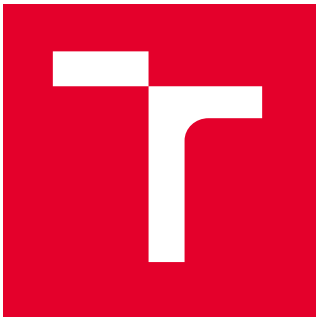
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2024 Michal Pavlíček



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

ŠIFROVÁNÍ V DATABÁZOVÉM SYSTÉMU POSTGRESQL

ENCRYPTION IN THE POSTGRESQL DATABASE SYSTEM

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — **Michal Pavlíček**

AUTHOR

VEDOUCÍ PRÁCE — **RNDr. Ing. Pavel Šeda, Ph.D.**

SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Michal Pavlíček ***ID:*** 240965

Ročník: 3 ***Akademický rok:*** 2023/24

NÁZEV TÉMATU:

Šifrování v databázovém systému PostgreSQL

POKYNY PRO VYPRACOVÁNÍ:

Hlavním cílem bakalářské práce je implementace nástroje automatizující nastavení šifrování (Transparent data encryption – TDE) v databázovém systému PostgreSQL. V teoretické části práce analyzujte možnosti šifrování v databázových systémech (DLE, FLE, FDE, TDE, ALE) a zhodnoťte vhodnost jejich použití a jejich dopady na bezpečnost databázových systémů a dat v nich uložených. Hlavní výstupem práce bude implementace a porovnání softwarových nástrojů automatizující nastavení TDE v databázovém systému PostgreSQL, včetně návodu všech nezbytných nastavení. Funkčnost řešení důkladně otestujte v experimentálním prostředí a porovnejte se zamýšleným řešením šifrování PostgreSQL na úrovni disku.

DOPORUČENÁ LITERATURA:

[1] GARCIA-MOLINA, Hector, Jeffrey D. ULLMAN a Jennifer WIDOM. Database system implementation. Upper Saddle River: Prentice Hall, 2000. xv, 653 s. ISBN 0-13-040264-8. (EN)

[2] Obe, Regina O., and Leo S. Hsu. PostgreSQL: up and running: a practical guide to the advanced open source database. " O'Reilly Media, Inc.", 2017.

Termín zadání: 5.2.2024 ***Termín odevzdání:*** 28.5.2024

Vedoucí práce: RNDr. Ing. Pavel Šeda, Ph.D.

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

ABSTRAKT

Bakalářská práce se zabývá problematikou šifrování v databázovém systému PostgreSQL. Teoretická část práce detailně vysvětluje základní pojmy spojené s databázemi a šifrováním, dat mimo využití a dat přenášených po síti. Popisuje frekventovaně využívaných šifrovacích algoritmů, je vysvětlena problematika typů klíčů a jejich managementu. Analyzovány jsou bezpečnostní funkce pěti hlavních databázových systémů. Praktická část práce zahrnuje implementaci tří nástrojů pro nastavení transparentního šifrování v PostgreSQL a jejich popis. Bylo zjištěno, že PostgreSQL neposkytuje možnost transparentního šifrování na úrovni databáze, ale lze ji šifrovat na úrovni souborového systému nebo pomocí modulu pgcrypto na úrovni sloupců v relační tabulce. Měření výkonu ukázalo, že transparentní šifrování na úrovni databáze dosahuje v průměru lepšího výkonu i zabezpečení než šifrování souborového systému, ačkoli jednotné nastavení konfigurace může ovlivnit výkon šifrování. Práce přináší ucelený pohled na problematiku šifrování v PostgreSQL a poskytuje relevantní informace pro rozhodování v oblasti databázové bezpečnosti.

KLÍČOVÁ SLOVA

PostgreSQL, šifrování, TDE, databáze

ABSTRACT

The bachelor thesis deals with the issue of encryption in database The theoretical part of the thesis explains in detail the basic concepts related to databases and encryption, data out of use and data transmitted over the network. It describes commonly used encryption algorithms, explains the issues of key types and their management. The security features of five major database systems are analyzed. The practical part of the work includes the implementation of three tools for setting up transparent encryption in PostgreSQL and their description. It was found that PostgreSQL does not provide the possibility of transparent encryption at the database level, but it can be encrypted at the file system level or by using the pgcrypto module at the column level in a relational table. Performance measurements have shown that transparent encryption at the database level achieves better performance and security on average than file system encryption, although uniform configuration settings can affect encryption performance. This paper presents a comprehensive view of encryption in PostgreSQL and provides relevant information for database security decision making.

KEYWORDS

PostgreSQL, encryption, TDE, database

Vysázeno pomocí balíčku `thesis` verze 4.09; `http://latex.feec.vutbr.cz`

PAVLÍČEK, Michal. *Šifrování v databázovém systému PostgreSQL*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2024, 60 s. Bakalářská práce. Vedoucí práce: RNDr. Ing. Pavel Šeda, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Michal Pavlíček
VUT ID autora:	240965
Typ práce:	Bakalářská práce
Akademický rok:	2023/24
Téma závěrečné práce:	Šifrování v databázovém systému PostgreSQL

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 bakalářské práce panu RNDr. Ing. Pavlu Šedovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

Tato práce se zabývá zabezpečením a šifrováním databází. V dnešní době, kdy se exponenciálně zvyšuje počet dat na internetu, přibývá i počet kybernetických útoků, tudíž otázka zabezpečení se stává důležitější. Šifrování dat je jedním z klíčových prvků zabezpečení moderních informačních systémů. V rámci databázových systémů, které často slouží jako úložiště citlivých informací, je proto důležité implementovat efektivní mechanismy zabezpečení.

V kontextu databázových systémů nabývá šifrování stále většího významu, tudíž by měly zabezpečené systémy být schopny zvládat svou každodenní zátěž i s šifrováním. Tradiční přístupy k šifrování však často znamenají zásah do běžného provozu aplikací a náročné správy klíčů. Jedním z nejpopulárnějších open-source databázových systémů je PostgreSQL, který se vyznačuje robustností, flexibilitou a širokou komunitou uživatelů [1].

Cílem této bakalářské práce je seznámit čtenáře se základními principy šifrování se zaměřením na šifrování v databázích, jež se v automatizované podobě nazývá transparentní šifrování. Transparentní šifrování umožňuje šifrování dat mimo využití, což zajišťuje jejich bezpečnost při neautorizovaném vstupu na úložiště, aniž by bylo nutné provádět změny na straně aplikací či klientů.

Teoretická část práce bude zaměřena zaměřena na databáze, jazyk pro komunikaci s databází, základy šifrování jako typy kryptografických klíčů, zacházení s nimi a budou popsány čtyři používané šifrovací algoritmy. Dále bude shrnut přehled vlastností a bezpečnostních funkcí pěti nejpopulárnějších databázových systémů dle DB-Engines Ranking [1]. Poslední kapitola teoretické části bude analyzovat typy šifrování, popíše jejich způsob šifrování dat a zaměří se i na vhodnost použití.

Praktická část se pokusí vysvětlit problém transparentního šifrování v databázi PostgreSQL a pokusí se nabídnout možná řešení problému ve spojitosti s aktuální situací. Nadále v praktické části budou prezentovány tři nástroje pro transparentní šifrování v PostgreSQLs, včetně návrhu instalace řešení společnosti Cybertech [2].

Neopomenutelnou součástí praktické části této bakalářské práce bude také testování funkčnosti navržených řešení, testování proběhne pomocí nástroje na zatěžovací testy pgbench. Měření bude porovnávat výkon čtyř šifrovaných databází s jednou nezašifrovanou databází, měření proběhne ve třech rozličných velikostech databází a při rozdílných druzích testů na čtení, zápis či obojí. V závěru budou výsledky měření zhodnoceny a porovnány.

1 Teoretický rámec

Bakalářská práce se zabývá PostgreSQL databází, možnostmi šifrování se zaměřením na transparentní šifrování databáze a šifrováním v databázových systémech. Teoretická část práce popisuje základní pojmy spojené s databází, úvod do šifrování, analýzu databázových systémů a uvedení jejich bezpečnostních funkcí.

1.1 Databáze

Databáze je definována jako: „Uspořádaný soubor strukturovaných informací nebo dat, obvykle uložených elektronicky v počítačovém systému“ [3]. Řízení databáze probíhá přes databázový software nebo systémem pro správu databáze (DBMS). Data jsou většinou strukturována do sloupců a řádků v tabulkách. Strukturovaný dotazovací jazyk se stará o veškeré zápisy, upravování, organizování a dotazování dat. Při definování databází je důležité rozlišovat, že Spreadsheet jako například Microsoft Excel není považován za databázi z důvodu rozdílu v uchování, manipulaci a zobrazení dat. Existují i nerelační databáze, které pracují hlavně s nestrukturovanými daty, což znamená, že data nemusí být definována pomocí relací před vložením [3, 4].

Databázový software je soubor programů, který usnadňuje správu databáze. Například telefonní seznam je příklad databáze bez databázového software. Software řídí více přístupů, zálohování, reportování a bezpečnost databáze. Slouží ke správě databázových záznamů pomocí vytváření, aktualizací a mazání. Grafické rozhraní pomáhá uživateli, navíc funkce zjednodušují správu databáze, zejména díky lehčí orientaci s daty. Rozhraní mezi uživatelem a databází může být taktéž označováno jako DBMS. Funkce DBMS umožňují sledovat výkon, dohlížet na chod, ladit, popřípadě obnovovat databázi [3].

Databáze si od svého vzniku prošly mnoha změnami v logice ukládání dat, pokud bychom se podívali na vývoj změn, tak bychom viděli následující změny. První databáze v 60. letech se podobaly spíše homogenním datovým strukturám. Hierarchické databáze napodobovaly stromovou strukturu, zatímco síťové databáze připomínaly grafovou strukturu. V 80. letech popularita relačních databází vzrostla. Během dalších deset let vznikaly objektově orientované databáze. Nerelační databáze neboli NoSQL nabraly na popularitě až kolem roku 2009 s rozvojem internetu a s ním souvisejícími spoustami nestrukturovaných dat [3].

1.2 Jazyk SQL

Structured query language (SQL) je již dlouho nedílnou součástí databází a tudíž i součástí světa informatiky. Jedná se o jazyk pro komunikaci a manipulaci s databází. Předchůdce novodobého jazyku SQL, jazyk „Structured English Query Language“ (SEQUEL), vznikl v 70. letech výzkumníky firmy IBM Raymondem Boyzcem a Donaldem Chamberlinem, výzkumníci se inspirovali článkem „A Relational Model of Data for Large Shared Data Banks“. SEQUEL měl být intuitivní a jednoduchý pro manipulaci s IBM relačním systémem pro správu databáze System R. Později byl SEQUEL přejmenován na SQL. Až v roce 1979 byl jazyk zpřístupněn veřejnosti firmou Relational Software, později Oracle, v podobě verze jazyka SQL s názvem Oracle V2. V blízkých letech začaly firmy jako Sybase nebo Microsoft prodávat své návrhy RDBMS, později se našly komunity rozvíjející projekty jako MySQL a PostgreSQL v podobně otevřeného zdrojového kódu, které jsou k dispozici i pod komerčními licencemi [5, 6].

Dotazy jazyku SQL se dělí do pěti hlavních kategorií [7].

Data Definition Language (DDL)
Jazyk pro definici dat strukturuje tabulky, indexy nebo pohledy. Jsou definovány pomocí příkazů na tvoření, mazání a změny. Představuje příkazy `CREATE`, `DROP`, `ALTER`, `TRUNCATE`.

Data Manipulation Language (DML)
Jazyk pro manipulaci s daty umožnuje upravovat data databáze pomocí vkládání, mazaní a úpravou. Upravuje veškeré změny modifikací dat v databázi. Zastupuje příkazy `INSERT`, `UPDATE`, `DELETE`.

Data Control Language (DCL)
Jazyk pro kontrolu dat umožňuje nastavovat práva a oprávnění uživatelů v prostoru databáze. Definuje příkaz na přidělení práv a odebrání práv. Představuje příkazy `GRANT`, `REVOKE`.

Transaction control language (TCL)
Jazyk pro řízení transakcí se zabývá transakcemi v databázi. Obsahuje příkaz pro uložení transakcí, vrácení neuložených transakcí a nastavení úložného bodu v transakci. Skládá se z příkazů `COMMIT`, `ROLLBACK` a `SAVEPOINT`.

Data Query Language (DQL)
Jazyk dotazování slouží k načítání dat z databáze. Je reprezentován pouze jedním příkazem `INSERT SELECT`, který načte data v struktuře tabulky. Klauzule jako `FROM`, `HAVING`, `WHERE`, `ORDER BY`, `GROUP BY`, `LIKE`, `AND` a `OR` pomáhají při dotazování se dat vybrat přesně chtěná data.

1.3 Šifrování

Šifrování je matematická transformace sloužící k zakódování citlivých dat vyžadujících ochranu důvěrnosti dat do nerozeznatelné podoby pro neoprávněné osoby. Šifrování disponuje i vratnou transformaci do původní čitelné podoby textu, avšak absence dešifrování udělá data nečitelná a nepoužitelná. Tento proces nazýváme jako dešifrování. Každá šifrovací a dešifrovací funkce vyžaduje kryptografický klíč, který se na své nejjednodušší úrovni skládá z řetězce binárních čísel [8].

1.3.1 Kryptografické klíče a jejich management

Asymetrické klíče

Kryptografie pomocí asymetrických klíčů generuje a rozlišuje dva klíče. Veřejný klíč není třeba uchovávat v tajnosti a mnohdy může být sdílen. Soukromý klíč je tvořen pouze pro autora a neměl by být šířen. Šifrování probíhá jedním z klíčů a funkce šifrovacího algoritmu, zatímco dešifrování probíhá pomocí druhého klíče a funkce dešifrovacího algoritmu, záleží zda se jedná o podpis. Asymetrické algoritmy jsou založené na matematických funkcích, nikoli na substitucích a permutacích, za což může vývoj v počítačových technologiích. Což lze použít jak pro důvěrnost, tak pro autentizaci [9, 10].

Symetrické klíče

Kryptografie pomocí symetrického klíče provádí šifrování a dešifrování pomocí stejného klíče. Jedná se o konvenční techniku, která bývala využívána před vývojem novodobé asymetrické kryptografie. Šifrovací funkce se může měnit při šifrování a dešifrování narozdíl od klíče. Výhodou symetrických klíčů je jejich rychlost transformace, avšak velikou nevýhodou je obtížnost distribuce klíče při komunikaci. Poslání klíče kanálem by mohlo být odposlechnuto a zneužito při následné komunikaci. Na symetrické klíče se dá útočit pomocí kryptoanalýzy, která je založena na posuzování vlastnosti šifry, nebo brute force útoku, který zkouší postupně všechny možnosti klíče [9, 10].

Správa klíčů

Správa klíčů udává, jak budou klíče generovány a jak s nimi bude zacházeno během jejich platnosti. Bezpečnost šifrovacích klíčů bude vždy tak silná, jako jejich správa a omezení přístupu k nim. Jedním ze snadných řešení při šifrování na úrovni databáze je ukládat klíče do souboru, který je šifrovaný hlavním (master) klíčem. Jenomže úskalím tohoto způsobu by bylo, že by se každý správce dostal ke všem klíčům a mohl

dešifrovat všechna data, proto tento problém řeší hardware security module (HSM). V HSM jsou uloženy veškeré klíče, kdy při požadavku jsou dešifrovány hlavním klíčem a po dešifraci potřebných dat odstraněny i z paměti serveru. Dalším řešením je použití vnějšího bezpečnostního serveru, který spravuje veškeré klíče, uživatele, role a oprávnění. Při úspěšném ověření jsou poskytnuty příslušné klíče. Použitím zmíněných technik můžeme zvýšit bezpečnost v databázi, avšak klíče se stále krátce objevují v paměti databázového serveru [11].

1.3.2 Algoritmy šifrování

Funkčních algoritmů pro šifrování dat existuje celá řada, každý však každý nabízí různou úroveň ochrany a každý disponuje rozdílnými nároky na výkon. Funkční algoritmy se dělí podle délky klíče, typu klíče a velikosti šifrovaných bloků. Mezi běžně používané šifrovací algoritmy patří:

AES

Advanced Encryption Standart je symetrická šifra, která byla v roce 2001 přijata Národním institutem pro standarty a technologie jako náhrada za stárnoucí a již prolomený Data Encryption Standart (DES). AES si rychle vydobyl v dnešním světě rychle oblíbenost zejména díky odstranění nedostatků svého předchůdce. Často bývá označován jako „zlatý standart šifrování dat“ a je využíván úřady po celém světě. Byl považován za budoucnost šifrování v aplikacích každodenního života. Jedná se o velmi efektivní metodu šifrování používanou hlavně pro zabezpečení souborů, aplikací, databází, přenosu Wifi, VPN a protokolů SSL/TLS [12, 13].

V době tvorby Data Encryption Standart měl návrh šifry pro tehdejší výkon výpočetní techniky smysl, avšak se zlepšením výkonu se čas prolomení šifry zkracoval. Prolomení v roce 1998 trvalo 56 hodin, v roce 2021 již pouhých 5 minut. Z finančních důvodů a pokusů vyřešit problémy DES byl vytvořen 3DES, který šifroval pomaleji než AES, tudíž se tolik nerozšířil. AES vyřešil chyby předchůdce, mezi něž můžeme zahrnout [12, 15, 16]:

- Podporu delších klíčů. AES podporuje délky klíče o velikosti 128, 192 a 256. Délka klíče dělá šifru odolnější proti útoku hrubou silou (hádáním). Prolomení AES-256 by dnešním moderním počítačům trvalo déle, než je stáří vesmíru.
- Odolnost proti moderním útokům. AES je moderní standart symetrické kryptografie, který byl široce analyzován, aby odolal útokům jako jsou lineární a diferenciální kryptoanalýza.
- Několik rund šifrování. AES jako bloková šifra provádí šifrování bloků dat nebo malých částí. Každá runda zahrnuje míchání dat a permutace, čímž stěžuje zpětné inženýrství nebo nalezení slabých míst v šifrovacím procesu. Počet

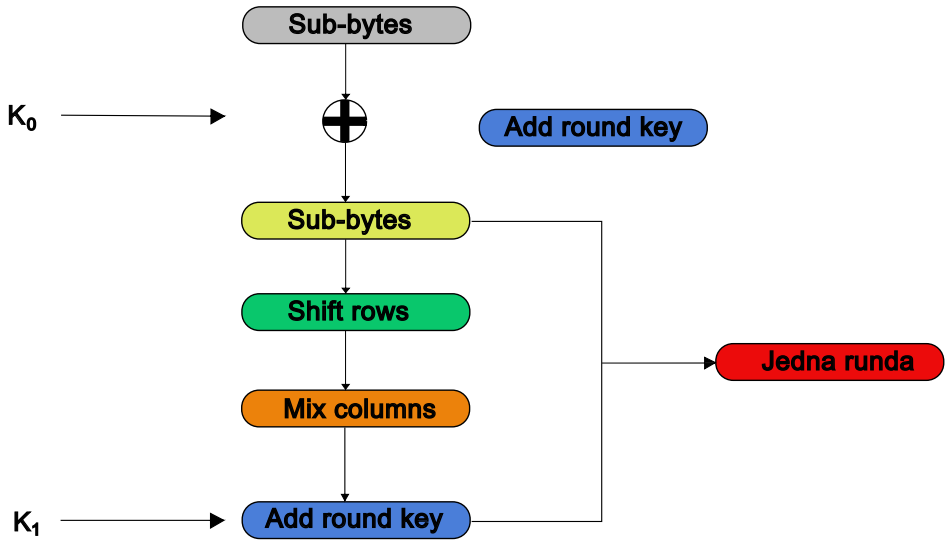
prováděných kol závisí na délce klíče použitého k šifrování dat. Klíč o velikosti 128 bitů má deset rund, klíč o velikosti 192 bitů má 12 rund a klíč o velikosti 256 bitů má 14 rund.

Systém AES převádí vstupní informace do bloků maticí, protože jeden vstupní blok má vždy 16 bajtů, matice je o velikosti 4 x 4, tudíž každá buňka matice obsahuje jeden Bajt informace dle 1.1. způsobem je ze vstupního klíče vytvářeno dalších $(n+1)$ klíčů, kde n značí počet rund. Tyto klíče nazýváme rundovní klíče.

Tab. 1.1: Dělení vstupu do bloku

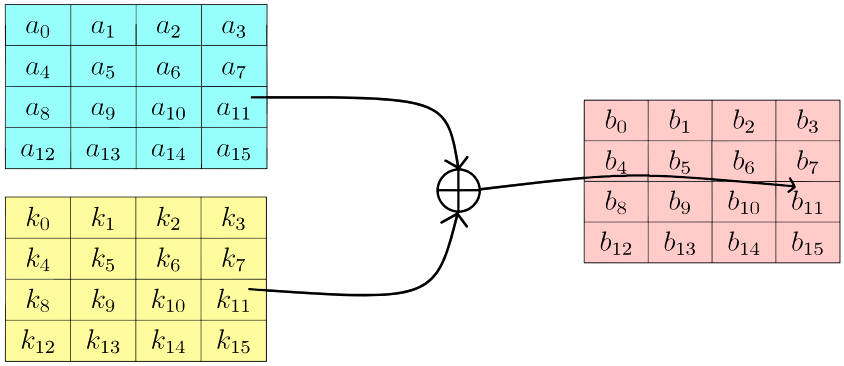
8b **1.**	8b **2.**	8b **3.**	8b **4.**
8b **5.**	8b **6.**	8b **7.**	8b **8.**
8b **9.**	8b **10.**	8b **11.**	8b **12.**
8b **13.**	8b **14.**	8b **15.**	8b **16.**

Na obr 1.1 je znázorněna jedna runda šifrování, která obsahuje jednotlivé vrstvy šifrování. K úspěšnému zašifrování musí být projíty všechny rundy, kdy poslední runda neobsahuje vrstvu Mix columns [12].



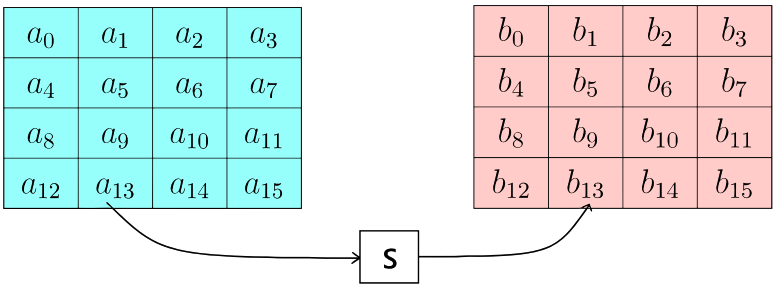
Obr. 1.1: Proces jedné rundy AES

- Add round key: data bloku projdou funkcí XOR s prvním vygenerovaným klíčem K_0 výsledný blok dat je předán další vrstvě. Každý Add round key využívá jiný klíč, jenž je odvozený z z původního klíče. V posledním kole je z výsledku funkce XOR sestaven šifrovaný text.



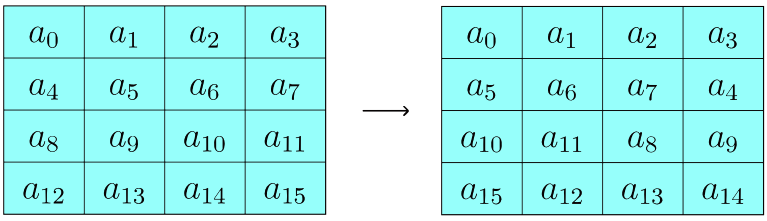
Obr. 1.2: Znázornění Add round key

- Sub-bytes: v této vrstvě se každý Bajt převede do šestnáctkové soustavy. Převedeným datům jsou namapovány nové hodnoty podle řádků a sloupců substitučního pole (S-box), které jsou zpětně převedeny znovu do matice.

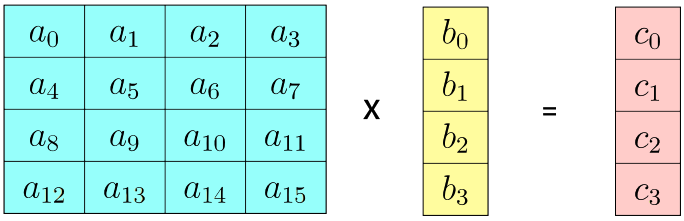


Obr. 1.3: Znázornění Sub-bytes

- Shift rows: posunuje řádky mezi sebou. První řádek je přeskočen. Druhý řádek je posunut o jednu pozici doleva. Třetí je posunut o dvě pozice doleva a poslední řádek je posunut o tři pozice doleva.
- Mix columns: matice hodnot je vynásobena stavovým polem, tímto krokem je získáno nové stavové pole pro další vrstvu Mix column a matice hodnot je upravena. Tato vrstva není prováděna v poslední rundě.



Obr. 1.4: Znázornění Shift rows



Obr. 1.5: Znázornění Mix columns

RSA

Rivest-Shamir-Adleman je asymetrická šifra, kterou vyvinul Ron Rivest, Adi Shamir a Leonard Adleman. Jejich práce byla publikována v roce 1977, algoritmus funguje na principu součinu prvočísel (faktorizace). Funkce musí být dostatečně složitá, aby odolala hrubé síle, přitom však byla dostatečně rychlá. Násobení dvou prvočísel je elementární úloha, avšak pro rozklad tak velkého čísla na dvě prvočísla neexistuje algoritmus v polynomiálním čase. Jedná se o první šifru, která je vhodná pro podepisování i šifrování. RSA se obvykle používá pro komunikaci přes internet z jednoho místa na druhé, což je vhodné i pro funkci databází. Doporučená délka klíče se pohybuje mezi 1024 až 3072 bity, přičemž od roku 2006 je délka klíče 1024 bitů prolomitelná do dne [14].

Při šifrování algoritmem RSA je použit veřejný klíč k šifrování a k dešifrovaní je použit soukromý klíč vlastníka, tudíž nedochází k výměně klíče mezi stranami. K zašifrování zprávy pomocí RSA musíme nejprve vytvořit klíčový pár a následně šifrovat data.

Před samotným šifrováním je nezbytné vytvořit veřejný a soukromý klíč, pomocí nichž budeme moci následně zašifrovat nešifrovaný text. Generaci klíčů pro RSA šifru můžeme vyjádřit následovně [14]:

1. **Zvolíme dvě velká prvočísla:** p and q
2. **Vypočítejme:**
   (a) $n = p \cdot q$
   (b) $\phi(n) = (p-1)(q-1)$
3. **Zvolíme číslo, kde:** $e[(\gcd(e)) - 1; 1 < e < \phi(n)]$
4. **Vypočteme:** $d = e - 1 \bmod (p-1)(q-1)$

Soukromé klíče jsou (n, d), veřejné klíče jsou (n, e).

Jakmile máme vypočítány klíče, použijeme parametry pro šifrování samotného textu pomocí příslušného klíče.

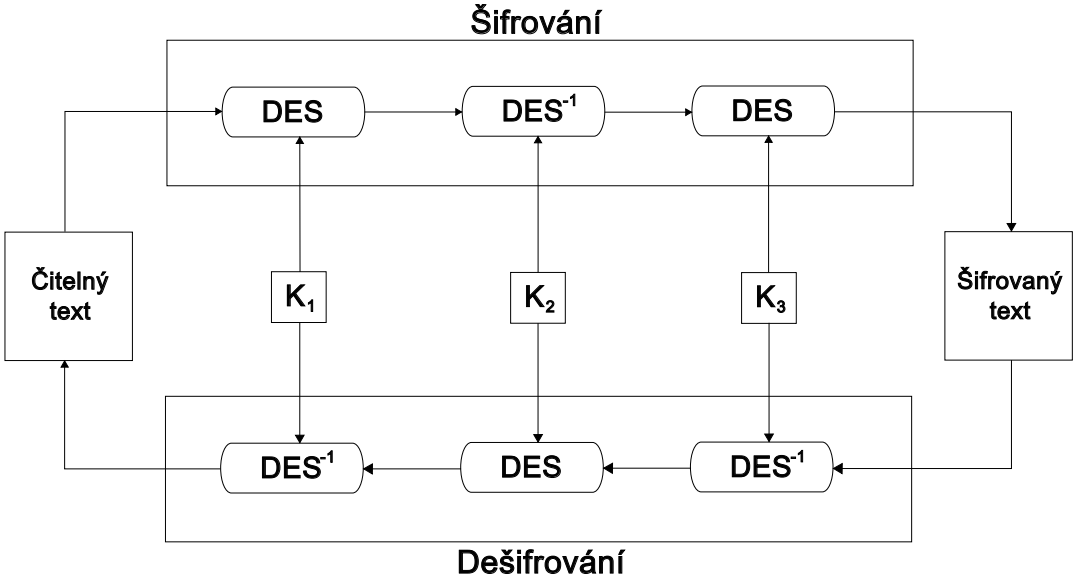
- Pro zašifrování použijeme veřejný klíč e, kde m je nezašifrovaná zpráva, tudíž: zašifrovaná zpráva $= me (\bmod n)$
- Pro zašifrování použijeme veřejný klíč d, kde c je zašifrovaná zpráva, pak: dešifrovaná zpráva $= cd (\bmod n)$

RSA je jednou z nejvyužívanějších asymetrických šifer kvůli mnoha výhodám při přenosu dat po internetu. Zachycení dat je zbytečné, protože útočník nebude mít soukromý klíč k dešifraci a při změně dat vlastník pozná, že data byla změněna, protože informace nebudou moct být dešifrovány. RSA je rychlejší v porovnání s jinými asymetrickými metodami šifrování než např. DSA. Delší velikost klíče vede k složitějším výpočtům a je méně efektivní než jiné metody. Tato šifra má svá omezení při šifrování s větším objemem dat, proto bývá používána hlavně pro kratší dokumentace, soubory, zprávy a platby nebo nachází své využití v hybridním schématu šifrování, kdy je asymetrická kryptografie používána hlavně pro digitální podpis, tudíž je používána silných stránek obou metod šifrování [14, 15, 16, 18].

3DES

Triple Data Encryption Standart je symetrická bloková šifra, která používá k šifrování bloků dat tři 56bitové klíče. Oficiálně byla tato šifra pojmenována Triple Data Encryption Algorithm (3DEA), avšak bývá nejčastěji nazývána 3DES, protože se jedná o bezpečnější a pokročilejší verzi DES, která používá DES na každý blok třikrát. DES je symetrická šifra založena na Feistelově síti, jež je totožná pro šifrování i dešifrování, což ji dělá efektivnější pro implementaci. Šifrování začíná prvním klíčem, následuje dešifrování druhým klíčem, a nakonec opětovné šifrování pomocí třetího klíče. Dešifrovaní je dle Feistelovy sítě invertovaná funkce, tudíž začíná dešifrování třetím klíčem, pokračuje šifrováním druhým klíčem a nakonec dešifruje prvním klíčem. Tento proces obr. 1.6 trojitého šifrování poskytuje vyšší bezpečnost

než jednoduché šifrování DES díky delší délce klíče. Nicméně algoritmus 3DES postupně ztratil popularitu kvůli své nižší rychlosti a byl nahrazen bezpečnějšími algoritmy, jako jsou AES-256 a XChaCha20 [17].



Obr. 1.6: Blokové znázornění 3DES

Přestože je algoritmus DES používán ve firmách, jako standard se nejspíše neudrží dlouho, dokonce i Národní institut pro standardy uvedl, že šifrování 3DES bude od 31. prosince 2023 zakázáno, povoleno bude pouze dešifrování již zašifrovaných dat. Přesto je 3DES nadále využíván pro platební systémy, v aplikacích jako Firefox, nebo Microsoft Office, či systém Unix. NIST označuje AES jako lepší variantu zabezpečení než 3DES [15, 16, 17].

Twofish

Twofish je další symetrickou blokovou šifrou s klíči 128, 192 nebo 256 bitů. Jejím předchůdcem je veřejně dostupná šifra Blowfish, avšak Twofish je rychlejší a použitelná na software i hardware. Twofish byla v roce 1997 přihlášena do soutěže NIST ve snaze vyvinout šifru, která nahradí DES. V soutěži přednostmi algoritmu měla být rychlost, dlouhá délka klíče a a velká délka šifrovaného bloku, což Twofish splňuje. V soutěži však vyhrál AES nad Twofish kvůli jednoduššímu designu, vyšší bezpečnosti a vyšší rychlosti.

Twofish používá Feistelovu síťovou strukturu s 16 koly, obsahuje S-boxy závislé na klíči a komplexní klíčový plán. Funkce Twofish zahrnuje předem vypočtené substituce závislé na klíči, což zvyšuje bezpečnost šifrování. Důležité je, že Twofish není patentován, což znamená, že je volně k dispozici pro použití bez omezení. Tato šifra

nabízí flexibilitu v kompromisu mezi výkonem, paměťovým využitím a dalšími parametry, což umožňuje různé implementace šifrování s různými prioritami. Twofish byl navržen pro šifrování na zařízení s nízkou výpočetní kapacitou, využívá vždy 16 šifrovacích rund bez ohledu na velikost klíče. Nejčastěji je používán pro šifrování souborů [15, 16].

1.4 Analýza databázových systémů

Databázové systémy byly porovnávány dle popularity na stránkách DB-Engines [1]. Stránka porovnává popularitu podle počtu zmínek na webu, počtu pracovnách nabídek, četnosti na diskuzních fórech a relevanci na sociálních sítí.

1.4.1 Porovnání nejpopulárnějších databázových systémů

Tabulka 1.2 nám pomůže pro stručné shrnutí následujícího popisu databázových systémů [4].

Tab. 1.2: Tabulka základních vlastností DBMS systémů

DBMS	Typ databáze	Licensování	Škálovatelnost	Křivka učení
Oracle	multimodelová, SQL	vlastní	vertikání, horizontální	složitá
MySQL	SQL	GNU general public license	vertikální	mírná
MSSQL	T-SQL	vlastní	vertikální	složitá
PostgreSQL	objektově relační, SQL	otevřený zdrojový kód	vertikální	složitá
MongoDB	NoSQL, dokumentově orientované	SSPL	horizontální	mírná

Oracle je relační databázový management systém (RDBMS), jenž je vlastněn firmou Oracle Corporation. Mezi konkurencí SQL Oracle vyniká. Ve svých posledních verzích se zaměřuje i na cloud computing. K dispozici má bezplatnou i placenou variantu, avšak bezplatná verze má velmi limitované funkce a i levnější z placených verzí za 17 500 dolarů má omezené funkce. Za vyšší cenu naopak poskytuje velkou kapacitu a vhodnou dokumentaci se zákaznickou podporou [4, 19].

MySQL je jedním z nejoblíbenějších RDBMS s otevřeným zdrojovým kódem, kterou vlastní a spravuje společnost Oracle. I jeho bezplatná verze je často aktualizována. Po aktualizacích byla přidána i podpora NoSQL pro větší konkurenceschopnost. Kvůli své jednoduchosti a vysoké kompatibilitě s ostatními systémy bývá doporučována pro malé až středně velké projekty s velkými rozpočtovými omezeními [19, 4].

Dle zdroje dalším nejoblíbenějším RDBMS je **Microsoft SQL Server**, který slouží jako komerční nástroj pro systém Windows a Linux. Verze, které jsou zdarma, fungují spíše pro testování platformy, přičemž placené verze jsou již dražší. Platforma je především tvořená komponenty zaměřenými pro podniky, jako poskytnutí extraction, transformation, load (ETL) řešení a tvoření znalostní báze. Microsoft SQL Server poskytuje rozsáhlou dokumentaci i podporu cloudových databází [4, 19].

PostgreSQL je bezplatný RDBMS s otevřeným zdrojovým kódem, který působí na trhu již od 90. let 20. století. Má schopnost podporovat jak relační, tak nerelační formáty dat. PostgreSQL umožňuje práci s lokálními servery i cloud servery. Je k dispozici na platformách jako je Microsoft, Android, iOS a další, lze dobře integrovat s daty z nejrůznějších databází. PostgreSQL má početnou síť stoupenců, kteří systém aktualizují a nabízí bezplatnou podporu, avšak dokumentace nedosahuje úrovně jiných placených DBMS. Jednou ze silných stránek je nadstandardní škálovatelnost a konzistentnost v zpracování požadavků na databázi [4, 19].

MongoDB je bezplatný nerelační DBMS s otevřeným zdrojovým kódem, který disponuje i placenou verzí. Jelikož většina systémů pracuje se strukturovanými daty, MongoDB je schopen pracovat i se strukturovanými daty se zpomalením výkonu. Propojuje nerelační databáze pomocí ovladačů s aplikacemi. Je velmi flexibilní pro vývojáře, protože je méně omezuje ve vkládání druhu dat a také lze škálovat i horizontálně [4, 19].

1.4.2 Hodnocení bezpečnostních funkcí DBMS

Oracle nabízí široké spektrum bezpečnostních funkcí. Po vytvoření uživatelského účtu může být heslo zabezpečeno, profilováno a upravováno různými způsoby, dle potřeb při tvorbě projektu. Různé autentifikační metody jako Kerberos, RADIUS, digitální certifikát a další, jsou implementovány. Bezpečnost šifrováním dat mimo využití a přenášených dat je podpořeno správou klíčů s možností využití Oracle Key Vault vnějších bezpečnostních serverů, který se dá využívat k funkci TDE. Funkce revize aktivit umožňuje sledovat veškerou aktivitu v databázi i mimo ni pomocí IP adres. Oracle vydává taktéž dokumentaci zabývající se vývojem bezpečné aplikace dle jejich aplikačních zásad [20].

MySQL poskytuje několik bezpečnostních opatření na ochranu dat. Řízení účtů je založeno na tabulkách řízení přístupu, které konfigurují přístup do databáze a veškeré proveditelné požadavky uživatelů. MySQL poskytuje v Enterprise verzi symetrickou a asymetrickou kryptografii, služby pro správu klíčů, podpis, validaci podpisu a transparentní šifrování dat. Pro auditování MySQL je mnoho pluginů, které se zaměřují na jiné úrovně, jakožto cloud, uživatele a databáze [21, 22].

Microsoft SQL Server je propojen jako platforma s Microsoft produkty, což z něj dělá jeden z nejoblíbenějších cílů útoků, protože většina malware nástrojů je cílena na Windows. [23] Přístup pomocí rolí je součástí zabezpečení, který obsahuje i předem specifikované role. U rolí lze definovat i přístup ke konkrétním řádkům a sloupcům. Data jsou přenášena standardně pomocí nešifrovaného protokolu Tabular Data Stream, u kterého je možnost šifrovat pomocí SSL/TLS a zabezpečit certifikátem. Data Protection API je šifrovací funkce v systému Windows, která zabezpečuje veškerá přihlašovací jména, role, pověření, certifikáty, funkce, šifrovací klíče, avšak je závislá na verzi Windows. Enterprise, Developer a Data center poskytují funkci transparentního šifrování s možností několika šifrovacích algoritmů. SQL Server Audit je součástí databáze, kde funguje jako nástroj pro auditování transakcí a změn dat v databázi [24, 25].

PostgreSQL i přes svoji nulovou cenu nabízí velmi kvalitní a sofistikovanou bezpečnost. Databáze zajisté nabízí systém rolí, který udává daná práva v databázi s povinným uživatelským jménem a volitelným heslem. Postgre také nabízí možnost přidělení role roli. Soubor pg_hba.conf řídí a nastavuje přístup k databázi, který dokáže vyžadovat od hosta i autentifikaci. Soubory databáze jsou přístupné pouze pro účet superuživatele databáze, přičemž neoprávněným účtům není povoleno ani jejich čtení. Rozšíření pgcrypto je kryptografické rozšíření PostgreSQL s širokých kryptogtafickým využitím, avšak šifruje pouze na úrovni sloupců, které je pro zabezpečení celé databáze nedostatečné. Plní účel šifrování a hašování dat v databázi. Podporováno je šifrování dat mimo využití pomocí šifrování na úrovni úložiště. Šifrování přenášených dat je řešeno pomocí SSL/TLS. Auditování aktivity databáze je zajištěno pomocí rozšíření třetí strany pgAudit [26, 27, 28].

Samo **MongoDB** má bezpečnost podobnou zabezpečení jako MySQL. Zakládá svoji bezpečnost na ověřování identity i s vícefaktorovým ověřováním pomocí mechanismů jako SCRAM, certifikátů x.509, LDAP nebo Kerberos. Definice rolí uživatelům umožňuje definovat povolená privilegia. Standardně nabízí i zabezpečení pomocí TLS/SSL, také zabezpečení dat mimo užití pomocí AES256-GCM, která je kompatibilní s využívaným storage enginem WiredTiger [29]. MongoDB nabízí funkci revize činností navrženou pro detekci neoprávněného přístupu k datům, která kontroluje operace vytvoření, čtení, změn, mazání, správy klíčů a ověřování rolí. Filtry činností lze přenastavit pro specifické události. Bezpečnostní funkce MongoDB nejsou sou-

částí výchozí bezplatné verze, pro dosažení základních bezpečnostních standardů je třeba si zaplatit alespoň komerční verzi [30, 31].

Tab. 1.3: Hodnocení bezpečnostních funkcí DBMS

DBMS	**Šifrování**	**Správa klíčů**	**Autentifikace**
Oracle	transparentní šifrování	Oracle Key Vault	✓
MySQL	transparentní šifrování	Oracle Key Vault	✓
MSSQL	transparentní šifrování	Extensible Key Management	✓
PostgreSQL	pgcrypto	pgcryptokey	✓
MongoDB	WiredTiger storage engine	KMIP, lokální správa klíče	✓

1.5 Šifrování v databázových systémech

1.5.1 Data at rest

Data at rest je označením pro data mimo využití, často bývají uložená na pevném disku, flash disku, v archivech, nebo cloudu. Data mimo využití bývají cennější avšak obtížnější k ukradení. Proto je doporučeno uchovávat šifrovací klíče mimo úložiště s ostatními daty. Tento pojem zahrnuje i data jakožto zálohy nebo logy [32, 33].

1.5.2 Data in transit

Data in transit je termín označující data přenášená po síti, a to ať privátní nebo veřejné. Nejčastějšími případy vyžadující zabezpečení jsou online transakce, e-maily, soukromé zprávy nebo přenosy souborů, jejichž bezpečnost bývá obvykle zajištěna pomocí Secure Sockets Layer nebo Internet Protocol security [32].

1.5.3 Přehled různých typů šifrování DLE, FLE, FDE, TDE a ALE

Database-level encryption

Database-level encryption je šifrování prováděno přímo v databázi pomocí kryptografického modulu. Můžeme vybrat specifické sloupce pro zašifrování a k nim i roz-

dílné klíče. Několik DBMS podporuje šifrování na úrovni databáze například Oracle, MySQL a Microsoft SQL Server. Pojem šifrování na úrovni databáze označuje i šifrování celé databáze nebo dat v ni. Bývá spojeno s bezpečnostními opatřeními jako transparentní šifrování a šifrování na úrovni klastru. Šifrování je nezávislé na aplikaci, je řešeno již v databázi, čímž však snižuje výkon databáze. V dnešní době je považováno šifrování na úrovni databáze jako „základ bezpečnosti“ při ochraně dat [11, 34]. Mezi silné stránky DLE patří [35]:

- Je velmi jednoduchý na implementaci do DBMS. Bývá součástí v databázových software a funguje automaticky.
- Šifrování jakkoli nezasahuje do aplikace.
- V porovnání s jinými metodami má nejmenší dopad na výkon databáze.
- Kvůli využití méně klíčů je i správa klíčů méně komplikovaná.
- V porovnání s šifrováním na nižších úrovních v databázi, šifruje i zálohy databáze.

Mezi slabé stránky šifrování na úrovni databáze patří:

- Omezená ochrana proti útokům útokům zevnitř a potencionálně i útokům na úrovni aplikace.
- Využívá jeden klíč na všechny operace databáze.
- Mnoho průniků velkého měřítka bylo provedeno na systémech šifrovaných na úrovni databáze.
- Řešení na úrovni databáze jsou drahá.
- Zabezpečuje data pouze mimo využití. Při přenosu dat po síti je potřeba je zašifrovat.

Filesystem-level encryption

Filesystem-level encryption je termín označující softwarové šifrování a dešifrování souborů, adresářů, pevných disků, výměnných medií a mailových zpráv. Při velkých objemech dat se bootování i práce se soubory zpomalí. Šifrování souborů umožňuje zálohu šifrovaných souborů, hash pro ověření integrity dat a symetrické šifrování [36]. Nástroj fscrypt je k dispozici v systému Linux, podporuje šifrování různých adresářů pomocí různých klíčů ve stejném souborovém systému. Nástroj má své hlavní využití pro víceuživatelské systémy, kde je potřeba uživatele od sebe kryptograficky oddělit. Nástroj funguje transparentně po zadání hesla složky. Metadata jako časové značky, velikosti a počty souborů. rozšířené vlastnosti se nešifrují [37]. Výhodami používání šifrování na úrovni souborů je [38]:

- Zajišťuje granuální kontrolu nad tím, co má a nemá být šifrováno. Toto může pomoci rozdělit kritické pracovní soubory od osobních méně důležitých.

- Při prolomení hesla účtu zůstávají data stále zašifrovaná a jsou pro útočníka nepoužitelná.
- Zašifrované data při přesunu po síti zůstávají stále zašifrovaná, což přidává navíc vrstvu ochrany dat při přenosu po síti.
- Při prolomení jednoho klíče nemůže útočník přistupovat k jiným datům, protože jsou zašifrována pomocí jiného klíče.

Nevýhodami používání šifrování na úrovni souborů je:

- Přehled o zašifrovaných a nezašifrovaných souborech je složitý, pokud je zašifrováno hodně souborů, zadávání hesel se může stát pro uživatele zdlouhavým úkolem.
- Zálohování souborů může být komplikováno mnoha šifrovanými složkami.

Full disk encryption

Full disk encryption je typ šifrování, který šifruje celý disk ve většině případů pomocí jednoho klíče s výjimkou master boot record. Implementuje transparentní šifrování. Je třeba před bootem autentizovat uživatele a zavést bezpečný systém na obnovu klíče pokud dojde k jeho ztrátě. Sám nesplňuje bezpečnostní standardy, je napadnutelný pomocí cold boot útoku, nebo pomocí krádeže v uspaném režimu. Od vydání Windows Vista umožňuje Windows šifrování celého disku pomocí nástroje BitLocker. Po odemknutí sytému heslem nebo například USB flash diskem s klíčem začne nástroj šifrovat a dešifrovat všechny operace se soubory, pomocí nízkoúrovňového ovladače, přičemž veškeré interakce aplikací se zašifrovaným úložištěm budou vypadat transparentně [39, 40]. Mezi hlavní výhody nástroje BitLocker patří:

- Jednoduché používání a správa. Lze jednoduše zapnout v Ovládacích panelech. Nástroj také poskytuje jednoduchou obnovu hesla, jelikož je propojen s Windows účtem.
- BitLocker poskytuje i několik možností šifrování jako AES-128, AES-256 nebo XTS-AES, které je vhodné vybrat dle potřeby zabezpečení a výkonu.

Hlavními nevýhodami nástroje BitLocker je:

- Nástroj Bitlocker není dostupný pro každou verzi Windows. Windows 10 Home nemá k dispozici BitLocker.
- BitLocker může zpomalit systém, pokud zařízení je starší nebo méně výkonné. Proces šifrování a dešifrování využívá kapacitu procesoru a disku.
- Některé aplikace nebo zařízení nemusí být kompatibilní s nástrojem BitLocker, mohou pak vyvolat chyby nebo narušovat šifrování. Tento problém se hlavní vyskytuje u bootovacích USB disků, antivirových programů nebo nástrojů pro zálohování.

Application-level encryption

Application-level encryption probíhá již v dané aplikaci, což vede k další vrstvě obrany a chrání data i během přesunu po síti na všech vrstvách. Vyžaduje komplexní infrastrukturu správy klíčů, která ji více autoritami v databázi hodně zatěžuje, a tím činí indexování a vyhledávání nemožným. Používání šifrovacích knihoven uvnitř aplikace jako libsodium, themis, gocrypto, Google Tink usnadňuje implementaci šifrování na úrovni aplikace. Tyto knihovny poskytují různé metody šifrování zjednodušující přenos zašifrovaných dat do databáze [11, 39, 41]. Hlavními výhodami používání šifrování na úrovni aplikace je:

- Šifrování přidává extra bezpečnostní ochranu TLS protokolu. Při přenosu dat je TLS nezbytné, využití šifrování v aplikaci uchovává šifrovaná data zašifrovaná, dokud je klient nedešifruje, ne pouze pro přenos dat.
- ALE umožňuje méně důvěřovat infrastruktuře. Poskytuje bezpečnost na všech základních vrstvách.
- Šifrování v aplikaci na straně klienta znamená, že veškeré klíče nejsou uloženy na straně databáze. Snižuje hrozby na straně databáze jako nepříznivý přistup do databáze, únik prostřednictvím snímků, záloh a protokolů.

Dle zmíněných výhod ALE lze konstatovat, že ALE poskytuje kvalitní zabezpečení, to avšak je za cenu následujících nevýhod:

- Implementace šifrování na úrovni aplikace je velmi složitá. Největší výzvou je správa klíčů. Mezi výzvy správy klíčů patří ukládání klíčů, propojení klíčů s identitou a životní cyklus klíčů.
- Při hacknutí klienta se hacker dostane k jeho datům uloženým v databázi. S největší pravděpodobností se podaří útočníkovi spíše získat přístup k databázi skrz klienta pomocí phishingového útoku než hacknutím samotné databáze.
- Velmi komplikuje vyhledávání a indexování v databázi, avšak nedělá ho nemožným.
- Je potřeba dodržovat konzistentnost při vývoji aplikace. Pro aplikaci přibývá krok šifrování mezi zápisem dat, který je potřeba dodržovat

Transparent Data Encryption

Transparent Data Encryption je používáno na šifrování celé databáze na úrovni dat mimo použití. Data nejsou chráněna při přesunu po síti. Transparentní systém pracuje na stránkové úrovni, data jsou tudíž šifrována v reálném čase před zapsáním na disk a dešifrována při přesunu z disku do systémové paměti. Šifrování nevyžaduje změny v aplikaci, aby TDE fungovalo správně. Transparentní šifrování neprovádí autorizaci před nepovoleným vstupem do databáze. TDE využívá symetrické klíče

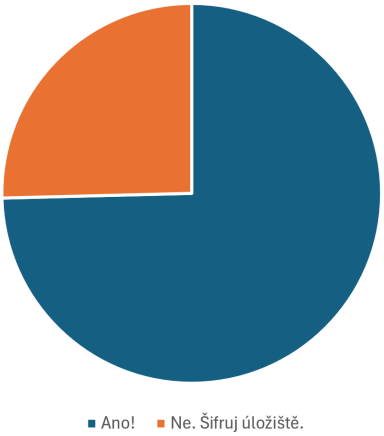
pro minimální snížení výkonu, k čemuž bývá typicky využita šifra AES. Klíče na šifrování databáze jsou zašifrovány Master klíčem, který je uložen bezpečně mimo databázi. Na rozdíl od jiných šifrovacích řešení transparentní šifrování se vyznačuje jednoduchou implementací. Mezi hlavní výhody transparentního šifrování patří granualita a flexibilita při šifrování databáze, díky nimž můžeme rozlišit, co vůbec budeme šifrovat a jakým klíčem. Transparentní šifrování nevyžaduje kombinaci se šifrováním na úrovni úložiště, avšak pokud existuje, přidává databázi další vrstvu zabezpečení do hloubky. Při kopírování nebo jiném vystavení dat v surové formě nedojde k nebezpečnému vystavení dat v jejich odhalené formě. Transparentní šifrování se vztahuje i na zálohy [43].

2 Praktická část

Praktická část se zaměřuje na implementaci transparentních šifrování v PostgreSQL. Popisuje úskalí transparentního šifrování v PostgreSQL a jejich možné řešení. Jednotlivé PostgreSQL DBMS řešení transparentního šifrování jsou popsány, jsou analyzovány jejich vlastnosti, výhody, nevýhody, dále je popsáno, jak implementovat dané řešení. Porovnání jednotlivých řešení bylo provedeno v experimentálním prostředí, kde byl měřen jejich výkon pod určitou zátěží a při určité velikosti databáze. Měření porovnává tři navrhované řešení transparentního šifrování v PostgreSQL se standardní nezašifrovanou verzí PostgreSQL a zašifrovanou verzí pomocí souborového systému.

2.1 Transparentní šifrování dat v PostgreSQL

Při průzkumu potencionálních řešení jsem zjistil, že PostgreSQL neumožňuje šifrovat pomocí transparentního šifrování. Problémem je, že vývojáři PostgreSQL zastávají své konzervativní názory, které naopak přispívají tomu, proč je PostgreSQL tak stabilní, přestože zpomalení inovace projektu může poškodit budoucnost databázového systému. Vývojáři PostgreSQL si stojí za názorem, že šifrování je problém na úrovni úložiště, a tak by měl být řešen pomocí souborového systému. Názor uživatelů se však značně liší od názoru vývojářů, což vyplývá například z hlasování komunity uživatelů na platformě X [42]. Otázkou hlasování bylo, zda-li PostgreSQL potřebuje TDE. Z výsledku na obr. 2.1 lze vidět, že skoro 75 % hlasovalo pro „Ano!“, ačkoli se hlasování v konečném součtu zúčastnilo pouze 63 hlasujících.



Obr. 2.1: Hlasování o důležitosti transparentního šifrování v PostgreSQL

Z toho vyplývá, že vývojáři určili technickou stránku šifrování jako pokrytou a nehodlají se zabývat otázkou složitosti implementace pro projekty organizací. Díky

těmto funkcím, které umožňují jednodušší organizaci, hodně firem zůstává u placených databázových systémů, této příležitosti si ale zároveň všímají firmy vytvářející externí nástroje pro PostgreSQL. Externí nástroje pro transparentní šifrování jsou v nových verzích PostgreSQL zpoplatněny.

2.2 Instalace PostgreSQL TDE

Pro interní účely bylo zvoleno řešení TDE od firmy Cybertec, které je jako jediné možné používat zdarma ve verzi 12.3. Jedná se o implementaci, která podporuje transparentní a kryptograficky bezpečné šifrování dat na úrovni klastru. Soubory obsahující data jsou přidány do klastru a zašifrovány. Při načtení jsou data automaticky dešifrována, avšak v paměti zůstávají data ve zranitelné podobě. Při tvorbě klastru je požadován parametr šifrovacího klíče, což je rozdíl, jímž se liší od základní verze PostgreSQL. Parametr šifrovacího klíče je definován skriptem nebo funkcí, která vrací přesně dvaatřicetimístný hexadecimální kód [44].

Databáze je šifrována pomocí 128 bitové AES-CTR (Counter mode neboli čítačový mód). Čítačový mód je pojmenován podle využívání čítače zvyšujícího číslo od 0 postupně k zašifrování s klíčem a xorováním s nešifrovaným textem. Módy jsou v AES využity pro znáhodnění zašifrované zprávy. Znáhodnění má chránit proti zranitelnosti přehrání zprávy. Výhodou CTR módu je paralelizace umožňující rychlé šifrování a dešifrování. Hardwarová podpora v procesorech od společností AMD a Intel ještě více snižuje dopad šifrování AES. Dle Cybertech by měl být dopad šifrování na výkon minimální, až zanedbatelný. Využitím moderního hardwaru by mělo být možno dosáhnout rychlosti šifrování až 5 GB/s, což je rychlost stejná, nebo rychlejší jako u moderních SSD disků [45].

Bezpečnost není pouze o zabezpečení dat mimo využití, z toho důvodu tohle řešení nabízí i několik dalších pomocných vrstev zabezpečení, aby byla pokryta většina bezpečnostních hrozeb. PostgreSQL TDE obsahuje všechny standardní funkce základního PostgreSQL. Mimoto nabízí extra funkce jakožto:

- Transportní šifrování klient/server prostřednictvím SSL.
- Šifrování procesu replikace.
- Zabezpečení replik.

Hlavními nevýhodami řešení je:

- Nemožnost měnit klíč při kompromitaci nebo vypršení životnosti. Při změně klíče by musel být inicializován nový klastr, a tím vložen nový klíč. Databáze by bylo vhodné obnovit pomocí SQL dump, a tím provést obnovu databáze do původního stavu.
- Nemožnost provedení vylepšení z PostgreSQL na PostgreSQL TDE. Řešení nepodporuje šifrování stávajících klastrů. Pro přechod na vylepšenou verzi bude

muset být použita buď replikace, migrace online, nebo SQL dump.

- Řešení je tvořeno pouze pro Linux a Mac OS. Pro maximální efektivitu databáze je doporučen SELinux.
- Jediný šifrovací klíč na celý klastr by měl být uložen externím programem kvůli maximální bezpečnosti jediného klíče.

2.2.1 Popis skriptu pro instalaci

Skript byl pro zkušební účely implementován na linuxové distribuci Ubuntu. Skript je psán v jazyce Shell, který překládá sérii příkazů příkazového řádku. Shell postupně čte soubor příkazů a postupně spouští jednotlivé příkazy. Vkládané příkazy Shellem se v příkazové řádce nezobrazují, avšak jejich výpis ano. Skript Shellu je možné poznat podle prvního řádku skriptu tzv. Shebangu. Na výpisu 2.1 je na prvním řádku definován Shell `#!/bin/sh`. Bash je následovník Shellu, který by byl například definován `#!/bin/bash`. Jazyk Shell kromě interpretace příkazů příkazové řádky přidává jednoduché funkce, jako například zadání vstupu proměnné.

Ve skriptu je požadován po uživateli vstup pojmenování účtu, který dostane přístup k databázi. Pro načtení vstupu byl použit příkaz `read` s argumentem `-p`, jenž definuje název výzvy a vypisuje vstup ve viditelné podobě.

Instalované balíčky skriptu jsou esenciální pro jeho instalaci. Nenačtení jednoho z balíčků povede k neúspěšné instalaci. Podle potřeb používání databáze by bylo doporučeno doinstalovat další potřebné balíčky. Argument `-y` automaticky odpovídá `yes` pro hladší průběh instalace. Pro instalaci potřebujeme zvýšená oprávnění `sudo`, pro přístup do systémových adresářů. Balíčky obsahující `-dev`, obsahují pomocné soubory pro vývoj. `Zlib1g-dev` je knihovna implementující kompresní metodu deflate obsaženou v knihovnách gzip a PKZIP. Balíček `libreadline-dev` obsahuje knihovny GNU (GNU's Not Unix) readline a history. Knihovna readline napomáhá konzistenci uživatelského rozhraní u programů, které fungují na rozhraní příkazového řádku. Knihovna history umožňuje vyvolávat dříve zadané řádky. `Libssl-dev` implementuje kryptografické protokoly jako SSL a TLS pro bezpečnou komunikaci přes internet. Obsahuje i hlavičkové soubory a příručky pro libssl a libcrypto. `Libperl-dev` obsahuje soubory pro vývoj aplikací interpretem Perl. `Bison` je univerzální generátor překladačů, který převádí bezkontextovou gramatiku na deterministický LR analyzátor, nebo zobecněný LR (GLR) analyzátor. LR analyzátory analyzují zdola nahoru deterministické bezkontextové jazyky v lineárním čase. GLR analyzátory zleva doprava rozšiřují LR analyzátory o zpracování nedeterministických jazyků. PostgreSQL TDE od verze 12.2 vyžaduje tento balíček. Balíček `flex` obsahuje skenery, tedy programy na rozpoznání lexikálních vzorů v textu.

Následovně budeme konfigurovat PostgreSQL pomocí `./configure` s podporou

SSL a moduly Perlu. Konfigurace je zodpovědná za přípravu softwaru k používanému systému. Při chybějících balíčcích bude konfigurace neúspěšná, což se může stát, protože proběhnou změny či aktualizace v instalovaných balíčcích. Pomocí `sudo make` příkazu spustíme řadu operacích definovaných v souboru Makefile, které sestaví program ze zdrojového kódu. Soubor Makefile vzniká konfigurací. Příkaz `sudo make install` zkopíruje programy a knihovny do správných umístění v systému. To znamená, že binární soubory jsou umístěny i do adresáře PATH.

Pro data klastru vytvoříme složku v `/usr/local/pg12tde/data`, které přiřadíme vlastníka dle zadaného jména ze začátku skriptu. Stejný postup zvolíme pro vytvoření složky obsahující vracející hodnotu klíče. Vlastnosti klíče byly popsány v odstavcích výše. Příkaz `touch` vytvoří spustitelný soubor *provide_key.sh*, do nehož vložíme pomocí roury Shebang `echo '#!/bin/sh' | sudo cat >> provide_key.sh`. Podobným příkazem vložíme samotný výpis klíče do spustitelného souboru. Následovně jsou přidány práva pro spuštění skriptu. Klíč by měl být uložen v externím programu jako vzdáleném zabezpečeném úložišti klíčů, není ideální uchovávat klíč v místním souborovém systému, avšak pro interní účely projektu si vystačíme s omezenější bezpečností klíče.

Po vytvoření provizorní správy klíče vytvoříme klastr databáze. Nastartujeme databázi a spustíme ji jako postgres. Pomocí příkazu `SHOW data_encryption;` vypíšeme příkaz, který nám potvrdí, jestli je databáze šifrovaná. Výpis příkazu lze vidět na obr. 2.2.

```
postgres=# SHOW data_encryption;
 data_encryption
-----------------
 on
(1 row)

postgres=#
```

Obr. 2.2: Výpis příkazu zobrazení šifrování

Listing 2.1: Skript pro instalaci PostgreSQL TDE

```
#!/bin/sh
echo This is script to install, PostgreSQL TDE 12.3
read -p 'Username: ' uservar
wget
 ↪ https://download.cybertec-postgresql.com/postgresql-12.3_TDE_1.0.tar.gz
tar xvfz postgresql-12.3_TDE_1.0.tar.gz
sudo apt-get update
sudo apt-get install -y zlib1g-dev libreadline-dev libssl-dev
 ↪ libperl-dev
sudo apt-get install -y bison flex
cd postgresql-12.3_TDE_1.0/
sudo ./configure --prefix=/usr/local/pg12tde --with-openssl
 ↪ --with-perl
sudo make
sudo make install
cd contrib
sudo make install
cd
sudo mkdir /usr/local/pg12tde/data
sudo chown $uservar /usr/local/pg12tde/data
sudo mkdir /usr/local/pg12tde/keypass
cd /usr/local/pg12tde/keypass
sudo touch provide_key.sh
sudo chown $uservar provide_key.sh
sudo echo '#!/bin/sh' | sudo cat >> provide_key.sh
sudo echo 'echo 882fb7c12e80280fd664c69d2d636913' | sudo cat >>
 ↪ provide_key.sh
chmod +x provide_key.sh
cd
/usr/local/pg12tde/bin/initdb -D /usr/local/pg12tde/data -K
 ↪ /usr/local/pg12tde/keypass/provide_key.sh
/usr/local/pg12tde/bin/pg_ctl -D /usr/local/pg12tde/data -l
 ↪ /usr/local/pg12tde/data/log start
/usr/local/pg12tde/bin/psql postgres
```

2.3 Fujitsu Enterprise Postgres

Fujitsu Enterprise Postgres je řešení třetí strany, které je rozvíjeno od roku 2004. Jedná se o enterprise rozšířenou verzi PostgreSQL, která se specializuje na implementaci projektů podniků. Fujitsu Enterprise Postgres poskytuje devadesátidenní zkušební verzi zdarma, což umožnilo tuto enterprise verzi testovat a porovnat s ostatními, ovšem velkou nevýhodou pro implementaci databáze je podpora pouze Red Hat Enterprise Linux (RHEL) a SUSE Linux pro implementaci databáze. Dokumentace od firmy Fujitsu slibuje lepší výkon, vyšší bezpečnost a posílenou spolehlivost. Lepší výkon je dle dokumentace zajištěn funkcí Vertikálních Clusterových Indexů. Clusterové indexy jsou indexy, jejichž pořadí řádků v tabulce odpovídá pořadí řádků v indexu. Vertikální tabulky clusterových indexů nejsou řazeny horizontálně do řádků jak v běžné tabulce ale vertikálně do sloupců, což při větším objemu databáze značně zvýší výkon. Pomocí koordinace fyzického hardware Fujitsu Enterprise Postgres snižuje riziko ztráty dat při selhání pomocí automatizovaných procesů [46].

Fujitsu Enterprise Postgres poskytuje i enterprise funkce pro zabezpečení databáze jako maskování dat, auditovací a transparentní zabezpečení databáze. Maskování dat může fungovat ve dvou verzích – online a offline. Offline maskování dat bývá použito pro testování partnerských prostředí, bez odhalení citlivých informací, protože data jsou přenášena z nemaskované databáze do maskované dle nastavených politik maskování. Online maskování je dynamicky načítáno na data při načtení z disku. Maskování dat může maskovat všechny hodnoty hodnotami náhradními, nebo pouze části sloupců, popřípadě nahrazovat regulárním výrazem. Postgres od Fujitsu má široké auditovací možnosti, které mohou být využity pro bezpečnostní a regulační požadavky. Auditování podporuje rozsáhlou podporu politiky správy dat [47].

Transparentní šifrování dat Fujitsu Postgres používá AES šifrování s 128 nebo 256 bitovým klíčem. Fujitsu Enterprise Postgres šifrování na úrovni databáze nabízí veškeré funkce, které se dají očekávat od transparentního šifrování [47]:

- Plnou podporu šifry AES od výrobců jako je Intel a AMD, která minimalizuje dopad na výkon při šifrovaní nebo dešifrování, tudíž není nutné upravovat rozsah šifrování pro udržení výkonu a veškerá data mohou být zašifrována.
- Je šifrován celý rozměr databáze s tabulkami, daty, funkcemi, zálohami, dočasnými soubory a WAL (Write-Ahead Log).
- Nevyžaduje upravovat fungující aplikaci, protože data jsou transparentně dešifrována při čtení z disku.
- Zašifrovaná data si ponechávají stejný objem velikosti, jako by byla nezašifrována.
- Podporuje proudovou replikaci, která z primárního serveru přenáší data v zašifrované podobě na záložní server.

Fujitsu Enterprise Postgres nabízí využití více klíčů k šifrování a jejich management zašifrováním master klíčem. Fujitsu spolupracuje s IBM na hardwarovém bezpečnostním modulu (HSM) pro správu šifrovacích klíčů a jeho robustního zabezpečení, který sama doporučuje kvůli jednoduchosti v porovnáním se softwarovým řešením zabezpečení klíčů. Transparentní šifrování je na úrovni databáze, avšak je třeba iniciovat v každém tablespace, kde je možnost rozlišit, zda má být šifrování použito či nikoli. Pro správné fungování transparentního šifrování je třeba přiřadit uživateli databáze alespoň vlastnictví datové složky a složky s uloženým klíčem, poté přiřadit práva přístupu k složce s daty databáze a složce obsahující klíče pomocí `chmod 700 /database/inst1 /database/tde/keystore`. Následovně musíme upravit soubor *postgresql.conf* dle 2.2 a restartovat databázi [48].

Listing 2.2: Úprava souboru postgresql.conf

```
1 [fepuser@hostname ~]$ vi /database/inst1/postgresql.conf
2 keystore_location = '/database/tde/keystore'
3 tablespace_encryption_algorithm = 'AES256'
```

Je třeba vytvořit master šifrovací klíč, pomocí kterého jsou zašifrovány klíče použité k šifrování databází. Složka s šifrovacími klíči musí být při spuštění databáze otevřena pomocí master klíče.

Listing 2.3: Nastavení klíčů databáze a zašifrování databáze

```
1 postgres=# SELECT pgx_set_master_key('pass-phrase');
2 [fepuser@hostname ~]$ pg_ctl -D /database/inst1 --keystore-passphrase
  ↪ restart
3 Enter passphrase:  <pass-phrase>
```

Po vytvoření klíčů, databáze a specifikaci klíčů zbývá pro nastavení transparentního šifrování specifikace tablespace, který má být šifrován. Toho lze dosáhnout několika způsoby dle 2.4. Na řádku 1 a 2 výpisu 2.4 je příklad příkazu pro šifrování pomocí AES256 nebo pro žádné šifrování, na řádku 3 je příklad příkazu, který rovnou definuje typ šifrování a zároveň vytváří tablespace. Pomocí příkazu **ALTER** můžeme zašifrovat již vytvořený tablespace, avšak pokud se v tablespace nachází data, tak vyvoláme error [48].

Listing 2.4: Způsoby šifrování tablespace v Fujitsu Enterprise Postgres

```
SET tablespace_encryption_algorithm = 'AES256';
SET tablespace_encryption_algorithm = 'none';
CREATE TABLESPACE secure_tablespace LOCATION '/My/Data/Dir' WITH
    (tablespace_encryption_algorithm = 'AES256' );
ALTER TABLESPACE db_tablespace SET
    (tablespace_encryption_algorithm=AES256);
```

2.4 Percona Transparentní šifrování dat

Firma Percona, která vytváří i vlastní verzi PostgreSQL, se snaží vyvinout bezpečnou metodu transparentního šifrování pro PostgreSQL. Stejně jako předem zmíněná firma Cybertech se snaží zlepšit situaci v šifrování v PostgreSQL databázi. Percona vyvíjí rozšíření s otevřeným kódem s názvem *PG_TDE*, které je dostupné v repozitáři, nebo je ke stažení na stránkách firmy Percona. Přestože je rozšíření vyvíjeno krátce, lze implementovat na verzi Percona Postgres a vanilla verzi PostgreSQL. Funkce jako transparentní šifrování vyžaduje důkladné testování a delší čas na vývoj, kterého v době psaní zatím nebylo dosaženo, tudíž časem může přibýt tomuto rozšíření nespočet funkcí, tedy *PG_TDE* je aktuálně vyvíjeno příliš krátkou dobu [49].

PG_TDE je rozšíření šifrování dat mimo využití pro PostgreSQL. Data jsou šifrována pomocí AES-CTR na úrovni souborového systému dle dokumentace Percony. Rozšíření ve své aktuální verzi šifruje [50]:

- Data v tabulkách, TOAST (The Oversized-Attribute Storage Technique) tabulky vytvořené tímto rozšířením.
- Data protokolu WAL.
- Veškeré dočasné tabulky během operací databáze.

Rozšíření v aktuální verzi vykazuje omezení:

- Klíče v lokálním úložišti nejsou šifrovány.
- Indexy a bitové mapy Null tuplů nejsou šifrovány.
- Metadata dat tabulek a TOAST tabulek jsou nezašifrovány.
- Logická replikace není dostupná u zašifrovaných tabulek.

Percona varuje, že jejich způsob šifrování má dopad na výkon databáze, který odpovídá zpomalení menšímu než 10 %. V následujících verzích je plán rozšířit rozšíření o rotace hlavních klíčů, logickou replikaci pro zašifrované tabulky, podporu více uživatelů, šifrování indexů a bitových map Null tuplů.

PG_TDE podporuje dva typy managementu klíčů. První metoda spočívá v ukládání master klíče do vlastního souborového systému, což nebývá doporučovaný postup, protože klíč poté není bezpečně uložen a může být zneužit. Ukládaní master klíče do souborového systému bývá používáno pouze pro testování rozšíření, jelikož je nejjednodušší na implementaci. Druhá metoda implementace klíčového managementu je HashiCorp Vault server. HashiCorp Vault lze využít jak lokálně, tak v podobě cloudu. Bezpečně uložený master klíč zašifrovává ostatní klíče, které šifrují tabulky dat. Pro každou tabulku je náhodně a automaticky vytvořen nový klíč, tudíž při každé funkci vložení nebo načtení dat je specifický klíč pro danou tabulku dešifrován a pomocí dešifrovaného klíče je dešifrována tabulka dat [50].

Rozšíření na transparentní šifrování od Percony je implementovatelné na jejich Percona Distribution for PostreSQL 16 a upstream PostgreSQL 16. Před instalací je potřeba nainstalovat knihovny make gcc libjson-c-dev postgresql-server-dev-16 libcurl4-openssl-dev. Ve výpisu 2.5 je popsána instalace rozšíření, v níž si jej naklonujeme z Github, změníme adresář na adresář rozšíření, provedeme konfiguraci a instalaci. Při instalaci PostgreSQL do nestandardní složky je třeba změnit v *PG_CONFIG* konfiguraci rozšíření PG_TDE adresu na aktuální adresář PostgreSQL označený *pg_config*.

Listing 2.5: Instalace PG_TDE

```
git clone https://github.com/Percona-Lab/pg_tde.git
cd pg_tde
./configure
make USE_PGXS=1
sudo make USE_PGXS=1 install
```

Nastavení PG_TDE vyžaduje přidání parametru v psql a následovné restartování PostgreSQL `ALTER SYSTEM SET shared_preload_libraries = 'pg_tde';`. Rozšíření musí být následně vytvořeno pomocí `CREATE EXTENSION pg_tde;` v psql. Použití příkazu pro vytvoření rozšíření vyžaduje oprávnění superuživatele nebo vlastníka databáze. Rozšíření je nutné vytvořit pro každou databázi, pokud chceme automaticky nechat šifrovat každou nově vytvořenou databázi, můžeme příkazem `psql -d template1 -c 'CREATE EXTENSION pg_tde;` upravit databázi *template1*.

Posledním krokem k nastavení šifrování je třeba nastavit konfiguraci klíčů. Do souboru *postgresql.conf* musíme přidat parametr s umístěním souboru s klíčem `pg_tde.keyringConfigFile = '/path/to/the/keyring.json'`, vytvořit jej v daném adresáři, přidat práva vlastnictví správci databáze a restartovat databázi [50].

2.5 pgbench

Nástroj pgbench je benchmarkový nástroj, který je součástí PostgreSQL. Pgbench umožňuje provádět zátěžové testy pomocí simulace provozu databáze, podle nastavené konfigurace, jako například počet relací připojených na databázi. Přístup pgbench je vytvořen podle testu TCP-B. Měření je založeno na průměrném výpočtu transakcí za sekundu. Ve výchozím nastavení zahrnuje jedna transakce nástroje pgbench 5 příkazů skládajících se z `SELECT`, `UPDATE` a `INSERT`. Pgbench nabízí možnost testovat dle svých vlastních skriptů, což nebylo využito pro toto měření. Výstup programu měření pgbench vypadá následovně 2.6:

Listing 2.6: Výpis měření pgbench

```
1  postgres@UbuntuVanilla:/home/vboxuser$ pgbench -c 12 -j 4 -T 10
   ↪  pgbenchTable
2  pgbench (16.2 (Ubuntu 16.2-1.pgdg22.04+1))
3  starting vacuum...end.
4  transaction type: <builtin: TPC-B (sort of)>
5  scaling factor: 500
6  query mode: simple
7  number of clients: 12
8  number of threads: 4
9  maximum number of tries: 1
10 duration: 10 s
11 number of transactions actually processed: 6375
12 number of failed transactions: 0 (0.000%)
13 latency average = 18.849 ms
14 initial connection time = 21.874 ms
15 tps = 636.648483 (without initial connection time)
```

Prvních deset řádků ve výpisu 2.6 popisuje důležité údaje nastavení měření. Jedenáctý řádek řádek informuje o počtu provedených transakcí během délky testu, jenž v tomto případě trval 10 s. Poslední řádek výpisu je nejdůležitější, protože udává výstup celého testu v podobě transakcí za sekundu.

Testování TCP-B transakčních testů vyžaduje vytvořit předem specifické tabulky pro testování. Můžeme spustit pgbench s `-i` (initialize) inicializační možností a specifikovat pomocí `-s` (scale) velikost databáze. Výchozí velikost databáze při nespecifikování je 16 MB, což odpovídá scale 1, každý další bod velikosti odpovídá

zhruba dalším 16 MB. Výpis 2.7 ukazuje, jak by vypadala inicializace databáze pro testování pgbench o definované velikosti scale 50 s 5000000 řádky dat.

Listing 2.7: Vytvoření a naplnění databáze daty

```
postgres@UbuntuVanilla:~$ createdb pgbenchDatabase
postgres@UbuntuVanilla:~$ pgbench -i -s 50 pgbenchDatabase
dropping old tables...
NOTICE:  table "pgbench_accounts" does not exist, skipping
NOTICE:  table "pgbench_branches" does not exist, skipping
NOTICE:  table "pgbench_history" does not exist, skipping
NOTICE:  table "pgbench_tellers" does not exist, skipping
creating tables...
generating data (client-side)...
5000000 of 5000000 tuples (100%) done (elapsed 12.20 s, remaining
↪  0.00 s)
vacuuming...
creating primary keys...
done in 20.18 s (drop tables 0.01 s, create tables 0.03 s,
↪  client-side generate 12.42 s, vacuum 0.29 s, primary keys 7.43
↪  s).
             List of relations
 Schema |       Name       | Type  |  Owner
--------+------------------+-------+----------
 public | pgbench_accounts | table | postgres
 public | pgbench_branches | table | postgres
 public | pgbench_history  | table | postgres
 public | pgbench_tellers  | table | postgres
```

Program pgbench provádí skripty dle náhodných hodnot a specifikací pro tvar daného testu. Výchozí skript (upřesněný pomocí -b) má tvar sedmi příkazů na jednu transakci. Hodnoty v transakci jako *aid, tid, bid a delta* jsou náhodně nastaveny pro každou transakci. Test je inspirován benchmarkem podle TCP-B testu, avšak se o něj nejedná. Pomocí specifikace -S (nebo select-only) lze specifikovat, zda ve skriptu bude prováděn pouze příkaz na výběr dat.

Listing 2.8: Transakční skript pgbench

```
1 BEGIN;
2 UPDATE pgbench_accounts SET abalance = abalance + :delta WHERE aid =
  ↪ :aid;
3 SELECT abalance FROM pgbench_accounts WHERE aid = :aid;
4 UPDATE pgbench_tellers SET tbalance = tbalance + :delta WHERE tid =
  ↪ :tid;
5 UPDATE pgbench_branches SET bbalance = bbalance + :delta WHERE bid =
  ↪ :bid;
6 INSERT INTO pgbench_history (tid, bid, aid, delta, mtime) VALUES
  ↪ (:tid, :bid, :aid, :delta, CURRENT_TIMESTAMP);
7 END;
```

2.6 Porovnání výkonu šifrovaných databází

Nedílnou součástí této práce je provedení měření pro srovnání typů šifrování. Jako reference pro zašifrované metody byl v měření porovnán nezašifrovaný standardní PostgreSQL, který byl srovnáván s transparentními šifrovacími metodami zmiňovanými výše, teda Cybertech TDE, rozšíření PG_TDE od Percony a TDE řešení od Fujitsu, které je součástí Fujitsu Enterprise Postgres. Šifrování PostgreSQL na úrovni disku bylo provedeno pomocí dm-crypt a LUKS.

Měření bylo provedeno na domácím zařízení, neboť nebylo k dispozici vhodnější zařízení ze strany školy. Procesor využitý pro měření je AMD Ryzen 5 4500U s 6 jádry a 6 thready. Použitý procesor disponuje podporou AES funkcí. Paměť systému je 16 GB DDR4 paměti v dual channelu. Databáze byly uloženy na úložišti KIOXIA SSD 256GB – M.2 PCIe Gen3 (NVMe). Pro měření byl využit virtualizovaný operační systém Ubuntu 22.04.4 s pamětí 8192 MB a 3 jádry s plným výkonem (systém s 3 z 6 jader produkoval nejvyšší výkon). Pro implementaci Fujitsu Enterprise Postgres musel být využit operační systém RHEL 9.0 (Red Hat Enterprise Linux), který byl podporován Fujitsu verzí Postgres, při stejném nastavení jako operační systém Ubuntu. Lze předpokládat, že změna operačního systému při měření má dopad na výkon databáze. Zkušební verze RHEL svým omezením nedovolila provést celé měření pouze na operačním systému RHEL.

Pro všechna provedená měření PostgreSQL databází bylo využito jednotné nastavení souboru *postgresql.conf* vypsané na 2.9, které bylo konfigurováno dle konfigurátoru [51]. Testy pomocí pgbench nástroje se mohou zdát velmi nepřesné, pokud jsou dělány nesprávně. Mezi doporučené praktiky patří dělat dlouhé testy pro vy-

rovnání vyšších a nižších výkonů databáze v časových periodách. Dlouhým testem je myšlen časový interval minimálně 30 minut až hodinu, což je rozsah nemožný pouze pro jeden test, proto měříme 300 testů. Testy byly měřeny v kraších intervalech 3 minut a jejich hodnota byla ověřována na několika testech s délkou 30 sekund, pokud hodnota byla podobná, byla zaznamenána hodnota minutového testu, při neshodě bylo dané měření prováděno znovu, dokud se výsledky neshodovaly.

Každý scale factor jednotlivých měřených databází byl iniciován dle výchozího nastavení. Pro měření byly zvoleny tři velikosti databází, které byly počítány dle přesné kalkulačky velikostí podle měření [52]. Velikost databáze se scale faktorem 50 odpovídá malé databázi o velikosti 786,4 MB, což představuje dle konfigurace 2.9 19,2 % velikosti nastavení *shared buffers*. Druhá zvolená databáze o velikosti 3932.16 MB se scale faktorem 250, která odpovídá středně malé databázi a odpovídá 96 % hodnoty *shared buffers*. Poslední měřená databáze o střední velikosti 7838.11 MB odpovídající scale faktoru 500, dosahovala hodnoty 191.36 % *shared buffers*, avšak velikost databáze se stále nachází pod hranicí RAM operačního systému.

Mezi doporučené praktiky testování pomocí TPC-B bývá využívat scale faktor minimálně velký, jako je počet simulovaných klientů připojených na databázi. Měření se zaměřuje na omezení výkonu při aktivním transparentním šifrování, což bude zobrazeno se zvyšující se poptávkou souběžnosti databáze. Souběžnost databáze je schopnost databáze současně provádět více požadavků zároveň, tudíž provést souběžný počet transakcí klientů bez jejich zahození pod stresem testu. Bylo provedeno 10 měření pro každou databázi od 12 klientů po 48 klientů.

Listing 2.9: Jednotná konfigurace postgresql.conf pro měření

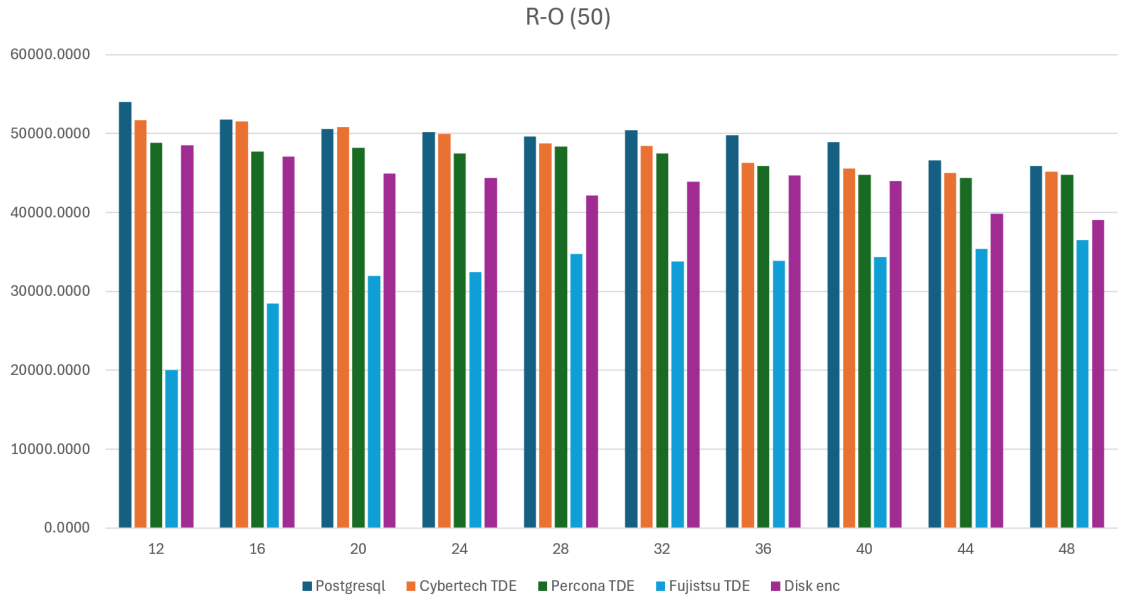
```
max_connections = 100
superuser_reserved_connections = 3
shared_buffers = '4096 MB'
work_mem = '32 MB'
maintenance_work_mem = '320 MB'
huge_pages = off
effective_cache_size = '6 GB'
effective_io_concurrency = 100
random_page_cost = 1.25
shared_preload_libraries = 'pg_stat_statements'
track_io_timing = on
track_functions = pl
wal_level = replica
max_wal_senders = 0
synchronous_commit = on
checkpoint_timeout = '15 min'
checkpoint_completion_target = 0.9
max_wal_size = '1024 MB'
min_wal_size = '512 MB'
wal_compression = on
wal_buffers = -1
wal_writer_delay = 200ms
wal_writer_flush_after = 1MB
bgwriter_delay = 200ms
bgwriter_lru_maxpages = 100
bgwriter_lru_multiplier = 2.0
max_worker_processes = 3
max_parallel_workers_per_gather = 2
max_parallel_maintenance_workers = 2
max_parallel_workers = 3
parallel_leader_participation = on
enable_partitionwise_join = on
enable_partitionwise_aggregate = on
jit = on
max_slot_wal_keep_size = '1000 MB'
track_wal_io_timing = on
maintenance_io_concurrency = 100
wal_recycle = on
```

2.6.1 Posouzení výsledků měření pro read only testy

Read only test se scale 50

Při měření read only testu na databázi malého rozměru byl rozdíl mezi transparentním šifrováním a nezašifrovanou databází minimální. Nejvyšší hodnota transakcí za sekundu byla naměřená na nezašifrované databázi při 12 připojených klientech s hodnotou 53989,35 tps, tato hodnota je na použitý hardware vynikající při virtualizaci, avšak neodpovídá hodnotám reálných serverů, které dosahují statisíců až milionů transakcí za sekundu.

Nejmenšího průměru rozdílů mezi nezašifrovanou a zašifrovanou databází dosahovalo transparentní šifrování od firmy Cybertech s hodnotou 97 %, jenž se v několika měření velmi blížilo hodnotám nezašifrované databáze, a jednou ji dokonce převýšilo, což bylo nejspíše způsobeno nepřesností měření. Druhý nejmenší rozdíl s nezašifrovanou databází má transparentní šifrování od firmy Percona, které dosahovalo také velmi zanedbatelných průměrných hodnot transakcí za sekundu a to 94 %. Nejnižšího průměrného výkonu 65 % dosahoval Fujitsu Enterprise Postgres, což nejspíše mohlo být způsobeno využitím jiného operačního systému. Průměrný rozdíl výkonu oproti nezašifrované databázi u šifrování v souborovém systému dosahoval 88 %, což se velmi blíží dvěma výkonným transparentním řešením. V trendu postupně zvyšujících se připojení lze vidět, že se počet transakcí snižuje, avšak u Fujitsu Enterprise Postgres se zvyšuje a to z 37 % na 79 % transakcí za sekundu nezašifrované databáze.



Obr. 2.3: Graf read only testu při scale faktor 50

Tab. 2.1: Porovnání výkonu PostgreSQL šifrování read only při scale 50

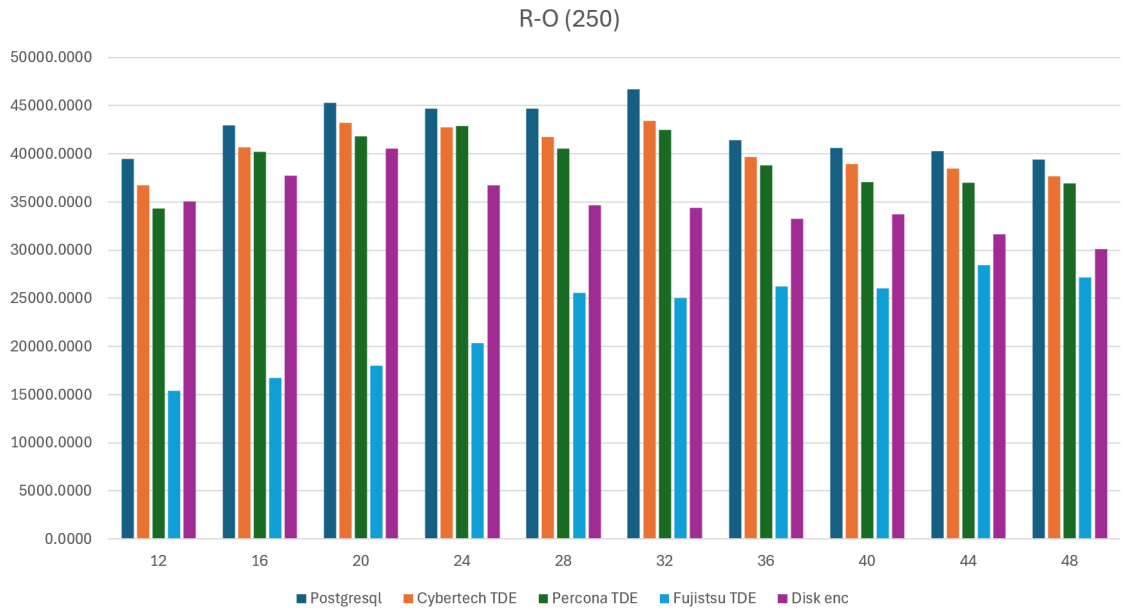
R-O (50)					
připojení	**PostgreSQL**	**Cybertech TDE**	**Percona TDE**	**Fujistsu TDE**	**Disk encryption**
12	53989.3518	51725.9227	48811.1399	19995.8487	48483.6411
16	51812.9028	51532.9764	47714.3924	28473.9776	47117.1215
20	50626.9621	50839.0110	48205.5198	31988.4687	44932.4240
24	50183.5142	49929.4874	47487.4903	32450.0926	44348.6516
28	49626.8836	48766.4194	48379.3862	34731.5768	42112.9336
32	50463.3872	48442.4816	47469.2388	33819.3561	43917.7356
36	49782.1065	46271.4963	45888.8329	33882.0011	44685.8038
40	48933.3961	45596.1547	44808.1089	34325.1149	43951.9929
44	46646.7079	44988.3765	44386.6015	35354.3552	39821.9247
48	45881.6793	45175.3954	44799.7040	36474.6525	39041.8171

Read only test se scale 250

Měření read only testu na databázi středně malého rozměru připomínalo výsledky dat z měření na databázi malého rozměru s výjimkou průběhu grafu. Na grafu lze vidět, že data dosáhly maxima 46706.06 tps při 32 připojených klientech, přičemž podobný trend se opakuje i u zašifrovaných databází, transakce za sekundu se postupně snižují s narůstajícím počtem připojených klientů. Tento pokles lze vysvětlit velikostí databáze, která se velmi přibližuje hodnotě *shared buffers*. Celkový pokles tps je způsoben zvětšením objemu databáze. Pro funkci čtení dat je těžší najít požadovanou hodnotu, což ukazuje na snížení několika tisíc tps. U Fujitsu Enterprise Postgres lze vidět postupný nárůst transakcí za sekundu stejně jako u scale 50. Šifrování v souborovém systému dosáhlo maxima při připojení 20 klientů a počet tps se následovně postupně mírně snižoval.

Šifrované databáze dosahovaly následujícího rozdílu transakcí za sekundu v porovnání s nezašifrovanou:

- Nejlepšího výsledku dosahoval PostgreSQL od Cybertech a to 95 %.
- Velmi nízký rozdíl tps 92 % byl naměřen u Percona TDE rozšíření.
- Šifrování v souborovém systému mělo rozdíl 81 %.
- Postupný nárůst u Fujitsu Enterprise Postgres byl naměřen od 38 % do 71 %.



Obr. 2.4: Graf read only testu při scale faktor 250

Tab. 2.2: Porovnání výkonu PostgreSQL šifrování read only při scale 250

R-O (250)					
připojení	**PostgreSQL**	**Cybertech TDE**	**Percona TDE**	**Fujistsu TDE**	**Disk encryption**
12	39495.7840	36734.7907	34312.0897	15366.5703	35030.8515
16	42948.5826	40679.1046	40186.9366	16712.1081	37725.7172
20	45320.0253	43234.0533	41800.5536	17976.9091	40571.6754
24	44692.2538	42759.1270	42899.4222	20370.2649	36736.1850
28	44684.8079	41775.8894	40527.1060	25572.1490	34645.9334
32	46706.0619	43421.7157	42499.6883	25044.9393	34370.1074
36	41415.3714	39677.9244	38793.9799	26196.6013	33244.5178
40	40614.6092	38966.3085	37064.2709	26021.0040	33712.9966
44	40259.8811	38474.4203	37022.4745	28460.7083	31680.5171
48	39392.6914	37647.5929	36905.0281	27166.7492	30098.7995

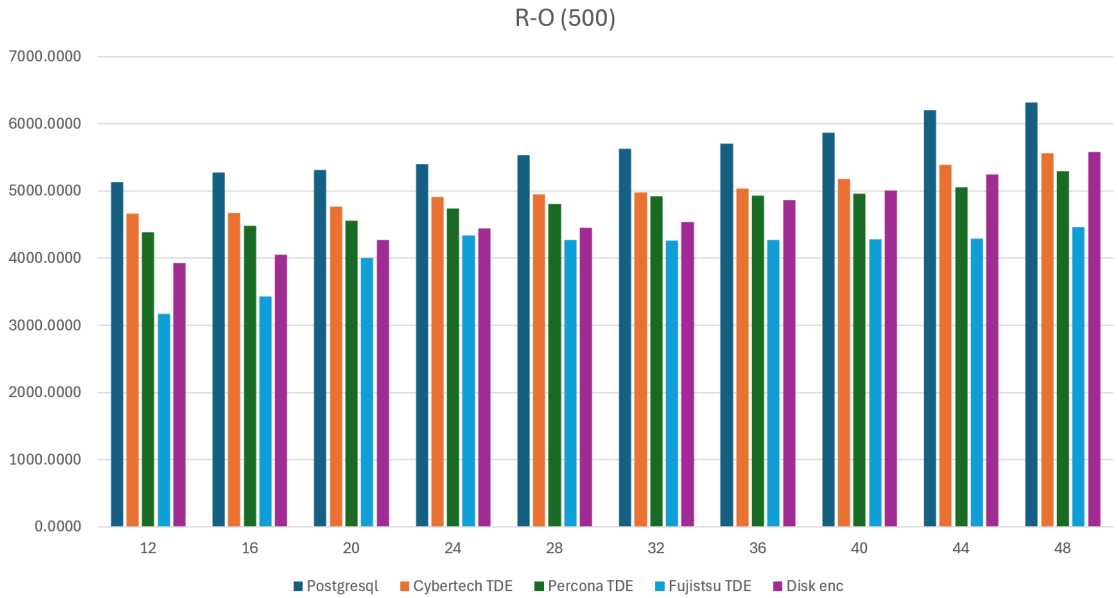
Read only test se scale 500

Poslední měření 2.3 bylo provedeno na databázi o střední velikosti scale faktor 500, kde můžeme vidět velký propad ve výkonu databáze. Propad výkonu je způsoben velikostí databáze v poměru s velikostí RAM pamětí operačního systému. Využití paměti SSD disku je mnohem pomalejší než pouze provedení operace v rozměru operační paměti systému. Každý dotaz má velkou latenci, tudíž pokud bychom

chtěli dosáhnout vyšších tps, museli bychom zvýšit počet připojených klientů. Na disku SSD lze dosáhnout vyšších transakcí za sekundu pomocí navýšení souběžnosti. Z tohoto důvodu lze vidět na grafu 2.5 to, že hodnota tps se zvyšujícím počtem klientů postupně stoupá.

U šifrovaných databází se projevil větší průměrný rozdíl transakcí za sekundu v porovnání s nezašifrovanou než u měření s menším objemem dat:

- PostgreSQL od Cybertech se svým transparentním šifrováním dosahoval v průměru 88 %.
- Rozšíření Percona TDE dosahovala 85 % výkonu nezašifrované.
- Šifrování v souborovém systému dosahovalo skoro výkonu transparentních řešení a to 82 %.
- Fujitsu Enterprise Postgres dosáhl 72 % výkonu.



Obr. 2.5: Graf read only testu při scale faktor 500

Tab. 2.3: Porovnání výkonu PostgreSQL šifrování read only při scale 500

R-O (500)					
připojení	**PostgreSQL**	**Cybertech TDE**	**Percona TDE**	**Fujistsu TDE**	**Disk encryption**
12	5130.8274	4663.1363	4383.6310	3168.9409	3928.9989
16	5278.7842	4671.1500	4478.4904	3428.1212	4046.8572
20	5313.5948	4767.0365	4554.1940	3999.4470	4272.1779
24	5396.8057	4916.5062	4744.4472	4339.8447	4439.1774
28	5531.4312	4955.1256	4809.4388	4275.2136	4453.2789
32	5634.0771	4976.8097	4923.8141	4266.7002	4538.4668
36	5701.4670	5032.2289	4936.1393	4274.7002	4867.4574
40	5868.2674	5178.8167	4964.4982	4283.9745	5007.5229
44	6198.7806	5393.4972	5056.0512	4295.3010	5245.4905
48	6316.6591	5566.4175	5296.5878	4464.6822	5583.2835

2.6.2 Posouzení výsledků měření pro read write testy

Read write test se scale 50

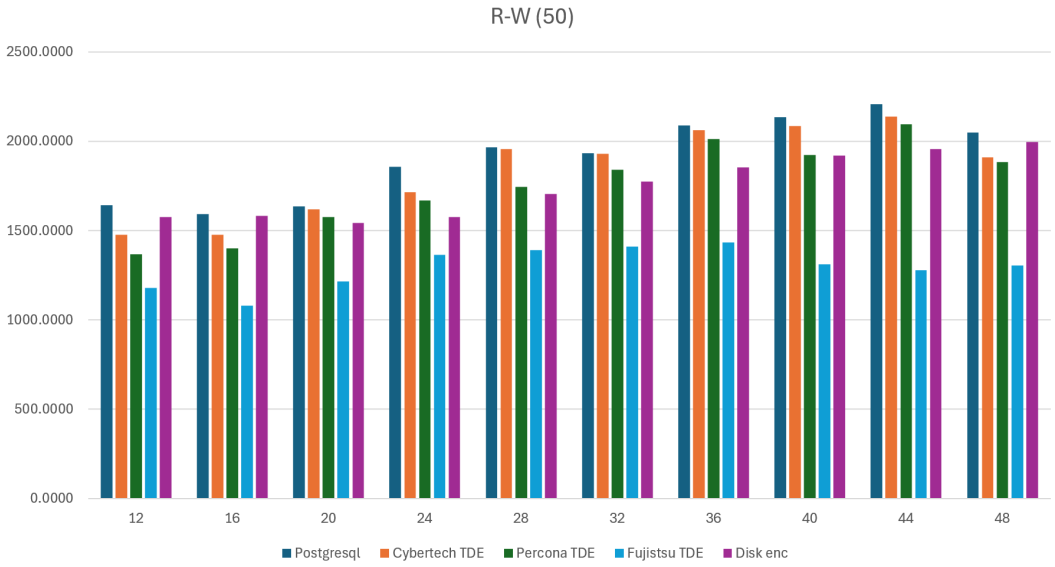
Při měření read write testu je třeba brát v potaz, že jedna výchozí read write transakce obsahuje 5 příkazů, tudíž je každá transakce náročnější pro zpracování. Změny na disku také více zatěžují systém, protože musí být využit fsync pro kontrolu zapsání fyzických dat a nejedná se o pouhé uložení do registru dat. Prováděním této synchronizace dokáže PostgreSQL zajistit, aby změny na disku byly provedeny i během havárie databáze. Z těchto dvou důvodů je počet transakcí nižší než u only read testů.

Read write testy by teoreticky měly mít větší rozdíl tps mezi zašifrovanou a nezašifrovanou databází, což se při měření neprokázalo. Rozdíl ve výkonu by měl být větší jak 10 %. Za menší rozdíl tps může mít nastavení parametru shared_buffers v souboru postgre.conf. Vyšší hodnota shared_buffers zvyšuje paměť databáze pro šifrování a dešifrování, z čehož transparentní šifrovaní těží větší výkon. Zvolená hodnota shared_buffers v postgre.conf je 50 % RAM, což může snížit potenciál výkonu nezašifrované PostgreSQL databáze nebo databázím šifrovaným souborovým systémem.

Rozdíl transakcí za sekundu v průměru mezi zašifrovanou a nezašifrovanou byl naměřen:

- Nejvyššího výkonu dosáhl Cybertech TDE 96 %, který si udržel během všech měření nejvyšší výkon.

- Stejného rozdílu dosáhl Percona TDE a šifrování souborovým systémem pomocí dm-crypt s 91 %.
- Opakovaně nejnižšího výkonu dosáhlo Fujitsu transparentní šifrování s 68 % rozdílu.



Obr. 2.6: Graf read write testu při scale faktor 50

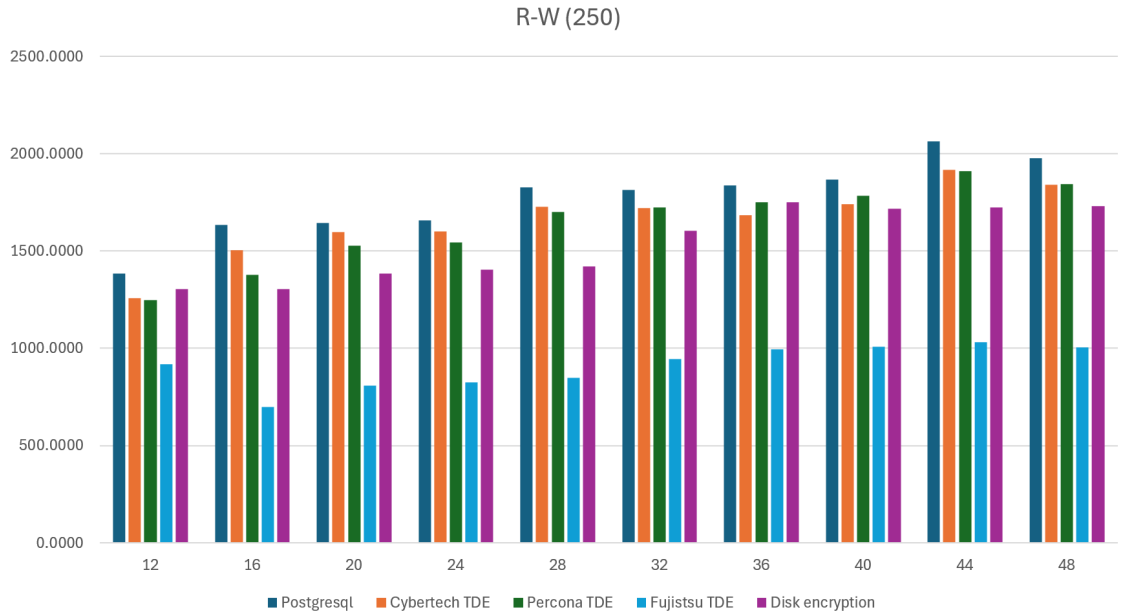
Tab. 2.4: Porovnání výkonu PostgreSQL šifrování read write při scale 50

R-W (50)					
připojení	**PostgreSQL**	**Cybertech TDE**	**Percona TDE**	**Fujistsu TDE**	**Disk encryption**
12	1640.3565	1475.8013	1367.4535	1179.3859	1574.3597
16	1592.0621	1476.7342	1399.1522	1080.5687	1582.3728
20	1634.6305	1618.2575	1575.8511	1216.2267	1543.5997
24	1856.1296	1714.5173	1666.7039	1364.4814	1576.7888
28	1967.0372	1954.3813	1743.8028	1390.9083	1704.5038
32	1933.3903	1930.6709	1838.9296	1408.7394	1774.7530
36	2087.1685	2060.8575	2011.9358	1434.4371	1851.7724
40	2133.5661	2085.7303	1924.1847	1312.3701	1920.8752
44	2205.5741	2137.6564	2095.3747	1278.2431	1956.5047
48	2047.1889	1910.7579	1883.7020	1305.5688	1997.0797

Read write test se scale 250

Read write test databází středně malé velikosti databáze může opětovně velmi připomínat výsledky testů na malé databázi. Rozdíly nezašifrované PostgreSQL databáze oproti zašifrovaným byly následující:

- Cybertech TDE dosáhl opětovně nejvyššího výkonu 93 %.
- Transparentní rozšíření Percony dosáhlo 92 % transakcí.
- Dm-crypt šifrování souborového systému dosáhlo 87 %.
- Nejnižšího výkonu ze všech měření dosáhl Fujitsu Enterprise Postgres s 51 % a minimem 42 % s 16 připojenými klienty.



Obr. 2.7: Graf read write testu při scale faktor 250

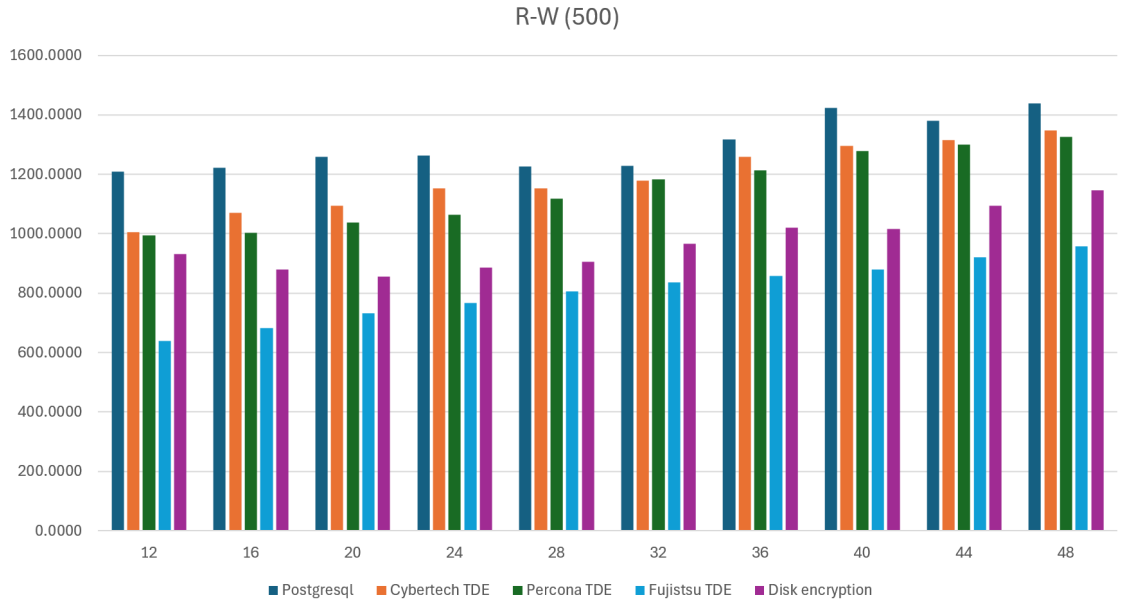
Tab. 2.5: Porovnání výkonu PostgreSQL šifrování read write při scale 250

R-W (250)					
připojení	**PostgreSQL**	**Cybertech TDE**	**Percona TDE**	**Fujistsu TDE**	**Disk encryption**
12	1383.6610	1258.3900	1247.6181	918.3037	1303.9390
16	1632.6387	1504.9129	1377.3167	697.4172	1303.5605
20	1643.4142	1598.4794	1525.9670	809.2387	1384.6777
24	1657.7027	1600.4379	1544.4315	824.4393	1405.2371
28	1825.2001	1725.9534	1700.2068	849.5297	1420.8712
32	1815.1695	1720.2650	1722.4286	945.3251	1605.3857
36	1837.5709	1682.5190	1750.6601	993.3391	1750.8464
40	1865.3540	1738.8339	1784.5650	1006.9769	1718.3765
44	2063.8466	1915.7843	1909.4576	1029.8752	1724.1126
48	1975.1260	1838.6541	1842.8111	1003.1410	1730.4510

Read write test se scale 500

Read write test na databázi o střední velikosti se scale faktorem mělo dle grafu 2.8 velmi lineární vývin během měření oproti ostatním průběhu v grafu. U read write testu se neprojevil velký propad ve výkonu jako u read only testu, akorát škálování transakcí za sekundu při navýšení připojených klientů je menší. Byl vypočítán největší rozdíl zašifrovaných databází oproti nezašifrované databázi:

- Cybertech TDE měl vývoj z 83 % až na 95 % s celkovým průměrem 91 %.
- Transparentní rozšíření Percony dosáhlo 88 % rozdílu oproti nezašifrované databázi.
- Transparentní šifrování souborového systému dosáhlo pouze 74 %.
- Opětovně nejnižšího výkonu oproti jiným druhům transparentního šifrování dosáhl Fujitsu Enterprise Postgres s 62 %.



Obr. 2.8: Graf read write testu při scale faktor 500

Tab. 2.6: Porovnání výkonu PostgreSQL šifrování read write při scale 500

R-W (500)					
připojení	**PostgreSQL**	**Cybertech TDE**	**Percona TDE**	**Fujistsu TDE**	**Disk encryption**
12	1209.0835	1005.0306	994.9364	639.3929	930.7822
16	1222.1897	1070.5878	1003.1529	683.2006	880.0031
20	1257.6703	1092.9517	1038.1348	732.9342	856.2378
24	1263.8642	1153.1954	1063.4576	766.8702	884.8709
28	1226.8240	1153.3716	1116.9294	805.5926	905.8545
32	1228.3378	1177.6034	1182.0976	836.4057	965.7928
36	1316.4249	1259.1018	1213.3640	858.7673	1020.9930
40	1423.0472	1296.0243	1279.0304	880.4748	1016.1414
44	1378.8589	1313.8590	1300.3894	920.7673	1092.9200
48	1438.8733	1347.3182	1325.6435	956.9105	1145.4174

Závěr

V této bakalářské práci jsme se zaměřili na problematiku šifrování v databázovém systému PostgreSQL. V teoretické části byly plně vysvětleny základní pojmy spojené s odbornou terminologií vztahující se k databázím jakožto termíny, databáze, software pro správu databáze (DBMS), jazyk Structured query language a 5 typů dotazů jazyku. Dále byly popsány čtyři aktuálně frekventovaně využívané šifrovací algoritmy, postup zašifrování dat i jejich silné a slabé stránky. Pro pochopení šifrovacího schématu byla popsána problematika typů klíčů, management šifrovacích klíčů, dat mimo využití a dat přenášených po síti. Následně byla provedena analýza pěti nejoblíbenějších databázových systémů jako Oracle, MySQL, Microsoft SQL Server, PostgreSQL a MongoDB a vysvětleny jejich bezpečnostní funkce. Byly rozlišeny různé typy šifrování na základě způsobu šifrování, pro jednotlivé typy bylo poukázáno na jejich silné stránky a využití.

V průběhu práce byla provedena implementace tří nástrojů pro nastavení transparentního šifrování v PostgreSQL. Při průzkumu možností transparentního šifrování bylo zjištěno, že dle názoru vývojářů PostgeSQL není transparentní šifrování potřeba, tudíž PostgreSQL neposkytuje možnost transparentního šifrování na úrovni databáze, avšak lze PostgreSQL databázi šifrovat transparentně na úrovni souborového systému, což může poskytnout minimální ochranu pro data mimo využití. Další z možností pro šifrování PostgreSQL je podle modulu pgcrypto, který funguje na úrovni sloupců v relační tabulce, nikoli na logovací soubory a celou databázi.

Jednotlivé nástroje pro transparentní šifrování byly popsány a byl uveden stručný popis implementace. Pro řešení firmy Cybertech byl napsán skript pro instalaci. Pro porovnání výkonů šifrovaných databází bylo provedeno měření transakcí za sekundu. Bylo vysvětleno chování změny výkonu databází při měnících se velikostech databází a vzrůstajícím počtu připojených klientů. V průměru výsledků napříč více velikostmi databází v experimentálním prostředí transparentního šifrování na úrovni databáze dosahovalo lepšího výkonu i lepšího zabezpečení než šifrování souborového systému, avšak jednotné nastavení konfigurace databází mohlo přidávat výkon šifrování na úrovni databáze.

V průměru všech měření v porovnání s nezašifrovanou databází PostgreSQL dosahoval nejlepšího výsledku 93 % transakcí za sekundu Cybertech TDE. Rozšíření PG_TDE od firmy Percona dosáhlo 90 % výkonu nezašifrované databáze a šifrování souborovým systémem dosáhlo 84 %. Nejnižšího výkonu dosáhl Fujitsu Enterprise Postgres s 62 %, což mohlo být způsobeno problémy s virtualizací systému.

Literatura

[1] *DB-Engines Ranking*. Online. 2023. Dostupné z: https://db-engines.com/en/ranking. [cit. 2023-10-30].

[2] CYBERTEC. *POSTGRESQL TRANSPARENT DATA ENCRYPTION*. Online. 2020. Dostupné z: https://www.cybertec-postgresql.com/en/products/postgresql-transparent-data-encryption/. [cit. 2024-05-06].

[3] ORACLE. *What Is a Database?* Online. 2021. Dostupné z: https://www.oracle.com/database/what-is-database/. [cit. 2023-11-28].

[4] ALTEXSOFT. *Comparing Database Management Systems: MySQL, PostgreSQL, MSSQL Server, MongoDB, Elasticsearch, and others*. Online. 2023. Dostupné z: https://www.altexsoft.com/blog/business/comparing-database-management-systems-mysql-postgresql-mssql-server-mongodb-elasticsearch-and-others/. [cit. 2023-10-30].

[5] ORACLE. *History of SQL*. Online. 2003. Dostupné z: https://docs.oracle.com/cd/B13789_01/server.101/b10759/intro001.htm. [cit. 2024-05-03].

[6] W3SCHOOLS. *SQL History*. Online. Dostupné z: https://www.w3schools.in/sql/history. [cit. 2023-11-28].

[7] PETERSON, Richard. *SQL Commands: DML, DDL, DCL, TCL, DQL with Query Example*. Online. 2023, 2023-10-3. Dostupné z: https://www.guru99.com/sql-commands-dbms-query.html. [cit. 2023-11-28].

[8] STINE, Kevin a DANG, Quynh. *Journal of AHIMA (American Health Information Management Association)*. Online. 2011. Dostupné z: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=908084. [cit. 2023-10-11].

[9] KASTNING, Jörg. *Basic concepts of encryption in cryptography*. Online. 2021. Dostupné z: https://www.redhat.com/sysadmin/basic-concepts-encryption-cryptography. [cit. 2023-10-14].

[10] STALLINGS, William. *Cryptography and Network Security Principles and Practices*. Fourth Edition. Prentice Hall, 2005. ISBN 0-13-187319-9.

[11] BOUGANIM, Luc a GUO, Yanli. Database Encryption. Online. *ResearchGate*. 2010, s. 10. Dostupné z: https://doi.org/10.1007/978-1-4419-5906-5_677. [cit. 2023-10-28].

[12] JENA, Baivab Kumar. *What Is AES Encryption and How Does It Work?* Online. 2023, 9. 2. 2023. Dostupné z: https://www.simplilearn.com/tutorials/cryptography-tutorial/aes-encryption. [cit. 2024-03-05].

[13] SHAH, Javed. *What Is AES Encryption? The Complete Guide*. Online. 2023, 25.4.2023. Dostupné z: https://www.1kosmos.com/authentication/aes-encryption/. [cit. 2024-03-05].

[14] SIMPLILEARN. *What Is RSA Algorithm and How Does It Work in Cryptography?* Online. 2023, 13. 2. 2023. Dostupné z: https://www.simplilearn.com/tutorials/cryptography-tutorial/rsa-algorithm. [cit. 2024-03-05].

[15] N-ABLE. *Types of Database Encryption Methods*. Online. 2019, 2019-5-10. Dostupné z: https://www.n-able.com/blog/types-database-encryption-methods. [cit. 2023-10-14].

[16] KIDD, Chrissy. *Data Encryption Methods & Types: Beginner-s Guide To Encryption*. Online. 2022. Dostupné z: https://www.splunk.com/en_us/blog/learn/data-encryption-methods-types.html. [cit. 2023-10-14].

[17] LAKE, Josh. *What is 3DES encryption and how does DES work?* Online. 2023, 31. 8. 2023. Dostupné z: https://www.comparitech.com/blog/information-security/3des-encryption/. [cit. 2024-03-05].

[18] KALISKI, Burt. *TWIRL AND RSA KEY SIZE*. Online. 2003, 6. 5. 2023. Dostupné z: https://web.archive.org/web/20170417095741/https://www.emc.com/emc-plus/rsa-labs/historical/twirl-and-rsa-key-size.htm. [cit. 2023-10-14].

[19] TOBIN, Donal. *Which Modern Database Is Right For Your Use Case?* Online. 2023. Dostupné z: https://www.integrate.io/blog/which-database/. [cit. 2023-10-30].

[20] ORACLE. *Introduction to Oracle Database Security*. Online. 2023. Dostupné z: https://docs.oracle.com/en/database/oracle/oracle-database/19/dbseg/introduction-to-oracle-database-security.html#GUID-41040F53-D7A6-48FA-A92A-0C23118BC8A0. [cit. 2023-10-30].

[21] SATORI. *MySQL Security — Common Threats And 8 Best Practices*. Online. 2022. Dostupné z: https://satoricyber.com/mysql-security/mysql-security-common-threats-and-8-best-practices/. [cit. 2023-10-30].

[22] ORACLE. *Security Guidelines*. Online. 2023. Dostupné z: https://dev.mysql.com/doc/mysql-security-excerpt/8.0/en/security-guidelines.html. [cit. 2023-10-30].

[23] CONSTANTIN, Lucian. *Why you should review the security of your MSSQL servers*. Online. 2023. Dostupné z: https://www.csoonline.com/article/643561/why-you-should-review-the-security-of-your-mssql-servers.html. [cit. 2023-10-30].

[24] SATORI. *SQL Server Security*. Online. 2021. Dostupné z: https://satoricyber.com/sql-server-security/sql-server-security/. [cit. 2023-10-30].

[25] MICROSOFT. *Securing SQL Server*. Online. 2023. Dostupné z: https://learn.microsoft.com/en-us/sql/relational-databases/security/securing-sql-server?view=sql-server-ver16. [cit. 2023-10-30].

[26] PAGE, Dave. *How to Secure PostgreSQL: Security Hardening Best Practices & Tips*. Online. 2023. Dostupné z: https://www.enterprisedb.com/blog/how-to-secure-postgresql-security-hardening-best-practices-checklist-tips-encryption-authentication-vulnerabilities. [cit. 2023-10-30].

[27] POSTGRESQL. *Security*. Online. 2000. Dostupné z: https://www.postgresql.org/docs/7.0/security.htm. [cit. 2023-10-30].

[28] SATORI. *3 Pillars of PostgreSQL Security*. Online. 2021. Dostupné z: https://satoricyber.com/postgres-security/3-pillars-of-postgresql-security/. [cit. 2023-10-30].

[29] MONGODB. *Encryption at Rest*. Online. 2023. Dostupné z: https://www.mongodb.com/docs/manual/core/security-encryption-at-rest/. [cit. 2023-10-30].

[30] MONGODB. *Strong Security Defaults. Protect your workloads confidently*. Online. 2023. Dostupné z: https://www.mongodb.com/products/capabilities/security. [cit. 2023-10-30].

[31] MONGODB. *Security Checklist*. Online. 2023. Dostupné z: https://www.mongodb.com/docs/manual/administration/security-checklist/. [cit. 2023-10-30].

[32] CHERRY, Denny. *Securing SQL Server: Protecting Your Database from Attackers*. 3rd ed. Syngress Media,U.S., 2015. ISBN 978-0-12-801275-8.

[33] AKEYLESS. *Data at Rest vs. Data in Transit*. Online. 2023. Dostupné z: https://www.akeyless.io/secrets-management-glossary/data-at-rest-vs-data-in-transit/. [cit. 2023-10-30].

[34] AHMED, Ibrar. *Transparent Data Encryption (TDE)*. Online. 2022. Dostupné z: https://www.percona.com/blog/transparent-data-encryption-tde/. [cit. 2023-12-12].

[35] STIEGLITZ, Jeremy. *Encryption: Pros and Cons*. Online. 2017. Dostupné z: https://www.imperva.com/blog/archive/encryption-pros-and-cons/. [cit. 2023-12-12].

[36] *Šifrování dat*. Online. In: Wikipedia: the free encyclopedia. San Francisco (CA): Wikimedia Foundation, 2001-. Dostupné z: https://cs.wikipedia.org/wiki/Åãifrovãąnãŋ_dat. [cit. 2023-10-14].

[37] *Filesystem-level encryption (fscrypt)*. Online. Dostupné z: https://www.kernel.org/doc/html/latest/filesystems/fscrypt.html. [cit. 2023-12-12].

[38] BAXTER, Brendon. *File-based encryption vs full-disk encryption*. Online. 2022. Dostupné z: https://www.hexnode.com/blogs/file-based-encryption-vs-full-disk-encryption/. [cit. 2023-12-12].

[39] *Disk encryption*. Online. In: Wikipedia: the free encyclopedia. San Francisco (CA): Wikimedia Foundation, 2001. Dostupné z: https://en.wikipedia.org/wiki/Disk_encryption. [cit. 2023-10-14].

[40] *BitLocker*. Online. In: Wikipedia: the free encyclopedia. San Francisco (CA): Wikimedia Foundation, 2001-. Dostupné z: https://en.wikipedia.org/wiki/BitLocker. [cit. 2023-12-12].

[41] PILYANKEVICH, Eugene. *Application Level Encryption for Software Architects*. Online. 2020. Dostupné z: https://www.infoq.com/articles/ale-software-architects/. [cit. 2023-12-12].

[42] ZAITSEV, Peter. *Does PostgreSQL Need TDE (Transparent Data Encryption) ?* Online. 2022. Dostupné z: Twitter Inc., https://twitter.com/PeterZaitsev/status/1584557364845637633. [cit. 2023-12-02].

[43] *Database Encryption*. Online. In: Wikipedia: the free encyclopedia. San Francisco (CA): Wikimedia Foundation, 2001-. Dostupné z: https://en.wikipedia.org/wiki/Database_encryption. [cit. 2023-10-30].

[44] CYBERTEC. *Transparent Data Encryption (TDE) is a CYBERTEC encryption patch for PostgreSQL*. Online. 2017, 30. 5. 2023. Dostupné z: `https://www.cybertec-postgresql.com/en/products/postgresql-transparent-data-encryption/`. [cit. 2023-12-04].

[45] MUSTAFEEZ, Anusheh Zohair. *What is CTR?* Online. 2023. Dostupné z: `https://www.educative.io/answers/what-is-ctr`. [cit. 2023-11-27].

[46] FUJITSU. *Fujitsu Enterprise Postgres - Overview*. Online. 2023. Dostupné z: `https://www.postgresql.fastware.com/fep-overview`. [cit. 2024-04-17].

[47] FUJITSU. *Enhanced data security with Fujitsu Enterprise Postgres*. Online. 2023. Dostupné z: `https://www.postgresql.fastware.com/enhanced-security-for-enterprises`. [cit. 2024-04-17].

[48] *FUJITSU Enterprise Postgres 12 on IBM LinuxONE: Operation Guide*. Online. 2020. FUJITSU, 2020. Dostupné z: `https://fast.fujitsu.com/hubfs/_Global/Manuals/V12onZ-OperationGuide.pdf`. [cit. 2024-04-17].

[49] STOKES, David. *Percona PG_TDE and Docker*. Online. 2024. Dostupné z: `https://www.percona.com/blog/percona-pg_tde-and-docker-please-test-transparent-data-encryption-for-postgresql/`. [cit. 2024-04-17].

[50] PERCONA. *Pg_tde documentation*. Online. 2023. Dostupné z: `https://percona-lab.github.io/pg_tde/main/tde.html`. [cit. 2024-04-17].

[51] CYBERTECH. *PostgreSQL Configurator*. Online. 2017. Dostupné z: `https://pgconfigurator.cybertec.at`. [cit. 2024-04-17].

[52] JSFIDDLE. *Pgbench-size-to-scale*. Online. 2018. Dostupné z: `https://jsfiddle.net/kmoppel/6zrfwbas/`. [cit. 2024-04-17].

Seznam symbolů a zkratek

DBMS	systém pro správu databáze – Database Management System
SQL	standardizovaný strukturovaný dotazovací jazyk – Structured Query Language
SEQUEL	anglický standardizovaný strukturovaný dotazovací jazyk – Structured English Query Language
DDL	jazyk pro definici dat – Data Definition Language
DML	jazyk pro manipulaci s daty – Data Manipulation Language
DCL	jazyk pro kontrolu dat – Data Control Language
TCL	jazyk pro řízení transakcí – Transaction control language
DQL	jazyk dotazování – Data Query Language
HSM	hardwarový bezpečnostní modul – Hardware Security Module
AES	standard pokročilého šifrování – Advanced Encryption Standart
VPN	virtuální privátní síť – Virtual Private Network
SSL	vrstva vložená mezi vrstvu transportní a aplikační – Secure Sockets Layer
TLS	zabezpečení na transportní vrstvě – Transport Layer Security
RSA	asymetrická šifra pojmenována jmény tvůrců – Rivest Shamir Adleman
3DES	trojnásobné šifrování dat – Triple Data Encryption
Nist	národní institut standardů a technologie – National Institute of Standards and Technology
ETL	extrakce transformace načítání – extraction transformation load
TDE	transparentní šifrování dat – Transparent Data Encryption
API	rozhraní pro programování aplikací – Application Programming Interface
LDAP	protokol pro ukládání a přístup k datům na adresářovém serveru – Lightweight Directory Access Protocol

DLE	šifrování na úrovni databáze – Database Level Encryption
FLE	šifrování souborů – Filesystem Level Encryption
FDE	šifrování celého disku – Full Disk Encryption
ALE	Šifrování na úrovni aplikace – Application Level Encryption
CTR	čítačový mód – Counter mode
SSD	polovodičový disk – Solid State Drive
RHEL	Red Hat Enterprise Linux
TOAST	Technika ukládání nadměrných atributů – The Oversized-Attribute Storage Technique
tps	transakce za sekundu – transactions per seconds
WAL	logování s předstihem – write-ahead logging