. Dostupné z URL: <http://www.atmel.com/dyn/resources/prod_documents/DOC0856.PDF>.

[23] HANKERSON, D; MENEZENS, A; VANSTONE, S. *Guide to Elliptic Curve Cryptography*. 2004, Springer [cit. 9. 10. 2017].

[24] RSA, L. *RSAREF: A cryptographic toolkit (version 2.0)*. 1994 [cit. 9. 10. 2017].

[25] CERTICOM, R. *Standards for efficient cryptography - SEC2: Recommended elliptic curve domain parameters* [online]. 2000, poslední aktualizace 2000 [cit. 9. 10. 2017]. Dostupné z URL: <`http://www.secg.org/download/aid-386/sec2_final.pdf`>.

[26] EASTLAKE, D; JONES, P. *US Secure Hash Algorithm 1 (SHA1)*. 2001, RFC 3174 [cit. 9. 10. 2017].

[27] MENEZES, A; OORSCHOT P, VANSTONE S. *Handbook of Applied Cryptography*. 1997, CRC Press [cit. 2. 10. 2017]. Dostupné z URL: <`https://zygoteiot.wordpress.com/2017/03/16/a-primer-to-elliptic-curve-diffie-hellman`>.

[28] TEXAS, I. *MSP430x1xx Family User's Guide (Rev. F)* [online]. 2006, poslední aktualizace 2006 [cit. 9. 10. 2017]. Dostupné z URL: <`http://focus.ti.com/lit/ug/slau049f/slau049f.pdf`>.

[29] *ARM v5TE Architecture Reference Manual* [online]. 2010, poslední aktualizace 2010 [cit. 9. 10. 2017]. Dostupné z URL: <`http://www.arm.com/documentation/Instruction_Set/index.html`>.

[30] *Crossbow* [online]. 2015 poslední aktualizace 2015 [cit. 9. 10. 2017]. Dostupné z URL: <`http://www.xbow.com`>.

[31] *OpenSSL* [online]. 1999, poslední aktualizace 2016 [cit. 9. 10. 2017]. Dostupné z URL: <`https://www.openssl.org/source/`>.

[32] *OpenSSL library* [online]. 2017, poslední aktualizace 2017 [cit. 9. 10. 2017]. Dostupné z URL: <`https://github.com/openssl/openssl`>.

[33] *libecc library* [online]. 2017, poslední aktualizace 2017 [cit. 14. 3. 2018]. Dostupné z URL: <`https://github.com/ANSSI-FR/libecc`>.

[34] *wolfSSL* [online]. 2017, poslední aktualizace 2017 [cit. 9. 10. 2017]. Dostupné z URL: <`https://www.wolfssl.com`>.

[35] *wolfSSL library* [online]. 2017, poslední aktualizace 2017 [cit. 9. 10. 2017]. Dostupné z URL: <`https://github.com/wolfSSL/wolfssl`>.

[36] *AVR-Crypto-Lib* [online]. 2014, poslední aktualizace 2014 [cit. 9. 10. 2017]. Dostupné z URL: <https://wiki.das-labor.org/w/AVR-Crypto-Lib/en>.

[37] *AVR-Crypto-Lib library* [online]. 2017, poslední aktualizace 2017 [cit. 9. 10. 2017]. Dostupné z URL: <https://github.com/Emilevanderlaan/AVR_CryptoLib>.

[38] *wiselib* [online]. 2017, poslední aktualizace 2017 [cit. 9. 10. 2017]. Dostupné z URL: <https://github.com/ibr-alg/wiselib>.

[39] DAI, W. *Crypto++® Library 5.6.5* [online]. 2017, poslední aktualizace 2017 [cit. 9. 10. 2017]. Dostupné z URL: <https://www.cryptopp.com>.

[40] *LibTomCrypt* [online]. 2017, poslední aktualizace 2017 [cit. 9. 10. 2017]. Dostupné z URL: <http://www.libtom.net/about>.

[41] *LibTomCrypt library* [online]. 2017, poslední aktualizace 2017 [cit. 9. 10. 2017]. Dostupné z URL: <https://github.com/libtom/libtomcrypt>.

[42] *MIRACL* [online]. 2017, poslední aktualizace 2017 [cit. 9. 10. 2017]. Dostupné z URL: <https://www.miracl.com/about-miracl>.

[43] *MIRACL library* [online]. 2017, poslední aktualizace 2017 [cit. 9. 10. 2017]. Dostupné z URL: <https://github.com/miracl/MIRACL>.

[44] *SEC 2: Recommended Elliptic Curve Domain Parameters* [online]. 2010, poslední aktualizace 2010 [cit. 3. 3. 2018]. Dostupné z URL: <http://www.secg.org/sec2-v2.pdf>.

[45] *Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation* [online]. 2010, poslední aktualizace 2017 [cit. 3. 3. 2018]. Dostupné z URL: <https://tools.ietf.org/html/rfc5639>.

[46] *GNU General Public License (Všeobecná veřejná licence GNU), verze 2* [online]. 2017, poslední aktualizace 2017 [cit. 9. 10. 2017]. Dostupné z URL: <https://gnu.org/licenses/old-licenses/gpl-2.0.html>.

[47] *Raspberry Pi Zero W* [online]. 2017, poslední aktualizace 2017 [cit. 23. 10. 2017]. Dostupné z URL: <http://rpishop.cz/rpi-zero-sady/647-raspberry-pi-zero-w.html>.

[48] *Raspberry Pi 1 Model B 512MB RAM Rev. 2.0* [online]. 2012, poslední aktualizace 2012 [cit. 23. 10. 2017]. Dostupné z URL: <http://rpishop.cz/raspberry-pi-pocitace/8-raspberry-pi-0766897151323.html>.

[49] *Raspberry Pi 2 Model B 1GB* [online]. 2015, poslední aktualizace 2015 [cit. 23. 10. 2017]. Dostupné z URL: <http://rpishop.cz/raspberry-pi-pocitace/170-raspberry-pi-2-1024-mb-ram.html>.

[50] *Raspberry Pi 3 Model B 64-bit 1GB RAM* [online]. 2016, poslední aktualizace 2016 [cit. 23. 10. 2017]. Dostupné z URL: <http://rpishop.cz/kategorie/283-raspberry-pi-3-model-b-64-bit.html>.

[51] *Raspberry Pi Zero W* [online]. 2017, poslední aktualizace 2017 [cit. 23. 10. 2017]. Dostupné z URL: <http://rpishop.cz/2458-thickbox_default/raspberry-pi-zero-w.jpg>.

[52] *Raspberry Pi 1 Model B 512MB RAM Rev. 2.0* [online]. 2012, poslední aktualizace 2017 [cit. 23. 10. 2017]. Dostupné z URL: <http://rpishop.cz/1845-thickbox_default/raspberry-pi.jpg>.

[53] *Raspberry Pi 2 Model B 1GB* [online]. 2015, poslední aktualizace 2015 [cit. 23. 10. 2017]. Dostupné z URL: <http://rpishop.cz/1849-thickbox_default/raspberry-pi-2-1024-mb-ram.jpg>.

[54] *Raspberry Pi 3 Model B 64-bit 1GB RAM* [online]. 2016, poslední aktualizace 2016 [cit. 23. 10. 2017]. Dostupné z URL: <http://rpishop.cz/1851-thickbox_default/raspberry-pi-3-model-b-64-bit.jpg>.

[55] VUT, BRNO. *Proceedings of the 24th Conference STUDENT EEICT 2018* [online]. 2018, poslední aktualizace 2018, ISBN 978-80-214-5614-3. [cit. 19. 4. 2018].

[56] *WolfSSL* [online]. 2017, poslední aktualizace 2017 [cit. 28. 11. 2017]. Dostupné z URL: <http://wolfssl.com/wolfSSL/download/downloadForm.php>.

[57] *OpenSSL* [online]. 2017, poslední aktualizace 2017 [cit. 28. 11. 2017]. Dostupné z URL: <https://www.openssl.org/source>.

[58] *NOOBS* [online]. 2017, poslední aktualizace 2017 [cit. 30. 10. 2017]. Dostupné z URL: <https://www.raspberrypi.org/downloads/noobs>.

[59] *Welcome to PyPy* [online]. 2017, poslední aktualizace 2017 [cit. 6. 11. 2017]. Dostupné z URL: <http://pypy.org>.

[60] RODOLA, G. *About Giampaolo Rodolà* [online]. 2014, poslední aktualizace 2014 [cit. 6. 11. 2017]. Dostupné z URL: <http://grodola.blogspot.com/p/about.html>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

AES Advanced Encryption Standard
API Aplication User Interface
BRP Brainpool křivka
CGAL The Computational Geometry Algorithms Library
DES Data Encryption Standard
ECC Elliptic Curve Cryptography
ECDH Elliptic Curve Diffie-Hellman
ECDS Elliptic Curve Digital Signature Algorithm
ECDSA Elliptic Curve Digital Signature Algorithm
ECIES Elliptic Curve Integrated Encryption Scheme
ECMQV Elliptic Curve Menezes-Qu-Vanstone
ECQV Elliptic Curve Qu-Vanston
GF Galoi's Field
GNU General Public License
MD5 Message-Digest algorithm
nesC The National Electrical Safety Code
NIST National Institute of Standards and Technology
PRNG Pseudorandom Number Generator
RFC Request For Comments
RSAREF RSA Reference
SECG Standards for Efficient Cryptography Group
SECP Securities & Exchange Commission of Pakistan
SSL Secure Sockets Layer
TDES Triple DES
TinyECC Tiny Eliptic Curves
XTEA eXtended TEA
wNAF W-ary Non Adjacent Form
WTFPL Do What the Fuck You Want To Public License

SEZNAM PŘÍLOH

A SEZNAM POUŽITÝCH KŘIVEK A JEJICH PARAMETRIZACE

```
# secp192k1
secp192k1 = ECcurve()
secp192k1.p = \
    hex2int('FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFEE37')
secp192k1.a = \
    hex2int('00000000 00000000 00000000 00000000 00000000 00000000')
secp192k1.b = \
    hex2int('00000000 00000000 00000000 00000000 00000000 00000003')
secp192k1.n = \
    hex2int('FFFFFFFF FFFFFFFF FFFFFFFE 26F2FC17 0F69466A 74DEFD8D')
secp192k1.G = ECpoint(curve=secp192k1,
                      x=hex2int('DB4FF10E C057E9AE 26B07D02 80B7F434 1DA5D1B1 EAE06C7D'
                      ),
                      y=hex2int('9B2F2F6D 9C5628A7 844163D0 15BE8634 4082AA88 D95E2F9D'
                      ))
# secp192r1
secp192r1 = ECcurve()
secp192r1.p = \
    hex2int('FFFFFFFF FFFFFFFF FFFFFFFE FFFFFFFF FFFFFFFF')
secp192r1.a = \
    hex2int('FFFFFFFF FFFFFFFF FFFFFFFE FFFFFFFF FFFFFFFC')
secp192r1.b = \
    hex2int('64210519 E59C80E7 0FA7E9AB 72243049 FEB8DEEC C146B9B1')
secp192r1.n = \
    hex2int('FFFFFFFF FFFFFFFF FFFFFFFF 99DEF836 146BC9B1 B4D22831')
secp192r1.G = ECpoint(curve=secp192r1,
                      x=hex2int('188DA80E B03090F6 7CBF20EB 43A18800 F4FF0AFD 82FF1012'
                      ),
                      y=hex2int('07192B95 FFC8DA78 631011ED 6B24CDD5 73F977A1 1E794811'
                      ))
# secp224k1
secp224k1 = ECcurve()
secp224k1.p = \
    hex2int('FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFE56D'
            )
secp224k1.a = \
    hex2int('00000000 00000000 00000000 00000000 00000000 00000000 00000000'
            )
secp224k1.b = \
    hex2int('00000000 00000000 00000000 00000000 00000000 00000000 00000005'
            )
secp224k1.n = \
    hex2int('00000000 00000000 00000000 0001DCE8 D2EC6184 CAF0A971 769FB1F7'
            )
secp224k1.G = ECpoint(curve=secp224k1,
                      x=hex2int('A1455B33 4DF099DF 30FC28A1 69A467E9 E47075A9 0F7E650E B6B7A45C'
                      ),
                      y=hex2int('7E089FED 7FBA3442 82CAFBD6 F7E319F7 C0B0BD59 E2CA4BDB 556D61A5'
                      ))
# secp224r1
secp224r1 = ECcurve()
secp224r1.p = \
    hex2int('FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 00000000 00000001'
            )
secp224r1.a = \
    hex2int('FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFE'
            )
secp224r1.b = \
    hex2int('B4050A85 0C04B3AB F5413256 5044B0B7 D7BFD8BA 270B3943 2355FFB4'
            )
secp224r1.n = \
    hex2int('FFFFFFFF FFFFFFFF FFFFFFFF FFFF16A2 E0B8F03E 13DD2945 5C5C2A3D'
            )
secp224r1.G = ECpoint(curve=secp224r1,
                      x=hex2int('B70E0CBD 6BB4BF7F 321390B9 4A03C1D3 56C21122 343280D6 115C1D21'
                      ),
                      y=hex2int('BD376388 B5F723FB 4C22DFE6 CD4375A0 5A074764 44D58199 85007E34'
                      ))
# secp256k1
secp256k1 = ECcurve()
secp256k1.p = \
    hex2int('FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFC2F'
            )
```

```
secp256k1.a = \
    hex2int('FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFC2F'
            )
secp256k1.b = \
    hex2int('00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000007'
            )
secp256k1.n = \
    hex2int('FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141'
            )
secp256k1.G = ECpoint(curve=secp256k1,
                      x=hex2int('79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798'
                      ),
                      y=hex2int('483ADA77 26A3C465 5DA4FBFC 0E1108A8 FD17B448 A6855419 9C47D08F FB10D4B8'
                      ))
# secp256r1
secp256r1 = ECcurve()
secp256r1.p = \
    hex2int('FFFFFFFF 00000001 00000000 00000000 FFFFFFFF FFFFFFFF FFFFFFFF'
            )
secp256r1.a = \
    hex2int('FFFFFFFF 00000001 00000000 00000000 FFFFFFFF FFFFFFFF FFFFFFFC'
            )
secp256r1.b = \
    hex2int('5AC635D8 AA3A93E7 B3EBBD55 769886BC 651D06B0 CC53B0F6 3BCE3C3E 27D2604B'
            )
secp256r1.n = \
    hex2int('FFFFFFFF 00000000 FFFFFFFF FFFFFFFF BCE6FAAD A7179E84 F3B9CAC2 FC632551'
            )
secp256r1.G = ECpoint(curve=secp256r1,
                      x=hex2int('6B17D1F2 E12C4247 F8BCE6E5 63A440F2 77037D81 2DEB33A0 F4A13945 D898C296'
                      ),
                      y=hex2int('4FE342E2 FE1A7F9B 8EE7EB4A 7C0F9E16 2BCE3357 6B315ECE CBB64068 37BF51F5'
                      ))
# secp384r1
secp384r1 = ECcurve()
secp384r1.p = \
    hex2int('FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFFFF 00000000
    00000000 FFFFFFFF'
            )
secp384r1.a = \
    hex2int('FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFFFF 00000000
    00000000 FFFFFFFC'
            )
secp384r1.b = \
    hex2int('B3312FA7 E23EE7E4 988E056B E3F82D19 181D9C6E FE814112 0314088F 5013875A C656398D 8A2ED19D
    2A85C8ED D3EC2AEF'
            )
secp384r1.n = \
    hex2int('FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF C7634D81 F4372DDF 581A0DB2 48B0A77A
    ECEC196A CCC52973'
            )
secp384r1.G = ECpoint(curve=secp384r1,
                      x=hex2int('AA87CA22 BE8B0537 8EB1C71E F320AD74 6E1D3B62 8BA79B98 59F741E0 82542A38
                      5502F25D BF55296C 3A545E38 72760AB7'
                      ),
                      y=hex2int('3617DE4A 96262C6F 5D9E98BF 9292DC29 F8F41DBD 289A147C E9DA3113 B5F0B8C0
                      0A60B1CE 1D7E819D 7A431D7C 90EA0E5F'
                      ))
# secp521r1
secp521r1 = ECcurve()
secp521r1.p = \
    hex2int('01FF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
    FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF'
            )
secp521r1.a = \
    hex2int('01FF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
    FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFC'
            )
secp521r1.b = \
    hex2int('0051 953EB961 8E1C9A1F 929A21A0 B68540EE A2DA725B 99B315F3 B8B48991 8EF109E1 56193951
    EC7E937B 1652C0BD 3BB1BF07 3573DF88 3D2C34F1 EF451FD4 6B503F00'
            )
secp521r1.n = \
    hex2int('01FF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFA 51868783
    BF2F966B 7FCC0148 F709A5D0 3BB5C9B8 899C47AE BB6FB71E 91386409'
            )
secp521r1.G = ECpoint(curve=secp521r1,
                      x=hex2int('00C6858E 06B70404 E9CD9E3E CB662395 B4429C64 8139053F B521F828
                      AF606B4D 3DBAA14B 5E77EFE7 5928FE1D C127A2FF A8DE3348 B3C1856A 429BF97E 7E31C2E5 BD66'
                      ),
```

```
                    y=hex2int('0118 39296A78 9A3BC004 5C8A5FB4 2C7D1BD9 98F54449 579B4468 17AFBD17
                    273E662C 97EE7299 5EF42640 C550B901 3FAD0761 353C7086 A272C240 88BE9476 9FD16650'
                    ))
# sect163k1
sect163k1 = ECcurve()
sect163k1.a = \
    hex2int('00000000 00000000 00000000 00000000 00000001')
sect163k1.b = \
    hex2int('00000000 00000000 00000000 00000000 00000001')
sect163k1.n = \
    hex2int('00000000 00000000 00020108 A2E0CC0D 99F8A5EF')
sect163k1.G = ECpoint(curve=sect163k1,
                    x=hex2int('13C0537B BC11ACAA 07D793DE 4E6D5E5C 94EEE802'
                    ),
                    y=hex2int('89070FB0 5D38FF58 321F2E80 0536D538 CCDAA3D9'
                    ))
# sect163r1
sect163r1 = ECcurve()
sect163r1.a = \
    hex2int('B6882CAA EFA84F95 54FF8428 BD88E246 D2782AE2')
sect163r1.b = \
    hex2int('13612DCD DCB40AAB 946BDA29 CA91F73A F958AFD9')
sect163r1.n = \
    hex2int('FFFFFFFF FFFFFFFF FFFF48AA B689C29C A710279B')
sect163r1.G = ECpoint(curve=sect163r1,
                    x=hex2int('979697AB 43897789 56678956 7F787A78 76A65400'
                    ),
                    y=hex2int('435EDB42 EFAFB298 9D51FEFC E3C80988 F41FF883'
                    ))
# sect163r2
sect163r2 = ECcurve()
sect163r2.a = \
    hex2int('00000000 00000000 00000000 00000000 00000001')
sect163r2.b = \
    hex2int('0A601907 B8C953CA 1481EB10 512F7874 4A3205FD')
sect163r2.n = \
    hex2int('00000000 00000000 000292FE 77E70C12 A4234C33')
sect163r2.G = ECpoint(curve=sect163r2,
                    x=hex2int('EBA16286 A2D57EA0 991168D4 994637E8 343E3600'
                    ),
                    y=hex2int('D51FBC6C 71A0094F A2CDD545 B11C5C0C 797324F1'
                    ))
# sect233k1
sect233k1 = ECcurve()
sect233k1.a = \
    hex2int('00000000 00000000 00000000 00000000 00000000 00000000 00000000')
sect233k1.b = \
    hex2int('00000000 00000000 00000000 00000000 00000000 00000000 00000001')
sect233k1.n = \
    hex2int('00000000 00000000 00000000 00069D5B B915BCD4 6EFB1AD5 F173ABDF')
sect233k1.G = ECpoint(curve=sect233k1,
                    x=hex2int('017232BA 853A7E73 1AF129F2 2FF41495 63A419C2 6BF50A4C 9D6EEFAD 6126'
                    ),
                    y=hex2int('01DB 537DECE8 19B7F70F 555A67C4 27A8CD9B F18AEB9B 56E0C110 56FAE6A3'
                    ))
# sect233r1
sect233r1 = ECcurve()
sect233r1.a = \
    hex2int('00000000 00000000 00000000 00000000 00000000 00000000 00000001')
sect233r1.b = \
    hex2int('647EDE6C 332C7F8C 0923BB58 213B333B 20E9CE42 81FE115F 7D8F90AD')
sect233r1.n = \
    hex2int('00000000 00000000 00000000 0013E974 E72F8A69 22031D26 03CFE0D7')
sect233r1.G = ECpoint(curve=sect233r1,
                    x=hex2int('00FAC9DF CBAC8313 BB2139F1 BB755FEF 65BC391F 8B36F8F8 EB7371FD 558B'
                    ),
                    y=hex2int('0100 6A08A419 03350678 E58528BE BF8A0BEF F867A7CA 36716F7E 01F81052'
                    ))
# sect239k1
sect239k1 = ECcurve()
sect239k1.a = \
    hex2int('00000000 00000000 00000000 00000000 00000000 00000000 00000000')
sect239k1.b = \
    hex2int('00000000 00000000 00000000 00000000 00000000 00000000 00000001')
sect239k1.n = \
    hex2int('00000000 00000000 00000000 005A79FE C67CB6E9 1F1C1DA8 00E478A5')
sect239k1.G = ECpoint(curve=sect239k1,
                    x=hex2int('29A0B6A8 87A983E9 730988A6 8727A8B2 D126C44C C2CC7B2A 65551930 35DC'
                    ),
                    y=hex2int('7631 0804F12E 549BDB01 1C103089 E73510AC B275FC31 2A5DC6B7 6553F0CA'
```

```
                    ))
# sect283k1
sect283k1 = ECcurve()
sect283k1.a = \
    hex2int('00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000')
sect283k1.b = \
    hex2int('00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000001')
sect283k1.n = \
    hex2int('01FFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFE9AE 2ED07577 265DFF7F 94451E06 1E163C61')
sect283k1.G = ECpoint(curve=sect283k1,
                    x=hex2int('0503213F 78CA4488 3F1A3B81 62F188E5 53CD265F 23C1567A 16876913 B0C2AC24
                    58492836'
                    ),
                    y=hex2int('01CCDA38 0F1C9E31 8D90F95D 07E5426F E87E45C0 E8184698 E4596236 4E341161
                    77DD2259'
                    ))
# sect283r1
sect283r1 = ECcurve()
sect283r1.a = \
    hex2int('00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000001')
sect283r1.b = \
    hex2int('027B680A C8B8596D A5A4AF8A 19A0303F CA97FD76 45309FA2 A581485A F6263E31 3B79A2F5')
sect283r1.n = \
    hex2int('03FFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFEF90 399660FC 938A9016 5B042A7C EFADB307')
sect283r1.G = ECpoint(curve=sect283r1,
                    x=hex2int('05F93925 8DB7DD90 E1934F8C 70B0DFEC 2EED25B8 557EAC9C 80E2E198 F8CDBECD
                    86B12053'
                    ),
                    y=hex2int('03676854 FE24141C B98FE6D4 B20D02B4 516FF702 350EDDB0 826779C8 13F0DF45
                    BE8112F4'
                    ))
# sect409k1
sect409k1 = ECcurve()
sect409k1.a = \
    hex2int('00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000')
sect409k1.b = \
    hex2int('00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000001')
sect409k1.n = \
    hex2int('7FFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFE5F 83B2D4EA 20400EC4 557D5ED3
    E3E7CA5B 4B5C83B8 E01E5FCF')
sect409k1.G = ECpoint(curve=sect409k1,
                    x=hex2int('0060F05F 658F49C1 AD3AB189 0F718421 0EFD0987 E307C84C 27ACCFB8 F9F67CC2
                    C460189E B5AAAA62 EE222EB1 B35540CF E9023746'
                    ),
                    y=hex2int('01E36905 0B7C4E42 ACBA1DAC BF04299C 3460782F 918EA427 E6325165 E9EA10E3
                    DA5F6C42 E9C55215 AA9CA27A 5863EC48 D8E0286B'
                    ))
# sect409r1
sect409r1 = ECcurve()
sect409r1.a = \
    hex2int('00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000001')
sect409r1.b = \
    hex2int(0021A5C2 C8EE9FEB 5C4B9A75 3B7B476B 7FD6422E F1F3DD67 4761FA99 D6AC27C8 A9A197B2 72822F6C
    D57A55AA 4F50AE31 7B13545F')
sect409r1.n = \
    hex2int('01000000 00000000 00000000 00000000 00000000 000001E2 AAD6A612 F33307BE 5FA47C3C
    9E052F83 8164CD37 D9A21173')
sect409r1.G = ECpoint(curve=sect409k1,
                    x=hex2int('015D4860 D088DDB3 496B0C60 64756260 441CDE4A F1771D4D B01FFE5B 34E59703
                    DC255A86  8A118051 5603AEAB 60794E54 BB7996A7'
                    ),
                    y=hex2int('0061B1CF AB6BE5F3 2BBFA783 24ED106A 7636B9C5 A7BD198D 0158AA4F 5488D08F
                    38514F1F DF4B4F40 D2181B36 81C364BA 0273C706'
                    ))
# sect571k1
sect571k1 = ECcurve()
sect571k1.a = \
    hex2int('00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000')
sect571k1.b = \
    hex2int(00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000001')
sect571k1.n = \
    hex2int('02000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 131850E1
    F19A63E4 B391A8DB 917F4138 B630D84B E5D63938 1E91DEB4 5CFE778F 637C1001')
sect571k1.G = ECpoint(curve=sect571k1,
                    x=hex2int('026EB7A8 59923FBC 82189631 F8103FE4 AC9CA297 0012D5D4 60248048 01841CA4
```

```
                    43709584 93B205E6 47DA304D B4CEB08C BBD1BA39 494776FB 988B4717 4DCA88C7 E2945283 A01C8972'
                    ),
                    y=hex2int('0349DC80 7F4FBF37 4F4AEADE 3BCA9531 4DD58CEC 9F307A54 FFC61EFC 006D8A2C
                    9D4979C0 AC44AEA7 4FBEBBB9 F772AEDC B620B01A 7BA7AF1B 320430C8 591984F6 01CD4C14 3EF1C7A3'
                    ))
# sect571r1
sect571r1 = ECcurve()
sect571r1.a = \
    hex2int('00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000001')
sect571r1.b = \
    hex2int(02F40E7E 2221F295 DE297117 B7F3D62F 5C6A97FF CB8CEFF1 CD6BA8CE 4A9A18AD 84FFABBD 8EFA5933
    2BE7AD67 56A66E29 4AFD185A 78FF12AA 520E4DE7 39BACA0C 7FFEFF7F 2955727A')
sect571r1.n = \
    hex2int('03FFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF E661CE18
    FF559873 08059B18 6823851E C7DD9CA1 161DE93D 5174D66E 8382E9BB 2FE84E47')
sect571r1.G = ECpoint(curve=sect571r1,
                    x=hex2int('0303001D 34B85629 6C16C0D4 0D3CD775 0A93D1D2 955FA80A A5F40FC8 DB7B2ABD
                    BDE53950 F4C0D293 CDD711A3 5B67FB14 99AE6003 8614F139 4ABFA3B4 C850D927 E1E7769C 8EEC2D19'
                    ),
                    y=hex2int('037BF273 42DA639B 6DCCFFFE B73D69D7 8C6C27A6 009CBBCA 1980F853 3921E8A6
                    84423E43 BAB08A57 6291AF8F 461BB2A8 B3531D2F 0485C19B 16E2F151 6E23DD3C 1A4827AF 1B8AC15B'
                    ))
# brainpoolP160r1
brainpoolP160r1 = ECcurve()
brainpoolP160r1.p = \
    hex2int('E95E4A5F737059DC60DFC7AD95B3D8139515620F')
brainpoolP160r1.a = \
    hex2int('340E7BE2A280EB74E2BE61BADA745D97E8F7C300')
brainpoolP160r1.b = \
    hex2int('1E589A8595423412134FAA2DBDEC95C8D8675E58')
brainpoolP160r1.n = \
    hex2int('E95E4A5F737059DC60DF5991D45029409E60FC09')
brainpoolP160r1.G = ECpoint(curve=brainpoolP160r1,
                            x=hex2int('BED5AF16EA3F6A4F62938C4631EB5AF7BDBCDBC3'
                            ),
                            y=hex2int('1667CB477A1A8EC338F94741669C976316DA6321'
                            ))
# brainpoolP192r1
brainpoolP192r1 = ECcurve()
brainpoolP192r1.p = \
    hex2int('C302F41D932A36CDA7A3463093D18DB78FCE476DE1A86297')
brainpoolP192r1.a = \
    hex2int('6A91174076B1E0E19C39C031FE8685C1CAE040E5C69A28EF')
brainpoolP192r1.b = \
    hex2int('469A28EF7C28CCA3DC721D044F4496BCCA7EF4146FBF25C9')
brainpoolP192r1.n = \
    hex2int('C302F41D932A36CDA7A3462F9E9E916B5BE8F1029AC4ACC1')
brainpoolP192r1.G = ECpoint(curve=brainpoolP192r1,
                            x=hex2int('C0A0647EAAB6A48753B033C56CB0F0900A2F5C4853375FD6'
                            ),
                            y=hex2int('14B690866ABD5BB88B5F4828C1490002E6773FA2FA299B8F'
                            ))
# brainpoolP224r1
brainpoolP224r1 = ECcurve()
brainpoolP224r1.p = \
    hex2int('D7C134AA264366862A18302575D1D787B09F075797DA89F57EC8C0FF')
brainpoolP224r1.a = \
    hex2int('68A5E62CA9CE6C1C299803A6C1530B514E182AD8B0042A59CAD29F43')
brainpoolP224r1.b = \
    hex2int('2580F63CCFE44138870713B1A92369E33E2135D266DBB372386C400B')
brainpoolP224r1.n = \
    hex2int('D7C134AA264366862A18302575D0FB98D116BC4B6DDEBCA3A5A7939F')
brainpoolP224r1.G = ECpoint(curve=brainpoolP224r1,
                            x=hex2int('0D9029AD2C7E5CF4340823B2A87DC68C9E4CE3174C1E6EFDEE12C07D'
                            ),
                            y=hex2int('58AA56F772C0726F24C6B89E4ECDAC24354B9E99CAA3F6D3761402CD'
                            ))
# brainpoolP256r1
brainpoolP256r1 = ECcurve()
brainpoolP256r1.p = \
    hex2int('A9FB57DBA1EEA9BC3E660A909D838D726E3BF623D52620282013481D1F6E5377'
            )
brainpoolP256r1.a = \
    hex2int('7D5A0975FC2C3057EEF67530417AFFE7FB8055C126DC5C6CE94A4B44F330B5D9'
            )
brainpoolP256r1.b = \
    hex2int('26DC5C6CE94A4B44F330B5D9BBD77CBF958416295CF7E1CE6BCCDC18FF8C07B6'
            )
brainpoolP256r1.n = \
```

```
    hex2int('A9FB57DBA1EEA9BC3E660A909D838D718C397AA3B561A6F7901E0E82974856A7'
            )
brainpoolP256r1.G = ECpoint(curve=brainpoolP256r1,
                            x=hex2int('8BD2AEB9CB7E57CB2C4B482FFC81B7AFB9DE27E1E3BD23C23A4453BD9ACE3262'
                            ),
                            y=hex2int('547EF835C3DAC4FD97F8461A14611DC9C27745132DED8E545C1D54C72F046997'
                            ))
# brainpoolP320r1
brainpoolP320r1 = ECcurve()
brainpoolP320r1.p = \
    hex2int('D35E472036BC4FB7E13C785ED201E065F98FCFA6F6F40DEF4F92B9EC7893EC28FCD412B1F1B32E27'
            )
brainpoolP320r1.a = \
    hex2int('3EE30B568FBAB0F883CCEBD46D3F3BB8A2A73513F5EB79DA66190EB085FFA9F492F375A97D860EB4'
            )
brainpoolP320r1.b = \
    hex2int('520883949DFDBC42D3AD198640688A6FE13F41349554B49ACC31DCCD884539816F5EB4AC8FB1F1A6'
            )
brainpoolP320r1.n = \
    hex2int('D35E472036BC4FB7E13C785ED201E065F98FCFA5B68F12A32D482EC7EE8658E98691555B44C59311'
            )
brainpoolP320r1.G = ECpoint(curve=brainpoolP320r1,
                            x=hex2int('43BD7E9AFB53D8B85289BCC48EE5BFE6F20137D10A087EB6E7871E2A10A599C710AF8D0D
                            39E20611'
                            ),
                            y=hex2int('14FDD05545EC1CC8AB4093247F77275E0743FFED117182EAA9C77877AAAC6AC7D35245D1
                            692E8EE1'
                            ))
# brainpoolP384r1
brainpoolP384r1 = ECcurve()
brainpoolP384r1.p = \
    hex2int('8CB91E82A3386D280F5D6F7E50E641DF152F7109ED5456B412B1DA197FB71123ACD3A729901D1A71874700133107EC53'
            )
brainpoolP384r1.a = \
    hex2int('7BC382C63D8C150C3C72080ACE05AFA0C2BEA28E4FB22787139165EFBA91F90F8AA5814A503AD4EB04A8C7DD22CE2826'
            )
brainpoolP384r1.b = \
    hex2int('04A8C7DD22CE28268B39B55416F0447C2FB77DE107DCD2A62E880EA53EEB62D57CB4390295DBC9943AB78696FA504C11'
            )
brainpoolP384r1.n = \
    hex2int('8CB91E82A3386D280F5D6F7E50E641DF152F7109ED5456B31F166E6CAC0425A7CF3AB6AF6B7FC3103B883202E9046565'
            )
brainpoolP384r1.G = ECpoint(curve=brainpoolP384r1,
                            x=hex2int('1D1C64F068CF45FFA2A63A81B7C13F6B8847A3E77EF14FE3DB7FCAFE0CBD10E8E826E03
                            436D646AAEF87B2E247D4AF1E'
                            ),
                            y=hex2int('8ABE1D7520F9C2A45CB1EB8E95CFD55262B70B29FEEC5864E19C054FF99129280E46462
                            177918111428203412 63C5315'
                            ))
# brainpoolP512r1
brainpoolP512r1 = ECcurve()
brainpoolP512r1.p = \
    hex2int('AADD9DB8DBE9C48B3FD4E6AE33C9FC07CB308DB3B3C9D20ED6639CCA703308717D4D9B009BC66842AECDA12AE6A380E
    62881FF2F2D82C68528AA6056583A48F3'
            )
brainpoolP512r1.a = \
    hex2int('7830A3318B603B89E2327145AC234CC594CBDD8D3DF91610A83441CAEA9863BC2DED5D5AA8253AA10A2EF1C98B9AC8B5
    7F1117A72BF2C7B9E7C1AC4D77FC94CA'
            )
brainpoolP512r1.b = \
    hex2int('3DF91610A83441CAEA9863BC2DED5D5AA8253AA10A2EF1C98B9AC8B57F1117A72BF2C7B9E7C1AC4D77FC94CADC083E6
    7984050B75EBAE5DD2809BD638016F723'
            )
brainpoolP512r1.n = \
    hex2int('AADD9DB8DBE9C48B3FD4E6AE33C9FC07CB308DB3B3C9D20ED6639CCA70330870553E5C414CA92619418661197FAC1047
    1DB1D381085DDADDB58796829CA90069'
            )
brainpoolP512r1.G = ECpoint(curve=brainpoolP512r1,
                            x=hex2int('81AEE4BDD82ED9645A21322E9C4C6A9385ED9F70B5D916C1B43B62EEF4D0098EFF3B1
                            F78E2D0D48D50D1687B93B97D5F7C6D5047406A5E688B352209BCB9F822'
                            ),
                            y=hex2int('7DDE385D566332ECC0EABFA9CF7822FDF209F70024A57B1AA000C55B881F8111B2DCD
                            E494A5F485E5BCA4BD88A2763AED1CA2B2FA8F0540678CD1E0F3AD80892'
                            ))
```

B OBSAH PŘILOŽENÉHO CD

Obsah přiloženého CD obsahuje zdrojové kódy, naměřené hodnoty a vektorové grafy z výsledků. Algoritmy v knihovně OpenSSL byly testovány ve verzích 1.0.2 a 1.1.0, v obou verzích jsou funkční. Knihovna libecc byla testována ve verzi 2017. Algoritmy v pythonu byly testovány v systémové verzi Raspberry Pi.

- `/` ..... kořenový adresář přiloženého CD
  - `zdrojové kódy`
    - `libecc`
      - `add` ..... addition dvou bodů metodou skalární na všech křivkách
      - `add_monty` ..... addition dvou bodů metodou Montgomery na všech křivkách
      - `dbl` ..... doubling bodu metodou skalární na všech křivkách
      - `dbl_monty` ..... doubling bodu metodou Montgomery na všech křivkách
      - `get_rnd_point` ..... generování náhodného bodu na eliptické křivce na všech křivkách
      - `mul` ..... multipling dvou bodů metodou skalární na všech křivkách
      - `mul_monty` ..... multipling dvou bodů metodou Montgomery na všech křivkách
    - `OpenSSL`
      - `BRAINPOOL_add` ..... addition dvou bodů metodou skalární na křivkách BRP
      - `BRAINPOOL_dbl` ..... doubling bodu metodou skalární na křivkách BRP
      - `BRAINPOOL_get_rnd_point` .... generování náhodného bodu na eliptické křivce na křivkách BRP
      - `BRAINPOOL_inv` ..... negace bodu na křivkách BRP
      - `BRAINPOOL_mulx` ..... multipling metodou wNAF na křivkách BRP
      - `SECP_add` ..... addition dvou bodů metodou skalární na křivkách SECP
      - `SECP_dbl` ..... doubling bodu metodou skalární na křivkách SECP
      - `SECP_get_rnd_point` ..... generování náhodného bodu na eliptické křivce na křivkách SECP
      - `SECP_inv` ..... negace bodu na křivkách SECP
      - `SECP_mulx` ..... multipling metodou wNAF na křivkách SECP
      - `SECT_add` ..... addition dvou bodů metodou skalární na křivkách SECT
      - `SECT_dbl` ..... doubling bodu metodou skalární na křivkách SECT
      - `SECT_get_rnd_point` ..... generování náhodného bodu na eliptické křivce na křivkách SECT
      - `SECT_inv` ..... negace bodu na křivkách SECT
      - `SECT_mulx` ..... multipling metodou wNAF na křivkách SECT
    - `python`
      - `BRAINPOOL_curves` ..... databázový soubor křivek BRP
      - `BRAINPOOL_curves_mul` ..... multipling metodou skalární na křivkách BRP
      - `SECP_curves` ..... databázový soubor křivek SECP
      - `SECP_curves_mul` ..... multipling metodou skalární na křivkách SECP
  - `naměřené hodnoty`
    - `libecc_mereni` ..... tabulky a grafy s naměřenými hodnotami z knihovny libecc
    - `OpenSSL_mereni` ..... tabulky a grafy s naměřenými hodnotami z knihovny OpenSSL
    - `pametove_srovnani` ..... tabulky paměťové srovnání knihoven
    - `proud_python` ..... tabulka a graf s proudovým měřením v pythonu
    - `prum_mod_med` ..... příklad průměr, modus a medián
    - `python_mereni` ..... tabulky a grafy s naměřenými hodnotami z pythonu
    - `srovnani_napric_knihovnami` ..... tabulky a grafy srovnání hodnot napříč knihovnami