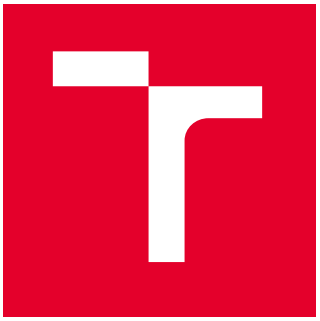


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

ANALÝZA KRYPTOGRAFICKÝCH OPERACÍ NA RŮZNÝCH ARCHITEKTURÁCH PROCESORU

ANALYSIS OF CRYPTOGRAPHIC OPERATIONS ON VARIOUS PROCESSOR ARCHITECTURES

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — **Hynek Kubík**

AUTHOR

VEDOUCÍ PRÁCE — **Ing. Tomáš Lieskovan**

SUPERVISOR

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Hynek Kubík ***ID:*** 221554

Ročník: 3 ***Akademický rok:*** 2021/22

NÁZEV TÉMATU:

Analýza kryptografických operací na různých architekturách procesoru

POKYNY PRO VYPRACOVÁNÍ:

Tématem bakalářské práce bude nastudovat, porovnat a zanalyzovat kryptografické operace na serverových architekturách procesorů Intel Xeon a AMD Epyc. Analýza bude probíhat na výpočetním clusteru. Samotná analýza bude realizována pomocí programu, který bude navržen a zhotoven pro potřeby tohoto zadání. Program zašifruje zvolené soubory pomocí několika šifer a následně dešifruje, dobu operací uloží pro následné analyzování. Stejný proces bude probíhat na všech zvolených architekturách procesorů s využitím vestavěných akcelerace kryptografických operací daných procesů. Pro výstup bakalářské práce bude taktéž navržen kód, který bude realizovat výstup ve formě tabulek a grafů, ve kterých bude znázorněn čas vypočtu šifrovaní a dešifrování v závislosti na velikosti souboru a zvolené šifry.

DOPORUČENÁ LITERATURA:

[1] GILGER, Johannes; BARNICKEL, Johannes; MEYER, Ulrike. GPU-acceleration of block ciphers in the OpenSSL cryptographic library. In: International Conference on Information Security. Springer, Berlin, Heidelberg, 2012. p. 338-353.

[2] BARIK, Rabindra K., et al. Performance analysis of virtual machines and containers in cloud computing. In: 2016 international conference on computing, communication and automation (iccca). IEEE, 2016. p. 1204-1210.

Termín zadání: 7.2.2022 ***Termín odevzdání:*** 31.5.2022

Vedoucí práce: Ing. Tomáš Lieskovan

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

UPOZORNĚNÍ:

Autor bakalářské práce nesmí při vytváření bakalářské práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

ABSTRAKT

Bakalářská práce analyzuje kryptografické operace a algoritmy na různých architektůrách serverových procesorů. Práce ve své teoretické části se zabývá rozborem jednotlivý kryptografických algoritmů, architektůrama procesoru, výpočetnímy clustru a HPC. Pro realizaci analýzy je zhotovena aplikace, která provede v závislosti na vstupních parametrech, zvolené architektůře a konfigurace zvolené uživatelem měření výkonosti jednotlivých operací. Výsledky z výkonnostních měření a různých kryptografických operacích a architekturách a s různými vstupními parametry jsou následně prezentovány v aplikaci ve formě grafů.

KLÍČOVÁ SLOVA

Kryptografie, Procesory, Cluster, Analýza dat, Sběr dat, Bezpečnost

ABSTRACT

Bachelor's thesis analyzes cryptographic operations and algorithms on different server processor architectures. The thesis in its theoretical part deals with the analysis of individual cryptographic algorithms, processor architecture, computing cluster and HPC. An application is made for the analysis, which, depending on the input parameters, selected architectures and configuration selected by the user, performs the measurement of the performance of individual operations.The results from performance measurements and various cryptographic operations and architectures and with various input application parameters are subsequently years in the form of graphs.

KEYWORDS

Cryptography, Processors, Cluster, Data Analysis, Data Acquisition, Security

Vysázeno pomocí balíčku `thesis` verze 4.07; `http://latex.feec.vutbr.cz`

KUBÍK, Hynek. *Analýza kryptografických operací na různých architekturách procesoru*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2021, 60 s. Semestrální práce. Vedoucí práce: Ing. Tomáš Lieskovan

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Hynek Kubík
VUT ID autora:	221554
Typ práce:	Semestrální práce
Akademický rok:	2021/22
Téma závěrečné práce:	Analýza kryptografických operací na různých architekturách procesoru

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné 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 uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom 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*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu semestrální práce panu Ing. Tomášovi Lieskovanovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci. Mé poděkovaní taktež patří panu Ing. Ondřeji Vysockému za odbornou konzultaci, trpělivost a podnětné návrhy k práci a taktéž IT4Innovations za přístup k superpočítačům.
Tato práce byla podpořena Ministerstvem školství, mládeže a tělovýchovy České republiky prostřednictvím e-INFRA CZ (ID:90140).

Obsah

Seznam obrázků

Úvod

S neustále rostoucím výkonem počítačů a výpočetních jednotek a také s rostoucím počtem zařízení, která se dokáží připojit k internetu, roste i počet komunikujících zařízení nebo uživatelů. Roste také množství informací v digitální podobě, ať už proudících skrz síť, nebo pouze digitalizovaných na lokálních úložištích. Tento jev má za následek, že přibývá i počet potenciálních útočníků a obětí v prostředí internetu nebo intranetu. Z tohoto důvodu je potřeba informace chránit. Samotnou ochranu lze chápat z několika pohledů. Jedním z nich je například zabránění v přečtení informace tím, komu není určena. Tomu lze třeba zabránit použitím šifry. Dalším způsobem jak chápat ochranu je, pokud chceme zabránit neoprávněné změně informací. K tomuto účelu se používá například hashovací funkce, která provede otisk původní informace. Kontrola se provádí tak, že se provede operace znovu a jestli se otisky shodují je informace nezměněna.

Tato práce ve své první kapitole rozebírá kryptografii převážně z teoretického pohledu. Vymezuje základní pojmy a seznamuje čtenáře s danou problematikou, se kterou se pak dále pracuje.

Druhá kapitola pojednává o procesorech v superpočítačích, konkrétně v Barboře a Karolíně, které se nachází v IT4Innovations, což je národní superpočítačové centrum České republiky, se kterým byla v rámci této práce vedena spolupráce. Dále je v této kapitole uvedeno seznámení s serverovými procesory, na kterých bylo prováděno měření.

Třetí kapitola je jak teoretická, tak i praktická. Teoretická část této kapitoly se věnuje vývojovému prostředí, programovacímu jazyku Python a knihovnám, které byly stěžejní pro návrh a vývoj aplikace. V praktické části této kapitoly se nachází návrh a vývoj samotné aplikace. Je zde popsáno, jakým způsobem byla aplikace implementována, jaké možnosti podporuje a poskytuje návod, jakým způsobem s ní může uživatel pracovat.

Ve čtvrté kapitole je rozebráno testování. Konkretně pojednává o tom, na jakých procesorech se aplikace testovala a jaké algoritmy byly pro testování zvoleny. Dále se věnuje samotnému měření. Je zde popsaný postup měření a deskripce nezbytných prerekvizit před samotným měřením.

V páté části práce jsou prezentovány výsledky měření a práce.

Cíle práce

Tato práce má za cíl nastudovat kryptografické algoritmy a jejich operace a zanalyzovat a porovnat, jakým způsobem se liší časová složitost vybraných kryptografických operací na vybraných typech serverových procesorů. Vyhodnocuje, jak velký vliv má výkon, potažmo modelová řada procesorů, na kryptografické algoritmy a zhodnocuje, který z vybraných procesorů je při použití daného kryptografického algoritmu nejideálnější, a který procesor dosahoval v průměru nejlepších výsledků. K tomuto je vhodné navrhnout a zhotovit aplikaci, která by usnadnila samotné měření hodnot a následnou analýzy naměřených hodnot. Analýza dat provedená touto aplikací by mohla mít do budoucna přínos jak pro tvorbu nových šifer, tak i pro případnou volbu stávájících kryptografických algoritmů.

1 Kryptografie

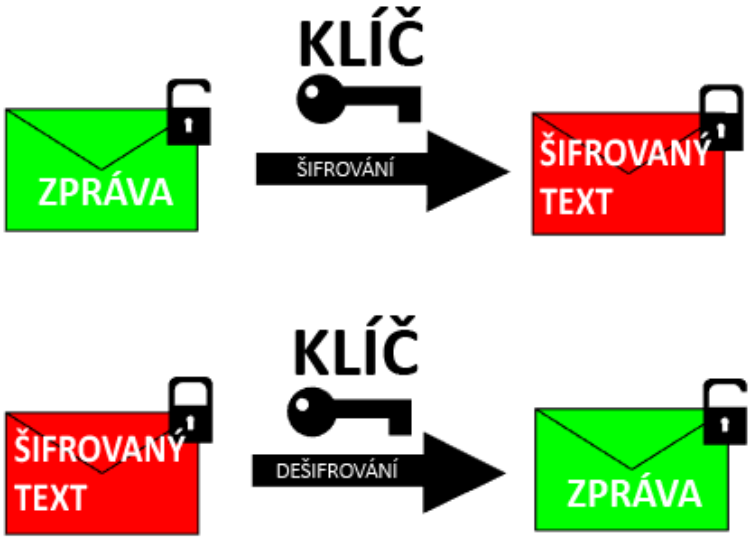
Kryptografie je nauka zabývající se šiframi a metodami utajovaní zpráv. Utajení smyslu zpráv je zajištěno převodem šifrovaného textu do podoby čitelného pouze se speciální znalostí (např. znalost klíče, konstrukce šifry). Protikladem kryptografie je kryptoanalýza, která hledá způsoby, jak prolomit kryptografické systémy [1]. Tyto obory mezi sebou soupeří ve vymýšlení nových algoritmů a následném hledání jejich slabin. Všechny šifry jsou prolomitelné, je tedy možné hrubou silnou vypočítat utajovanou zprávu. Cílem šifer je, aby čas potřebný k tomuto výpočtu byl natolik velký, aby nebyl v rozumném čase proveditelný. Výkon výpočetních jednotek a počítačů neustále exponenciálně roste [2] a šifry je tedy nutné této skutečnosti přizpůsobit. Bezpečnost lze navýšit například použitím delšího šifrovacího klíče, zopakováním algoritmu vícekrát po sobě nebo návrhem nové bezpečnější šifry. Tento proces ale kromě zvýšení bezpečnosti přináší i zvýšení nároků na výpočetní výkon zařízení, kde je šifrování prováděno.

1.1 Časová složitost

Pro řešení operací rozkladu velkých čísel na prvočísla je velmi důležitým kritériem časová a paměťová složitost. Při řešení těchto operací za pomocí výpočetní techniky se pro tuto složitost (náročnost) používá termín asymptotická složitost. Vyjadřuje způsob, jakým se chová daný algoritmus v závislosti na tom, jak velká jsou zpracovávaná data. Tyto kritéria se rozdělují na třídy složitosti. Nejpoužívanějším hlediskem pro určení je v jeho nejhorším případě. Třídy složitostí se dělí na P, NP a NP těžké. Jednou z největších současně nevyřešených otázek matematiky a teoretické informatiky je, zda se třídy P a NP rovnají. S největší pravděpodobností se nerovnají, ale vyloučit se to nedá, ačkoli tomu všechno nasvědčuje. Moderní šifry a kryptografické algoritmy využívají složitost při konstrukci matematických operací a zvyšují podíl výpočtů pro šifrování, respektive dešifrování, k tomu, aby nebyly prolomitelné v polynomiálním čase [3].

1.2 Kryptografické algoritmy

Základní kryptografické nástroje se obecně mohou rozdělit do dvou skupin, a to na schválené a dosluhující. Schválené kryptografické algoritmy (Approved, Recommended, Future) jsou algoritmy, u kterých se předpokládá bezpečnost alespoň ve střednědobém horizontu. Dosluhující kryptografické algoritmy (Legacy) jsou algoritmy, u kterých se doporučuje přestat s jejich používáním v horizontu několika málo let [4]. Mezi základní rozdělení patří rozdělení na tři základní kategorie. Symetrické,



Obr. 1.1: Schéma symetrické šifry

asymetrické (využívající veřejný klíč) a bez klíče [1]. Nedílnou současní kryptografie a kryptografických algoritmů je americký Národní institut standardů a technologie (NIST), který přijímá nové algoritmy za standardy a vyhlašuje tendry na nové algoritmy. Jedná se o autoritu, která prohlašuje šifry a algoritmy za bezpečné, potažmo prolomené.

1.2.1 Symetrické šifry

Symetrická kryptografie je založena na stejném, popřípadě lehce spočitatelném, klíči pro šifrování i dešifrování [5]. Její nepopiratelnou výhodou je rychlost, která může dosahovat i několika set násobku oproti asymetrické kryptografii. Nevýhodou je distribuce klíče mezi jednotlivými stranami. Princip fungování symetrických šifer je možno vyčíst ze schématu na Obr. 1.1.

Úroveň odolnosti symetrického kryptosystému určuje primárně délka klíče (tajného), počet opakování (rund) algoritmu a složitost konstrukce zvyšují odolnost vůči kryptoanalýze. Rundy označují počet opakovaní aplikovaní šifrovacích funkcí na jeden blok dat. Pro udržení utajení zprávy je zásadní, aby šifrovací klíč neboli tajný klíč zůstat v rukou subjektů mezi, kterými komunikace probíhá ideálně tedy v utajení před okolními vlivy. V případě odhalení klíče, může útočník odhalit obsah utajované zprávy popřípadě kompromitovat data. Jednou z nejvýznamnějších moderních symetrických šifer je Advanced Encryption Standard (AES) .

Data Encryption Standard (DES)

Algoritmus DES je předchůdce modernější šifry AES. Jako standard byl přijat v roce 1976. Délka klíče používaná u tohoto algoritmu je 56 bitů [6]. K prolomení došlo v roce 1999 s časovou náročnosti okolo 22 hodin a 15 minut [7]. DES je založen na dvou základních atributech a to na substituci (záměna) a transpozici (Diffusion).S rostoucím výpočetním výkonem se algoritmus DES s délkou 56 bitu stával nedostatečným a v roce 1995 vznikl Triple Data Encryption Algorithm, neboli TDES, TDEA, Triple DEA (3DES), který podporuje šifrovací klíč o délce 168 bitů jedná se tedy o ztrojnásobení [8]. Podle National Institute of Standards and Technology (NIST) není ani tato varianta považována za bezpečnou a jedná se o dosluhující algoritmus a měl by být kompletně nahrazen do roku 2023 [4].

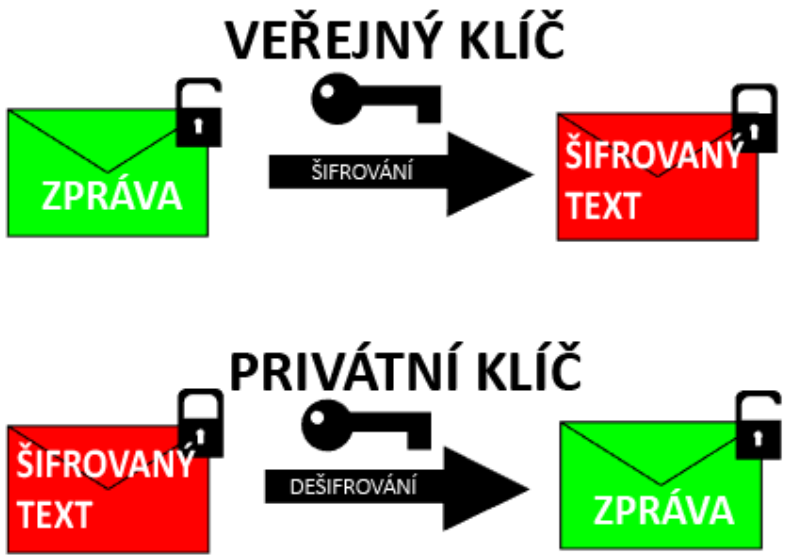
Advanced Encryption Standard (AES)

Kryptografický algoritmus AES je nástupcem šifry DES. Autory jsou pánové Joan Daemen a Vincent Rijmen [9]. Přijatá za standard pro šifrování byla v roce 2001. Algoritmus AES je symetrická bloková šifra, která dokáže šifrovat a dešifrovat informace. Šifrování převádí data do nesrozumitelné formy zvané šifrovaný text a dešifrování šifrovaného textu převede data zpět do původní podoby, tedy na původní zprávu. Každý vstup i výstup algoritmus AES je složen ze sekvence 128 bitů (číslice s hodnotou 0 nebo 1). Tyto sekvence bitů jsou označovány jako bloky a jejich počet bitů je označován jako jejich délka. Šifrovací klíč pro algoritmus AES je o délce posloupnosti 128, 192 nebo 256 bitů [10]. Jiné délky vstupu, výstupu a šifrovacího (tajného) klíče nejsou podporovány. Bity v takových sekvencích jsou číslovány počínaje nulou a o jednu menší, než je délka sekvence (délka bloku nebo klíče). I v nejslabší variantě je AES považován za bezpečný minimálně do roku 2030 [11]. Základem algoritmu je substitučně-permutační síť. AES je využíván například v protokolu Transport Layer Security(TLS), který je jedním z nejrozšířenějších kryptografických protokolů v praxi. Chrání například webové a e-mailové přístupy. Protokol TLS umožňuje klientovi a serveru vzájemné ověření a vytvoření klíče [12].

1.2.2 Asymetrické šifry

Jedná se o skupinu kryptografických metod, ve kterých se používají odlišné klíče pro šifrování a dešifrování (privátní a veřejný) [13]. Může být též označována jako kryptografie dvou klíčů. Data šifrované veřejným klíčem je možno dešifrovat soukromým klíčem. Princip šifrování a dešifrování je zobrazen na Obr. 1.2. Veřejný šifrovací klíč je volně dostupný pro prakticky kohokoli (majitel ho zveřejní) a kdokoli ho může použít pro šifrování dat určených pro majitele klíčů. Privátní, neboli soukromý klíč,

je držen v tajnosti majitelem a dešifrovat data může pouze majitel za pomocí privátního klíče. V praxi to může vypadat tak, že majitel klíčů má svůj privátní klíč bezpečně schovaný a svůj veřejný klíč sdílí na internetu.



Obr. 1.2: Schéma asymetrické šifry

Rivest–Shamir–Adleman (RSA)

RSA neboli Rivest–Shamir–Adleman je jedním z prvních algoritmů založených na myšlence asymetrické kryptografie. Byl patentován, jíž v roce 1983 [14]. Lze jej využít při podepisování i šifrování. V současné době je podle NIST stále považován za bezpečný za předpokladu použití dostatečně dlouhých klíčů o délce alespoň 2048 bitů [11]. Bezpečnost má základ ve faktorizaci (součin prvočísel). Princip problematiky faktorizace spočívá na rozložení velkého čísla na součin prvočísel a je to časově i výkonnostně velmi obtížná úloha. Z čísla $n = pq$ je prakticky nemožné zjistit činitele p a q v polynomiálním čase.

Digital Signature Algorithm (DSA)

Jak již název napovídá, jedná se o algoritmus pro digitální podepisování. Byl navržen institutem NIST v roce 1991. Rozdíl mezi DSA a RSA spočívá jak v základní matematické operaci, kde je založen na problematice diskrétního logaritmu, tak v užívaní DSA, kde se používá pouze k podepisovaní nikoli k šifrovaní [15]. Problematika diskrétního logaritmu je založena na modulární aritmetice a je poměrně jednoduché ji

spočítat jedním směrem, ale velmi obtížné opačným směrem. Například $8/3 = 2$ a zbytkem 2. Tuto rovnici lze také zapsat jako $8 \mod 3 = 2$.

Diffie–Hellman key exchange (DH)

Diffie–Hellman pojmenovaný podle autorů Whitfieldem Diffiem a Martinem Hellmanem byl uveřejněn již v roce 1976 [16]. Jedná se o kryptografický protokol pro ustanovení tajného klíče mezi stranami prostřednictvím nezabezpečeném kanálu. Je postaven na problému diskrétního logaritmu. Podrobnější popis problematiky diskrétního logaritmu se nachází v popisu algoritmu DSA. Je považován za bezpečný při použití dostatečně velké grupy alespoň 2048 bitů [11].

1.2.3 Hashovací funkce

Hashovací, neboli hash, funkce jsou speciálním druhem kryptografických algoritmů, jejichž mechanismus je založen na transformaci libovolných vstupních dat o libovolné délce na výstupní data fixní délky (otisk, fingerprint). Hash funkce zaručuje především kontrolu integrity (celistvosti) souborů. Tyto funkce jsou jednosměrné, neboli jednocestné, to znamená, že je zajištěna nemožnost odvodit data vstupní z výstupních dat [17]. Další důležitou vlastností z pohledu kryptografie, kterou by měla hash funkce obsahovat, je bezkoliznost. Bezkoliznostní je myšleno, že z různých vstupních dat by neměl vzniknout stejný otisk (hash). Tato vlastnost je v praxi téměř nedosažitelná. Z tohoto důvodu se požaduje, aby případné nalezení shody (kolize) bylo tak výpočetně velmi náročné (časově i výkonnostně), aby bylo prakticky neproveditelné, nebo aby zabralo dostatečně dlouhý čas. Jelikož jsou hash funkce, co se rychlosti týče, mnohem efektivnější než symetrické šifry, jsou hojně využívané k autentizaci v autentizačních systémech nebo pro zabezpečení dat elektronickými podpisy (integrity dat). Schéma principu fungování hash funkce je znázorněna na Obr. 1.3. Na obrázku lze vidět transformace libovolně velkého vstupu na fixní velikost výstupu. Na levé straně jsou červenou barvou znázorněná libovolná vstupní data o libovolné velikosti a použitím hashovací funkce, která je znázorněná černou šipkou, jsou transformována do takzvaného hashe, neboli otisku, tedy na výstupní data o pevně dané délce, na Obr. 1.3 znázorněno zeleným čtvercem. Nejznámější hashovací funkce jsou SHA-1 (Secure Hash Algoritm), SHA-2, SHA-3, MD4, MD5, RIPEMD, SHA-0 a HAVAL-128 [18]. Jako prolomené jsou považovány MD4, MD5, RIPEMD, SHA-0 a HAVAL-128 a v ohrožení je funkce SHA-1, která ale zatím prolomena nebyla. V teoretické části budou dále rozebírané některé z výše zmíněných.

Obr. 1.3: Schéma hashovací funkce

SHA-2

SHA-2 vytváří otisk o délce 224 až 512 bitů. Podle institutu NIST jsou bezpečné všechny verze [11]. Konstrukčně se velmi podobá funkci SHA-1 a používají i stejné principy. Implementuje jej Domain Name System Security Extensions (DNSSEC) nebo systém datových schránek v České republice [19].

SHA-3

SHA-3 funguje odlišně oproti svému předchůdci SHA-2. Vnitřní konstrukci lze nazvat jako „houbu“. Jedná se nejnovější hashovací funkce od NIST konkrétně pochází z roku 2015 [21]. Stejně jako SHA-2 jsou podporovány otisky o různé bitové délce od 224 až 512 bitů a je považována za bezpečnou ve všech verzích [11].

MD5

MD5 je nástupcem MD4 [20]. Hashovací funkci provádí po blocích o velikosti 512 bitů. Výsledný hash má délku 128 bitů. Bylo nalezeno několik zranitelností, z tohoto důvodu není pro většinu aplikací doporučován [21]. Ačkoli se dá funkce považovat za prolomenou, i nadále se využívá, například při hashování hesel, proto, aby hesla nebyla v počítači uložena v čitelné podobě, ale pouze jako otisk (hash). Ověřování přístupového hesla probíhá porovnáním hashe. Další využití je například při kontrole integrity stažených souborů, kdy se předpočítaný hash ze serveru porovná s uživatelem vypočítaným hashem staženého souboru. V případě shodě nedošlo k chybě při stahovaní.

2 Procesory (CPU)

Central Processing Unit (CPU), v českém jazyce Centrální procesorová jednotka, je řídící část a základ celého počítače. Základní činností CPU je vykonávání strojových instrukcí, ze kterých jsou tvořeny počítačové programy. Z programů přijímá vstupy a realizuje výstupy. Možností kategorizovat CPU je mnoho, ať už podle výrobce, technologie využité k výrobě nebo podle toho, k čemu jsou určeny. Tato práce se převážně zabývá High Performance Computing (HPC) serverovými procesory. Konkrétně se práce zaměřuje na procesory, které jsou využity v superpočítačích IT4Innovations Karolína a Barbora. Serverové procesory HPC se vyznačují velkým počtem jader, kde jádra dokáží pracovat na vysokých frekvencích, velkým počtem paměťových kanálů s vysokou propustností. Taktéž je pro ně typické vysoké TDP a z tohoto důvodu disponují výkonným a efektivním chlazením [22]. V případě procesorů obsažených v superpočítačích Karolina a Barbora je chlazení realizováno pomocí vody. Na poli výrobců CPU, ať už pro servery, osobní počítače nebo pro notebooky, se nacházejí dva největší hráči, a to společnost Intel a AMD. V posledních letech se do popředí dostává i společnost Apple, a to hlavně díky svým procesorům s názvem M1, které jsou určeny do osobních počítačů a notebooků. Serverové CPU zatím společnost Apple nevyrábí, z tohoto důvodu nebude již dále zmiňována. Mezi další významné výrobce serverových procesorů patří IBM s procesorovou řadou OpenPOWER, nebo Fujitsu s procesory A64FX (ARM) [23]. Tato práce se bude nadále věnovat procesorům od společnosti AMD a Intel, které jsou využity v superpočítačích národního superpočítačového centra IT4Inovations.

2.1 IT4Inovations a superpočítače

IT4Innovations národní superpočítačové centrum zřízené při VŠB – Technické univerzitě Ostrava je provozovatelem nejvýkonnějších superpočítačových systémů v České republice. Jedná se především o inovativní výzkumné a vývojové centrum v oblastech vysoce výkonného počítání (HPC), datových analýz nebo umělé inteligence. V současné době IT4Innovations provozuje hned tři superpočítače, a to Karolina, Barbora a NVIDIA DGX-2 [24]. IT4Innovations umožnilo přístup k procesorům využívaných v superpočítačích Karolina a Barbora, jedná se o velmi výkonné HPC serverové procesory, nejmodernější architektury [25]. Tato práce operuje v prostředí superpočítačového clusteru.

2.2 Intel

Intel je předním světovým výrobcem procesorů, přičemž na poli superpočítačů byl v poslední dekádě jasnou jedničkou [23]. Intel vyrábí monolitické procesory (procesor je tvořen jedním kusem křemíku) [26] s desítkami výpočetních jader zajišťujících vysoký výpočetní výkon, který je pro superpočítače nezbytný. V rámci HPC se hyper-threadované vlákna vypínají z důvodu, že můžou snížit výkon spouštěné aplikace.

Intel Cascade Lake 6240

Intel Cascade Lake 6240 je osmnácti jádrový a třiceti šesti vláknový serverový procesor z druhé generace škálovatelných procesorů Intel Xeon [27]. Procesor je základem standardních výpočetních uzlů na superpočítači Barbora [25]. Byl představen společností Intel začátkem roku 2019. Xeon Gold 6240 je 64 bitový, je založen na architektuře Cascade Lake pro servery a je vyráběn 14 nm technologií. Tento procesor podporuje až 1 TB paměti RAM typu DDR4 o rychlosti 2933 MHz, pracuje na frekvenci 2,6 GHz s turbo boost až 3,9 GHz. TDP, česky nejvyšší možný tepelný výkon procesoru, činí 150 W [28].

Intel Skylake Gold 6126

Intel Skylake Gold 6126 je dvanácti jádrový a dvaceti čtyř vláknový serverový procesor, který je základem akcelerovaných výpočetních uzlů na superpočítači Barbora [25]. Intel Xeon Gold 6126 je 64 bitový, je založen na architektuře Skylake pro servery a je vyráběn 14 nm technologií. Tento procesor podporuje až 768 GB paměti RAM typu DDR4 s rychlostí 2666 MHz, pracuje na frekvenci 2,6 GHz s turbo boost až 3,7 GHz. TDP procesoru činí 125 W [28] [29].

Intel Skylake Platinum 8153

Intel Xeon Platinum 8153 je, stejně jako předešlé dva zmíněné procesory, využit v superpočítači Barbora, konkrétně v tlustém uzlu [25]. Jedná se o 64 bitový 16 jádrový a 32 vláknový procesor a byl nejvýkonnějším serverovým procesorem představeným společností Intel v polovině roku 2017. Xeon Platinum 8153 je založen na serverové konfiguraci architektury Skylake a je vyráběn 14 nm technologií [30]. Tento procesor pracuje na frekvenci 2 GHz s turbo boostem až na frekvenci 2,8 GHz a podporuje až 768 GB šesti kanálové RAM paměti typu DDR4 s maximální rychlostí 2666 MHz. TDP dosahuje až 125 W [28].

Intel Cascade Lake Platinum 8268

Intel Xeon 8268 je jediným procesorem od společnosti Inte,l který je využíván v superpočítači Karolína [25]. Tento procesor je postaven na serverové konfiguraci architektury Cascade Lake a je vyráběn 14 nm technologií. Obsahuje 24 jader a 48 vláken [27]. Pracuje na frekvenci 2,90 GHz s turbo boost frekvencí až 3,90 GHz, podporuje až 1 TB RAM typu DDR4 v šesti kanálech s maximální rychlostí až 2933 MHz. TDP tohoto procesoru dosahuje hodnot 205 W [28].

2.3 AMD

Společnost AMD je dalším z předních světových výrobců procesorů, nicméně v minulosti jejich procesory nebyly zastoupeny v předních superpočítačích. Díky řadě procesorů EPYC, která byla představena v roce 2017 a je založená na technologii chipletů (procesor je tvořen několika kusy křemíků, potenciálně každý kus o jiné litografii), byly výrazně sníženy výrobní náklady těchto procesorů a jejich zastoupení v superpočítačích se v posledních letech výrazně zvýšilo [23, 31].

AMD EPYC 7H12

Procesor AMD EPYC 7H12 od firmy AMD je v podstatě páteří superpočítače Karolina, jelikož je zde zastoupen v počtu 1512 kusů. Jedná se o serverový procesor z řady AMD EPYC, konkrétně tedy druhá generace [32]. Tento procesor je vyráběn 7 nm technologií a obsahuje 64 jader a 128 vláken. Standardní frekvence, na které tento procesor pracuje, je 2,6 GHz a turbo boost dosahuje až hodnoty 3,3 GHz. Podporuje až 4 TB RAM v osmi kanálech a rychlosti 3200 MHz. Základní TDP je 280 W [33].

AMD EPYC 7763

Serverový procesor AMD EPYC 7763 z třetí generace AMD EPYC je, stejně jako předešlý procesor, umístěn v superpočítači Karolína. Skládá se ze 64 jader a 128 vláken. Frekvence v základu činí 2,45 GHz a při tubro boostu dosahuje 3,5 GHz [34]. Procesor je vyráběn 7 nm výrobní technologií [35]. Podporuje paměť RAM až o velikosti 4 TB a rychlosti 3200 MHz. TDP procesoru je 280 W [36].

3 Návrh a realizace aplikace

Aplikace je navržena tak, aby bylo možné testovat výkonost jednotlivých kryptografických operací na různých platformách a různých procesorech. Jelikož je aplikace zamýšlena jako aplikace pro jakéhokoliv uživatele, bylo myšleno na uživatelskou přívětivost, z tohoto důvodu bylo zvoleno, že aplikace bude mít grafické uživatelské rozhraní pro konfiguraci, výběr vstupních dat nebo pro výběr místa ukládaní naměřených hodnot. Aplikace je programována v programovacím jazyce Python.

Python je hybridní (paradigmatický) jazyk s podporou objektově orientovaného, procedurálního, ale i funkcionálního programování. Jedná se o vysokoúrovňový interpretovaný skriptovací programovací jazyk. Je vyvíjen jako open-source software. Funguje na většině běžných platformách operačních systémů, jako jsou Unix, Windows, Mac OS a ve většině distribucí operačního systému Linux je dokonce součástí prvotní instalace. Ačkoli je označován za skriptovací jazyk, je navržen tak, aby umožňoval tvorbu rozsáhlých komplexních aplikací včetně grafického uživatelského rozhraní [37, 38].

3.1 Knihovny jazyka Python

NumPy

NumPy je knihovna, která se využívá především pro práci s vektory, maticemi a vícerozměrnými poli. Mimo datové struktury nabízí i mnoho matematických funkcí pracujících s těmito strukturami, například lineární algebru nebo diskrétní Fourierovu transformaci. Podporuje i mnoho dalších funkcí pro vědecké výpočty a pro práci nad datovými strukturami [39].

Pandas

Knihovna Pandas má využití v analýze dat, která lze reprezentovat 2D tabulkou. Tento „tvar" dat obsahují SQL databáze, soubory CSV [40].

Matplotlib

Modul Matplotlib je jedním z vizualizačních nástrojů. Nabízí široké možnosti pro vizualizaci dat. Mezi podporované grafy patří kromě klasických spojitých nebo bodových grafu ve 2D poli i velké množství 3D grafů, popřípadě grafy různých funkcí [41].

Python Cryptography Toolkit

Python Cryptography Toolkit je soubor knihoven, obaluje implementace různých knihoven s kryptografickými funkcemi, algoritmy či operacemi. Z hashovacích funkcí lze využít MD2, MD4, MD5, SHA-1, SHA-256 a RIPEMD. Mezi symetrické kryptografické algoritmy, které lze použít, patří například DES, Triple DES a AES. Pro asymetrickou kryptografii pro digitální podepisování může být použito například DSA nebo RSA [42].

pyCryptodome

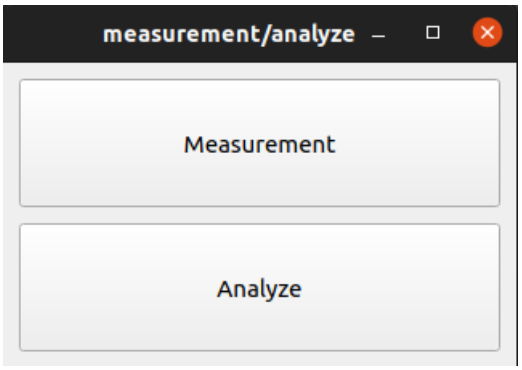
Knihovna pyCryptodome je sbírka standardů kryptografických operací a algoritmů a také funkcí obvykle používaných v kryptografii, mezi nimiž lze nalézt šifrovací algoritmy (symetrické a asymetrické šifry), hashovací funkce, algoritmy pro detekci a opravu chyb a generování pseudonáhodných čísla [43]. Tato knihovna je moderní verzí pyCrypto zahrnuto v PyPI (Python Package Index).

3.2 Vývoj aplikace

Vývoj aplikace probíhá na operačním systému Linux. Aplikace je navržena jako desktopová aplikace s uživatelským rozhraní s využitím PyQt [44]. Návrh aplikace zahrnuje dvě relativně samostatné aplikace. První z nich je určená pro měření času, a to z pohledu doby trvání dané operace, a pro zápis naměřených hodnot do souboru. Druhá je pak určená pro vizualizaci naměřených hodnot. Samotný vývoj probíhá postupným přidáváním možnosti konfigurace a vizualizace.

Jednotlivé komponenty aplikace byly programovány samostatně. Jádrem celé aplikace je spustitelný soubor main.py, ve kterém se nachází čtyři třídy. Každá třída má na starost jedno menu, respektive jedno okno. První třída s názvem Window (Úvodní tlačítkové menu) má na starost tlačítkové menu pro základní rozdělení aplikace podle toho, jestli chce uživatel měřit (čas), nebo zobrazovat naměřené hodnoty. Tlačítkové menu je zobrazeno na Obr. 3.1 Po stisknutí jednoho z tlačítek přejde aplikace do zvolené sekce měření, respektive analýzy.

Druhá třída Window2 (Konfigurační menu) je mnohem rozsáhlejší co se rozsahu i funkcionality týče. Její hlavní funkcionalitou je konfigurace měření a kontrola správnosti zadané konfigurace. Samotná konfigurace je realizována přes widgety (QtWidgets), tlačítka (QPushButton), textová pole(QLineEdit), nadpisy (QLabel), checkBoxy (QCheckBox) a menu (menuBar()). Kontrola správné konfigurace je realizována přes podmínky, skrze které se kontroluje, zda je zvoleno vše, co je potřeba pro správný běh měření. Kontroluje se, zda je zvolená cesta k adresáři se vstupními



Obr. 3.1: První menu

daty, cesta k místu ukládaní naměřených dat, respektive k místu pro vytvoření souboru pro ukládání naměřených hodnot, zda jsou zvolené algoritmy a zda je zvolená metrika pro měření (čas). V případě, že konfigurace není v pořádku, se objeví okno s chybovou zprávou, ve které se nachází, co je zvoleno špatně, respektive co chybí nakonfigurovat, a návod, jakým způsobem provést konfiguraci správně. V opačném případě se objeví okno s informací, že konfigurace proběhla úspěšně a stisknutím tlačítka 'ok' se spustí samotné měření. Tyto zprávy jsou realizovány pomocí QMessageBox a vhodných parametrů (v závislosti na tom, jestli se jedná o zprávu chybovou nebo informační), jako je nastavení ikony nebo tlačítka. Konfigurace je uložena do slovníku, ve kterém se nachází list pro algoritmy, které byly nakonfigurovány. Dále se zde nachází list pro měřenou metriku, kde se uloží doba běhu algoritmu (Time[s]), a dva stringy, ve kterých je uložena cesta k adresáři s daty a cesta k umístění souboru s výsledky měření.

Pro potřeby měření a spouštění měření bez grafického uživatelského rozhraní byl zhotoven spustitelný soubor run_terminal.py, který provádí vše výše zmíněné, s tím rozdílem, že se jedná pouze o konzolovou verzi, tedy bez grafického uživatelského rozhraní.

Další nedílnou součástí aplikace je třída Measure, která se nachází v souboru meric_data_load.py. Tato třída obstarává většinu funkcionality měřící části aplikace. Konfigurace je jí předána z třídy Window2 ve formě slovníku. Nejprve projde adresář se vstupními daty, nalezne všechny soubory nacházející se v daném adresáři a uloží je do listu. Následně proběhne funkce, která zapíše do souborů název procesoru. Název procesoru aplikace zjistí v adresáři /proc a v souboru cpuinfo. Poté projde list s algoritmy a zavolá funkci pro daný algoritmus. V této funkci projde list s cestami k jednotlivým souborům a uloží název souboru, zjistí velikost souboru, spustí časovač a zavolá funkci zajištující provedení daného algoritmu se vstupním parametrem, kterým je cesta k souboru. Po ukončení algoritmu se vypočte čas běhu

operace a výsledek se zapíše do souboru. Data jsou zapisována ve dvou řádcích a to ve formátu line1 = # filename; size [B] a line2 = algorithmname; time [s]. Toto se opakuje pro všechny soubory nacházející se v adresáři a pro všechny zvolené algoritmy. Hash funkce jsou implementovány pomocí knihovny hashlib, jedná se o funkci SHA-256 a MD5. K implementaci šifrovacích algoritmů jsou využity knihovny k tomu určené jako je například PyCrypto [42] nebo pycryptodome [43]. Symetrická šifra AES je implementována v několika možnostech, a to jak s využitím knihoven, tak i nativně. Pro optimalizaci nativní implementace byla původní verze napsaná v programovacím jazyce Python3, zkompilovaná pomocí Cythonu na kód jazyka C. V rámci kompilace bylo nejprve potřeba změnit příponu souboru z .py na .pyx. a následně vytvořit kompilační soubor. Ukázka tohoto souboru je znázorněna ve výpisu 3.2, kde první dva řádky jsou importy modulů. Na třetím řádku se nachází volání importovaného modulu a jsou mu přidány parametry.

Výpis 3.1: Ukázka kompilace

```
from setuptools import setup
from Cython.Build import Cythonize

setup(
    ext_modules = Cythonize('aes.pyx')
)
```

Výpis 3.2: Příkaz v terminálu

```
$python3 setup.py build_ext --inplace
```

Třetí třída ze souboru main.py s názvem Window3 (Menu pro výběr grafu) je třídou pro analýzu. Tato třída nejprve otevře dialogové okno pro výběr souboru s naměřenými daty. Následně vytvoří slovník, do kterého si uloží cestu k tomuto souboru, soubor projde a data nacházející se v tomto souboru uloží. Následně se zobrazí tlačítkové menu pro výběr grafů, které je opět realizované pomocí tlačítek. Při kliknutí na tlačítko se zavolá třída odpovídající grafu, který je reprezentovaný daným tlačítkem. Třídě se předá slovník, který je následně zpracován. Okna a plátna pro vykreslení grafu jsou realizovaná pomocí matplotlib [41] FigureCanvas a PyQt5 [44]. O ovládaní a nastavení grafu se starají widgety (QtWidgets), tlačítka (QPushButton), kombinovaná pole (QComboBox) a toolbar (NavigationToolbar2QT).

Každá třída pracuje samostatně, jediné společné, krom vzhledu, mají slovník, předaný z třídy Window3, ze kterého si každá třída vyčte cestu k souboru a pro sebe důležité informace, jako jsou algoritmy a názvy procesorů. Následné tyto informace propíše do kombinovaných polí určených pro výběr vizualizace. Data ze

souboru jsou vyčítána až ve chvíli, kdy je provedena akce pro vykreslení grafu. Samotné vyčítaní dat je realizováno přes funkci a vyčítá se pouze to, co je zvoleno. Následně jsou data uložena do listu a vykreslena. V případě spojitého, respektive bodového grafu, je akce vyvolána stiskem tlačítka 'Add to plot'. Ve sloupcových grafech je akce realizována výběrem prvku v kombinovaném poli. Přepínaní mezi spojitým a bodovým grafem zajišťuje podmínka, která změní parametr vykreslovaní z self.ax.scatter na self.ax.plot, respektive z self.ax.plot na self.ax.scatter. Změna velikosti fontu a velikost bodu při bodovém grafu je zajištěna spinboxy.

3.3 Spuštění aplikace a konfigurace

Před prvním spuštěním aplikace je nejprve potřeba připravit si prostředí a provést instalaci několika knihoven a toolkitů. V rámci této práce byl vytvořen bash skript, který nastaví virtuální prostředí pro Python do požadované podoby z toho důvodu, že v prostředí superpočítače nejsou přidělená práva pro instalaci.

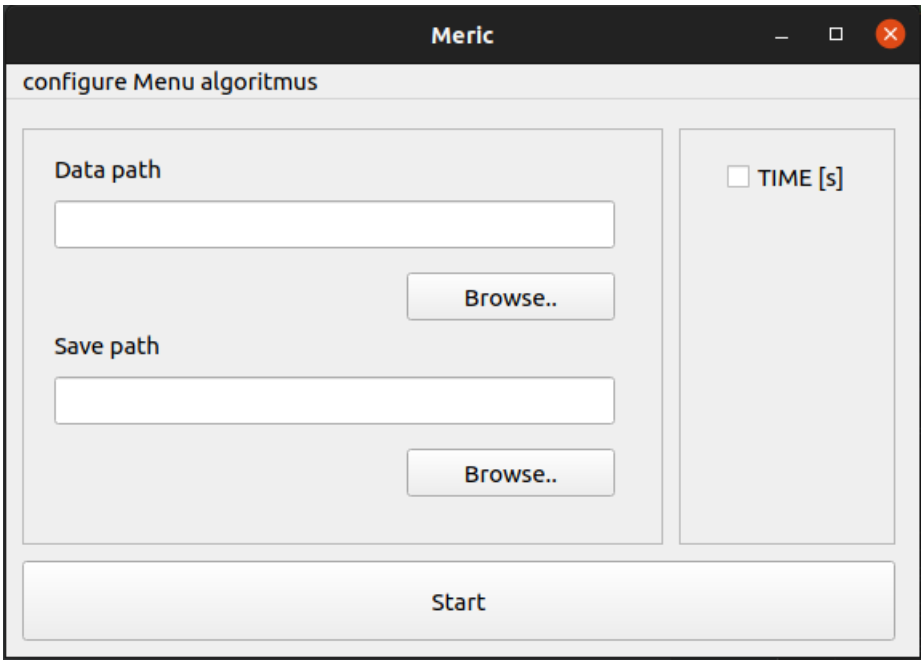
Po spuštění aplikace se uživateli zobrazí první okno se dvěma tlačítky, vyobrazeno na Obr. 3.1, kdy na prvním z nich je **Measurement** (měření) 3.3.1 a na druhém tlačítku je **Analyze** 3.3.2. Po stisknutí tlačítka Measurement se uživatel přepne do sekce measurement neboli měření (viz Obr. 3.2).

3.3.1 Measurement

V sekci measurement si může uživatel vybrat, v jaké konfiguraci si přeje měření provádět. Možnost výběru vstupních dat pro měření se provede kliknutím na tlačítko s textem **Browse**, které se nachází pod nápisem **Data path** a textovým polem. Po kliknutí na toto tlačítko se objeví dialogové okno pro výběr souborů a složek. Další možností, kterou může uživatel využít, je zadat cestu na místo, na které chce, aby byl výsledek měření uložen. To provede obdobně jako u volby vstupních dat, s tím rozdílem, že se tato operace provádí kliknutím na tlačítko Browse pod nápisem Save path a pod textovým polem.

Možnost pro volbu metriky, kterou chce uživatel měřit, se nachází v pravém menu ohraničeném obdélníkem, ve kterém se nachází zaškrtávací pole s názvem metriky (viz Obr. 3.2 v pravé části). V tomto případě se zde nachází čas (TIME [s]), ten je defaultně nastaven.

Menu pro výběr kryptografických algoritmů se nachází v levém horním rohu a to konkrétně pod tlačítkem configure menu algorithm, kde se po kliknutí objeví menu s zaškrtávacími poli, ve kterém si uživatel může vybrat, které algoritmy si přeje provést nad vstupními daty (znázorněno na Obr. 3.3, na kterém jsou zaškrtnuté hashovací funkce SHA-256 a MD5).

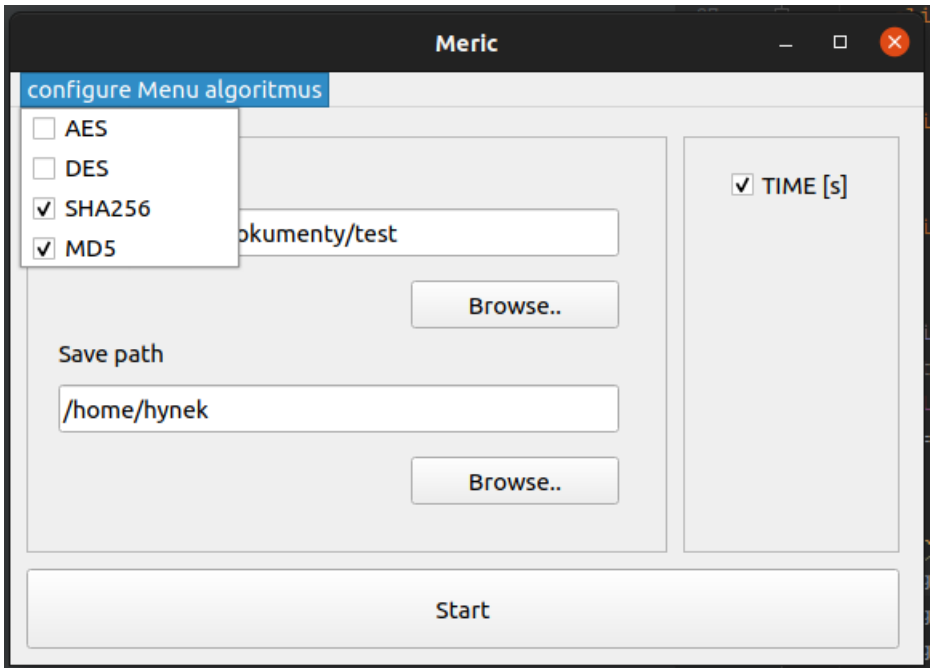


Obr. 3.2: Sekce measurement (měření)

Jestliže je všechno nakonfigurováno a vybráno, může uživatel provést akci start, která se provede kliknutím na tlačítko **Start**. V případě, že vznikne nějaká chyba v konfiguraci, a uživatel provede kliknutí na tlačítko Start, objeví se okno s popisem, kde se daná chyba nachází a jak ji případně napravit. V opačném případě, tedy pokud konfigurace proběhla v pořádku, se objeví okno, ve kterém je napsána zpráva, že je vše připraveno na měření, kliknutím na tlačítko ok se měření spustí. Po ukončení měření se uživateli objeví okno s informací, že měření proběhlo.

Naměřená data jsou uložena v souboru v podobě podle Obr. 3.4.

Struktura dokumentu s naměřenými daty je následovná: # je identifikátorem měření, následuje název souboru s označením typu souboru a velikostí souboru v bitech([b]). Na dalším řádku se nachází název algoritmu prováděného nad tímto souborem a čas, který uvádí, jak dlouho daná funkce trvala (runtime) v sekundách.



Obr. 3.3: Menu pro výběr algoritmů

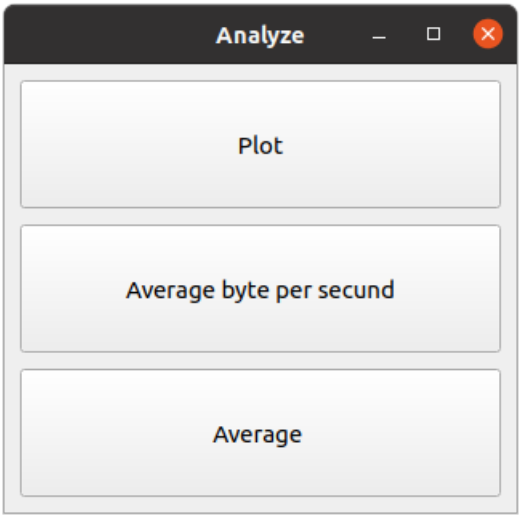
```
 1 # landscape-of-mountains-and-forest-4k-vaporwave.jpg;298093[b]
 2 SHA256; 0.00031495094299316406
 3 # desky.pdf;0[b]
 4 SHA256; 4.553794860839844e-05
 5 # text.csv;329[b]
 6 SHA256; 1.9550323486328125e-05
 7 # key.txt;90[b]
 8 SHA256; 1.5497207641601562e-05
 9 # landscape-of-mountains-and-forest-4k-vaporwave.jpg;298093[b]
10 MD5; 0.0006389617919921875
11 # desky.pdf;0[b]
12 MD5; 2.1696906982421888e-05
13 # text.csv;329[b]
14 MD5; 3.0040740966796875e-05
15 # key.txt;90[b]
16 MD5; 2.2649765014648438e-05
```

Obr. 3.4: Struktura naměřených dat

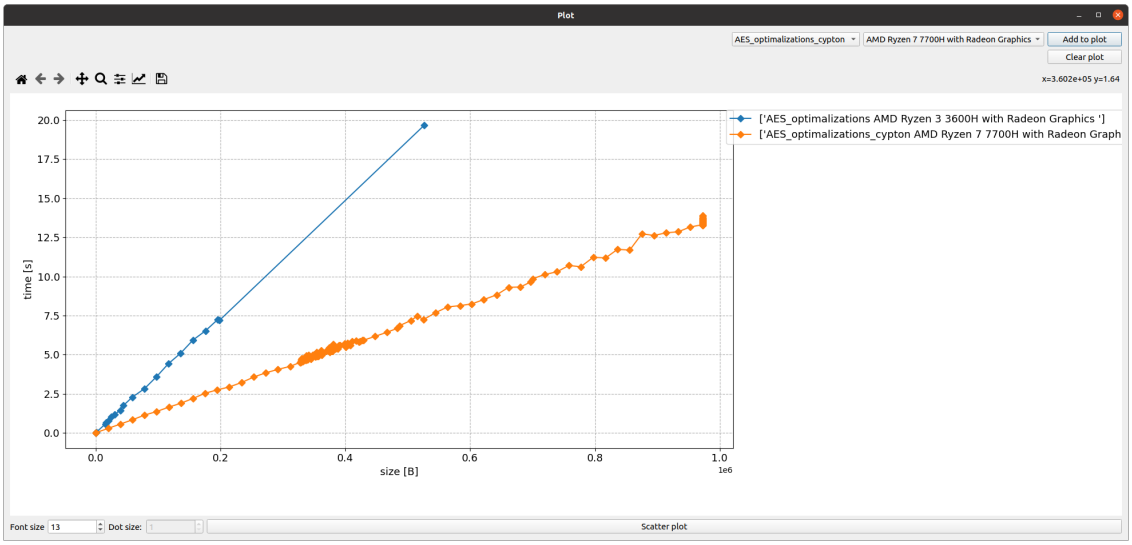
3.3.2 Analyze

V sekci Analyze si uživatel nejprve vybere data s naměřenými hodnotami v dialogovém okně. Následně se zobrazí tlačítkové menu s třemi tlačítky pro tři typy grafů. Obr. 3.5).

Po stisknutí tlačítka **Plot** se zobrazí okno s polem pro vykreslení spojitého grafu s vynesenými body, ve kterém je na ose x znázorněna velikost souboru a na ose y je znázorněn čas.

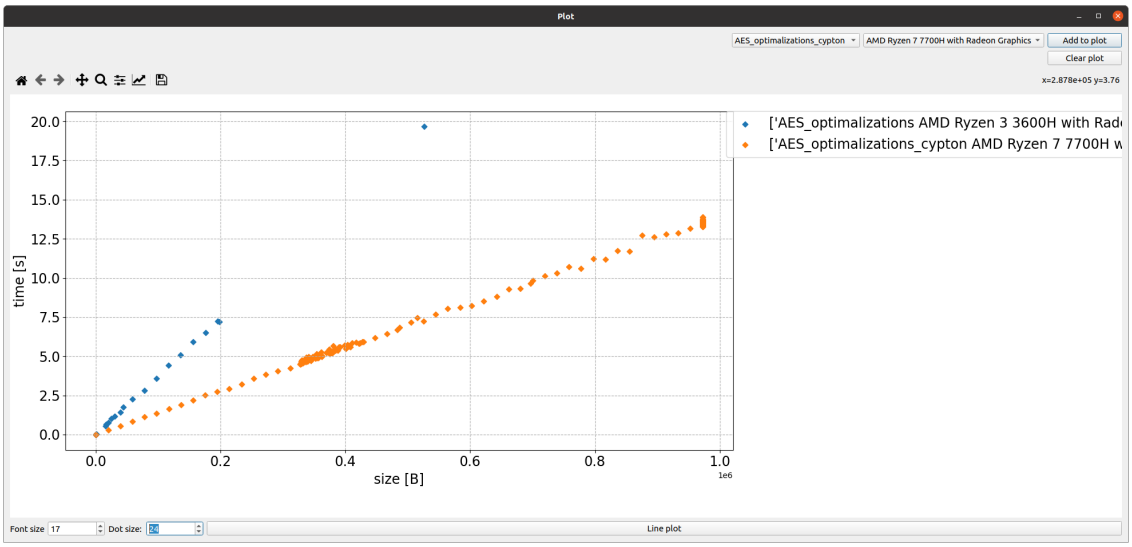


Obr. 3.5: Analyze Menu se třemi tlačítky



Obr. 3.6: Znázornění vykreslení spojitého grafu a legendy.

V horní části se nachází roletka, ve které si uživatel může vybrat, jaký algoritmus z jakého procesoru si přeje zobrazit. Taktéž se zde nachází dvě tlačítka a to **Add to plot** a **Clear**. Jak už názvy napovídají, jedno je pro přidávaní algoritmu do grafu a druhé je pro vyčištění celého pole pro vykreslování, znázorněno na Obr. 3.6. Ve spodní části okna se nachází menu pro úpravu grafu. Zde si uživatel může zvolit, zda chce bodový a nebo spojitý graf (viz Obr. 3.6), velikost fontu os, velikost fontu v legendě a popřípadě změnu velikosti bodů, pokud je tedy zvolený graf bodový, znázorněno na Obr. 3.7). Při porovnaní Obr. 3.6 s Obr. 3.7 je možné si všimnout jak

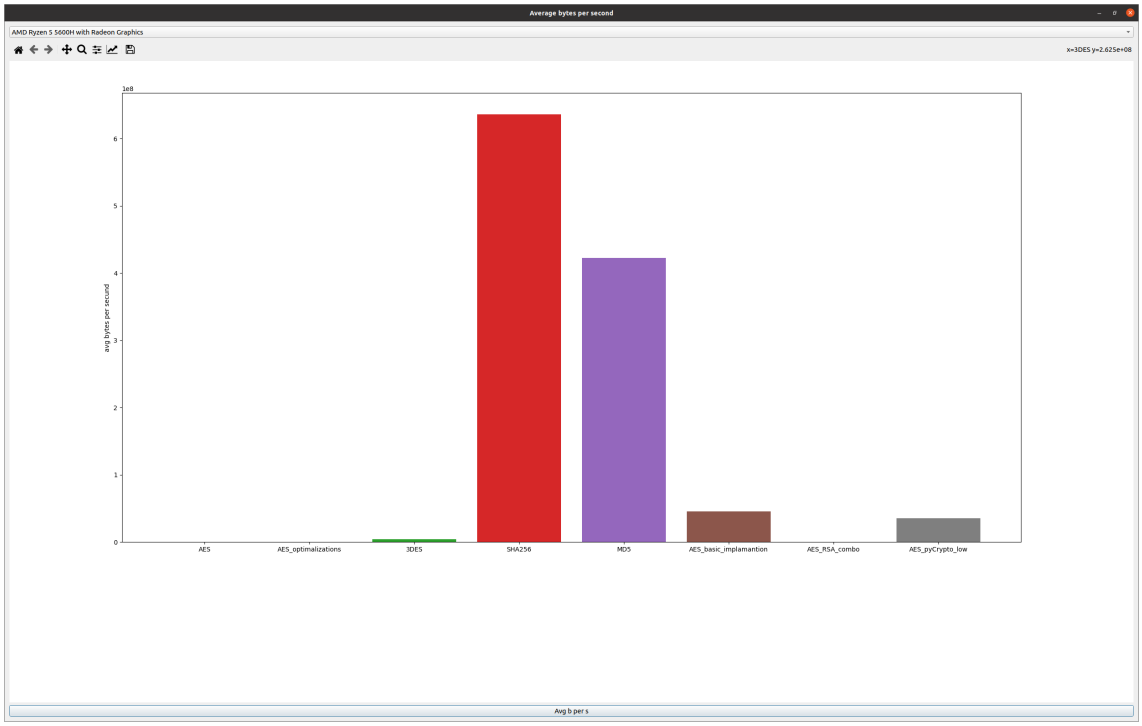


Obr. 3.7: Znázornění vykreslení bodového grafu a změny velikosti fontu a bodu.

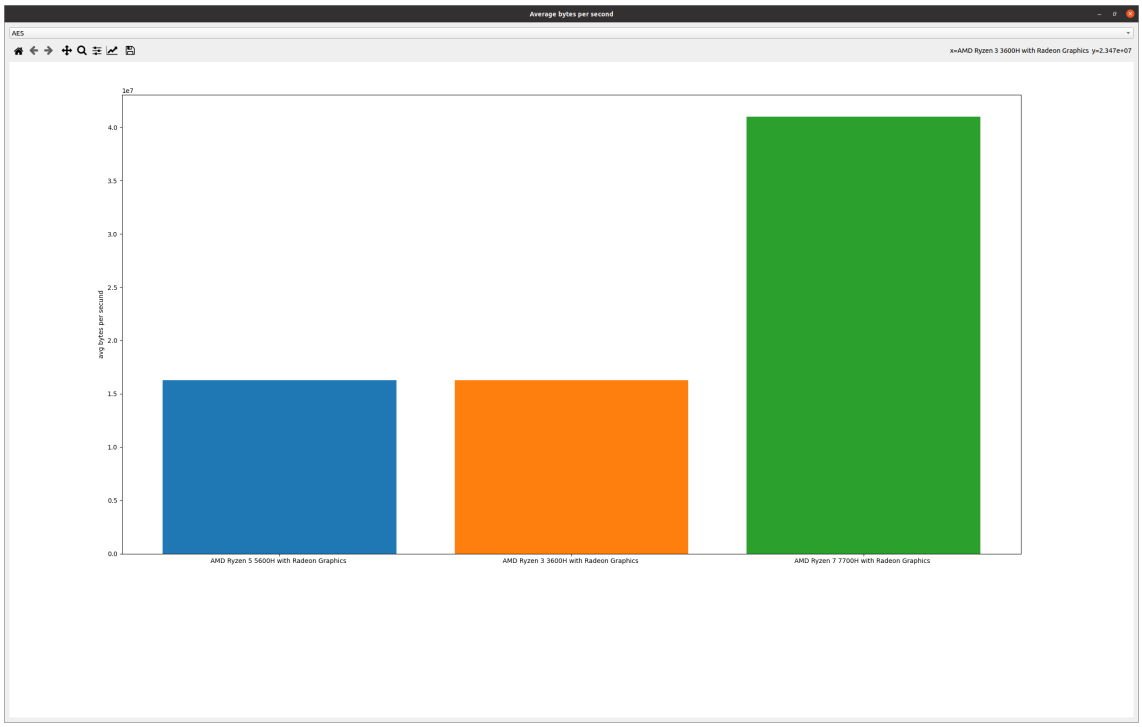
změny hodnot ve spinboxech ve spodní části obrázku, tak i změny velikosti fontu. Algoritmy z jednotlivých procesorů jsou od sebe odlišeny různými barvami. Barvy a názvy algoritmu spolu s názvem procesoru jsou zobrazeny v legendě.

V případě stisknutí tlačítka **Average byte per second** se zobrazí okno, obdobně jako u předchozího grafu (plotu), s tím rozdílem, že zde si uživatel může vybírat mezi procesory (roletka), na kterých měřil. Po zvolení procesoru se zobrazí sloupcový graf všech algoritmů měřených na daném procesoru. Data reprezentují výkonnost algoritmu ve formě průměru byte za sekundu, viz obrázek 3.9.

Po stisknutí **Average** se obdobně zobrazí okno s polem pro vykreslení grafu tak, jako u předešlých dvou možností, s rozdílem, že uživatel zde má na výběr mezi algoritmy (roletka). Po zvolení algoritmu je vykreslen sloupcový graf, ve kterém každý sloupec reprezentuje zvolený algoritmus a procesor, na kterém byl algoritmus měřen. Data ve sloupcích jsou ve formě průměrný byte za sekundu. Graf je znárodněn na Obr. 3.9).



Obr. 3.8: Znázornění vykreslení grafu Average byte per second.



Obr. 3.9: Znázornění vykreslení grafu Average.

4 Testování a měření

V rámci testování proběhlo i poměrně podrobné zhodnocení procesorů, na kterých bude následně samotná aplikace pro testování kryptografických operací spouštěna. V rámci vývoje byla aplikace testována na lokálním počítači/procesoru, na kterém je vyvíjena, konkrétně se jedna o AMD RYZEN 5 5600H. Jedná se 6 jádrový a 12 vláknový notebookový procesor s maximální frekvencí turbo boost 4,2 GHz a TDP 45 W.

Pro testování byly zvolené hashovací funkce konkrétně SHA256 a MD5, symetrická šifra AES a 3DES a kombinace RSA a AES, kde RSA se používá pro šifrování klíčů AES a samotný AES pro šifrování souborů. Vstupní data pro testování byla zvolena ve formátu obrázku typu .jpg, textového .txt soubor a soubor .pdf v různých velikostech.

4.1 Měření

Samotné měření by se dalo rozdělit na dvě skupiny, a to na měření na notebookových a na serverových procesorech. V rámci notebookových procesorů proběhlo měření na již dříve zmiňovaném AMD RYZEN 5 5600H a na Intel Core i7-5500U (2 jádrový a 4 vláknový s maximálním turbo boost frekvencí 3.00 GHz). Ze serverových procesorů měření proběhlo na dvou procesorech. Na superpočítači Barbora se konkrétně jednalo o procesory Intel Xeon Gold 6126 2.2 a Intel Xeon Gold 6240 2.2. Na superpočítači Karolina proběhlo na procesorech AMD EPYC 7H12 2.3 a AMD EPYC 7763 2.3.

Pro potřeby měření bylo vytvořeno několik skriptů, které měly za úkol připravit prostředí pro měření a nainstalovat potřebné moduly. Konkretně se jedná o skripty install.sh, install_env.sh, set_env_karolina.sh a set_env_barbora.sh (součást přílohy). Pro měření se využívala možnost spuštění bez grafického uživatelského rozhraní.

4.1.1 Průběh měření

Průběh měření se dá rozdělit do několika fází. První fáze je přípravná. V této fázi proběhlo spouštění instalačních a nastavovacích skriptů, které nainstalovaly potřebné moduly. V případě měření na serverových procesorech je nejprve nutné se připojit skrze protokol SSH k takzvanému login uzlu a až poté spustit instalační skripty, které vytvoří virtuální prostředí, kde budou moduly následně nainstalovány. Další částí přípravné fáze je kompilace programu do binárního spustitelného souboru. Druhou fází je samotné spuštění aplikace a měření. V rámci měření na notebookových

procesorech probíhá spuštění měření zadáním příkazu do terminálu a to ./<název zkompilovaného binárního souboru>. V případě měření na serverových procesorech je spuštění složitější. Nejprve je potřeba vybrat procesor/uzel, který se musí na určitou dobu alokovat pomoc plánovače úloh daného výpočetního klastru. Pro tyto potřeby byl vytvořen bash skript, který předává informace o úloze systému Portable Batch System (PBS) **??**, který je využíván na systémech IT4Innovations. Ukázka bash skriptu je znázorněna ve Výpisu 4.1.1.

Výpis 4.1: Bash skript pro nastavení parametru na superpočítači Karolina a spuštěni aplikace

```
#!/bin/sh
#PBS -A PROJECTNAME
#PBS -q qprod
#PBS -l select=1:ncpus=128
#PBS -l walltime=8:00:00


cd Krypto_meric_radar/
. set_env_karolina.sh
./run_terminal_karolina_final
```

Tento skript se spouští s příkazem pro frontu (qsub) na začátku *qsub <název_skriptu>*. Řádky začínající #PBS jsou parametry pro alokaci uzlu. Po přidělení uzlu následuje přikaž change directory (cd), jeho vykonáním se přejde do adresáře ve, kterém se nachází skript pro nastavení virtuálního prostředí ten se spustí a následně se spustí aplikace.

5 Výsledky studentské práce

Výsledky bakalářské práce přináší náhled do rozdílností výkonností procesorů AMD a Intel z pohledu kryptografických operací.

5.1 Změřené závislosti

Prezentované výsledky ve formě grafů jsou výsledky měření provedených aplikací KMP. Jako vstupní data bylo vybráno celkem 300 souborů (100 textových souborů .txt, 100 obrázků .jpg a 100 .pdf souborů). Pro výslednou analýzu byly vybrány dva typy grafů. Prvním typem grafů je spojitý graf s vynesenými body, ve kterém jsou na ose x vyznačeny velikosti souborů a na ose y je znázorněna doba trvání operace (šifrování + dešifrovaní). Velikosti souborů jsou seřazeny od nejmenšího souboru po nejvyšší. Křivky zde vynesené vznikly vyznačením času pro každou velikost souboru a následným propojením těchto bodů. Každá křivka znázorňuje všech 300 naměřených hodnot pro daný procesor. Název procesoru a název vykreslovaného algoritmu se nachází v legendě spolu s barvou dané křivky. Druhým grafem zařazeným do analýzy je sloupcový graf průměrů byte za vteřinu. V tomto grafu se na ose x nachází názvy procesorů a na ose y se nachází průměrná hodnota byte za vteřinu. Procesory jsou od sebe rozlišeny barvami.

5.2 Výsledky měření na serverový procesorech

Tato kapitola prezentuje výsledky měření, která proběhla na serverových procesorech AMD EPYC 7763, AMD EPYC 7H12, Intel Xeon Gold 6126, a Intel Xeon Gold 6240.

5.2.1 AES

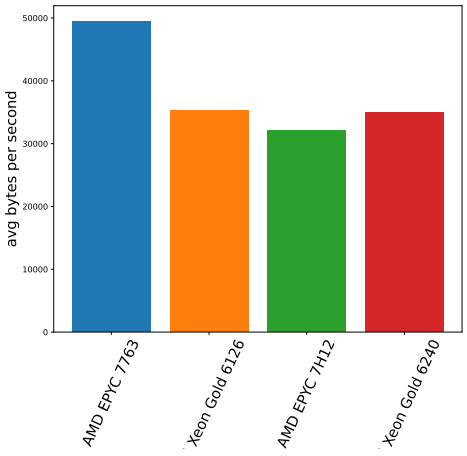
Prvním analyzovaným algoritmem je nativní implementace symetrické šifry AES s délkou klíče 128 bitů. Naměřené hodnoty jsou znázorněné na Obr. 5.3 a 5.1. Z naměřených hodnot lze usuzovat, že čas roste přímo úměrně v závislosti na velikosti souborů. Z obou grafů je patrné, že nejlepších hodnot dosahoval procesor AMD EPYC 7763 (v grafech znázorněn modrou barvou) a naopak nejhorších výsledků dosáhl AMD EPYC 7H12 (v grafech znázorněn zelenou barvou). Mezi nimi se nachází oba procesory Intel, u kterých jsou naměřené hodnoty téměř totožné, rozdíl mezi nimi je nepatrný. Naopak rozdíl mezi procesory AMD je znatelný a dosahuje až k 30 %. Druhým analyzovaným algoritmem je optimalizovaná nativní implementace symetrické šifry AES s délkou klíče 128 bitů. Jedná se stejnou implementaci, pouze s rozdílem optimalizace pomocí Cythonu. Na Obr. 5.4 a 5.2 lze na první pohled zaznamenat jistou podobnost v křivkách, respektive ve sloupcích s grafy na Obr. 5.3 a 5.1. Lze si zde všimnout změny času a tím i změny průměrné hodnoty. Podobně jako u neoptimalizované implementace, dosahoval nejlepších hodnot procesor AMD EPYC 7763 (v grafech znázorněn modrou barvou) a naopak nejhorších výsledků dosahoval AMD EPYC 7H12 (v grafech znázorněn zelenou barvou). Mezi nimi se nachází oba procesory Intel, u kterých jsou naměřené hodnoty téměř totožné, rozdíl mezi nimi je nepatrný. Podobný je i rozdíl mezi procesory AMD, který dosahuje až k 30 %.

Agregace výsledků AES-128 bez optimalizace a s optimalizací

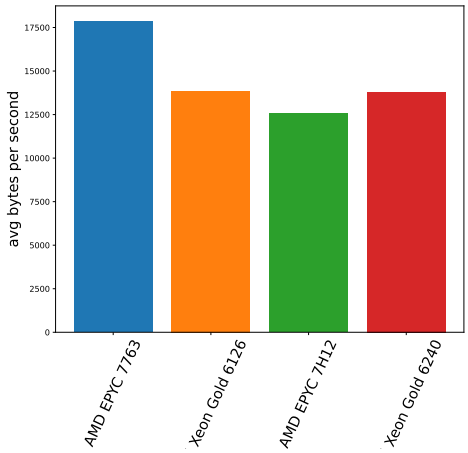
Po propojení grafů AES-128 nativní implementace v závislosti na velikosti souboru zobrazeného na Obr. 5.3 a AES-128 optimalizace Cython v závislosti na velikosti souboru zobrazeneho na Obr. 5.4 vznikl spojitý graf s osmi křivkami zobrazeny na Obr 5.5. Křivky jsou zde, opět jako v předešlých případech, rozlišeny barevně a názvy procesorů jsou umístěny v legendě. V tomto grafu je rozdíl mezi optimalizovanou implementací a neoptimalizovanou poměrně znatelný. Rozdíl je patrný i v Tab. **??**, kde se v prvním sloupci nachází název procesoru. V ostatních dvou sloupcích se nachází průměr byte za vteřinu pro danou implementaci.

Tab. 5.1: Přehled výsledků měření s optimalizací a bez optimalizace hodnoty jsou uvedeny jako průměr byte/vteřina.

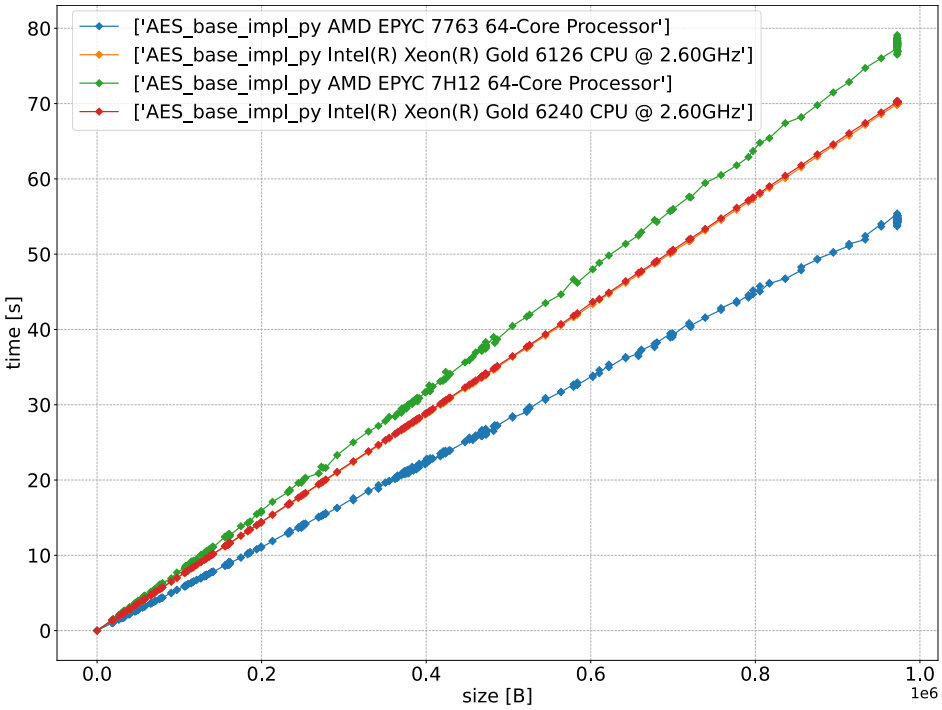
Název procesoru	Bez optimalizace	S optimalizací
`AMD EPYC 7763`	`17848.72`	49491.16
`Intel Xeon Gold 6126`	`13867.32`	35216.35
`AMD EPYC 7H12`	`12562.42`	32099.32
`Intel Xeon Gold 6240`	`13805.34`	34999.08



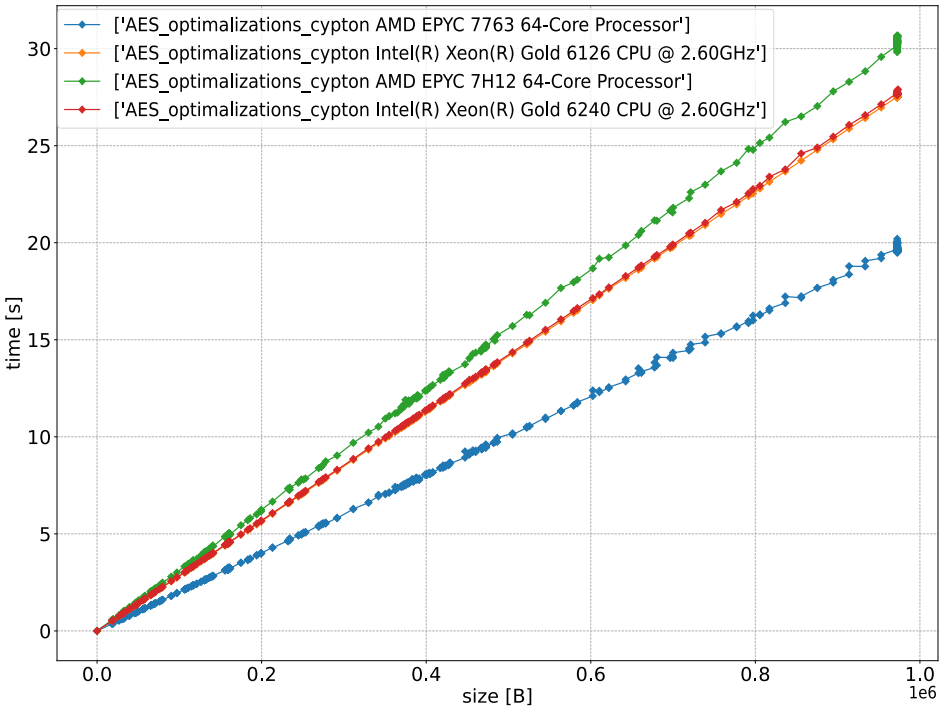
Obr. 5.1: Graf průměrů byte za vteřinu AES-128 s optimalizací Cython.



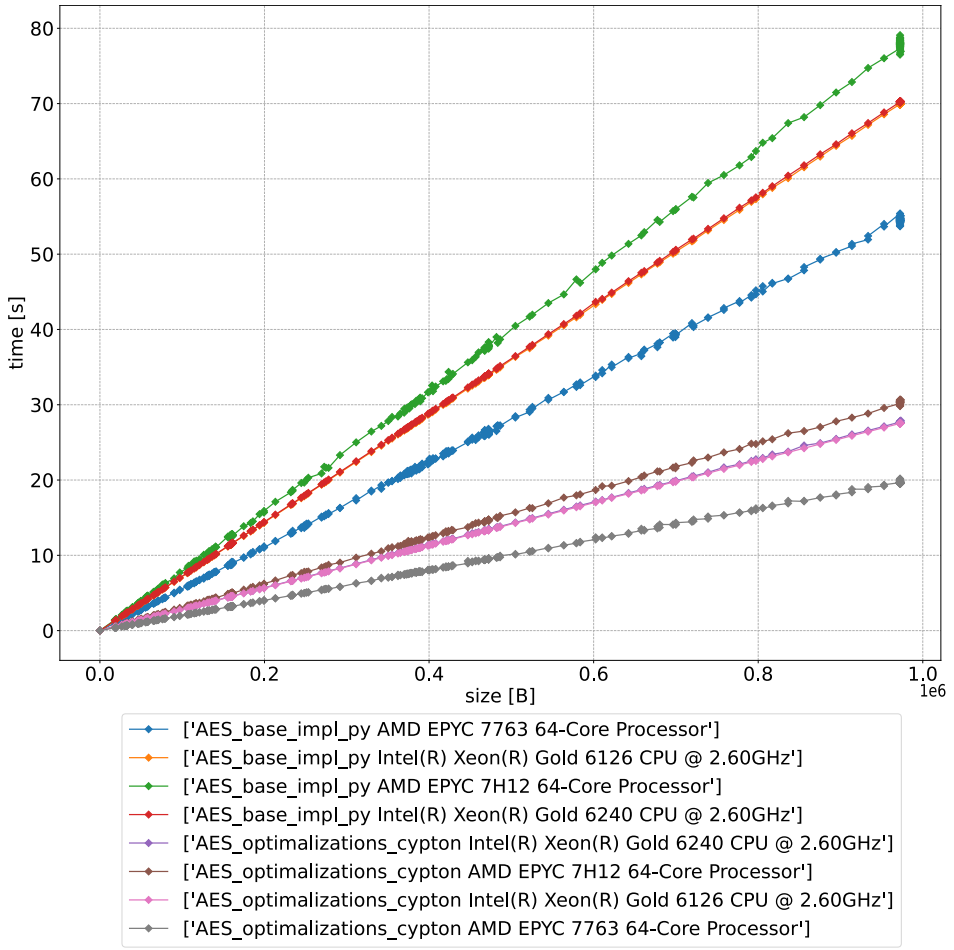
Obr. 5.2: Graf průměrů byte za vteřinu AES-128.



Obr. 5.3: Graf AES-128 nativní implementace v závislosti na velikosti souboru.



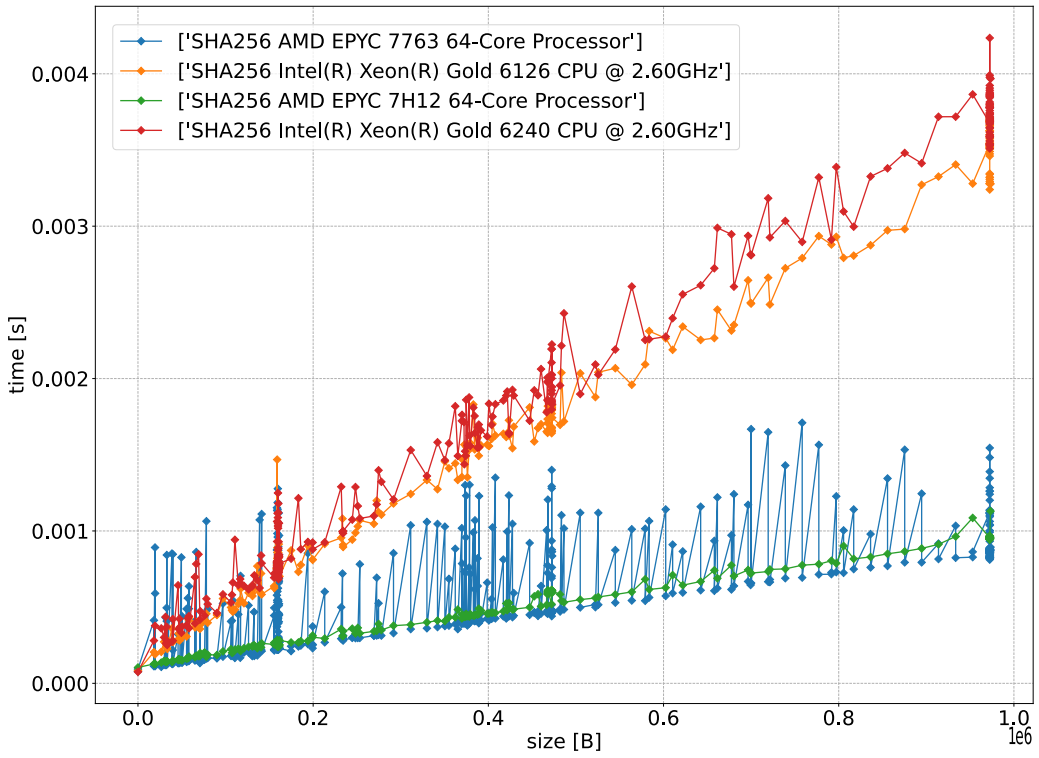
Obr. 5.4: Graf AES-128 s optimalizací pomocí Cython v závislosti na velikosti souboru.



Obr. 5.5: Graf agregace výsledků AES bez optimalizace a s optimalizací

5.2.2 SHA-256

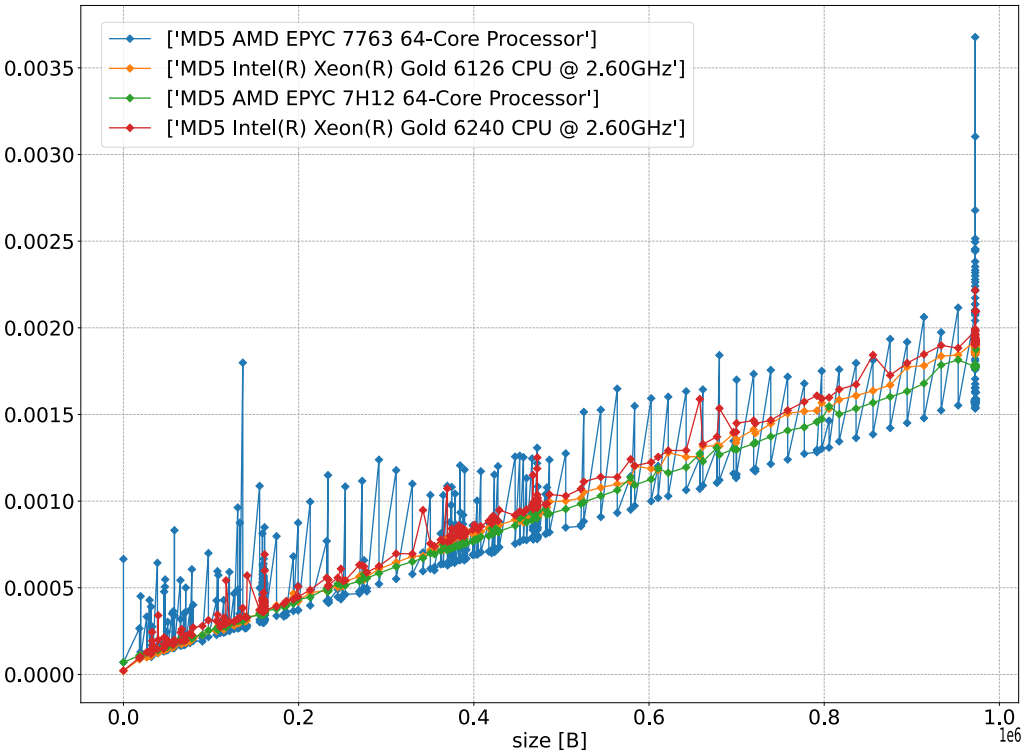
Prvním analyzovaným algoritmem z hash funkcí je SHA-256. Naměřené hodnoty jsou znázorněné na Obr. 5.6 a 5.8. Na spojitém grafu umístěném na Obr. 5.6 je při pohledu na větší vzorek dat opět patrná jistá stoupající tendence v závislosti na stoupající velikosti souborů. V případě menšího vzorku dat je možné si všimnout i nepřímé úměry, kdy operace s větším souborem trvala kratší dobu než s menším souborem. Taktéž je zde patrná jistá inkonzistence, především u grafu procesoru AMD EPYC 7763 (v grafech znázorněn modrou barvou), který v některých případech dosáhl nejnižšího času, ale naopak v jiných případech i nejvyššího. Pro objektivní hodnocení byl zvolen průměrný byte za vteřinu, znázorněný na Obr. 5.8. Nejlepších průměrných hodnot dosáhl AMD EPYC 7H12 (v grafech znázorněn modrou zelenou), druhý nejlepší byl AMD EPYC 7763 (v grafech znázorněn modrou barvou), třetí Intel Xeon Gold 6126 (v grafech znázorněn oranžovou barvou) a čtvrtý Intel Xeon Gold 6240 (v grafech znázorněn červenou barvou).



Obr. 5.6: Graf SHA-256 v závislosti na velikosti souboru.

5.2.3 MD5

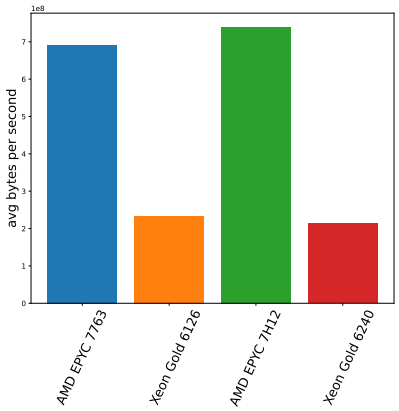
Druhým analyzovaným algoritmem je hash funkce MD5. Výsledky měření jsou znázorněny na Obr. 5.7 5.9. Na Obr. 5.7 je možno si všimnout, že spolu s růstem velikosti souboru na ose x roste i čas, který je zaznačen na ose y. Nejlepších průměrných hodnot dosáhl AMD EPYC 7H12 (v grafech znázorněn modrou zelenou), druhý byl AMD EPYC 7763 (v grafech znázorněn modrou barvou), třetí Intel Xeon Gold 6126 (v grafech znázorněn oranžovou barvou) a čtvrtý Intel Xeon Gold 6240 (v grafech znázorněn červenou barvou). Na Obr. 5.9 si lze taktéž všimnout, že výsledky dosažené všemi procesory jsou poměrně vyrovnané.



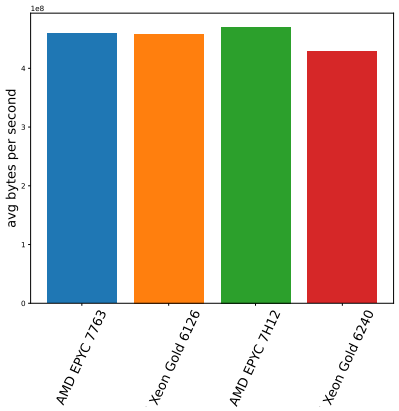
Obr. 5.7: Graf MD5 v závislosti na velikosti souboru.

Srovnání Hash funkcí

Pro srovnání hash funkcí MD5 a SHA-256 byl použit průměr byte za vteřinu zobrazený na Obr. 5.9, respektive 5.8. Hodnoty vykreslené v těchto grafech jsou zobrazeny v Tab. **??**. Kde se v prvním sloupci nachází název procesoru a v ostatních dvou se nachází průměr byte za vteřinu pro danou funkci. Taktéž byl pro toto porovnaní propojen graf 5.7 spolu s grafem 5.6 do Obr. 5.6, ve kterém se nachází osm křivek (pro každý procesor a funkci jedna). Na Obr. 5.6 lze pozorovat stoupající tendence spolu se stoupající velikostí souboru umístěnými na ose x. Nejlepších průměrných výsledků dosáhl algoritmus SHA-256 na procesoru AMD EPYC 7H12. Na Obr. 5.9 a 5.8 je možno si všimnout, že při měření algoritmu MD5 dosáhly všechny procesory podobných výsledků, kdežto výsledky SHA-256 jsou poměrně rozdílné. Na základě měření lze určit, že při obou hash funkcích dosahoval nejlepších výsledků AMD EPYC 7H12 (na Obr. 5.9 a 5.8 znázorněn modrou barvou) a nejhorších Intel Xeon Gold 6240 (na Obr. 5.9 a 5.8 znázorněn červenou barvou). Největší rozdíl nastává u procesoru Intel Xeon Gold 6240 (na Obr. 5.9 a Obr. 5.8 znázorněn oranžovou barvou), kde rozdíl mezi SHA-256 a MD5 je znatelný.



Obr. 5.8: Graf průměrů byte za vteřinu SHA-256



Obr. 5.9: Graf průměrů byte za vteřinu MD5

Tab. 5.2: Přehled výsledků měření hash funkcí hodnoty jsou uvedeny jako průměr byte/vteřina.

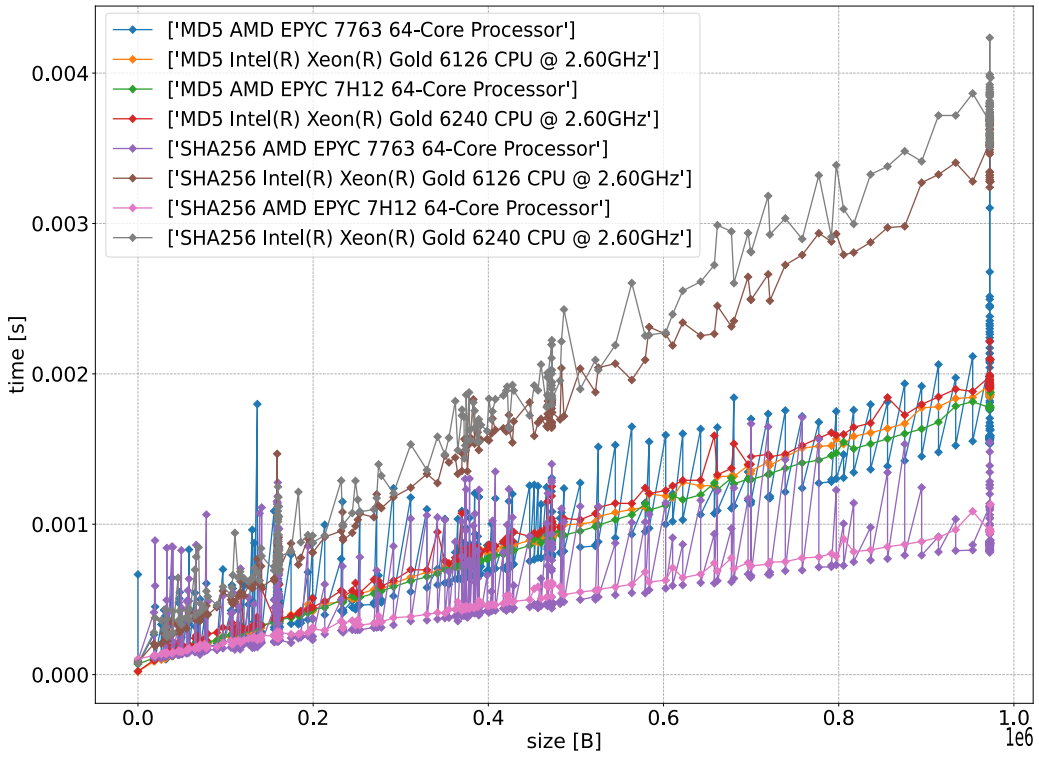
Název procesoru	SHA265	MD5
AMD EPYC 7763	692084770.08	459550314.56
Intel Xeon Gold 6126	233343577.48	458194510.42
AMD EPYC 7H12	739663094.98	470378880.28
Intel Xeon Gold 6240	214056000.17	428935859.73

5.3 Výsledky měření na notebookových (mobilních) procesorech

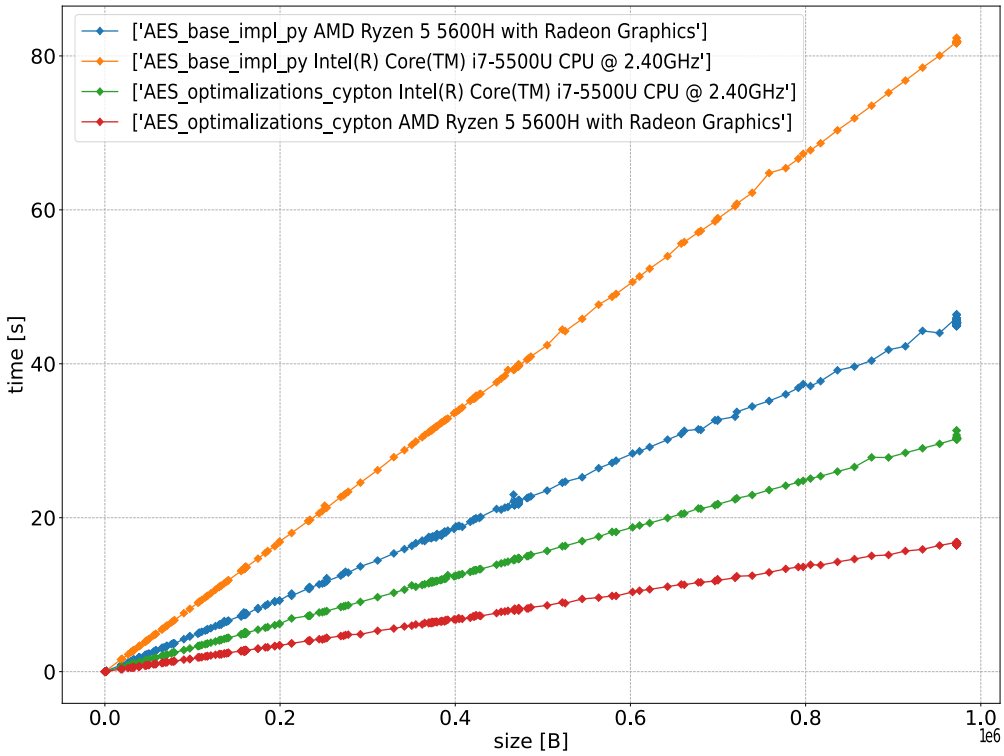
Pro analýzu na notebookových typech procesoru byla použita totožná vstupní data a jsou měřené totožné závislosti.

5.3.1 AES

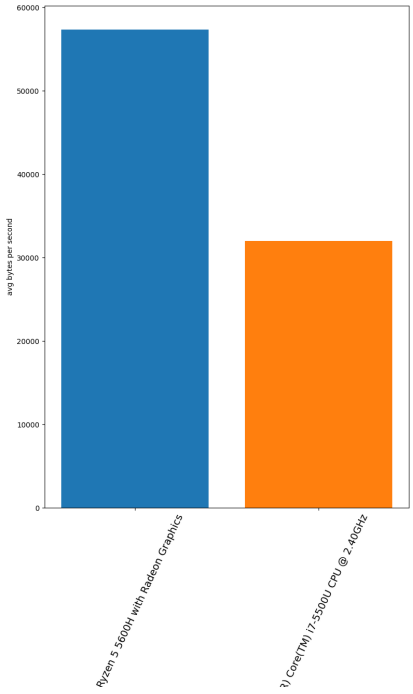
Na notebookových procesorech byla analyzována symetrická šifra AES s délkou klíče 128 bitů s použitím optimalizace pomocí Cythonu a bez něj. Měření proběhlo na dvou procesorech. Konkrétně se jednalo o procesory AMD RYZEN 5 5600H a Intel I7 5500U. Výsledky měření jsou vyneseny do sloupcových grafů Obr. 5.12 a 5.13, kde je znázorněn průměrný byte za vteřinu a agregovaný graf jak algoritmu AES s optimalizací pomocí Cythonu, tak neoptimalizovaná implementace zobrazeno na Obr. 5.11. Z těchto grafů je patrné, že procesor AMD RYZEN 5 5600H (v grafu znázorněný modře a červeně) dosahoval mnohem lepších výsledků, než procesor Intelu I7 5500U.



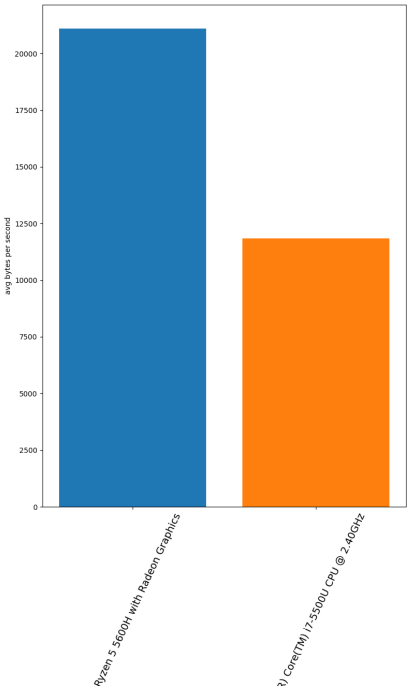
Obr. 5.10: Graf srovnání hash funkcí MD5 a SHA-256 v závislosti na velikosti souboru.



Obr. 5.11: Graf Agregace výsledků AES bez optimalizace a s optimalizací



Obr. 5.12: Graf průměrů byte za vteřinu AES optimalizace Cython.



Obr. 5.13: Graf průměrů byte za vteřinu AES

Závěr

V této bakalářské práci byla rozebrána a popsána témata, jako jsou například superpočítače, serverové procesory nebo kryptografie, která popisuje, jaké druhy šifer existují, jaké klíče používají, jakým způsobem pracují nebo k čemu se využívají.

Hlavním cílem práce bylo zanalyzovat vhodnost využití procesorů různých architektur, výrobců a modelů pro vykonávání kryptografických operací náročných na výkon a čas. Pro naplnění těchto cílů byla navržena a následně zhotovená aplikace pro měření doby běhu jednotlivých operací. Aplikace obsahuje kromě samotného měření také možnost vizualizace naměřených hodnot ve formě grafů. Aplikace disponuje grafickým uživatelským rozhraním a poměrně širokým množstvím možností konfigurace měření i následné vizualizace. Pro měření lze využít i možnost spuštění v rámci terminálu (CLI) bez nutnosti využití grafického rozhraní (GUI), ale se zachováním všech konfiguračních možností. Aplikace zvládá zpracovat libovolné typy souborů o libovolné velikosti. K aplikaci byly dále vytvořeny i instalační skripty, které zajišťují instalaci všech potřebných modulů pro správnou funkčnost aplikace.

Při implementaci šifry AES se povedlo vhodným využitím Cythonu optimalizovat čas běhů šifrovaní a dešifrovaní. Díky tomu bylo možné dosáhnout zvýšení průměrného byte za vteřinu až o 177 %, v průměru o 163,62 %.

Díky aplikaci se podařilo změřit a následně zanalyzovat kryptografické operace na šesti procesorech, na třech od společnosti AMD a třech od společnosti Intel. Jednalo se o čtyři serverové procesory (dva AMD EPYC a dva Intel Xeon) a dva notebookové, konkrétně AMD RYZEN 5 5600H a Intel I7 5500U. Na základě měření na serverových procesorech a následné analýze výsledků nelze jednoznačně určit, zda je jeden procesor nejlepší ve všech měřených operacích. Avšak lze konstatovat, že obecně lepších výsledků dosáhly procesory AMD EPYC.

Nejlepších průměrných výsledků dosáhl procesor AMD EPYC 7763, který dominoval především v rámci měření šifry AES-256, ale v rámci hash funkcí jeho výsledky tak výrazné oproti ostatním nebyly. Taktéž zaznamenal největší rozdíl mezi průměrným byte za vteřinu a mezi optimalizovanou a neoptimalizovanou implementací AES, a to o již dříve zmíněných 177 %.

V rámci hash funkcí dosahoval nejlepších výkonů druhý procesor od společnosti AMD a to konkrétně AMD EPYC 7H12. Naopak při měření šifry AES-256 dosahoval nejhorších výsledků z procesorů zapojených do měření.

Oba procesory od společnosti Intel, konkrétně Intel Xeon Gold 6126 a Intel Xeon Gold 6240, dosahovaly podobných výsledků. Z výsledku lze vyčíst, že tyto procesory zaostávaly za konkurenty od společnosti AMD především u hash funkcí SHA-256, a to poměrně výrazně. U druhé měřené hash funkce (MD5) již rozdíl nebyl tak enormní, ale i tak procesory Intel nedokázaly procesory AMD překonat. V případě

šifry AES se umístily mezi procesory AMD, ale poměrně výrazně zaostávaly za nejlepším procesorem v rámci výsledků tohoto algoritmu.

Co se výsledků měření na notebookových procesorech týče, tak lze jednoznačně určit, že lepších výsledků dosahl procesor AMD Ryzen 5 5600H oproti Intel Core i7 5500U. V rámci tohoto měření měl procesor od společnosti Intel nespornou nevýhodu, jelikož byl výrazně starší, než procesor AMD Ryzen. S přihlédnutím k tomuto faktu je výsledek měření poměrně v souladu s očekáváními a nelze jednoznačně říci, jestli by byl výsledek stejný i při měření na srovnatelné generaci procesoru, popřípadě na stejném modelovém roce. Toto měření bylo doplňkového charakteru.

Přínos této práce spočívá ve faktu, že aplikace, na které tato práce spočívá, je pouze jednou z mála, ne-li jedinou, z dostupných aplikací. Z práce s aplikací vyplývá, že je užitečná jak při analýze stávajících kryptografických algoritmů, tak při vývoji nových algoritmů.

Literatura

[1] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 2018.

[2] Technological progress. [online], 2013. URL: `https://ourworldindata.org/technological-progress`.

[3] Eliška Ochodková. Matematické základy kryptografických algoritmů, 2011.

[4] Kryptograficke prostredky doporuceni. [online], 2018. URL: `https://www.nukib.cz/download/uredni_deska/Kryptograficke_prostredky_doporuceni_v1.0.pdf`.

[5] Jawahar Thakur and Nagesh Kumar. Des, aes and blowfish: Symmetric key cryptography algorithms simulation based performance analysis. *International journal of emerging technology and advanced engineering*, 1(2):6–12, 2011.

[6] FIPS Pub. Data encryption standard (des). *FIPS PUB*, pages 46–3, 1999.

[7] David McNett. Us government's encryption standard broken in less than a day, 1999.

[8] Elaine Barker and Allen Roginsky. Transitioning the use of cryptographic algorithms and key lengths. Technical report, National Institute of Standards and Technology, 2018.

[9] Joan Daemen and Vincent Rijmen. *The design of Rijndael: AES — the Advanced Encryption Standard*. Springer-Verlag, 2002.

[10] James Nechvatal, Elaine Barker, Lawrence Bassham, William Burr, Morris Dworkin, James Foti, and Edward Roback. Report on the development of the advanced encryption standard (aes). *Journal of Research of the National Institute of Standards and Technology*, 106(3):511, 2001.

[11] Elaine Barker and Quynh Dang. Nist special publication 800-57 part 1, revision 4. *NIST, Tech. Rep*, 16, 2016.

[12] Tim Dierks, Christopher Allen, et al. The tls protocol version 1.0, 1999.

[13] Faiqa Maqsood, Muhammad Ahmed, Muhammad Mumtaz Ali, and Munam Ali Shah. Cryptography: A comparative analysis for modern techniques. *International Journal of Advanced Computer Science and Applications*, 8(6):442–448, 2017.

[14] Ronald L Rivest, Adi Shamir, and Leonard M Adleman. Cryptographic communications system and method, 9 1983.

[15] Cameron F Kerry and Patrick D Gallagher. Digital signature standard (dss). *FIPS PUB*, pages 186–4, 2013.

[16] Martin E Hellman, Bailey W Diffie, and Ralph C Merkle. Cryptographic apparatus and method, 4 1980. US Patent 4,200,770.

[17] Bart Preneel. Cryptographic hash functions. *European Transactions on Telecommunications*, 5(4):431–448, 1994.

[18] Je Sen Teh, Kaijun Tan, and Moatsum Alawida. A chaos-based keyed hash function based on fixed point representation. *Cluster Computing*, 22(2):649–660, 2019.

[19] Datové schránky přešly na sha-2. [online], 2010. URL: `https://www.lupa.cz/clanky/datove-schranky-presly-na-sha-2/`.

[20] Ronald Rivest. Rfc1321: The md5 message-digest algorithm, 1992.

[21] Juanita Blue, Eoghan Furey, and Joan Condell. A novel approach for secure identity authentication in legacy database systems. In *2017 28th Irish Signals and Systems Conference (ISSC)*, pages 1–6. IEEE, 2017.

[22] Mateusz Jarus, Sébastien Varrette, Ariel Oleksiak, and Pascal Bouvry. Performance evaluation and energy efficiency of high-density hpc platforms based on intel, amd and arm processors. In *European Conference on Energy Efficiency in Large Scale Distributed Systems*, pages 182–200. Springer, 2013.

[23] TOP500. List statistics. [online], 2021. URL: `https://www.top500.org/statistics/list/`.

[24] Vít Vondrák. It4innovations v národní a evropské infrastruktuře open science. *DSpace VŠB-TUO*, 2021.

[25] Michal Svatos, Petr Vokac, and Jiri Chudoba. Updates on usage of the czech national hpc center. Technical report, ATL-COM-SOFT-2021-036, 2021.

[26] Intel. *Data Center Solutions, IoT, and PC Innovation.* URL: `https://www.intel.com/content/www/us/en/homepage.html`.

[27] Mohamed Arafa, Bahaa Fahim, Sailesh Kottapalli, Akhilesh Kumar, Lily P Looi, Sreenivas Mandava, Andy Rudoff, Ian M Steiner, Bob Valentine, Geetha

Vedaraman, et al. Cascade lake: Next generation intel xeon scalable processor. *IEEE Micro*, 39(2):29–36, 2019.

[28] Intel. and ia-32 architectures software developer's manual. *Volume 3A: System Programming Guide, Part*, 1(64):64, 64.

[29] Hermes Senger, Jaime F de Souza, Edson S Gomi, Fabio Luporini, and Gerard J Gorman. Performance of devito on hpc-optimised arm processors. *arXiv preprint arXiv:1908.03653*, 2019.

[30] Simon Hammond, Courtenay Vaughan, and Clay Hughes. Evaluating the intel skylake xeon processor for hpc workloads. In *2018 International Conference on High Performance Computing & Simulation (HPCS)*, pages 342–349. IEEE, 2018.

[31] Samuel Naffziger, Kevin Lepak, Milam Paraschou, and Mahesh Subramony. 2.2 amd chiplet architecture for high-performance server and desktop products. In *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*, pages 44–45, 2020. doi:10.1109/ISSCC19947.2020.9063103.

[32] Tsai-Wei Wu, Stephen Lien Harrell, Geoffrey Lentner, Alex Younts, Sam Weekly, Zoey Mertes, Amiya Maji, Preston Smith, and Xiao Zhu. Defining performance of scientific application workloads on the amd milan platform. In *Practice and Experience in Advanced Research Computing*, pages 1–4. The ACM Digital Library, 2021.

[33] Mitchell Olsthoorn, Pouria Derakhshanfar, and Annibale Panichella. Hybrid multi-level crossover for unit test case generation. In *International Symposium on Search Based Software Engineering*, pages 72–86. Springer, 2021.

[34] Michael Mattioli. Rome to milan, amd continues its tour of italy. *IEEE Micro*, 41(4):78–83, 2021. doi:10.1109/MM.2021.3086541.

[35] Samuel Naffziger, Noah Beck, Thomas Burd, Kevin Lepak, Gabriel H. Loh, Mahesh Subramony, and Sean White. Pioneering chiplet technology and design for the amd epyc™ and ryzen™ processor families : Industrial product. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, pages 57–70, 2021. doi:10.1109/ISCA52012.2021.00014.

[36] Chenxi Li and Ji Li. Passive cooling solutions for high power server cpus with pulsating heat pipe technology. *Frontiers in Energy Research*, page 664, 2021.

[37] Guido Van Rossum et al. Python programming language. In *USENIX annual technical conference*, volume 41, page 36, 2007.

[38] Mark Lutz. *Programming python*. "O'Reilly Media, Inc.", 2001.

[39] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in science & engineering*, 13(2):22–30, 2011.

[40] Jeff Reback, Wes McKinney, Joris Den Van Bossche, Tom Augspurger, Phillip Cloud, Adam Klein, Matthew Roeschke, Simon Hawkins, Jeff Tratner, Chang She, et al. pandas-dev/pandas: Pandas 1.0. 3. *Zenodo*, 2020.

[41] Ekaba Bisong. Matplotlib and seaborn. In *Building machine learning and deep learning models on google cloud platform*, pages 151–165. Springer, 2019.

[42] S Vidya. Network security using python, 2018.

[43] Ricardo Martınez Santa and Fernando Martınez Santa Holman Montiel Ariza. Secure information transmission device implemented on an embedded system using 3des and aes algorithms. *International Journal of Engineering Research and Technology*, 12:1950–1956, 2019.

[44] Dmitri I Chitalov. Development of an application with a graphical user interface (gui) to compute in parallel in the openfoam environment. In *Journal of Physics: Conference Series*, volume 1399, page 033001. IOP Publishing, 2019.

[45] Val2017. URL: `https://cocodataset.org/`.

[46] Kaggle. *my receipts (pdf scans)*. URL: `ttps://www.kaggle.com/datasets/jenswalter/receipts`.

Seznam symbolů a zkratek

NIST	National Institute of Standards and Technology
AES	Advanced Encryption Standard
DES	Data Encryption Standard
TDES	Tree Data Encryption Standard
TDES	Tree Data Encryption Standard
3DEA	Tree Data Encryption Standard
TDEA	Tree Data Encryption Standard
RSA	iniciály autorů Rivest, Shamir, Adleman
DSA	Digital Signature Algorithm
SHA	Secure Hash Algorithm
MD	Message-Digest algorithm
DNSSEC	Domain Name System Security Extensions
CPU	central processing unit
HPC	High-performance computing
TDP	Thermal Design Power
AMD	Advanced Micro Devices
IBM	International Business Machines Corporation
VŠB	Vysoká škola báňská
TB	Terabyte
RAM	Random Access Memory
DDR4	Double Data Rate 4 Synchronous Dynamic Random Access Memory
nm	Nanometr
GHz	Gigahertz
MHz	Megahertz

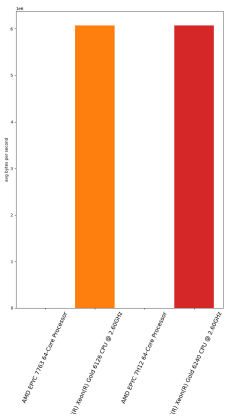
OS	Operační systém
SQL	Structured Query Language
CSV	Comma-separated values
PyPi	Python Package Index
GUI	Graphic User Interface

Seznam příloh

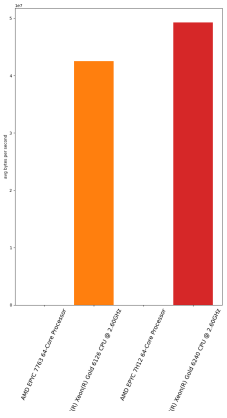
A Další grafy

Měřeni proběhlo i s jinými algoritmy a naměřené hodnoty jsou zobrazeny následujících grafech A.1 A.2 A.3. Jedná se o grafy ve kterých je vynesená průměrná hodnota byte za vteřinu. Konkrétně se jedné o algoritmy 3DES, AES s použitím knihovny pyCrypto a kombinaci AES a RSA, kde RSA se používá pro šifrovaní klíčů a AES pro šifrovaní souboru. Tyto algoritmy byly měřeny na procesorech Intel. V případě algoritmu 3DES zobrazeného v grafu je patrné, že rozdíl mezi procesory je velmi malý. Lepších výsledků dosahl procesor Intel Xeon Gold 6126 (znázorněny oranžovou barvou). V případě algoritmu AES s využitím knihovny pyCrypto je rozdíl mezi procesory znatelnejší, tento rozíl je patrý v grafu A.3. Lepších vyslédků dosáhl procesor Intel Xeon Gold 6240(znázorněný červeně). V případě kombinace šifer RSA a AES dosáhl lepších výsledků procesor Intel Xeon Gold 6126 (znázorněny oranžovou barvou) rozbrazeno v grafu A.2.

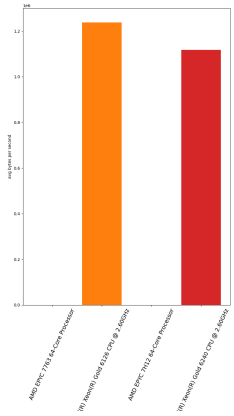
V dalších grafech konkrétně A.4 A.6 A.8 A.10 A.5 A.7 A.9 A.11 jsou vykresleny, všechny měřené algoritmy. Výsledky jsou prezentovány vždy jako dvojice grafu. Dvojice pro každá procesor. V levém grafu jsou výsledky znázorněny jako průměrný byte za vteřinu. V pravém grafu josu výsledky jako průměrná vteřiny za byte. Toto zobrazení bylo zvoleno na základě, že naměřený hodnoty pro měřené algoritmy jsou velmi rozdílné. V grafech nacházejících se vlevo je lze pozorovat algoritmy, které měly největší počet byte za jednu vteřinu, kdežto v pravých grafech lze pozorovat kolik vteřin bylo potřeba na byte pro jaký algoritmus.



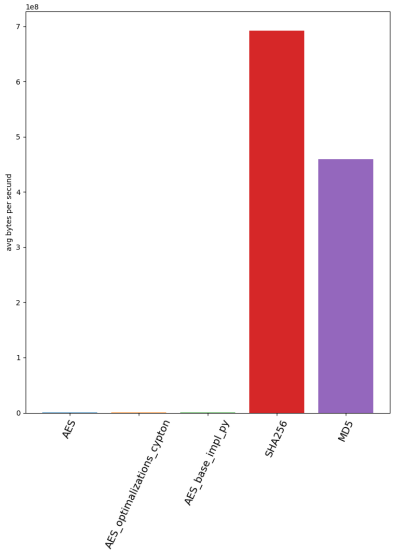
Obr. A.1: Graf průměrů byte za vteřinu 3DES. .



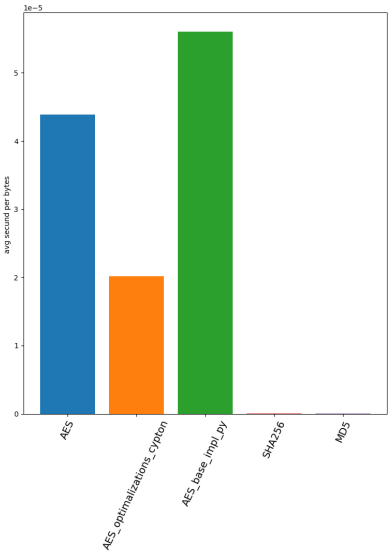
Obr. A.2: Graf průměrů byte za vteřinu AES s využitím knihovny Pycrypto.



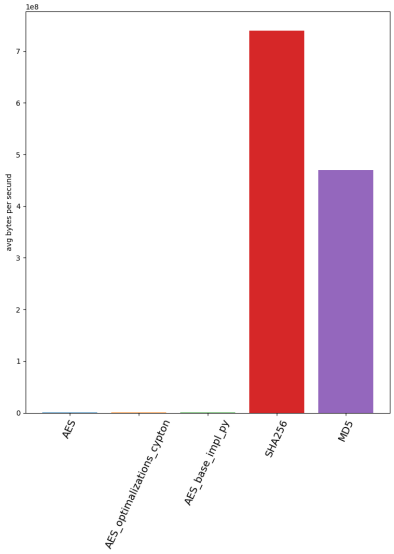
Obr. A.3: Graf průměrů byte za vteřinu kombinace AES pro šifrovaní souboru a RSA pro šifrovaní klíčů s využitím knihovny Pycrypto.



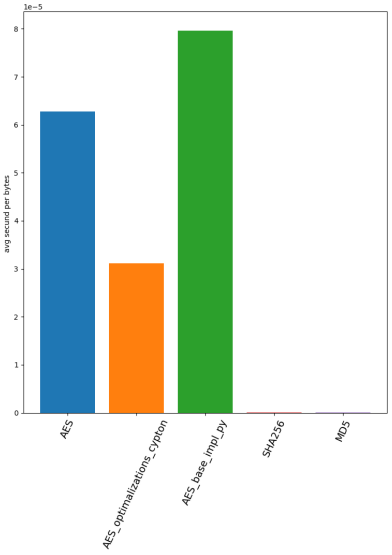
Obr. A.4: Graf průměrů byte za vteřinu AMD EPYC 7763



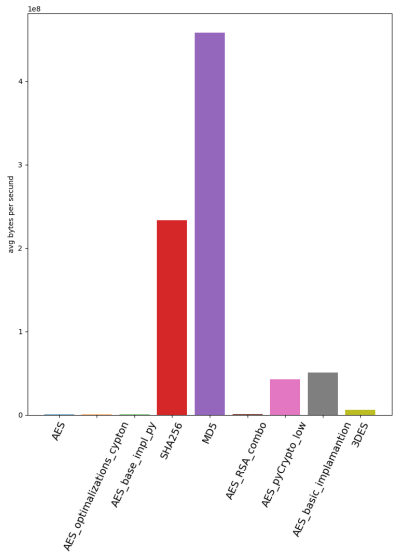
Obr. A.5: Graf průměrů vteřiny za byte AMD EPYC 7763



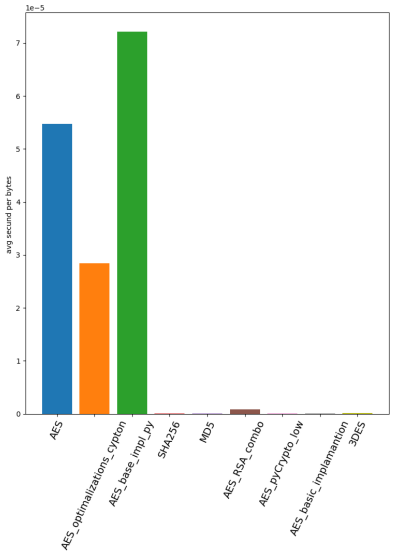
Obr. A.6: Graf průměrů byte za vteřinu AMD EPYC 7H12



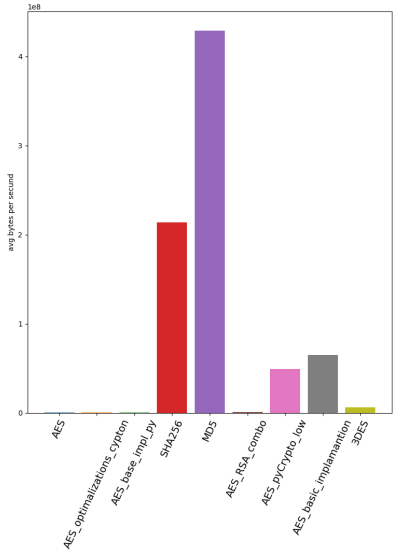
Obr. A.7: Graf průměrů vteřiny za byte AMD EPYC 7H12



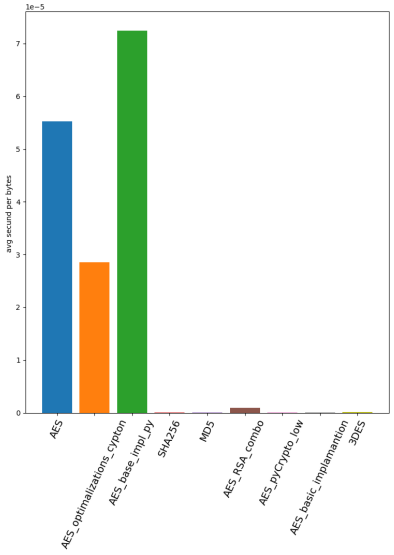
Obr. A.8: Graf průměrů byte za vteřinu Intel Xeon Gold 6126



Obr. A.9: Graf průměrů vteřiny za byte Intel Xeon Gold 6126



Obr. A.10: Graf průměrů byte za vteřinu Intel Xeon Gold 6240



Obr. A.11: Graf průměrů vteřiny za byte Intel Xeon Gold 6240

B Obsah elektronické přílohy

- `/` ..... kořenový adresář přiloženého archivu
  - `install_env.sh` .. instalační skript pro vytvořeni virtuálního prostředí a instalaci modulu
  - `install.sh` ..... instalační skript
  - `main.py` ..... soubor pro spuštěni aplikace s grafickým uživatelským rozhraním
  - `qsub_barbora_qnvidia.sh` ..... skript pro nastavení parametru na superpočítači Barbora pro akcelerovaný uzel a spuštěni aplikace
  - `qsub_barbora.sh` .. skript pro nastavení parametru na superpočítači Barbora pro výpočetní uzel a spuštěni aplikace
  - `qsub_karolia_qnvidia.sh` ..... skript pro nastavení parametru na superpočítači Karolina pro akcelerovaný uzel a spuštěni aplikace
  - `qsub_karolia.sh` .. skript pro nastavení parametru na superpočítači Barbora pro výpočetní uzel a spuštěni aplikace
  - `run_terminal_barbora_final`
  - `run_terminal_final`
  - `run_terminal_karolina_final`
  - `run_terminal.py` .... soubor pro spuštěni aplikace pouze v terminálu a pouze pro měření
  - `set_env_barbora.sh` ... skript pro spustení virtuálního prostředí na superpočítači Barbora
  - `set_env_karolina.sh` . skript pro spustení virtuálního prostředí na superpočítači Karolina
  - `tr_install.sh` ..... instalační skript
  - `src` ..... adresář se zdrojovými soubory
    - `aes_base_impl.py`
    - `AesFile.py`
    - `aes_low_pycrypto.py`
    - `analyze.py`
    - `average_s_b.py`
    - `avg_bytes_per_sec_algoritmus.py`
    - `DES3.py`
    - `meric_data_load.py`
    - `rsa_aes_file.py`
    - `aes_c` ..... adresář se zdrojovými soubory k optimalizaci Cython
      - `aes.c`
      - `aes.pyx`
  - `data` ..... adresář s daty
    - `testing_data` ..... adresář s daty se kterýma probíhalo měření
      - `Obrazky` .. adresář s obrázky se kterýma probíhalo měření dostupné na [45]
      - `Text` ..... adresář s textovými soubory se kterýma probíhalo měření
      - `PDF` ..... adresář s pdf se kterýma probíhalo měření dostupné na [46]
    - `results` ..... adresář s výsledky měření
      - `servery.csv` ..... soubor s výsledky měrení na serverových procesorech

└── `notebook.csv` ..... soubor s výsledky měření na serverových procesorech