Postkvantové algoritmy se zaměřením na homomorfní šifrování
Post-quantum algorithms with focus on homomorphic encryption

Jan Nedbal

Bakalářská práce
2023

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jan Nedbal**
Osobní číslo:	**A20443**
Studijní program:	**B0613A140020 Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Postkvantové algoritmy se zaměřením na homomorfní šifrování**
Téma práce anglicky:	**Post-quantum Algorithms with Focus on Homomorphic Encryption**

Zásady pro vypracování

1. Shrňte současný stav homomorfního šifrování.
2. Vysvětlete důležitost kvantových šifer.
3. Popište matematický aparát potřebný pro homomorfní šifrování.
4. Vytvořte ukázkový program s využitím homomorfního šifrování.
5. Uveďte využití homomorfního šifrování a možnou budoucnost.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. SINGH, Simon, Dita ECKHARDTOVÁ a Petr KOUBSKÝ. Kniha kódů a šifer. 2. vydání. Praha: Dokořán, 2009. ISBN 9788073632687.
2. JUKL, Marek, Univerzita PALACKÉHO a Přírodovědecká FAKULTA. Lineární algebra. 2. upravené vydání. Olomouc: Univerzita Palackého v Olomouci, 2010. ISBN 978-80-244-2522-1.
3. KAREL, Burda. Kryptografie okolo nás. internet: CZ.NIC, 2019. ISBN 978-80-88168-49-9.
4. HOFFSTEIN, Jeffrey, Jill PIPHER a Joseph H. SILVERMAN, 2008. An introduction to mathematical cryptography. New York: Springer. ISBN 978-0-387-77993-5.
5. STANOVSKÝ, David, 2010. Základy algebry. Praha: Matfyzpress. ISBN 978-80-7378-105-7.
6. STANOVSKÝ, David a Libor BARTO, 2017. Počítačová algebra. Druhé, upravené vydání. Praha: Matfyzpress. ISBN 978-80-7378-340-2.

Vedoucí bakalářské práce: **Mgr. Jan Krňávek, Ph.D.**
Ústav matematiky

Datum zadání bakalářské práce: **2. prosince 2022**
Termín odevzdání bakalářské práce: **26. května 2023**

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 25.5.2023 …….

podpis studenta

ABSTRAKT

Tato bakalářská práce se zabývá post kvantovou kryptologií se zaměřením na homomorfní šifrování. Součástí práce bude krátký program, který ukáže princip homomorfního šifrování. Důležitou součástí bude ukázka matematického aparátu pomocí kterého bude ukázaný princip homomorfního šifrování s RSA. Hlavní část práce se věnuje popisu současného stavu homomorfního šifrování se shrnutím důležitých průlomů. Na závěr bude uvedeno využití homomorfního šifrování a jeho možná budoucnost.

Klíčové slova: homomorfní šifrování, kryptologie, RSA

ABSTRACT

This bachelor thesis deals with post quantum cryptology with a focus on homomorphic encryption. The thesis will include a short program that will demonstrate the principle of homomorphic encryption. An important part will be a demonstration of the mathematical apparatus by which the principle of homomorphic encryption with RSA will be shown. The main part of the thesis is devoted to a description of the current state of homomorphic encryption with a summary of important breakthroughs. Finally, the use of homomorphic encryption and its possible future will be presented.

Postquantum algorithms: homomorphic encryption, cryptology, RSA

Chtěl bych poděkovat svému vedoucímu práce panu magistru Janu Krňávkovi, Ph.D. za jeho ochotu a vstřícnost.

Prohlašuji, že odevzdaná verze bakalářské/diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Homomorfní šifry jsou specializované šifry, které umožňují provádět aritmetické operace nad zašifrovanými daty bez nutnosti dešifrování. Jinými slovy, nad zašifrovanými daty lze provádět výpočty a výsledky získat v zašifrované podobě, aniž by bylo nutné dešifrovat původní data [1].

Základní princip spočívá v tom, že data jsou nejprve zašifrována veřejným klíčem. Poté se nad zašifrovanými daty provedou výpočty a výsledky se opět zašifrují stejným veřejným klíčem. Výsledek šifrování se shoduje s výsledkem výpočtu na původních datech, takže není nutné žádné dešifrování.

Existuje několik typů homomorfních šifer, které umožňují různé operace. Existují například homomorfní šifry jako sčítání, násobení a XOR [2]. Tyto typy homomorfních šifer mají lišící se vlastnosti a jsou vhodné pro různé aplikace.

Homomorfní kryptografie má mnoho praktických aplikací v oblastech, jako je cloud computing [3] (používání vzdálených serverů pro uložení, spravování a zpracování dat místo lokálního uložiště) a ochrana soukromí. Homomorfní šifrování umožňuje uživatelům ukládat data v zašifrované podobě a zároveň na nich provádět výpočty bez odhalení citlivých informací.

Významným příkladem, který navazuje na již zmíněné téma cloud computingu je použití homomorfního šifrování je takzvaný bezpečný výpočet více stran (secure multiparty computation [4]). Ten umožňuje různým stranám společně provádět výpočty nad zašifrovanými daty, aniž by došlo k odhalení citlivých informací. Tento koncept se často používá pro zpracování dat ve zdravotnictví a finančnictví, kde je důležité zachovávat soukromí. Dalším příkladem je homomorfní šifrování pro strojové učení. Jedná se o trénování modelů na zašifrovaných datech s cílem chránit soukromí a zajistit, aby data nebyla prozrazena. Tento přístup je užitečný například při zpracování citlivých údajů o pacientech nebo finančních transakcích.

Homomorfní šifrování však není bez problémů. Zaprvé je výpočetně nákladné. Proces homomorfního šifrování je obecně pomalejší než běžné operace s nešifrovanými daty, což může omezovat jeho použití v některých aplikacích. Dalším problémem je omezená podpora operací, které lze provádět nad šifrovanými daty. Tento přístup není zatím příliš efektivní pro šifrování velkého množství dat.

Homomorfní šifrování je však stále aktivní oblastí výzkumu a vývoje a mnozí se domnívají, že má velký potenciál pro ochranu soukromí v různých oblastech. Homomorfní šifry jsou výzvou pro výzkumníky a vývojáře, kteří se snaží zlepšit jejich efektivitu a rozšířit oblasti použití. V posledních letech se začaly objevovat i nové přístupy jako je třeba plně homomorfní šifrování, což je nejkomplexnější způsob, který umožňuje provádět libovolné aritmetické operace nad zašifrovanými daty.

I. TEORETICKÁ ČÁST

1 ÚVOD DO HOMOMORFNÍHO ŠIFROVÁNÍ:

1.1 Co je to homomorfní šifrování

Účelem homomorfní kryptografie je umožnit výpočet na zašifrovaných datech. To umožňuje zachovat důvěrnost dat během zpracování a provádět užitečné úlohy s daty v nedůvěryhodném prostředí. Ve světě distribuovaných výpočtů a heterogenních sítí je to neocenitelné.

Homomorfní kryptosystémy se podobají jiným formám otevřené kryptografie v tom, že šifrují data pomocí veřejného klíče a k nezašifrovaným datům mají přístup pouze ti, kteří mají odpovídající soukromý klíč. Od ostatních šifrovacích schémat se však liší tím, že používá algebraický systém, který může provádět různé výpočty (operace) nad šifrovanými daty.

Homomorfní šifrování je považováno za svatý grál šifrování [5], protože by mohlo poskytnout neprolomitelnou ochranu dat. Nicméně má velké nedostatky, protože potřebuje obrovskou výpočetní kapacitu.

Nejlépe bude vysvětlit princip homomorfního šifrování na příkladu:

Vezměme si tedy situaci, že jako lékařský výzkumník chceme vypočítat popisnou statistiku populace pacientů s rakovinou plic. Komplikace se vyskytne v podobě neschopnosti nemocnice sdílet soukromé lékařské záznamy, kvůli pravidlům ochrany osobních údajů. Řešením tedy je, že nemocnice zašifruje svá citlivá data pomocí plně homomorfního šifrování, takže jsou data chráněná, ale zároveň na nich lze počítat. Průběh by měl probíhat přes cloudové výpočetní centrum, kde nemocnice odešle zašifrovaná data, poté provede výzkumník požadované analytické funkce dokud nezíská požadovaný výsledek, který je ovšem zašifrovaný. Nakonec data dešifruje.

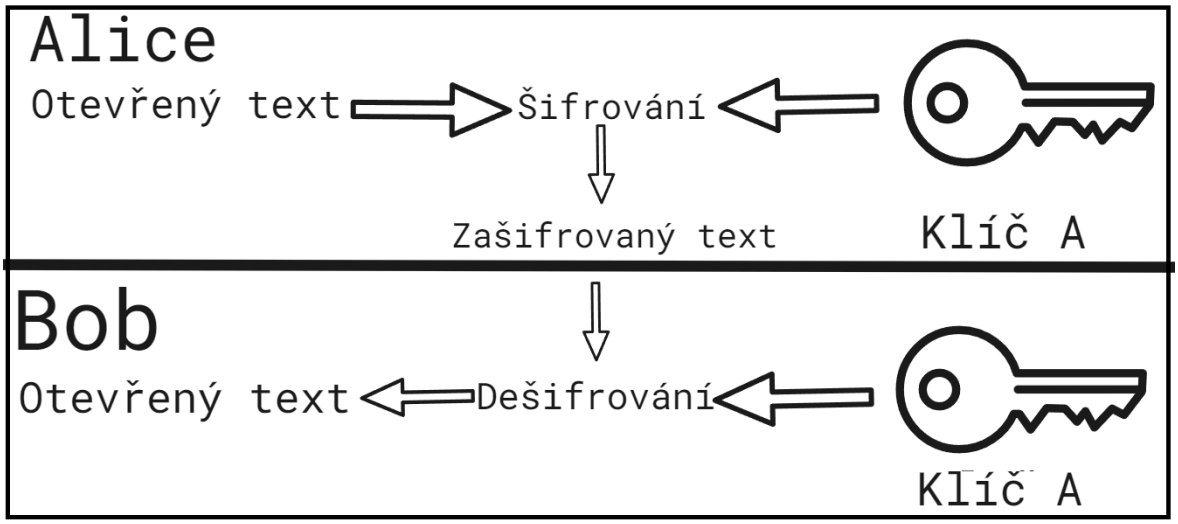
Je důležité si uvědomit, že data jsou šifrována koncovým šifrováním [6] (end-to-end encryption). Koncové je označení pro takové šifrování, při kterém je přenos dat zajištěn proti odposlechu správcem komunikačního kanálu i správcem serveru, přes který uživatelé komunikují.

Znalosti následujícího rozdělení na klasickou a moderní kryptografii a jejich příklady byly získány v předmětu AP3KR, který byl přednášen prof. Ing. Roman Šenkeřík, Ph.D. a cvičícími Ing. Milan Oulehla, Ph.D. a Ing. Petr Žáček, Ph.D.. Dále jsem čerpal z následujících pramenů: [7] [8].

1.1.1 Klasická kryptologie

Principem klasické kryptografie je šifrování (zakódování) a dešifrování (dekódování) zpráv pomocí různých technik a algoritmů, které je chrání před neoprávněným přístupem nebo čtením. Klasická kryptografie se zabývá především šifrováním a dešifrováním pomocí různých forem permutací (mění pořadí prvků v množině), substitucí (nahrazení) a polyalfabetickou kryptografii (šifrování různých částí zprávy pomocí klíčů z více než jedné abecedy).

Symetrické šifrování: Tato metoda používá stejný klíč k šifrování a dešifrování zpráv. Odesílatel a příjemce se musí předem dohodnout na společném klíči. Příklady symetrického šifrování zahrnují použití Caesarovy nebo Vigenèrovoy šifry.



Obrázek 1: Schéma symetrického šifrování

Tyto techniky jsou kombinovány a upravovány tak, aby vytvořily různé šifrovací algoritmy. Mezi nejznámější šifry patří například:

- Caesarova šifra – jednoduchá substituční šifra, kde je každé písmeno abecedy posunuto o pevný počet pozic (3). Například aplikováním na písmeno „A“ získáme „D“, šifra je jednoduše prolomitelná pomocí frekvenční analýzy
- Vigenèrova šifra – substituční šifra, která používá klíčové slovo nebo frázi k posunu písmen. Vzniká zde vzorec, který je způsoben opakovaným posouváním každého písmene o odpovídající pozici v klíčovém slově.
- Playfairova šifra – využívá 5 x 5 matici, která je naplněna neopakujícími se písmeny z abecedy (v anglickém jazyce se vynechává nebo nahrazuje písmeno „J“). Je

důležité, aby vstupní text měl sudý počet písmen, pokud nemá doplníme jedno (většinou „X“), vytvoříme dvojce neboli bigramy. Následně je každý pár nahrazen jiným párem na základě pravidel pro stejný řádek (nahradíme písmeny v pravo od nich), stejný sloupec (nahradíme písmeny pod nimi) a poslední případ, kdy dvojce nemá ani stejný řádek ani sloupec (první znak bigramu nahradíme průsečíkem řádku prvního znaku se sloupcem druhého, druhou část bigramu nahradíme průsečíkem řádku druhého znaku a sloupce prvního).

- Enigma – elektromechanická šifra používaná Německem, která využívala rotačních disků k nahrazení písmen. Rotační disky obsahovaly vnitřní obvod s abecedou. Při vstupu do rotorů byla písmena nahrazováno jinými. Poté byl šifrovaný text vložen do reflektoru, kde pokračovala symetrická substituce. Následně absolvoval cestu zpět v opačném směru v reflektoru, což způsobilo inverzní substituci. Stroj byl obsluhován stisknutím klávesy (rozsvítilo se písmeno reprezentující zašifrovanou verzi klávesy). Každý rotor se otáčel po určitém počtu stisknutých kláves (změna interního kabelování). Enigma byla složitá šifra na prolomení, protože měla mnoho možností konfigurace (počáteční pozice rotorů, jejich pořadí a také nastavení kabeláže).

1.1.2 Moderní kryptologie

V moderní době jsou šifrovací algoritmy a protokoly používané v systémech k zabezpečení komunikace a ukládání dat naprostou samozřejmostí, takže už si mnohdy ani neuvědomujeme jejich všudy přítomnost. Tyto šifry jsou navrhnuty, aby odolávaly sofistikovaným útokům, splňovaly bezpečnostní normy a optimalizovány pro okamžité použití. Mezi nejznámější patří blokové šifry, proudové šifry, hašovací funkce, asymetrické šifry, homomorfní šifry.

- Blokové šifry – fungují na principu symetrické šifry, kde je přesně definovaná délka bloku dat (nejčastěji 128 bitů), V případě většího množství dat jsou rozděleny na více bloků (pokud jsou prázdné místa je využita výplň – umělé rozšíření). Následně je každý blok nezávisle na ostatních zašifrován. Blokové šifry využívají substituci a permutaci. Dešifrování probíhá stejně jako u ostatních substitučních šifer. Pokud je opakovaně používaný stejný klíč muže dojít k prolomení pomocí kryptoanalýzy, z čehož vyplývá nutnost použití operačního módu blokové šifry (přidá sekvenci

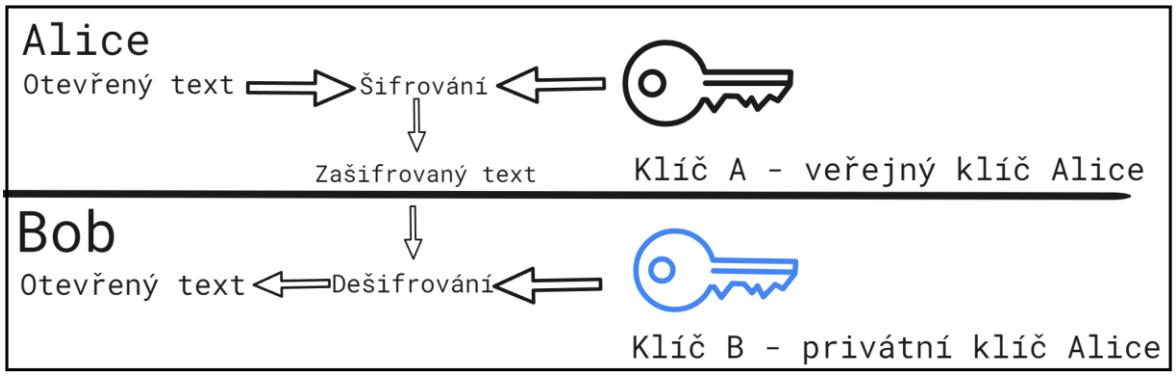
náhodnost). První bloková šifra byla DES (Data Encryption Standart) vyvinutá v roce 1977 společností IBM, která používá Feistelovu síť (rozdělíme do dvou polovin a následně opakujeme rundy – kola, ve kterých se bude provádět stejná operace). Za její pokračování můžeme označit AES (Advanced Encryption Standart), která byla schválena v roce 2001 a používá SP-síť (vstupní blok je nejdříve transformován substitucí na základě tabulky, poté se provádí permutace – mění se pořadí podle schématu, tento proces se opakuje ve vrstvách, proto se nazývá substitučně-permutační). Další příklad blokové šifry je Blowfish.

- Proudové šifry – fungují na principu šifrování nebo dešifrování jednotlivých bitů nebo bitů datových proudů. Na rozdíl od blokových šifer zpracovávají data postupně (bit po bitu). Proudové šifry využívají generátor proudu, který generuje klíčový proud (pseudonáhodná posloupnost bitů), ten je postupně kombinován s našim datovým proudem pomocí XORu. Důležité je, aby klíčový proud byl dostatečně dlouhý. Výhoda tohoto druhu šifrování spočívá v rychlosti zpracování dat a využívá se například při komunikaci v reálném čase nebo streamování videa. Mezi nejznámější zástupce patří: RC4 (byl využíván u protokolu WEP), ChaCha20 (používá se na šifrování pevného disku, ochrana datového toku při zapnuté VPN, zabezpečení při komunikaci přes mobilní aplikace).

- Hašovací funkce – je matematická funkce, která převádí vstupní data libovolné délky a převádí je na výstup určité délky (většinou malý, proto se označuje jako výtah nebo otisk). Klíčové vlastnosti jsou jednosměrnost (je téměř nemožné získat původní text), unikátnost (pokud uděláme sebemenší změnu oproti původnímu text haš bude jiný), rychlost (rychlé získání haše ze vstupních dat), stejná velikost výstupu (pokud byl použitý stejný algoritmus, vždy dostaneme stejnou délku). Mezi nejznámější haše patří: MD5 (Message-digest algorithm – algoritmus digitalizace správ, který na vstupu vytvoří 128 bitovou hašovací hodnotu), SHA-1 (Secure Hash Algorithm – bezpečnostní hašovací algoritmus, který na vstupu vytvoří 160 bitovou hašovací hodnotu), SHA-256 (Secure Hash Algorithm – bezpečnostní hašovací algoritmus, který na vstupu vytvoří 256 bitovou hašovací hodnotu). Využitím je zabezpečení hesel (využívá se na ukládání hesel, přičemž se nemusí ukládat jejich hodnota, ale pouze haš hodnota, která je porovnána se zadanou haš hodnotou), kontrolní součty (slouží k ověření integrity dat, vypočítaná hodnota přijatých dat se musí rovnat

očekávané), digitální podpisy (vytváří se v kombinaci s asymetrickým šifrováním ke zjištění autenticity).

- Kryptografie veřejného klíče (asymetrická) – Tato metoda používá dva různé klíče, veřejný klíč a soukromý klíč. Veřejný klíč se používá k šifrování zpráv a soukromý klíč se používá k dešifrování zpráv. Veřejný klíč je dostupný komukoli, ale soukromý klíč zůstává pouze u příjemce. Příkladem asymetrického šifrování je RSA (Rivest-Shamir-Adleman).



Obrázek 2: Schéma asymetrického šifrování

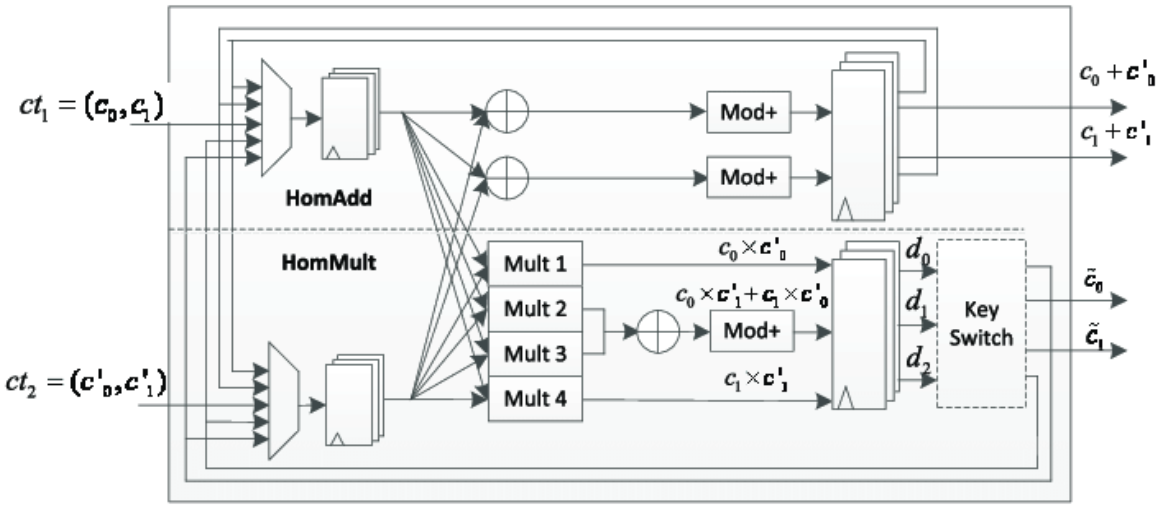
1.2 Dělení homomorfního šifrování

Homomorfní šifrování se dělí podle typu operací, které umožňuje provádět nad zašifrovanými daty.

1.2.1 Fully Homomorphic Encryption (FHE)

Plně homomorfní šifrování umožňuje provádět libovolný počet a typ operací. Dále se rozděluje na normální FHE a FHE s levelem, přičemž rozdílem je, že FHE s levelem je schopno vyhodnotit okruhy s omezenou hloubkou. Existují různá schémata například: BGV, LWE nebo CKKS.

1.2.1.1 BGV



Obrázek 3: Schéma BGV

Schéma (Obrázek 3) pochází z výzkumné práce [9] a je založeno na myšlence ideální mřížkové kryptografie, která umožňuje provádět operace kryptografického sčítání a kryptografického násobení na zašifrovaných datech.

Struktura schématu BGV (obrázek 3) je založena na použití různých matematických objektů, jako jsou mřížky, ideální mřížkové úlohy a speciální operace.

Mříž: Mříž je matematická struktura skládající se z vektorů v n-rozměrném prostoru. Pro BGV se mřížka používá pro q modulo celé číslo, kde q je velké prvočíslo. Tyto mřížky slouží jako základ pro šifrování a provádění operací nad šifrovanými daty. Ideální problém sítě: BGV využívá obtížnosti řešení problému ideální sítě. Jedním z takových problémů je Learning with Errors (LWE). LWE je problém pokusu získat informace o náhodných vektorech z mřížky se záměrně přidanými chybami. Obtížnost tohoto problému zajišťuje bezpečnost homomorfního šifrování BGV.

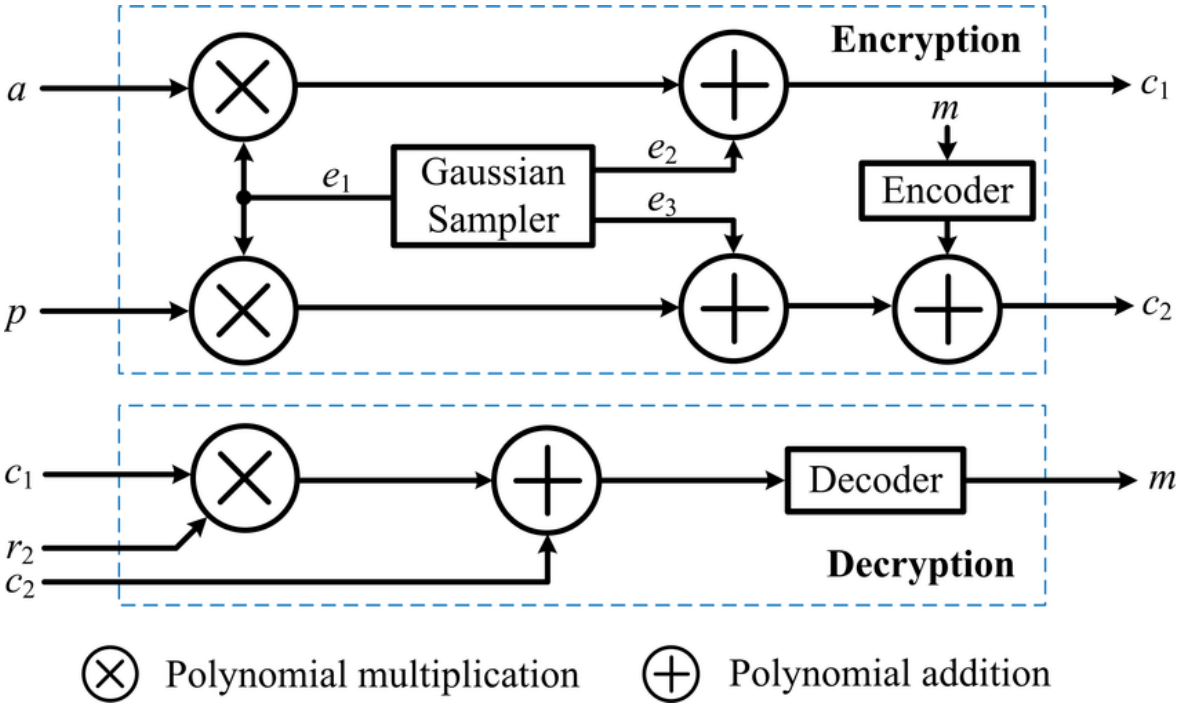
Šifrovací klíče: BGV používá veřejné a soukromé klíče. Veřejný klíč obsahuje parametry mřížky a další informace, zatímco soukromý klíč se skládá z tajných parametrů potřebných k provedení dešifrování.

Šifrování: Šifrování převede zprávu na vektor a vynásobí vektor maticí. Tato matice obsahuje náhodná celá čísla. K výslednému vektoru je pak přidána chyba založená na problému ideální mřížky. Výsledkem bude zašifrovaný vektor obsahující zašifrovanou zprávu. Homomorfní operace: BGV umožňuje provádět homomorfní operace na zašifrovaných datech. Operace Encrypt-Add umožňuje přidat dvě zašifrovaná čísla a získat

zašifrovaný výsledek. Operaci Encrypt-Multiply můžete použít k vynásobení zašifrovaného čísla konstantou, abyste získali zašifrovaný výsledek.

Dešifrování: K dešifrování se používá soukromý klíč. Vynásobte zašifrovanou zprávu soukromou mříží, abyste získali vektor. Tento vektor se dekódováním přemění na zprávu. Zprávy jsou generovány odečtením chyby a zaokrouhlením na nejbližší celé číslo.

1.2.1.2 LWE



Obrázek 4: Schéma LWE

LWE [9] je asymetrický kryptosystém založený na obtížnosti hledání vektorových chyb v lineárních kombinacích se zachyceným šumem.

Zde se pokusím vysvětlit princip LWE.

Generování veřejných a soukromých klíčů:

Vygeneruje se náhodná matice A a chybový vektor e představující náhodné hodnoty založené na rozdělení pravděpodobnosti. Matice A a vektor e se používají k vytvoření veřejných klíčů. šifrování:

Vstupní zpráva je před začátkem šifrování zakódována do vektoru zprávy m. Provede se lineární kombinace matice A a vektoru m, aby se vytvořila šifrovaná zpráva s chybou e, která do výsledku vnáší šum. Výsledkem je zašifrovaná zpráva c.

Dešifrování: příjemce zašifrované zprávy c používá svůj soukromý klíč, který obsahuje tajné informace potřebné k dešifrování. Soukromý klíč se vynásobí zašifrovanou zprávou c a vytvoří se nový vektor. Tento vektor je poté zaokrouhlen a převeden na zprávu. Ve výše uvedeném procesu je obtížné získat informace o původní zprávě m ze zašifrované zprávy c bez znalosti soukromého klíče. Problém je v tom, že lineární kombinace matice A a vektoru má chyba e vnáší do výsledku náhodný šum, což komplikuje extrakci původní informace.

1.2.2 Partially Homomorphic Encryption (PHE)

Částečně homomorfní šifrování umožňuje provádět pouze jeden typ operace (např. sčítání nebo násobení).

1.2.2.1 RSA

Představme si, že jsou dva účastníci, Alice a Bob. Alice nechce odhalovat citlivé informace a sdílet je s ostatními. Zároveň ale chce, aby Bob mohl tato data vypočítat, aniž by je odhalil.

V inicializační fázi vygeneruje Alice pár klíčů RSA (skládajícího se z veřejného a soukromého klíče, následně pošle svůj veřejný klíč Bobovi.

Ve fázi šifrování použije Alice svůj soukromý klíč na zprávu, kterou následně pošle Bobovi. Bob obdrží od Alice zašifrovaná data, na kterých provede homomorfní násobení.

V dešifrovací fázi pošle Bob zašifrovaný výsledek Alici. Alice dešifruje výsledek pomocí svého soukromého klíče a tím získá výsledek.

1.2.2.2 Pallier

Představte si, že jsou tam tři lidé, Alice, Bob a Charlie. Alice chce vypočítat součet dvou čísel, ale nechce, aby Bob a Charlie vypočítali hodnotu těchto čísel.

Proces vypadá takto:

Alice generuje veřejné a soukromé klíče pro Pallierovu šifru. Alice zašifruje první číslo svým veřejným klíčem a pošle ho Bobovi. Alice zašifruje druhé číslo svým veřejným klíčem a pošle ho Charliemu. Bob a Charlie dostanou zašifrované číslo, které nemohou dešifrovat bez soukromého klíče Alice. Bob a Charlie pošlou zašifrované číslo zpět Alici. Alice přidá číslo zašifrované pomocí svého soukromého klíče. Alice dekóduje výsledek a získá součet původních čísel.

1.2.2.3 ElGamal

Alice a Bob se předem dohodnou na společných parametrech pro šifru ElGamal, jako je velikost klíče a veřejný klíč. Bob má nějaká data, která by rád předal Alici, ale chce je také ponechat v soukromí.

Bob tato data zašifruje pomocí šifrování ElGamal s veřejným klíčem A. Bob pošle zašifrovaná data Alici. Doposud byla data předána Alici a Bobovi, ale Alice nezná skutečnou hodnotu dat.

Alice chce provádět aritmetické operace se zašifrovanými daty. Řekněme například, že chcete provést součet s jinými dostupnými šifrovanými daty.

Alice využívá homomorfní vlastnosti šifry ElGamal k provedení operace homomorfního součtu na zašifrovaných datech.

To lze provést například pomocí techniky zvané „aditivní homomorfní šifrování“, která umožňuje provádět homomorfní operace na zašifrovaných datech. Pro šifry ElGamal to znamená přidat zašifrovanou hodnotu a zašifrovat také výsledek. Důležité je, že Alice nemusí data dešifrovat, aby je sečetla. Alice pošle zašifrovaný výsledek Bobovi.

Bob nyní vlastní zašifrovaný výsledek, ale Alice nevidí skutečnou hodnotu výsledku.

Bob dešifruje výsledek pomocí svého soukromého šifrovacího klíče ElGamal, aby získal konečný výsledek operace.

1.2.3 SomeWhat Homomorphic Encryption (SWHE)

Poněkud homomorfní šifrování umožňuje provádět výpočty bez dešifrování zašifrovaných dat; SWHE podporuje homomorfní operace sčítání a násobení, ale pouze omezeně. Jinými slovy, SWHE dokáže zpracovávat pouze určité funkce nad šifrovanými daty; SWHE je běžnější než částečně homomorfní šifrování (PHE), které podporuje pouze jednu operaci (sčítání nebo násobení). SWHE je také důležitým krokem k plně homomorfnímu šifrování (Fully Homomorphic Encryption, FHE), které umožňuje provádět libovolné funkce na šifrovaných datech.

2 HISTORIE

Slovo homomorfismus pochází z řeckého jazyka (homo – stejný, morphe – forma/tvar). Termín byl poprvé použit v roce 1892 německým matematikem Felixem Kleinem. První zmínky o konceptu privátního homomorfního šifrování pochází z roku 1978 po zhruba třiceti letech začíná první období, ve kterém představil své schéma pan Gentry (2009). V roce 2010 bylo vydáno schéma DGHV. Poté následovaly další schémata například BGV. [10]

2.1 Počátky homomorfní kryptologie

Homomorfní šifrování je relativně nový koncept v kryptografii, který umožňuje provádět výpočty na zašifrovaných datech bez jejich dešifrování. Tento koncept se začal rozvíjet na přelomu 20. a 21. století, přičemž první významný krok v této oblasti se uskutečnil v roce 1978 s knihou Shafi Goldwassera a Silvia Micaliho s názvem „Pravděpodobnostní kryptografie“. byla zveřejněna.

V tomto článku autoři poprvé navrhli kryptografické schéma, které dokáže generovat náhodné šifrovací klíče a používat je k šifrování zpráv. Schéma se nazývalo „Goldwasser-Micali“ a bylo založeno na matematickém problému známém jako problém kvadratického zbytku.

2.2 Co se stalo od roku 2009

Celkově můžeme říci, že od roku 2009 bylo dosaženo mnoha důležitých milníků v oblasti homomorfní kryptografie, díky čemuž je užitečná v široké škále aplikací. Tato zjištění vedla k novým způsobům ochrany citlivých dat a zlepšení soukromí uživatelů v digitálním světě. Mezi důležité milníky patří následující.

V roce 2009 Craig Gentry představil první plně homomorfní šifrovací schéma, které umožňuje sčítání a násobení zašifrovaných dat. Toto schéma však bylo velmi pomalé a v praxi nepraktické. Gentryho schéma bylo založeno na tzv. ideální mřížce. Jedná se o matematický koncept, který nám umožňuje definovat operace se zašifrovanými daty. Toto schéma bylo založeno na kombinaci dvou typů šifrování, veřejného a soukromého. K zašifrování vstupních dat byla použita veřejná šifra a k dešifrování soukromá šifra. Gentryho schéma však bylo tak pomalé, že kryptografický výpočet byl prakticky nemožný. Ačkoli to bylo nepraktické, Gentryho schéma udělalo velké pokroky na poli homomorfního šifrování. V následujících letech bylo navrženo několik nových homomorfních šifrovacích schémat, která byla rychlejší a efektivnější než původní Gentryho schéma. Tato nová schémata

využívala různé techniky, jako např. na základě polynomiálních operací nebo ideálních sítí. Díky těmto vylepšením se homomorfní šifrování stává stále praktičtějším a používá se v různých aplikacích, jako je cloud computing a strojové učení.

Od chvíle, kdy Craig Gentry v roce 2009 představil první plně homomorfní šifrovací schéma, bylo v oblasti homomorfního šifrování dosaženo mnoha důležitých milníků. To zahrnuje navržení nového homomorfního šifrovacího schématu, které je rychlejší a efektivnější než původní Gentryho schéma.

Jedním z takových schémat je například schéma Brakerski a Vaikuntanathan, které umožňuje sčítání a násobení zašifrovaných dat s výrazně nižšími nároky na výpočetní zdroje než Gentryho schéma. Dalším příkladem je schéma Colon a Lupoint, které kombinuje různé techniky homomorfního šifrování pro dosažení vyšší účinnosti a rychlosti.

V roce 2012 představili Bos a Lauter ideální schéma založené na mřížce, které umožňuje homomorfní sčítání a násobení s mnohem lepší účinností než Gentryho schéma. To umožňuje efektivnější využití homomorfního šifrování v aplikacích, jako je cloud computing a bezpečný vzdálený výpočet.

Dalším důležitým milníkem v oblasti homomorfního šifrování bylo vydání knihovny HElib v roce 2013. Tato knihovna obsahuje implementaci homomorfního šifrování založenou na Gentryho schématu, díky čemuž je homomorfní šifrování praktické pro různé aplikace. Knihovna HElib se používá v různých aplikacích, jako je vzdálené zpracování dat a zabezpečení citlivých dat.

V roce 2014 Gentry a další autoři publikovali nový přístup strojového učení k homomorfnímu šifrování. Tento přístup umožnil efektivní aplikaci homomorfního šifrování v oblasti strojového učení a otevřel nové možnosti pro zpracování citlivých dat.

V roce 2018 byla vydána knihovna Microsoft SEAL s implementací homomorfního šifrování pro moderní aplikace. Tato knihovna podporuje různé druhy homomorfního šifrování a umožňuje efektivní využití homomorfního šifrování v různých aplikacích. B. Zabezpečení vzdáleného zpracování dat nebo citlivých dat.

V posledních letech došlo k ještě většímu pokroku v oblasti homomorfního šifrování. V roce 2019 bylo představeno nové homomorfní šifrování s názvem TFHE (Fully Homomorphic Encryption with Torus), které umožňuje rychlé homomorfní sčítání a násobení. TFHE byl vyvinut pro poskytování homomorfního šifrování v reálném čase pro aplikace, jako je výpočetní technika v neuronové síti.

2.3 Nejnovější pokroky od roku 2020

Od roku 2020 došlo v oblasti homomorfní kryptografie k několika důležitým pokrokům, které přinesly nové příležitosti pro efektivní využití homomorfní kryptografie v různých aplikacích.

Jedním z nejdůležitějších pokroků byl vývoj nového homomorfního šifrování nazvaného FHEW (Fully Homomorphic Encryption over the Wire). FHEW je algoritmus, který využívá techniky algebraické geometrie a algebraických kódů k vytvoření vysoce účinného homomorfního šifrování. To umožňuje FHEW provádět homomorfní sčítání a násobení velmi rychle a přesně, díky čemuž je homomorfní kryptografie použitelná ve výpočetně náročných aplikacích.

Dalším velkým pokrokem v homomorfní kryptografii byl vývoj nového algoritmu nazvaného CKKS (Cheon-Kim-Kim-Song), který umožňuje homomorfní sčítání a násobení dat ve spojité doméně. Algoritmus je založen na technikách algebraické geometrie a umožňuje efektivní využití homomorfního šifrování v aplikacích, které vyžadují zpracování nepřetržitých dat, jako jsou audio a video signály.

V roce 2021 byl představen nový algoritmus nazvaný FHE-MC (Fully Homomorphic Encryption with Matrix Compression). Využívá techniky z oblasti lineární algebry k vytvoření vysoce účinného homomorfního šifrování. Výsledkem je, že FHE-MC může provádět homomorfní sčítání a násobení rychle a přesně, což umožňuje použití homomorfního šifrování ve výpočetně náročných aplikacích.

V současné době je homomorfní šifrování stále aktivním výzkumným tématem a očekáváme, že se v blízké budoucnosti objeví další nové techniky a algoritmy, které umožní efektivní využití homomorfního šifrování v různých aplikacích.

3 DŮLEŽITOST A DRUHY KVANTOVÝCH ŠIFER

Kvantová kryptografie je speciální typ šifrování, který využívá principy kvantové mechaniky k ochraně komunikace před odposloucháváním a bezpečnostními útoky. Kvantová kryptografie je v kybernetické bezpečnosti velmi důležitá, protože nabízí vysoký stupeň soukromí a bezpečnosti.

Kvantová kryptografie využívá vlastnosti kvantových systémů, jako jsou kvantové bity (qubity) a kvantové stavy, k zajištění bezpečnosti komunikací. Pomocí kvantové kryptografie jsou informace zakódovány v kvantových stavech. Jinými slovy, dokáže odhalit pokusy o odposlouchávání. Když se někdo pokusí zachytit komunikaci, změní to stav kvantového systému, který lze detekovat, což nám umožní včas reagovat na potenciální hrozby.

3.1 Výhody kvantových šifer

Jednou z hlavních výhod kvantové kryptografie je její schopnost poskytovat vysokou úroveň zabezpečení. Kvantová kryptografie využívá kvantovou mechaniku k vytvoření klíčů, které jsou matematicky nerozbitné a téměř nemožné. Kvantová kryptografie umožňuje bezpečnou výměnu klíčů mezi dvěma stranami, které chtějí komunikovat přes nezabezpečenou síť. Matematický důkaz, který stojí za tímto výrokem se nazývá Shorův algoritmus.

Další výhodou kvantové kryptografie je, že může poskytnout zabezpečení proti počítačovým útokům. Tradiční metody šifrování jsou citlivé na výpočetní výkon útočníka, který dokáže šifru prolomit pomocí velmi výkonné počítačové technologie. Kvantová kryptografie je na druhou stranu vůči těmto útokům imunní. Je to proto, že výpočetní výkon potřebný k prolomení kvantové kryptografie není srovnatelný s výkonem současných počítačů. Tato skutečnost vyplývá ze skutečnosti, že kvantová kryptografie využívá kvantovou mechaniku. Kvantová mechanika se vyznačuje vlastnostmi, které se velmi liší od klasické fyziky, která je základem klasické kryptografie.

Výhodou kvantové kryptografie je potenciál vytvářet nové, efektivnější metody šifrování. Kvantová kryptografie umožňuje nové matematické metody pro ochranu dříve neznámých informací. To otevírá nové příležitosti pro rozvoj kybernetické bezpečnosti a nabízí nové a inovativní způsoby ochrany informací. Matematické důkazy pro existenci výpočetních problémů, které nelze vyřešit klasickými počítači, ale lze je vyřešit pomocí kvantových počítačů.

V neposlední řadě výhodou kvantové kryptografie je, že může poskytnout ochranu proti útokům na distribuci klíčů. Tradiční kryptografie vyžaduje výměnu klíčů na obou stranách komunikace. To vytváří potenciální zranitelnost, protože klíč může být zachycen nebo zkopírován útočníkem. Naproti tomu kvantová kryptografie umožňuje bezpečnou distribuci klíčů přes kvantové kanály. Tyto kanály využívají kvantovou mechaniku k vytvoření bezpečného prostoru pro výměnu informací a klíčů. Kvantové kanály jsou imunní vůči odposlechu, takže útočník nemůže získat informace o klíči, aniž by byl detekován.

Za zmínku stojí, že výhodou kvantové kryptografie je její škálovatelnost. Kvantovou kryptografii lze rozšířit do mnoha oblastí, jako je blockchain, bezdrátové sítě a internet věcí. Tyto nové oblasti budou využívat různé formy komunikace a budou vyžadovat nové a inovativní kryptografické metody. Kvantová kryptografie je pro tyto oblasti ideální, protože nabízí vysokou úroveň zabezpečení a je flexibilní a škálovatelná.

3.2 Nevýhody kvantových šifer

První nevýhodou kvantové kryptografie jsou její vysoké hardwarové nároky. Algoritmy kvantové kryptografie jsou založeny na principu využívání kvantových výpočtů, což klade vysoké nároky na hardwarové vybavení. Zatímco klasické počítače pracují na bitech, kvantové počítače pracují na kvantových bitech neboli tzv. qubitech. Tyto qubity je obtížné vytvořit a udržovat v kvantovém stavu, což vyžaduje speciální techniky, jako jsou supravodivé čipy a magnetická pole. Navíc jsou kvantové počítače stále velmi drahé a dnes nejsou komerčně dostupné. Existují snahy vyvinout komerční kvantové počítače, ale stále jsou velmi drahé a nákladné na výrobu a údržbu. To znamená, že kvantová kryptografie není v současnosti běžně používána ve většině oborů, které vyžadují šifrování dat.

Další nevýhodou kvantové kryptografie je omezená velikost dat, která lze šifrovat. Vzhledem ke zvláštním vlastnostem kvantové kryptografie je množství dat, která lze zašifrovat, v současné době omezeno na malé množství. Toto omezení je způsobeno tím, že kvantová kryptografie je založena na principu superpozice, který funguje pouze na malém počtu qubitů. To znamená, že kvantovou kryptografii nelze použít pro velké datové toky, jako jsou video toky nebo velké soubory. Tento problém se však aktivně řeší a výzkumníci pracují na rozšíření kapacity kvantové kryptografie tak, aby ji bylo možné použít pro větší datové toky. Doufáme, že zlepšení v této oblasti výrazně rozšíří praktické aplikace kvantové kryptografie.

Kvantová kryptografie je složitější než klasické šifry, což znamená, že jsou náchylnější k chybám a vyžadují více investic do vývoje a implementace. Jedním z hlavních důvodů je, že kvantová kryptografie vyžaduje zcela jiný matematický postup než klasická kryptografie. Většina lidí nemá potřebné znalosti z kvantové fyziky a matematiky, což ztěžuje implementaci a použití kvantové kryptografie. Kvantová kryptografie navíc vyžaduje speciální datový formát pro práci s kvantovými počítači, což může být problém pro organizace využívající nejrůznější hardware a software.

Další výzvou je omezená kompatibilita se stávající infrastrukturou a protokoly. Kvantovou kryptografii lze používat pouze s novými, kvantově kompatibilními protokoly, což vyžaduje obrovské úsilí pro upgrade stávajících sítí a infrastruktury. Pro společnosti a organizace, které spoléhají na stávající systémy a infrastrukturu, to může být obrovská výzva.

Kvantová kryptografie je relativně novým oborem kryptografie a výzkumníci stále objevují nové způsoby, jak tyto šifry prolomit. Kvantová kryptografie je odolná vůči mnoha klasickým útokům, ale existují také nové kvantové algoritmy, které lze použít k prolomení kvantové kryptografie. Například Shorův algoritmus lze použít k faktorizaci velkých čísel, což potenciálně ohrozí některé typy kvantové kryptografie, jako je RSA. Vědci však vyvíjejí novou kvantově bezpečnou kryptografii, která by mohla být v budoucnu použita ke zmírnění těchto rizik a zajištění bezpečnosti při používání kvantových technologií.

3.3 Druhy kvantových šifer

3.3.1 NTRU

NTRU (N-th Degree Truncated Polynomial Ring Units) je asymetrické kvantové post-šifrování založené na matematické struktuře polynomů. Ústředním matematickým konceptem, který je základem NTRU, je tzv. Ideal Lattice-Based Ring Learning with Errors (RLWE). V NTRU je důležité porozumět třem hlavním prvkům: tělo, okruhy NTRU a problém

NTRU těleso: V NTRU jsou tělesa obvykle založena na modulo uložených polynomech. Toto těleso obsahuje polynom s koeficienty modulo čísla q. Polynomiální koeficienty jsou obvykle celá čísla v rozsahu {-1, 0, 1} a tak dále.

NTRU okruh: NTRU šifrování funguje na speciální podmnožině těla, NTRU okruh. Tyto okruhy jsou konstruovány z párů polynomů nazývaných soukromý a veřejný klíč. Soukromý klíč se používá k dešifrování zpráv a veřejný klíč se používá k šifrování zpráv.

Problém NTRU: Základní matematický problém, na kterém je NTRU postaven. Problém je najít soukromý klíč z veřejného klíče. Tento problém je těžko řešitelný konvenčními výpočetními technikami, ale musíme zajistit, aby byl veřejný klíč dostatečně náhodný a neobsahoval žádné informace o soukromém klíči. NTRU využívá vlastnosti ideálního základního RLWE k zajištění kryptografické odolnosti proti kvantovým útokům, zejména útokům založeným na faktorizaci čísel nebo Schollově algoritmu rozkladu komplexních čísel.

3.3.2 Multivariate polynomial-based šifry

Kryptografie založená na mnohorozměrných polynomech je založena na matematické struktuře polynomů s více proměnnými. Tyto šifry využívají jako základ pro pevnost šifry složitost řešení systému polynomiálních rovnic.

Hlavní myšlenkou mnohorozměrné polynomiální kryptografie je, že její bezpečnost je založena na obtížnosti řešení složitého polynomiálního systému pro útočníka. Kryptografické operace v těchto šifrách se provádějí pomocí mnohorozměrných polynomů a algoritmů pro manipulaci s nimi.

K dosažení bezpečnosti v kryptografii na bázi mnohorozměrných polynomů je vybrán obtížně řešitelný matematický problém. Obvykle se jedná o problémy související s invertováním polynomiální funkce nebo hledáním jejích kořenů. Řešení těchto problémů je pravděpodobně obtížné a vyžadovalo by výpočetní zdroje přesahující možnosti současných kvantových počítačů.

Šifrování využívá transformací a operací s polynomy, které zaručují důvěrnost zpráv a produkci šifrovaného textu. Mezi tyto operace patří například sčítání polynomů, násobení polynomů, proměnná permutace a další matematické transformace.

Potíž při prolamování šifer založených na mnohorozměrných polynomech spočívá v tom, že útočník musí najít řešení systému polynomických rovnic, které mu umožní získat původní zprávu nebo dešifrovat šifrovaný text. Tento problém se nazývá problém odhadu nebo inverzní transformace.

3.3.3 Code-based šifry

Šifrování založené na kódu je založeno na konceptu kódů pro opravu chyb, což jsou matematické nástroje pro detekci a opravu chyb v přenášených datech. Tyto kódy jsou založeny na algebře nad konečnými poli (Galoisova pole) a používají lineární aritmetiku ke kódování a dekódování zpráv.

Základním prvkem kryptografie codebase je veřejný klíč, který je vytvořen pomocí generátorové matice. Matice generátoru obsahuje informace o kódu, což mu umožňuje zakódovat zprávu a vytvořit šifrovanou verzi. Tato matice se stává součástí veřejného klíče. Kombinace veřejného klíče a zprávy se používá pro šifrování a transformuje se pomocí generátorové matice. V důsledku toho lze odesílat šifrované zprávy.

Dešifrování používá soukromý klíč, který obsahuje informace potřebné k dešifrování zprávy. Soukromý klíč obsahuje inverzní matici generátoru. To umožňuje dešifrovat zašifrovanou zprávu a získat původní prostý text. Bezpečnost kódově založené kryptografie spočívá v obtížnosti prolomení lineárních kódů, což jsou NP-úplné problémy. K útoku na tuto šifru musí útočník získat informace o matici generátoru a úspěšně dokončit proces dešifrování. Kombinatorická povaha problému však dešifrování komplikuje a vyžaduje výpočetní zdroje, které v kvantových počítačích nenajdeme.

3.4 Druhy kvantových protokolů

3.4.1 BB84

Protokol BB84 je protokol distribuce kvantových klíčů navržený Bennettem a Brassardem v roce 1984. Protokol BB84 používá k distribuci klíčů kvantové zapletení. Protokol BB84 používá k distribuci klíčů náhodně generované bity a kvantové stavy. Alice vygeneruje náhodnou bitovou sekvenci a náhodně vybere jeden ze čtyř kvantových stavů (polarizací) pro každý bit. Tyto stavy jsou odeslány Bobovi prostřednictvím kanálu. Bob měří polarizaci každého kvantového stavu pomocí náhodně vybraných filtrů. Pokud je Bobův filtr správný, měření proběhne úspěšně a Bob získá hodnotu Alice. Pokud je Bobův filtr špatný, měření se nezdaří a Bob dostane náhodnou hodnotu.

3.4.2 B92

B92 je protokol distribuce kvantového klíče vyvinutý panem Bennett v roce 1992. Protokol B92 používá k distribuci klíčů kvantové zapletení. Zapletení(entanglement) je kvantová

vlastnost, která nastává, když je skupina částic interaguje, je generována nebo sdílí prostor v takové blízkosti, že kvantový stav každé částice skupiny nelze popsat bez ostatních, a to i v případě větší vzdálenosti. Téma provázání se nevyskytuje v klasické fyzice, ale pouze v kvantové.

Protokol B92 se skládá ze tří fází:

- Inicializace: Alice a Bob náhodně vyberou bity a připraví kvantový stav.
- Měření: Alice a Bob měří kvantové stavy.
- Oprava: Alice a Bob porovnávají bity a odstraňují bity, které se neshodují.

3.4.3 SARG

Protokol SARG využívá symetrické kódování a asymetrické měření. Symetrické šifrování znamená, že Alice a Bob používají stejný klíč k šifrování a dešifrování zpráv. Asymetrické měření znamená, že Alice a Bob měří kvantový stav odlišně. Protokol SARG je založen na protokolu BB84.

BB84 je protokol distribuce kvantového klíče vyvinutý Bennettem a Brassardem v roce 1984. Protokol BB84 používá k distribuci klíčů kvantové zapletení.

4 KNIHOVNY PRO HOMOMORFNÍ ŠIFROVÁNÍ

Samotné homomorfní šifrování je obtížné implementovat a vyžaduje speciální znalosti a zkušenosti. Homomorfní šifrovací knihovny poskytují programátorům a vývojářům přístup k hotovým řešením, která lze snadno integrovat do stávajících aplikací.

Výhodou knihovny je, že programátoři nemusí implementovat homomorfní šifrovací algoritmy od začátku, což snižuje náklady a čas potřebný k vytvoření nových aplikací. Knihovny také poskytují vývojářům lepší kontrolu nad šifrováním a zabezpečením dat, včetně správy a konfigurace klíčů.

Homomorphic Encryption Library umožňuje vývojářům rychleji vytvářet nové aplikace, které používají homomorfní šifrování. Není potřeba žádné hluboké porozumění matematice za technologií. To znamená, že i programátoři bez speciálního školení v kryptografii nebo matematice mohou ve svých aplikacích používat homomorfní šifrování.

Homomorfní šifrovací knihovny jsou nyní k dispozici pro několik programovacích jazyků, včetně Pythonu, C++ a Java, díky čemuž je homomorfní šifrování pro programátory snadno dostupné a praktické.

4.1 Knihovny v jazyce Python

Python má několik knihoven pro homomorfní šifrování. Některé z nejoblíbenějších jsou: Pyfhel, TenSEAL, PySEAL, Helib-Python.

4.1.1 Pyfhel

Pyfhel [11] je homomorfní šifrovací knihovna, která využívá principy plně homomorfního šifrování (FHE) a používá algoritmus FHE, který je založen na Gentryho bootstrap technice. Tato technika umožňuje „obnovit“ zašifrovaná data tak, aby byla šifra stále platná, a zároveň eliminuje hromadění chyb, které mohou nastat při opakovaném šifrování a dešifrování.

Pyfhel také používá různé typy šifrování, včetně symetrického a asymetrického šifrování. Tyto šifry umožňují efektivní šifrování a dešifrování dat podle konkrétních požadavků aplikace. Jedním z klíčových prvků Pyfhelu je třída EncryptedNumber, která představuje zašifrovaná uložená čísla. Tato třída umožňuje provádět aritmetické operace se zašifrovanými daty. Sčítání, odčítání, násobení atd. Výsledkem těchto operací je číslo, které je uloženo v zašifrované podobě.

Pyfhel obsahuje také mnoho dalších funkcí a nástrojů pro práci s homomorfním šifrováním. B. Schopnost generovat veřejné a soukromé klíče pro homomorfní šifrování a šifrovat a dešifrovat data v homomorfní podobě. Knihovna také podporuje různé aritmetické operace se zašifrovanými daty, jako je sčítání, odčítání, násobení, dělení, modulo a umocňování. Umožňuje také výpočet homomorfních agregačních funkcí, jako je součet, průměr, min, max. Pyfhel také podporuje serializaci a deserializaci šifrovaných dat do různých formátů a paralelní výpočet šifrovaných dat, což umožňuje efektivní výpočty velkých datových sad a zkracuje dobu výpočtu.

S těmito funkcemi a nástroji knihovna Pyfhel usnadňuje a zefektivňuje práci s homomorfním šifrováním v programovacím jazyce Python a umožňuje vývojářům provádět výpočty na zašifrovaných datech bez jejich dešifrování. To je užitečné v různých oblastech, jako je ochrana dat, analýza dat a strojové učení.

4.1.2 TenSEAL

Knihovna TenSEAL [12] umožňuje vývojářům vytvářet a používat FHE na vysoké úrovni abstrakce, aniž by vyžadovali podrobné znalosti matematických principů homomorfního šifrování. Knihovna poskytuje jednoduché rozhraní pro šifrování a dešifrování dat, které umožňuje provádět operace, jako je sčítání, násobení a další matematické operace se zašifrovanými daty.

Knihovna TenSEAL poskytuje mnoho funkcí a nástrojů, které umožňují programátorům praktikovat homomorfní šifrování. Jednou z jeho funkcí je podpora různých šifrovacích schémat, jako je BFV a CKKS, což umožňuje programátorům vybrat si nejlepší schéma pro konkrétní použití. Knihovna navíc umožňuje automatický výběr optimálních parametrů pro danou aplikaci, což výrazně zjednodušuje vývoj a implementaci homomorfního šifrování.

Další funkcí je podpora různých typů dat, jako jsou plovoucí, vektory a matematická data. Knihovna také umožňuje paralelní výpočet šifrovaných dat, což umožňuje rychlejší výpočty na více CPU nebo GPU. Kromě toho vám tato knihovna umožňuje šifrovat datové toky, což vám umožňuje šifrovat a analyzovat data v reálném čase.

Knihovna TenSEAL je navržena pro snadnou integraci s dalšími nástroji a knihovnami, jako jsou NumPy, PyTorch a TensorFlow. To umožňuje programátorům snadno používat homomorfní šifrování ve stávajících projektech.

4.1.3 PySEAL

PySEAL [13] (Python bindings for Simple Encrypted Arithmetic Library) je homomorfní šifrovací knihovna založená na Simple Encrypted Arithmetic Library (SEAL) jazyka C, umožňuje programátorům používat SEAL v Pythonu a snadno integrovat homomorfní šifrování do svých aplikací.

PySEAL implementuje různé homomorfní šifrovací algoritmy, jako je Fully Homomorphic Encryption (FHE), Partially Homomorphic Encryption (PHE) a Homomorphic Secret Sharing (HSS).

Knihovna umožňuje programátorům vytvářet a používat kryptografické klíče, vytvářet a šifrovat data, provádět aritmetické operace se zašifrovanými daty a dešifrovat výsledky. Dále také podporuje různé úrovně zabezpečení, což vám umožňuje nastavit parametry pro šifrování a dešifrování dat. PySEAL vám umožňuje vytvořit tři typy šifrovacích klíčů: veřejný, soukromý a šifrovací klíč zpráv. Veřejné a soukromé klíče se používají pro homomorfní šifrování a klíče pro šifrování zpráv umožňují bezpečné odesílání šifrovaných zpráv mezi různými stranami.

PySEAL také umožňuje různé druhy kryptografických a aritmetických operací. B. Aritmetické operace s celými čísly, polynomy nebo maticemi. To umožňuje programátorům používat knihovnu pro kryptografii a výpočty v různých aplikacích, jako je strojové učení, analýza dat a cloud computing.

4.1.4 Helib-Python

Jednou z dalších knihoven je HElib-Python [14]. Je založen na knihovně HElib a navržen pro programovací jazyk Python. Knihovna HElib-Python usnadňuje programátorům používat homomorfní šifrování v Pythonu bez hlubokých znalostí matematiky nebo kryptografie.

Provoz HElib-Python je založen na několika základních principech: asymetrické LWE šifrování, homomorfní operace, generování a správa klíčů a bezpečnost. Šifrování je založeno na ideálním LWE šifrování, které umožňuje šifrování a dešifrování pomocí speciálních klíčů. Homomorfní operace umožňují provádět výpočty se zašifrovanými daty bez jejich dešifrování. Pro šifrování a dešifrování se používá speciální klíč vygenerovaný speciálním algoritmem. Kromě toho je HElib-Python odolný vůči mnoha typům útoků. Díky

knihovně můžete využít výhody homomorfního šifrování, i když nemáte odborníka s hlubokými znalostmi matematiky nebo kryptografie.

4.2 Knihovny v jazyce C++

Existuje spousta knihoven pro jazyk C++, mezi nejpopulárnější patří: SEAL, HElib, TFHE, HEAAN.

4.2.1 SEAL

SEAL (Simple Encrypted Arithmetic Library [15]) je homomorfní šifrovací knihovna vyvinutá společností Microsoft Research. Tato knihovna, navržená tak, aby efektivně prováděla výpočty na šifrovaných datech, nabízí mnoho skvělých funkcí.

SEAL podporuje plně homomorfní šifrování (FHE) pomocí algoritmů CKKS a BFV. Kromě toho je navržen tak, aby byl velmi rychlý a efektivní. To zahrnuje mnoho výpočetních optimalizací pro velké soubory dat, včetně rychlého násobení a sčítání v polynomiálních obvodech. Díky své flexibilitě umožňuje SEAL uživatelům přizpůsobit šifrování jejich specifickým potřebám. Můžete si vybrat různé parametry šifrování, jako je velikost klíče, velikost polynomu atd. Z hlediska bezpečnosti poskytují SEAL také vysokou úroveň ochrany šifrovaných dat. Využívá nejnovější šifrovací algoritmy, jako je AES (Advanced Encryption Standard) a obsahuje mnoho bezpečnostních funkcí, jako je kontrola integrity a ověřování.

Kromě toho je SEAL přívětivý pro vývojáře díky intuitivnímu rozhraní API a sadě vzorových kódů, které demonstrují základní funkce knihovny. Tyto vlastnosti dělají ze SEAL jednu z nejpopulárnějších homomorfních šifrovacích knihoven a jsou široce používány v různých aplikacích, jako jsou kryptografické aplikace, průmyslové systémy a strojové učení.

4.2.2 HElib

HElib [16] je knihovna plně homomorfního šifrování (FHE) vyvinutá na Kalifornské univerzitě v Los Angeles (UCLA). To, co dělá HElib výjimečným, je jeho flexibilita a škálovatelnost pro různé úkoly. Knihovna podporuje šifrování pomocí Gentryho algoritmu a může provádět homomorfní operace s libovolnými polynomy, což umožňuje efektivní zpracování větších datových sad. Kromě toho HElib poskytuje nástroje pro optimalizaci výkonu, jako například: B. Techniky "Bootstrap". To umožňuje transformaci šifrovacích dat

z úrovně homomorfního šifrování na úroveň s více operacemi. To umožňuje aplikace v oblastech, jako je analýza dat, strojové učení a hledání vzorců v datech. HElib také podporuje multi-threading, což zlepšuje efektivitu a zrychluje výpočet při zpracování velkého množství dat. Tyto vlastnosti dělají z HElib jednu z nejpoužívanějších a nejvýkonnějších knihoven pro homomorfní šifrování.

4.2.3 TFHE

Knihovna TFHE (Fully Homomorphic Encryption over Torus [17]) se specializuje na provádění booleovských operací a je napsána v C. TFHE používá pro homomorfní šifrování algoritmus GSW (Gentry-Sahai-Waters) a může provádět homomorfní operace AND, OR a XOR.

Zajímavým aspektem knihovny TFHE je její schopnost manipulovat s šifrovaným textem uloženým v tom, co se nazývá torus. Tento torus je ve skutečnosti matematický objekt, který lze popsat jako dvourozměrnou periodickou mřížku bodů, z nichž každý má dvě souřadnice, jejichž hodnoty jsou omezeny na určitý rozsah. Díky použití torusu může knihovna TFHE zpracovat větší datové sady než jiné homomorfní šifrovací knihovny.

Zajímavé také je, že knihovna TFHE je maximálně efektivní a lze ji snadno integrovat s jinými knihovnami a aplikacemi. To znamená, že vývojáři mohou používat běžné funkce pro homomorfní šifrování a dešifrování dat, aniž by museli implementovat samotné složité algoritmy.

4.2.4 HEAAN

Knihovna HEAAN (Homomorphic Encryption of Approximate Number Operations [18]) se specializuje na homomorfní šifrování aritmetických operací pomocí přibližných čísel. To je důležitá vlastnost, která ji odlišuje od ostatních homomorfních šifrovacích knihoven. HEAAN využívá Gentryho algoritmus Fully Homomorphic Encryption (FHE) a provádí výpočty pomocí přibližných polynomů, což umožňuje efektivní homomorfní šifrování na velkých souborech dat.

V praxi to znamená, že knihovna HEAAN je vhodná pro aplikace, které vyžadují výpočty s velkým množstvím numerických dat, například v oblasti strojového učení a statistiky. Použití přibližného čísla také snižuje počet potřebných homomorfních operací, čímž se zkracuje doba výpočtu, což je důležité pro aplikace v reálném čase. HEAAN také poskytuje snadno použitelné nástroje, jako je převod dat mezi standardními formuláři a těmi, které se

používají v homomorfním šifrování, a podporuje širokou škálu homomorfních operací, jako je sčítání, násobení a umocňování.

II. PRAKTICKÁ ČÁST

5 MATEMATICKÝ ZÁKLAD

Na následující informace bylo čerpáno z pramenů: [19] [20] [21] [22] [23] [24] [25] [26] [27] [28].

5.1 Homomorfismus

Definice homomorfismu: homomorfismus je zobrazení mezi dvěma algrami stejného typu, která zachovává operace daných algeber to znamená, že zobrazení $f: A \to B$ mezi dvěma matematickými algebrami A, B se stejným typem, pro zjednodušení předpokládejme s jednou binární operací ×, pak platí: $f(x \times y) = f(x) \times f(y)$, pro každý pár x, y z prvků A. Tj. f zachovává operaci ×. Obecněji zobrazení $f: A \to B$ zachovává $k$ární operaci μ, pokud $f\big(\mu_A(a_1, a_2, \dots, a_k)\big) = \mu_B\big(f(a_1), f(a_2), \dots, f(a_k)\big)$, pro všechny prvky $a_1, a_2, \dots, a_k$ v A. Mezi operace, které musí být homomorfismem zachovány patří i 0ární (konstanty).

Příklad homomorfismu mezi dvěma grupami:

Grupa je algebra tvořená množinou spolu s jednou binární operací, která je asociativní a má neutrální prvek. Každý prvek má k sobě inverzní. Příkladem grup jsou třeba: nenulová racionální čísla s operací násobení nebo celá čísla s operací sčítání.

Homomorfismus Ⱶ mezi dvěma grupami$(G, \blacksquare)$ a $(H, *)$ je takové zobrazení, že: $\text{Ⱶ}(x \blacksquare y) = \text{Ⱶ}(x) * \text{Ⱶ}(y)$, pro $x, y \in G$.

Příklad. Přirozený logaritmus ln je homomorfismus z grupy $(R^+, \cdot)$ na grupu $(R, +)$. Logaritmus součinu je roven součtu logaritmů činitelů. Tj. platí, že:

$\ln(a \cdot b) = \ln a + \ln b$, kde a, b je z R^+

Na ukázkovém příkladu:

$$\log_2 8 + \log_2 16 = \log_2(8 \times 16)$$

$$3 + 4 = 7$$

U logaritmů platí isomorfismus, což je speciální typ homomorfismu, u kterého může být provedeno inverzní zobrazení.

Mezi další příklady patří homomorfismy algebraických vektorových prostor nad R. Můžeme si definovat T: $R^3 \to R^2$ jako $T\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x + z \\ y + z \end{pmatrix}$.

Jako konkrétní příklad: $T\begin{pmatrix}1\\0\\1\end{pmatrix}=\begin{pmatrix}2\\1\end{pmatrix}$ nebo $T\begin{pmatrix}3\\1\\5\end{pmatrix}=\begin{pmatrix}8\\6\end{pmatrix}$.

Pro zobrazení T platí:

$$T\begin{pmatrix}x_1+x_2\\y_1+y_2\\z_1+z_2\end{pmatrix}=\begin{pmatrix}x_1+x_2+z_1+z_2\\y_1+y_2+z_1+z_2\end{pmatrix}=\begin{pmatrix}x_1+z_1\\y_1+z_1\end{pmatrix}+\begin{pmatrix}x_2+z_2\\y_2+z_2\end{pmatrix}=T\begin{pmatrix}x_1\\y_1\\z_1\end{pmatrix}+T\begin{pmatrix}x_2\\y_2\\z_2\end{pmatrix}$$

$$T\left(a\begin{pmatrix}x\\y\\z\end{pmatrix}\right)=T\begin{pmatrix}ax\\ay\\az\end{pmatrix}=\begin{pmatrix}ax+az\\ay+az\end{pmatrix}=\begin{pmatrix}a(x+z)\\a(y+z)\end{pmatrix}=a\begin{pmatrix}x+z\\y+z\end{pmatrix}=aT\begin{pmatrix}x\\y\\z\end{pmatrix}$$

5.2 Matematika v ukázce

Matematický základ funguje na principu algoritmu RSA(Rivest-Shamir-Adleman). Tento asymetrický algoritmus je založený na matematických vlastnostech modulární aritmetiky, teorii čísel a prvočísel, přičemž hlavní „zbraní“ je složitost faktorizace velkých čísel, který tvoří bezpečnost algoritmu RSA.

Prvočíslo je přirozené číslo, které je větší než 1 a je dělitelné beze zbytku pouze samo sebou a číslem 1. Mezi nejmenší používaná prvočísla od 0 do 20, patří 2 ,3 ,5 ,7 ,11 ,13 ,17 ,19, nicméně jich existuje nekonečně mnoho. Využívají je algoritmy, prvočíselné faktorizace a také kryptologie. Existují různé způsoby, kterými můžeme získat prvočísla, v mém programu je použití testu prvočíselnosti, kde využívám faktu, že pokud číslo není prvočíslo, bude mít dělitele menší nebo rovny jeho odmocnině.

Mezi další způsoby hledání prvočísel patří Eratosthenovo síto, což je jednoduchý algorimus, který funguje na prosívání ze seznamu čísel. Vytvoříme si seznam čísel od 2 do požadovaného limitu, ze kterého budeme vyškrtávat násobky čísel. Jako jednoduchý příklad si můžeme vzít seznam čísel od 2 do 6, tedy čísla 2,3,4,5,6. Při prvním projití seznamem zapíšeme 2 jako prvočíslo a vyškrtneme 4 a 6. Nově vytvořený seznam vypadá takto: 3,5 a seznam prvočísel: 2. Po následovných dvou provedeních máme seznam prvočísel: 2,3,5.

Formální definice prvočísel: číslo $n \in N$ je prvočíslem, pokud platí:

$$n > 1 \wedge \forall \mathrm{k} \in N\colon (k|n \rightarrow (k = 1 \vee \mathrm{k} = \mathrm{n}))$$

Euklidův algoritmus se používá pro určení největšího společného dělitele u dvou přirozených čísel (takové číslo, které beze zbytku dělí obě čísla). Algoritmus probíhá tak, že máme dvě čísla $a, b(a \geq b)$ a dokud b není rovno nule opakujeme vypočítání zbytku z, kde

$z = a\%b$, přiřadíme hodnotu a do b, b do z. Po skončení cyklu bude největší společný dělit v proměnné a. Konkrétní příklad: máme čísla $a = 48, b = 18$. Po první iteraci máme hodnoty: $48\%18 = 12 = z$, $a = 18, b = 12$. Po druhé iteraci: $18\%12 = 6 = z, a = 12, b = 6$. Po třetí iteraci: $12\%6 = 0 = z, a = 6, b = 0$, v momentě, kdy je $b = 0$ končí cyklus a hodnoto NSD je v a, tedy 6.

Rozšířený Euklidův algoritmus, kromě vyhledání NSD, je schopen vypočítat koeficienty vyhovující Bézeutově identitě, ta říká, že pro dvě celá čísla a, b existují celá čísla x, y, pro která platí: $ax + by = NSD(a, b)$. Příklad je uveden v kódu v kapitole 6.3.5.

5.2.1 Šifrování a dešifrování

Šifrování probíhá pomocí vzorce $c = m^e \ mod \ n$, kde c je šifrovaný text, m je zpráva, e, n jsou hodnoty z veřejného klíče.

Dešifrování probíhá pomocí vzorce $m = c^d \ mod \ n$, kde c je šifrovaný text, m je zpráva, d, n jsou hodnoty z privátního klíče. Získání tohoto výpočtu z původní zprávy je možné díky těmto rovnostem: $c^d = (m^e)^d \ = m \quad (mod \ n)$

a protože $e \times d = 1 \quad (mod \ p - 1)$ a $e \times d = 1 \quad (mod \ q - 1)$, díky malé Fermatově platí, že: $(m^e)^d = m^{1+e(p-1)} = m^1(m^{p-1})^c = m \quad (mod \ p)$

a zároveň $(m^e)^d = m \quad (mod \ q)$, čínská věta o zbytcích udává, že pokud jsou p a q různá čísla, tak platí: $(m^e)^d = m \quad (mod \ pq)$ z toho vyplývá, že $c^d = m \quad (mod \ n)$

5.2.2 Matematika za homomorfním RSA

Pokud by se jednalo pouze homomorfní násobení vystačili bychom si s rovnicí:

$\text{Encrypt}(m_1) \blacksquare \text{Encrypt}(m_2) = \text{Encrypt}(m_1 * m_2)$, kde dešifrování na obou stranách dává součin otevřených textů.

V tomto případě se jedná o homomorfní šifrování s násobením za použití RSA.

Pokud máme dva otevřené texty m_1, m_2 a dva zašifrované texty $c_1 = Encrypt(m_1)$ a $c_2 = Encrypt(m_2)$ můžeme provést násobení

$$\text{Encrypt}(m_1) \blacksquare \text{Encrypt}(m_2) = (m_1^e \times \text{m}_2^e) \bmod \text{n} = (\text{m}_1 \times \text{m}_2)^e \bmod \text{n}$$
$$= \text{Encrypt}(m_1 * \text{m}_2)$$

Což je ověřeno ve výstupu ukázkového programu v tabulce 1 (v kapitole 6.3.11).

6 NÁSTROJE

Pro ukázku homomorfního šifrování s RSA jsem si vybral programovací jazyk Python a vývojové prostředí Visual Studio Code. Tyto nástroje jsem zvolil, protože jsem s nimi už obeznámen a práce v nich mi připadá přehledná a jednoduchá.

6.1 Python

Python je velmi populární a je často doporučován jako dobrá volba pro začátečníky i odborníky. Hlavní důvody pro programování v Pythonu jsou:

- Jednoduchá syntaxe: Python má čistou a čitelnou syntaxi, která je snadno pochopitelná. Je navržen tak, aby byl stejně čitelný jako angličtina, což usnadňuje psaní a porozumění kódu. To je obrovská výhoda zejména pro začátečníky.
- Velká komunita a dostupnost knihoven: Python má velkou a aktivní komunitu programátorů. Existuje mnoho dostupných knihoven a nástrojů s otevřeným zdrojovým kódem, které mohou výrazně zjednodušit vývoj aplikací v Pythonu. Například oblíbené knihovny jako NumPy, Pandas, Matplotlib a TensorFlow poskytují bohaté funkce pro analýzu dat, strojové učení, vývoj webu a další oblasti.
- Všestrannost: Python je všestranný programovací jazyk, který lze použít v mnoha oblastech. Vhodné pro vývoj webových aplikací, vývoj desktopových aplikací, vědecký výzkum, analýzu dat, automatizaci atd. Tato všestrannost vám umožňuje vytvářet různé typy projektů v Pythonu a snadno přepínat mezi různými vývojovými doménami. Lepší čitelnost a udržovatelnost kódu: Jednoduchá syntaxe a zaměření na čitelnost činí kód napsaný v Pythonu snazším čtením a porozuměním. To je důležité jak pro vás, tak pro ostatní programátory pracující s vaším kódem. Svůj projekt můžete snadno udržovat a rozšiřovat.
- Podpora napříč platformami: Python je k dispozici na většině hlavních operačních systémů včetně Windows, macOS a Linux. Jinými slovy, napsat kód Pythonu na jednom operačním systému a spustit jej na jiném není tak obtížné. Vysoká produktivita: Python má mnoho nástrojů a knihoven, které můžete použít k úspoře času při vývoji.

6.2 Visual Studio Code

Visual Studio Code (VS Code) je oblíbené integrované vývojové prostředí (IDE) vyvinuté společností Microsoft. Je to multiplatformní nástroj dostupný pro Windows, MacOS a Linux. Existuje mnoho důvodů, proč programovat ve VS Code.

- Rozšiřitelnost: VS Code je extrémně rozšiřitelný. Podporuje širokou škálu rozšíření a doplňků, které lze nainstalovat prostřednictvím tržiště rozšíření. Přizpůsobte si vývojové prostředí a přidejte funkce, které vyhovují vašim potřebám. Existuje obrovská komunita vývojářů, kteří přispívají k vytváření nových rozšíření a aktualizaci stávajících rozšíření.
- Podpora více jazyků a frameworků: VS Code nabízí podporu pro mnoho programovacích jazyků a frameworků, včetně Pythonu, JavaScriptu, C, Java, PHP a dalších. To vám umožňuje používat jedno IDE pro různé projekty bez přepínání mezi různými vývojovými prostředími.
- Vestavěná správa verzí: VS Code nabízí vestavěnou podporu pro správu verzí pomocí Git. Přímo ve VS Code můžete snadno sledovat změny svého kódu, provádět commity, vytvářet a slučovat větve a spolupracovat s ostatními členy týmu. Výkonný editor kódu: VS Code nabízí pokročilé funkce úpravy kódu, jako je automatické dokončování kódu, refaktorování, rychlá navigace v kódu, odsazení, zvýraznění syntaxe, chmýří a další. Tyto funkce vám pomohou být produktivnější při psaní a úpravách kódu.
- Ladění a odladění: VS Code poskytuje možnost ladit vaši aplikaci. Můžete nastavit body ladění, sledovat hodnoty proměnných, procházet kódem a analyzovat chyby a výjimky. To usnadňuje nalezení a opravu problémů v kódu.
- Komunita a dokumentace: Visual Studio Code má velkou a aktivní komunitu vývojářů. Existuje mnoho zdrojů, jako jsou diskusní fóra, blogy, výukové programy a videolekce, které vám pomohou získat znalosti a sdílet své zkušenosti s ostatními programátory.

6.3 Struktura programu

6.3.1 Použité knihovny

Pomocí slova import můžeme využít některé funkce z předem připravených knihoven. Ve své ukázce používám dvě knihovny, a to jsou math a random.

Knihovna math je importována pro použití matematických funkcí, jako je sqrt (druhá odmocnina) v rámci funkce is_prime. Tato funkce testuje, zda je číslo prvočíslo, a ve svém algoritmu používá druhou odmocninu.

Knihovna random je importována pro generování náhodných čísel v mém programu. Používá se ke generování náhodných prvočísel (p,q) při generování klíčů RSA, ale používá se také v rámci funkce generation_keypair ke generování náhodného šifrovacího klíče e.

6.3.2 Generování náhodných prvočísel

Tato funkce funguje pomocí smyčky while ve které se generuje náhodné číslo v rozsahu 32768 až 65636 pomocí knihovny random. Dále se ověří, jestli je číslo prvočíslem pomocí funkce is_prime.

```
# Funkce pro generování náhodných prvočísel
def generate_prime():
    while True:
          num = random.randint(2**15, 2**16)   # Vygenerujeme náhodné
číslo mezi 2^15 a 2^16
        if is_prime(num):
            return num
```

6.3.3 Ověření prvočísla

Funkce vrací False pokud je číslo num menší než 2. V cyklu for, který iteruje od 2 odmocniny čísla num až do zadaného čísla num, se testuje dělitelnost daného čísla, a pokud v daném rozsahu nebylo možné vydělit beze zbytku vrací True (je prvnočíslo) v opačném případě vrátí False.

```
# Funkce pro ověření, zda je číslo prvočíslo
def is_prime(num):
    if num < 2:
        return False
    for i in range(2, int(math.sqrt(num)) + 1):
        if num % i == 0:
            return False
    return True
```

6.3.4 Výpočet největšího společného dělitele

Spočívá v opakovaném odečítání menšího čísla od většího, dokud si obě čísla nejsou rovny. Podrobněji je vysvětleno v kapitole 5.2.

```
# Funkce pro výpočet největšího společného dělitele (NSD) pomocí Euklidova
algoritmu
def gcd(a, b):
    while b != 0:
        a, b = b, a % b
    return a
```

6.3.5 Výpočet inverzního prvku

Funkce má dva vstupní parametry: a (číslo u kterého hledáme inverzní prvek), m (které je modulo). Nejprve zkontrolujeme, že se největší společný dělitel nerovná 1 (inverzní prvek neexistuje). Poté si vytvoříme a inicializujeme proměnné podle ukázky. V cyklu while se vypočítá podíl (quotient) pomocí operace celočíselného dělení, podle kterého se aktualizují hodnoty proměnných ve vzorcích, cyklus probíhá, dokud proměnná v3 není rovna 0. Po skončení cyklu se vrátí hodnota $u\%m$, což je hodnota inverzního prvku m.

```
# Funkce pro výpočet inverzního prvku pomocí rozšířeného Euklidova algoritmu
def mod_inverse(a, m):
    if gcd(a, m) != 1:
        return None  # Inverzní prvek neexistuje
    u1, u2, u3 = 1, 0, a
    v1, v2, v3 = 0, 1, m
    while v3 != 0:
        quotient = u3 // v3
       v1, v2, v3, u1, u2, u3 = (u1 - quotient * v1), (u2 - quotient
* v2), (u3 - quotient * v3), v1, v2, v3
    return u1 % m
```

6.3.6 Zašifrování zprávy

Zpráva se zašifruje pomocí vzorce $c = m^e \ mod \ n$, kde c je šifrovaný text, m je zpráva, e, n jsou hodnoty z veřejného klíče.

```
# Funkce pro šifrování zprávy
def encrypt(message, public_key):
    n, e = public_key
    ciphertext = pow(message, e, n)
    return ciphertext
```

6.3.7 Dešifrování zprávy

Zpráva se dešifruje pomocí vzorce $m = c^d \ mod \ n$, kde c je šifrovaný text, m je zpráva, d, n jsou hodnoty z privátního klíče.

```
# Funkce pro dešifrování zprávy
def decrypt(ciphertext, private_key):
    n, d = private_key
    message = pow(ciphertext, d, n)
    return message
```

6.3.8 Generování RSA klíčů

V této funkci si necháme vygenerovat potřebná prvočísla, ze kterých vytvoříme privátní a veřejný klíč.

```
# Generování RSA klíčů
def generate_keypair():
    p = generate_prime()
    q = generate_prime()
    n = p * q
    phi = (p - 1) * (q - 1)
    while True:
        e = random.randint(2, phi - 1)
        if gcd(e, phi) == 1:
            break
    d = mod_inverse(e, phi)
    public_key = (n, e)
    private_key = (n, d)
    return public_key, private_key
```

6.3.9 Homomorfní násobení

Bude vysvětleno v kapitole 5.2.2

```
# Funkce pro homomorfní násobení šifrovaných textů
def homomorphic_multiply(ciphertext_a, ciphertext_b, public_key):
    n, e = public_key
    result = (ciphertext_a * ciphertext_b) % n
    return result
```

6.3.10 Volání funkcí a generace výpisů pro ověření správnosti programu

V následující části kódu si necháme vygenerovat klíče pomocí funkce generate_pair(), následně deklarujeme a inicializujeme proměnné openTextA a openTextB. Získáme zašifrované hodnoty a provedeme homomorfní násobení. Následně provedeme výpisy.

```
# Volání funkcí
public_key, private_key = generate_keypair()

openTextA = 42
openTextB = 31
# Šifrování původního otevřeného textu
ciphertext_a = encrypt(openTextA, public_key)
ciphertext_b = encrypt(openTextB, public_key)

# Násobení šifrových textů
ciphertext_result       =       homomorphic_multiply(ciphertext_a,
```

```
ciphertext_b, public_key)

# Dešifrování výsledku
result_message = decrypt(ciphertext_result, private_key)

# Výpis výsledků
print("Zašifrovaná zpráva A:", ciphertext_a)
print("Zašifrovaná zpráva B:", ciphertext_b)
print("Původní zpráva A:", decrypt(ciphertext_a,private_key))
print("Původní zpráva B:", decrypt(ciphertext_b, private_key))
print("výsledek homomorfního šifrování:", ciphertext_result)
print("Výsledek po dešifrování:", result_message)
print("Vynásobení A*B:", openTextA*openTextB)
```

6.3.11 Ukázka proměnných při spuštění

V tabulce 1 je ukázán konkrétní příklad jednoho průběhu programu, kde jsem ověřil pravdivost rovnice zmíněné v matematice za homomorfním RSA. Alice chce po Bobovi vynásobit dvě čísla 42 a 31, pomocí veřejného klíče (1878322133, 1269137861) data zašifruje na 1525317598, 821237017. Následně zašifrovaná data pošle Bobovi, který je provede operaci modulárního násobení a získá hodnotu 718762811. Tuto hodnotu pošle zpět Alici, která si data privátním klíčem (1878322133, 1421802509) rozšifruje. Alice dostane výslednou hodnotu 1302, to je 42 * 31. Alice získala výsledek, aniž by Bob věděl původní čísla.

Tabulka 1: Konkrétní příklad z RSA

Názvy proměnných:	**Hodnoty:**
Otevřený text A	42
Otevřený text B	31
p	39157
q	47969
n	1878322133
phi	1878235008
e	1269137861
d	1421802509
Public_key	(1878322133, 1269137861)
Private_key	(1878322133, 1421802509)
Zašifrovaný text A	1525317598
Zašifrovaný text B	821237017
Zašifrovaný výsledek homomorfního násobení	718762811
Dešifrovaný text A	42
Dešifrovaný text B	31
Dešifrovaný výsledek násobení	1302
Vynásobení A * B	1302

ZÁVĚR

Cílem bakalářské práce bylo nastudovat současný stav homomorfního šifrování a následně ho představit všem čtenářům. Pokusil jsem se téma zpracovat zjednodušeněji, tak aby si jej mohli číst i lidé z jiného oboru než je matematika. Uvedl jsem příklad využití z reálného života, kde by bylo možné homomorfní šifrování využít. Poté jsem se věnoval klasické a moderní kryptologie, kterou jsem vysvětlil a uvedl příklady.

Uvedl jsem rozdělení homomorfního šifrování, jeho současný stav. Zpracoval jsem druhy kvantového šifrování a jejich protokoly.

Dále jsem představil knihovny nabízející homomorfní šifrování v jazyku Python a C++, které značně zjednodušují aplikaci homomorfního šifrování.

Nakonec jsem představil matematický základ homomorfismu s několika příklady a uvedl matematiku potřebnou pro provedení programu z ukázky – částečné homomorfní šifrování s RSA. Jednotlivé složky kódu jsou důkladně popsány.

SEZNAM POUŽITÉ LITERATURY

[1] DZURENDA, Petr a Jan HAJNÝ. *Techniky homomorfního šifrování a jejich praktické využití. Elektrorevue* [online]. Brno: Fakulta elektrotechniky a komunikačních technologií VUT v Brně, 2014, 20.04.2014, (2), 53-60 [cit. 2023-05-25]. ISSN ISSN 1213-1539. Dostupné z: http://www.elektrorevue.cz/cz/clanky/informacni-technologie/20/techniky-homomorfniho-sifrovani-a-jejich-prakticke-vyuziti--techniques-of-homomorphic-encryption-and-their-practical-usage-/

[2] MIKEAZO. Homomorphic encryption based on XOR. *Crypto.stackexchange.com* [online]. 30. 9 2014 [cit. 2023-05-25]. Dostupné z: https://crypto.stackexchange.com/questions/19391/homomorphic-encryption-based-on-xor

[3] GOOGLE. What is Cloud Computing?. *Cloud.google.com* [online]. 2020 [cit. 2023-05-25]. Dostupné z: https://cloud.google.com/learn/what-is-cloud-computing

[4] HEYELLIOTT. Secure multi-party computation. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 4. 4 2023 [cit. 2023-05-25]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Secure_multi-party_computation&action=history

[5] SLOUKA, David. Svatý grál šifrování: Intel a DARPA na výpravě za neprolomitelným šifrováním. *Www.computerworld.cz* [online]. 9. 3 2021 [cit. 2023-05-25]. Dostupné z: https://www.computerworld.cz/clanky/svaty-gral-sifrovani-intel-a-darpa-na-vyprave-za-neprolomitelnym-sifrovanim/

[6] JELIČ, Pavel. Co je to end-to-end šifrování. *Www.letemsvetemapplem.eu* [online]. 23. 2 2020 [cit. 2023-05-25]. Dostupné z: https://www.letemsvetemapplem.eu/2020/02/23/co-je-to-end-to-end-sifrovani/

[7] BURDA, Karel. *Kryptografie okolo nás*. 1. internet: CZ.NIC, 2017. ISBN 978-80-88168-49-9.

[8] SINGH, Simon, Dita ECKHARDOVÁ a Petr KOUBSKÝ. *Kniha kódů a šifer*. 2. Praha: Dokořán, 2009. ISBN 9788073632687.

[9] YANG, Su, Bailong YANG a Chen YANG. FPGA-Based Hardware Accelerator for Leveled Ring-LWE Fully Homomorphic Encryption. *Www.researchgate.net* [online]. 2020, 24. 9 2020 [cit. 2023-05-25]. Dostupné z: https://www.researchgate.net/publication/346288254_FPGA-Based_Hardware_Accelerator_for_Leveled_Ring-LWE_Fully_Homomorphic_Encryption

[10] JOYE, Marc. *Homomorphic Encryption 101* [online]. internet, 2021 [cit. 2023-05-25]. Dostupné z: https://www.zama.ai/post/homomorphic-encryption-101

[11] IBARRONDO, Alberto a Alexander VIAND. Pyfhel. *Github.com* [online]. 2022, 19.4.2022 [cit. 2023-05-25]. Dostupné z: https://github.com/ibarrond/Pyfhel

[12] OPENMINDED. TenSEAL. *Github.com* [online]. 20.8.2021 [cit. 2023-05-25]. Dostupné z: https://github.com/OpenMined/TenSEAL

[13] LAB41. PySEAL. *Github.com* [online]. 21.12.2017 [cit. 2023-05-25]. Dostupné z: https://github.com/Lab41/PySEAL

[14] BERGAMASCHI, Flavio. HElib. *Github.com* [online]. 21.8.2020 [cit. 2023-05-25]. Dostupné z: https://github.com/homenc/HElib

[15] MICROSOFT. SEAL. *Github.com* [online]. 4.4.2021 [cit. 2023-05-25]. Dostupné z: https://github.com/microsoft/SEAL

[16] BERGAMASCHI, Flavio. HElib. *Github.com* [online]. 21.8.2020 [cit. 2023-05-25]. Dostupné z: https://github.com/homenc/HElib

[17] CHILLOTTI, Ilaria, Nicolas GAMA, Mariya GEORGIEVA a Malika IZABACH. TFHE: Fast Fully Homomorphic Encryption Library. *Tfhe.github.io* [online]. 2016 [cit. 2023-05-25]. Dostupné z: https://tfhe.github.io/tfhe/

[18] HOUSEBLASTER. HEAAN. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2023-05-26]. Dostupné z: https://en.wikipedia.org/wiki/HEAAN

[19] Eukleidův algoritmus. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2023-05-26]. Dostupné z: https://cs.wikipedia.org/wiki/Eukleid%C5%AFv_algoritmus

[20] HOFFSTEIN, Jeffrey. *An introduction to mathematical cryptography*. 1. New York: Springer, 2008.ISBN 978-0-387-77993-5.

[21] Rozšířený Eukleidův algoritmus. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-

[cit. 2023-05-26]. Dostupné z: https://cs.wikipedia.org/wiki/Roz%C5%A1%C3%AD%C5%99en%C3%BD_Eukleid%C5%AFv_algoritmus

[22] *Lineární algebra*. 2. Olomouc: Univerzita Palackého v Olomouci, 2010. ISBN 987-80-244-2522-1.

[23] ILMARI, Karonen. Homomorphic cryptosystems in RSA. *Crypto.stackexchange.com* [online]. 18. 3 2017 [cit. 2023-05-26]. Dostupné z: https://crypto.stackexchange.com/questions/3555/homomorphic-cryptosystems-in-rsa

[24] STANOVSKÝ, David. *Základy algebry*. 1. Praha: Matfyzpress, 2010. ISBN 978-80-7378-105-7.

[25] STANOVSKÝ, David a Libor BARTO. *Počítačová algebra*. 2. Praha: Matfyzpress, 2017. ISBN 978-80-7378-340-2.

[26] ČÍŽEK, Jakub. Homomorfní kryptografie: Málokdo ji rozumí, ale může vyřešit naši touhu po absolutním soukromí na internetu. *Www.zive.cz* [online]. 2021, 7. 1 2021 [cit. 2023-05-26]. Dostupné z: https://www.zive.cz/clanky/homomorfni-kryptografie-malokdo-ji-rozumi-ale-muze-vyresit-nasi-touhu-po-absolutnim-soukromi-na-internetu/sc-3-a-207782/default.aspx

[27] Homomorfismus. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2023-05-26]. Dostupné z: https://cs.wikipedia.org/wiki/Homomorfismus

[28] ROCCA, Charles. Isomorphisms and Homomorphisms of Vector Spaces. *OER - Western CT Linear Algebra* [online]. [cit. 2023-05-26]. Dostupné z: https://sites.wcsu.edu/mbxml/OER_Linear_Alg/section_morphisams.html

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

RSA	Rivest, Shamor, Adleman
XOR	eXkluzivní OR
AP3KR	Zkratka předmětu kryptologie
DES	Data Encryption Standart
AES	Advanced Encryption Standart
WEP	Wired Equivalent Privacy
VPN	Virtual Private Network
FHE	Fully Homomorphic Encryption
BGV	Brakerski-Gentry-Vaikuntanathan
LWE	Learning with Errors
PHE	Partially Homomorphic Encryption
SWHE	SomeWhat Homomorphic Encryption
FHEW	Fully Homomorphic Encryption over the Wire
CKKS	Cheon-Kim-Kim-Song
FHE-MC	Fully Homomorphic Encryption with Matrix Compression
NTRU	N-th Degree Truncated Polynomial Ring Units
RLWE	Ring Learning with Errors
HSS	Homomorphic Secret Sharing
BFV	Brakerski-Fan-Vercauteren
TFHE	Fully Homomorphic Encryption with Torus
HEAAN	Homomorphic Encryption of Approximate Number Operations
NSD	Největší společný dělitel

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: NÁZEV PŘÍLOHY

https://github.com/jonakk2/Homomorpic-en-RSA