УДК 681.04 DOI: 10.26565/2519-2310-2022-2-01

АНАЛІЗ МЕТОДІВ ПОШУКУ ДАНИХ У КРИПТОГРАФІЧНО ЗАХИЩЕНІЙ БАЗІ ДАНИХ

Теймур Махмудов, Віталій Єсін

Харківський національний університет імені В.Н. Каразіна, майдан Свободи, 4, Харків, 61022, Україна
timkin552364964@gmail.com, v.i.yesin@karazin.ua

Надійшла: серпень 2022. Прийнята: жовтень 2022.

Анотація: *Питання забезпечення безпеки даних, а саме їх конфіденційності та цілісності, як правило, у теперішній час вирішується за рахунок використання відповідних криптографічних примітивів з урахуванням розвитку обчислювальних потужностей. Але, у зв'язку із специфічним способом зберігання (у хмарі), виникає питання ефективності пошуку необхідної інформації. Проблема, яка розглядається у цій роботі, полягає у тому, що шифрування унеможливлює доступ до даних без ключів для зловмисника, позбавляючи при цьому, власника даних, можливості проведення пошуку за цією інформацією. В роботі розглянуто декілька методів шифрування з можливістю пошуку. Для кожного з них приведені алгоритми, приклади використання цих методів і надані пояснювальні рисунки та таблиці. Розглянуті методи є симетричними та динамічними, завдяки чому вони ефективні та мають відносно високий рівень безпеки, але низьку виразність запитів, через що знаходять найбільше використання у NoSQL базах даних. Проведено аналіз для виявлення складності та рівня безпеки методів, а також розглянута продуктивність практичних реалізацій. Зроблено висновки про доцільність використання того чи іншого методу шифрування з можливістю пошуку на практиці. Запропоновані рекомендації щодо комбінації описаних методів для отримання очікуваного результату.*

Ключові слова: *шифрування з можливістю пошуку, ключове слово, індекс, динамічне симетричне шифрування з можливістю пошуку, токен пошуку, лазівка.*

1. Вступ

За останні роки хмарні послуги стали важливою частиною життя більшості людей, замінивши та полегшивши виконання їхньої щоденної рутини. Тепер необов'язково мати на своєму персональному комп'ютері набір інструментів від Office, так як у будь-який момент можна скористатися безкоштовними аналогами Google Docs, Google Sheets або Office 365. Так само на ринку з'явилося безліч рішень для зберігання даних у хмарі: Google, Microsoft, Dropbox, Mega та багато інших, але у всіх з них стоїть питання безпеки даних, що зберігаються. Адже дані не лише передаються через Інтернет, але ще й зберігаються у третіх осіб, тому, не дивлячись на запевнення осіб, які надають послуги, необхідно пам'ятати про безліч новин про витоки даних.

Для здійснення пошуку можна завантажити всю базу даних (БД) і виконати розшифрування. Проте для більшості застосунків цій підхід є недоцільним. Якщо ж покласти на сервер розшифрування даних, щоб він міг виконувати запити і надсилати користувачеві лише результати, то знижується рівень безпеки, оскільки дані, що захищені шифруванням, розкриваються серверу. Тому слід мати можливість проведення пошуку у повному обсязі на стороні сервера без розшифрування даних, що зменшує ризик втрати конфіденційності даних.

У роботі запропоновано огляд і аналіз предметної області, що розглядається, саме, шифрування з можливістю пошуку (ШМП), проводиться його класифікація та наводяться моделі безпеки, які характеризують рівень захищеності методів ШМП. Крім того проведено стислий розгляд конструкції і аналіз відповідних методів за часовою, комунікаційною та просторовою складностями, а також за рівнем забезпечуваної ними безпеки.

Метою роботи є аналіз методів пошуку даних у криптографічно захищених БД для визначення більш ефективних та безпечних реалізацій, а також формулювання рекомендацій стосовно вибору певного методу в залежності від потреб проекту.

2. Основні поняття та тематично пов'язані роботи

Шифрування з можливістю пошуку (ШМП, *англ. searchable encryption*) – це вид шифрування, при якому мається можливість робити пошук за зашифрованими даними з мінімальним витоком інформації.

Ця особливість робить його дуже зручним при використанні хмарних сховищ, які зазвичай сприймаються як чесний-але-допитливий (honest-but-curious) супротивник – це законний учасник комунікаційного протоколу, який не буде відхилятися від визначеного протоколу, але намагатиметься дізнатися всю можливу інформацію з законно отриманих повідомлень [1].

Можливість пошуку у ШМП досягається декількома способами:

1) вироблення таким чином шифртексту, щоб за ним можна було виконувати пошук;

2) створення індексів, за якими виконуватиметься пошук.

Між цим, самі індекси можуть бути двох видів: прямі – кожному документу зіставляється множина ключових слів, та інвертовані – кожному ключовому слову зіставляються документи. Приклад цих індексів наведено на рис. 1. Інвертовані індекси працюють швидше за прямі, оскільки в БД кількість записів значно більше кількості ключових слів.

документ	ключові слова
d1	w2, w5, w7
d2	w1, w2
...	...
dn	w6, w9, wk

Прямий індекс

ключові слова	документ
w1	d2, d5, d11
w2	d1, d2
...	...
wk	d6, d12, dn

Інвертований індекс

Рис. 1 – Види індексів

Для шифрів з можливістю пошуку можна провести класичну для шифрів дихотомію на симетричне на асиметричне ШМП. Проте для таких систем є певна особливість: для симетричної системи є тільки один ключ, який є секретним, відповідно є тільки один користувач – власник ключа, що може здійснювати запис та зчитувати дані, але, якщо йде мова про асиметричну систему, то в ній наявний відкритий ключ, за допомогою якого багато користувачів можуть записувати дані на сервер, а здійснювати пошук може тільки власник секретного ключа. Більш того, якщо використовувати розподілений секретний ключ, то це дозволить створити систему, в якій буде більш однієї людини, що може робити пошук за шифртекстом. Отже, по типу «письменники/читачі» системи можуть бути 1/1, */1, 1/* та */*.

Схеми ШМП також поділяються на статичні та динамічні. Динамічні схеми ШМП – це такі, що дозволяють проводити оновлення структури даних. Тобто, якщо є деяка множина документів, то є можливість додавати нові елементи та видаляти вже присутні. При цьому реалізація цього може бути різною, деякі схеми підтримують додавання, але не підтримують видалення. Для того, щоб такі операції проводити, як правило, генерується відповідний токен оновлення (подібно до токена пошуку), в якому зазначаються ідентифікатор файлу та ключові слова, пов'язані з ним, що треба додати чи видалити. Але зрозуміло, що ці операції також можуть бути небезпечними та уможливлювати витік інформації. З цього приводу у [2] було визначено декілька понять безпеки в контексті ШМП:

1. Пряма секретність або *forward privacy* (*FP*) – гарантує, що сервер не дізнається, чи містять додані документи ключові слова, пошук за якими проводився раніше.
2. Зворотна секретність або *backward privacy* (*BP*) – гарантує, що сервер не може застосовувати запити к видаленим документам.

Більш того, у роботі [3] автори виділили три рівня зворотної секретності: перший є найпотужнішим, на цьому рівні схема при додаванні нового документу розкриває лише документи, що містять ключове слово w, що наявне у запиті, момент часу, коли вони були додані, а також загальну кількість додавань за ключовим словом w; другий рівень розкриває інформацію першого рівня, а також момент часу, коли було виконано будь-яке оновлення за ключовим словом w; третій, найслабкіший, рівень розкриває всю інформацію другого рівня, а також яке саме видалення скасувало певне попереднє додавання ключового слова.

Ще одним аспектом безпеки ШМП є шаблони пошуку та доступу. Шаблон пошуку – це інформація про те, чи створено будь-які два запити за одним ключовим словом чи ні. Шаблон доступу – це інформація про те, які документи містять ключове слово для кожного запиту користувача. Як зазначено в [4], дуже багато ШМП «страждають» від того, що з зазначених вище шаблонів витікає інформація. У симетричних схемах ця проблема ґрунтується на тому, що для генерування лазівки зазвичай використовуються детерміновані алгоритми, тобто для певного ключового слова завжди буде генеруватись однаковий запит.

Т.ч. очевидно, що супротивник може без зусиль встановити, чи мають 2 різні запити одне й те ж саме ключове слово. З іншого боку, є шаблон доступу, який, в свою чергу, може розкрити шаблон пошуку. Якщо шаблон доступу однаковий, то, ймовірно, що два пошукових запити містять однакове ключове слово. В роботі [5] зазначається, що майже у всіх симетричних ШМП витікає шаблон доступу.

Для визначення того, чи є метод ШМП безпечним, використовуються моделі безпеки. У роботі [6] були визначені моделі *IND1-CKA* та *IND2-CKA* (*від англ. indistinguishability against chosen keyword attack*), зазначивши, що врахування лазівок є обов'язковим, так як вони нерозривно пов'язані з безпекою індексів. Перша модель передбачає нерозрізненість проти атак з неадаптивно підібраними ключовими словами, друга – з адаптивно. Обидві моделі гарантують, що ні індекси, ні лазівки не розкривають жодну інформацію про вміст документу та ключові слова, асоційовані з документом (*за виключенням тієї, яку можна вивести з шаблонів пошуку та доступу*).

Проте все зазначене вище стосується тільки симетричних схем шифрування з можливістю пошуку. Що стосується асиметричних ШМП (АШМП), в таких схемах з'являється ще один ключ – відкритий, за допомогою якого генеруються лазівки, отже, безпека лазівок не враховується. Першою роботою стосовно моделі безпеки АШМП стала стаття Дена Боне та інших [7], у якій вони запропонували визначати безпеку асиметричних схем нерозрізненістю двох зашифрованих ключових слів, доки супротивник не має лазівки до цих слів, тобто цю модель також можна назвати *IND-CKA* або *PK-CKA* (від англ. *public key*). Аналогічно 1 версія буде для не адаптивно підібраних ключових слів, а друга – для адаптивно.

3. Конструкція методів шифрування з можливістю пошуку

Розглянемо побудову методів ШМП та криптопримітиви, на яких базуються ці методи. Всі зазначені методи відносяться до симетричного шифрування з можливістю пошуку, а також всі, окрім одного, є динамічними. Такі методи мають перевагу у ефективності та безпеці схем, проте поступаються виразністю запитів, через що найчастіше знаходять використання у NoSQL базах даних. При цьому увага не приділяється методам, що вже були розглянуті та проаналізовані у інших роботах, в яких доведена їх неспроможність чи то з погляду безпеки, чи то з боку ефективності. Наприклад, у роботі [8] було проведено серйозний аналіз великої кількості методів, однак більшість з них виявилися недостатньо ефективними для їх практичного використання. Деякі з зазначених схем є досить повільними при прийнятному рівні безпеки, інші ж працюють досить швидко, проте мають неприйнятний витік інформації.

3.1 Паралельне та динамічне шифрування з можливістю пошуку (PDSSE)

Цей метод (позначимо його як абревіатуру від англійської назви *Parallel and Dynamic Symmetric Seacrable Encryption*, тобто *PDSSE*) розглянуто в роботі [9]. Він заснований на червоно-чорних деревах (ЧЧД). Проте в методі використовуються деяка модифікація ЧЧД: листя дерева зберігають покажчики на документи, а інші вузли – ідентифікатори документів. Також кожен вузол u зберігає бітовий вектор $data_u$ довжиною m біт, де i-ий біт відображає чи наявне ключове слово w_i у його нащадках. Тут встає питання того, що вузли мають по два нащадки (припустимо, що лівий нащадок – це v, а правий – це z), отже, бітовий вектор кожного вузла u визначається як «побітове або» векторів його нащадків, тобто за формулою:

$$data_u = data_v \,\&\, data_z\,.$$

Тоді якщо i-ий біт дорівнює одиниці, то є хоча б один шлях, який приводить до документу d_j, що містить ключове слово w_i. А для знаходження всіх документів, які містять ключове слово, просто необхідно перевіряти вектори нащадків, доки не зустрінемо 0 в i-ії позиції вектора, або не доберемося до листа дерева.

Для проведення оновлення визначається ідентифікатор документа, та операція (додавання чи видалення), яку треба виконати. Після цього визначається частина дерева, яку це оновлення торкнеться, та проводиться перерахунок бітових векторів у вузлах, згідно доданого чи видаленого документу.

На рис. 2 зображено реалізацію методу приведену авторами, в якій є 8 документів та 5 ключових слів. Виходячи з кількості ключових слів, кожен вузол зберігає по два 5-ти бітних вектори, один з яких містить справжню інформацію о ключових словах у своїх нащадках, проте у зашифрованому вигляді. Червоними стрілками позначений пошук 5-го ключового слова, в результаті отримуємо, що 5 ключове слово знаходиться у 3, 6 та 7 документах.

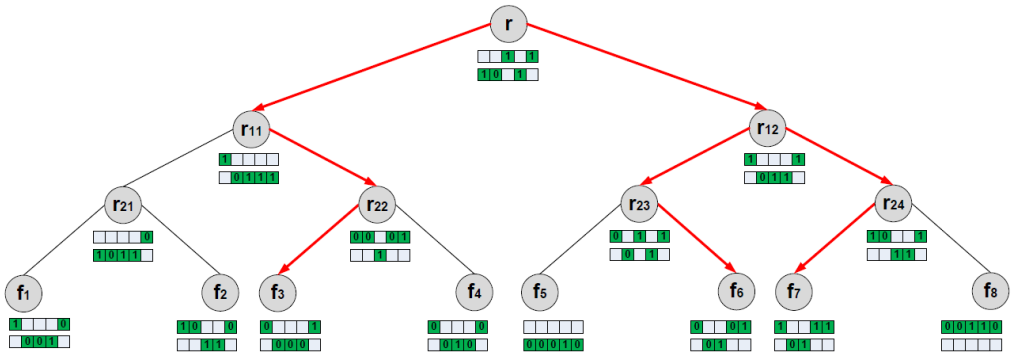

Рис. 2 – Приклад реалізації методу *PDSSE*

3.2 Метод Dyn2Lev

Цей метод визначено у [10]. Він взяв свою назву від принципу роботи: - він є динамічним, а головна ідея полягає у створенні 2 рівнів блоків покажчиків. Базис методу заснований на словнику, ключем у ньому виступають ключові слова, а значенням буде деякий блок довжиною B, що буде містити ідентифікатори записів. Тоді припустимо, що для ключового слова w та бази даних записів DB, існує $|DB(w)|$ записів, що потрібно витягнути з БД, отже, таких блоків буде $\frac{|DB(w)|}{B}$. А останній блок доповнюється до розміру B, якщо це потрібно, щоб їх було неможливо відрізнити.

Таке групування дозволяє значно скоротити кількість необхідних витягнень при пошуку. Але постає питання того, що для різних ключових слів кількість записів, в яких вони при-

сутні, може значно відрізнятись. Тобто для деяких слів, котрим відповідають значні обсяги записів, витягнення все ще може бути неефективним. З іншого боку, якщо B обирається надто великим, то будуть страждати ключові слова, котрим відповідає маленька кількість записів, тому як доповнення до розміру блоку B буде створювати багато надлишкової інформації при доповненні.

Спочатку було запропоновано ввести блок додавати у словник блок не ідентифікаторів, а покажчиків, що посилаються на блоки ідентифікаторів, які зберігаються у зовнішньому масиві «A». Але ця модифікація все ще не вирішує питання щодо наявності як ключових слів з великою кількістю відповідних записів, так і з незначною. З цього приводу планується розрізняти множини на маленькі, середні та великі. Визначається все через розміри блоків: блоки розміром b у словнику та блоки розміром «B» у деякому зовнішньому масиві. Отже, маленькі множини мають кількість записів менше або дорівнює b, середні множини мають кількість записів, що лежить у межах $b < |DB(w)| \leq Bb$, великі – $Bb < |DB(w)| \leq B^2b$. Верхня границя для великих множин дорівнює B^2b, з чого випливає, що необхідно обирати «B» та «b» таким чином, щоб не існувало множини $DB(w)$ потужністю більше ніж B^2b.

Для маленьких множин ідентифікатори будуть зберігатися напряму у словнику, таким чином, не буде потреби у використанні покажчиків. Для середніх будуть створюватися блоки покажчиків, що вказують на відповідні блоки ідентифікаторів. Для великих множин є два рівні (*2 lev*): - у словнику зберігаються покажчики, що вказують на блоки покажчиків, які, в свою чергу, вказують на блоки ідентифікаторів.

Отже, є три способи, розглянемо приклад (*див. табл. 1*), щоб зрозуміти як змінюється кількість витягнень в залежності від підходу для підвищення ефективності. Візьмемо $B = 500$ та $b = 100$.

Таблиця 1 – Кількість витягнень з словника в залежності від підходу

$\|D(w)\|$	Блоки ідентифікаторів	Блоки покажчиків	Два рівня блоків покажчиків
100	1	1	1
200	2	1	1
1100	11	1	1
5100	51	1	1
10100	101	1	1
50100	501	2	1
500100	5001	11	1
5000100	10001	101	1

Таким чином, при застосуванні комбінованого підходу з трьома видами множин, нам завжди необхідно зробити тільки 1 витягнення. Для пошуку клієнт генерує спеціальну позначку на базі ключового слова. За ключем з словника витягується блок, якщо цей блок містить ідентифікатори, то вони відправляються клієнту, якщо ж покажчики, то він використовує покажчики для отримання блоків з масиву A. Якщо ці блоки містять ідентифікатори, то він їх відправляє клієнту, інакше знову використовує позначки, але тепер вже точно отримує блоки ідентифікаторів, які відправляє клієнту.

Для можливості видалення генерується окрема структура даних – множина S_{rev}, що зберігає так звані ідентифікатори відкликання, значення яких отримується від псевдовипадкової функції (ПВФ) з спеціальним ключем ключового слова. Тобто буквального видалення з словника не проводиться. Для можливості додавання генерується ще один словник D^+, що має саму базову конструкцію методу. Якщо треба додати запис, то клієнт відправляє запит

до серверу, що містить ідентифікатор запису і множину ключових слів. Для кожної пари «ідентифікатор/ключове» слово, сервер перевіряє, чи наявний відповідний ідентифікатор відкликання у множині S_{rev}, і якщо наявний, то він його видаляє, а якщо ні – то додає пару до словника D^+.

У зв'язку з можливістю додавання та видалення пошук також дещо змінюється: після перевірки основного словника, проводиться перевірка у словнику для додавання записів, а також перевіряється чи є у множині відповідний ідентифікатору запису ідентифікатор відкликання. Якщо так, то знайдений ідентифікатор прибирається. Отриманий список ідентифікаторів при пошуку відправляється клієнту.

3.3 Логічне симетричне шифрування з можливістю пошуку

Наступний метод визначено у роботі [11], в якій зосереджується увага на виразності методу. Два попередньо розглянуті методи дозволяють по одному ключовому слову отримати певний перелік ідентифікаторів документів або записів БД, що містять це ключове слово. Проте іноді зручним є пошук тих документів чи записів, що містять в собі деяку множину ключових слів. Перед тим, як перейти до безпосереднього розгляду методу, визначимо декілька понять, важливих для розуміння реалізації методу.

Мультипвідображення (МВ) або *multimap* (*MM*) – це відображення, яке для кожного ключа зіставляє більше одного значення. Зокрема, у схемі використовується кортеж значень.

Фільтр Блума – це ймовірнісна структура даних, яку запропонував Бартон Блум у 1970 році [12]. Ця структура даних має дві операції: додавання елемента до множини та перевірка наявності елементу у множині. Імовірність структури полягає у тому, що можливі відповіді на питання наявності елементу – це «можливо» або «ні». Тобто фільтр Блума припускає хибно позитивні відповіді, але хибно негативні неможливі. Ідея полягає у тому, що існує бітовий масив розміру m та k геш-функцій, що видають значення від 0 до $m-1$. Якщо треба додати деякий елемент, то від нього розраховуються всі геш-функції, та відповідні розрахованим значенням біти масиву встановлюються в одиницю. А при перевірці також розраховуються значення геш-функцій та витягуються відповідні значення з масиву, і, якщо всі значення дорівнюють одиниці, то елемент може бути присутнім, а якщо маємо хоча б один нуль, то елементу точно немає у структурі даних.

Онлайн шифр – це блочний шифр, визначений у [13]. Його особливістю є те, що шифрування може проводитись в онлайн або у потоковому режимі. Такі шифри мають наступну вимогу: якщо є деяке повідомлення $M = M_1||M_2|| \dots ||M_l$ та відповідний йому шифртекст $C = C_1||C_2|| \dots ||C_l$, то для розрахування деякого блоку шифртексту C_i достатньо знати блоки відкритого тексту з 1 до i. Таким чином, блок шифртексту C_i не залежить від блоків відкритого тексту $M_{i+1}, \dots, M_l$.

Inclusion-exclusion principle (*IEP*), тобто принцип включень-виключень. Це техніка, що дозволяє визначати потужність об'єднань кінцевого числа кінцевих множин, завдяки ж цьому можна отримати множину унікальних елементів цього об'єднання. Принцип, як випливає з назви, полягає у почерговому застосуванні включення та виключення: спочатку включаються всі множини; після цього виключаються попарні перетини; далі включаються потрійні перетини; після виключаються четверні перетини і так далі до n-го порядку, де n – кількість множин.

Розглянемо приклад *IEP* для випадку перетину множин A, B і C, що наведено на рис. 3. При підсумовуванні трьох множин, двічі враховуються елементи, що наявні в двох множинах, та тричі враховуються елементи, що наявні в усіх трьох множинах. Тому, після підсумовування всіх множин, необхідно виключити подвійні перетини. Таких перетини три, і після

їх віднімання елементи, що наявні рівно у двох множинах, будуть представлені у єдиному екземплярі.

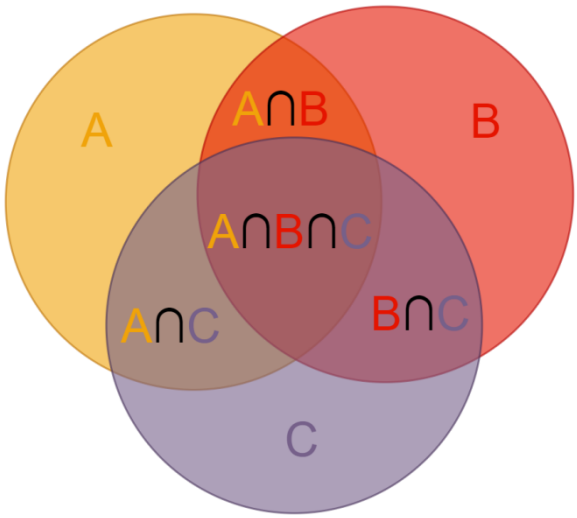

Рис. 3 – Перетин трьох множин

Але, як зазначалось вище, елементи, що наявні у всіх трьох множинах, були враховані тричі, проте на минулому кроці їх тричі відняли, тобто тепер їх не залишилось. Отже, необхідно знову включити їх до загального результату. Таким чином, було продемонстровано застосування *IEP* для отримання унікальних елементів об'єднання множин A, B і C.

Перейдемо до визначення головної конструкції методу – *IEX* (*inclusion-exclusion encryption*), назва якої відсилає на принцип «включень-виключень» (*IEP*). У методі визначаються теги, котрі обчислюються, як шифрування пари ідентифікатор та ПВФ від конкатенації ідентифікатору і ключового слова *(див. (1))*. Т.ч., кожен тег має унікальне значення, навіть якщо його згенеровано для одного ключового слова, або для одного ідентифікатору запису (1):

$$tag_{id} = SKE.Enc_{K_1}\left(id; F_{K_2}(id||w)\right). \quad (1)$$

Як базис методу використовуються три структури даних: два мультивідображення та словник. Перше мультивідображення називається глобальним, в якості ключа приймає ключове слово, а в якості значення – теги тих записів, що містять це ключове слово.

Розглянемо приклад такого мультивідображення на прикладі рис. 4.

MMg

w_1	tag_1, tag_4, tag_{10}
w_2	tag_3, tag_5
w_3	tag_1, tag_9
w_4	tag_6
w_5	tag_4, tag_5, tag_7
w_6	tag_1, tag_4, tag_8

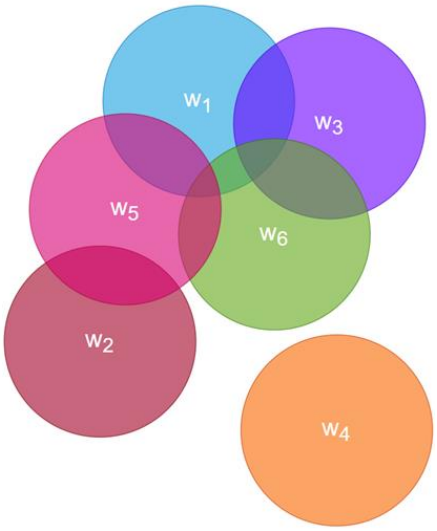

Рис. 4 – Приклад побудови глобального мультивідображення

Також для кожного ключового слова генерується локальне мультивідображення, як ключ приймаються ключові слова $v \in co(w)$. В множині $co(w)$ зберігаються ключові слова, що наявні у записі/документі разом з ключовим словом w. Значення у цьому мультивідображенні – «теги записів», в яких наявні обидва ключових слова. При цьому ці локальні МВ записуються у третю структуру даних – «словник», ключем виступає ключове слово w. Приклади «словника» та локального МВ наведені нижче, на рис. 5.

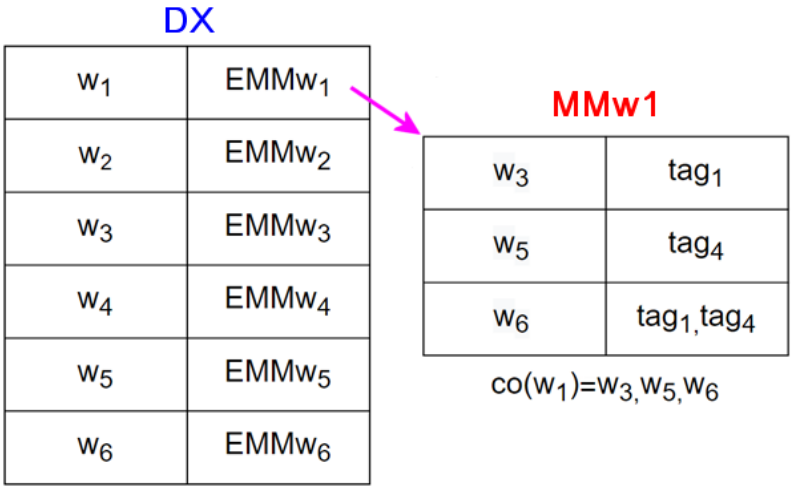

Рис. 5 – Приклад побудови словника та локального мультивідображення

Розберемо як проходить пошук, спочатку для диз'юнктивного виразу. Нехай маємо запит вигляду $w_1 \vee \ldots \vee w_q$. Результат такого запиту отримуємо за принципом включень-виключень: для кожного ключового слова (окрім останнього) спочатку звертаємось до глобального МВ, отримуємо всі теги та додаємо до загальної множини. Далі перебираємо локальні мультивідображення з тими ключовими словами, що йдуть у запиті після поточного, та прибираємо з множини всі теги, що наявні у значеннях цих локальних МВ. Для останнього у запиті ключового слова додаються всі теги. Таким чином, ми отримуємо відповідь на запит без дублювання тегів.

Якщо на вхід подається довільний логічний запит, то його завжди можна представити у кон'юнктивній нормальній формі, тобто $\Delta_1 \wedge \ldots \wedge \Delta_l, \Delta_i = w_{i,1} \vee \ldots \vee w_{i,q}$. А набір кон'юнкцій – це знаходження перетину деякої кількості множин, кожна з котрих може бути представлена деяким набором диз'юнкцій Δ_i. Причому треба відмітити, що перетин цих множин завжди буде підмножиною Δ_1. По суті, на початку маємо множину елементів Δ_1, після цього видаляємо всі елементи, котрих немає у Δ_2, далі всі елементи, котрих нема у Δ_3 і так до Δ_l.

У *динамічному* варіанті відмінність полягає у тому, що реалізації глобального мультивідображення та словника повинні підтримувати оновлення, локальне МВ залишається статичним. Сам процес оновлень теж майже повністю співпадає з цими реалізаціями, окрім генерації відповідних лазівок.

Слід відзначити, що описана схема є деякою надбудовою над іншими схемами ШМП: у якості реалізації МВ використовуються саме симетричні методи шифрування з можливістю пошуку. Хоча *IEX* і дозволяє виконувати логічні запити, його використання значно збільшує займаний простір. Тому автори запропонували компактний метод ШМП – *ZMF*, назва якого присвячена конструкціям на яких він базується - метод *Z-IDX* та *Matryoshka filter*.

Ця схема використовується як реалізація для локальних МВ. Вона має лінійну складність пошуку, проте значно меншу просторову складність, тому що базується на фільтрах Блума. Точніше, для методу розроблена нова структура даних – фільтр-матрьошка, який базується на фільтрах Блума. Визначимо коротко сутність: мається декілька множин з різною кількістю елементів (*наприклад, кортежи тегів у локальному МВ*). Серед цих множин ми визначаємо найбільшу та генеруємо для неї геш-функції та фільтр Блума. Для всіх інших множин фільтри Блума будуть виводитись з цього максимального фільтру (*тобто вони вкладені в максимальний фільтр*), так само як і геш-функції, в залежності від потужності кожної множини. Для безпеки даних, фільтр додається за модулем 2 з маскою. Значення цієї

маски повинні залежати від конкретної бітової позиції фільтру *(інакше це порушить коректність при перевірці наявності елементу*), а також від конкретного фільтра Блума. Безпосередньо маска генерується наступним чином: від елемента розраховується ПВФ, від отриманого значення обчислюється геш-функція. Кожен біт геш-значення доповнюється нулями до блоку розміром B, отриманий рядок шифрується за допомогою онлайн шифру, після чого усікається в залежності від розміру множини. Отримане значення комбінується з ідентифікатором множини (*фільтру Блума*) та подається до випадкового оракулу. Вихідне значення і є маскою для деякого біту фільтру.

3.4 Метод Σοφος

Σοφος – це симетрична схема шифрування з можливістю пошуку, що відповідає поняттю прямої секретності, запропонована у [14]. Як базис схеми використовуються відображення. Перед початком розгляду схеми визначимо основний її криптопримітив – перестановку лазівки або *trapdoor permutation (TDP).*

TDP – це така перестановка π над деякою множиною D, що за допомогою відкритого ключа PK, π може бути обчислена без будь-яких проблем за прийнятний час, але інверсія π^{-1} може бути ефективно обчислена лише за допомогою секретного ключа SK.

Розглянемо основну ідею методу: маємо деяку множину ключових слів W, для кожного $w \in W$ існує відповідний список індексів документів, що містять це ключове слово – $(ind_0, \dots, ind_{n_w-1})$, де $n_w = |DB(w)| - d_w$ – це кількість документів, що містять ключове слово w, за винятком видалених. Кожний елемент цього списку шифрується та записується у логічному місці, яке виводиться від ключового слова w та номеру ідентифікатора документа – c. Це логічне місце позначається як $UT_c(w)$ та, по суті, є токеном оновлення, відповідно, якщо клієнт захоче додати новий документ до БД, який містить ключове слово w, то йому треба зашифрувати індекс та розрахувати логічну позицію для нього, тобто токен оновлення.

Для пошуку клієнт повинен згенерувати токен пошуку $ST(w)$, який дозволяє серверу згенерувати відповідний токен оновлення за допомогою геш-функції, цей токен дозволяє витягнути зашифрований індекс документу. При цьому, мається n_w індексів $(c = n_w - 1)$, що відповідають ключовому слову w, які необхідно отримати при пошуку, але таким чином, щоб сервер не зміг отримати індекси оновлення $UT_i, i > c$, тобто логічні місця, де будуть зберігатися додані у майбутньому ідентифікатори.

Тут вступає у гру саме *TDP*, що зазначалась вище: за допомогою токену пошуку $ST_i(w)$ сервер зможе розрахувати $ST_{i-1}(w)$, використовуючи відкритий ключ PK, але для розрахунку $ST_{i+1}(w)$ знадобиться секретний ключ.

Для додавання можливості видалення елементів пропонується створити ще один екземпляр описаної схеми. А пошук буде різницею алгоритмів пошуку цих двох екземплярів.

3.5 Метод Fides

Розглянемо підхід та схему на його основі, що були запропоновані у роботі [15]. Підхід описує, як з звичайної схеми симетричного ШМП Σ (англ. *Symmetric Searchable Encryption* або *SSE*) отримати схему, що відповідає поняттю зворотної секретності. Ця схема є деякою «надбудовою» над іншими схемами, вона пропонує зберігання не індексів, а комбінації індексу та операції *(див. (2))*, після чого ця комбінація шифрується ключем відповідного ключового слова w.

$$E_{K_w}(ind, op) \tag{2}$$

В іншому, базова схема Σ працює як зазвичай. Проте є декілька важливих моментів щодо зазначеної схеми: - при пошуку сервер не зможе ідентифікувати необхідні записи за ключовим словом, оскільки він не бачить індексів (тому що вони зашифровані). З цього приводу

вводиться ще один раунд протоколу: після того, як клієнт отримає (2), він розшифровує цю структуру та отримує індекси, які відсилає серверу, після чого вже проводиться безпосередній витяг документів.

Тут рішення досить просте: отримуючи відповідь, клієнт прибирає всі індекси, що були видалені, та відправляє решту серверу у відкритому вигляді, але, поряд з цим, він відправляє ці ж самі індекси, зашифровані новим ключем. Таким чином, сервер зможе видалити всі старі індекси, а на заміну додасть нові, утримуючу базу даних в актуальному стані.

Застосовуючи описані вище операції до різних *SSE* схем, можна отримати різні екземпляри *BP*- надійної схеми. *Fides* базується на схемі *Σοφος*.

3.6 Метод Diana

Метод отримується на базі фреймворку *FS-RCPRF* (*forward secure scheme based on the range-constrained pseudorandom function*), то спочатку визначимо його. В цьому фреймворку використовується обмежена псевдовипадкова функція (ОПВФ), яку було паралельно розроблено та представлено у роботах [16] та [17]. Ця ПВФ особлива тим, що асоціюється з сімейством логічних схем $\mathcal{C}$. Для кожної з схем цього сімейства $C \in \mathcal{C}$ мається обмежений ключ K_C, який можна обчислити за допомогою майстер-ключа ПВФ. Цей обмежений ключ дозволяє проводити обчислення тільки за такими значеннями x, для яких $C(x) = 1$.

Ідея полягає у тому, щоб генерувати токени оновлення для ключового слова w за допомогою ОПВФ у режимі лічильника, який збільшується після кожного додавання ідентифікатора, що містить задане ключове слово. Для пошуку клієнту потрібно відправити серверу ключ для ОПВФ, який дозволяє проводити обчислення значень від 0 до $n_w - 1$.

Для створення *Diana* використовується ОПВФ, яка утворюється з псевдовипадкової функції *GGM*, що базується на бінарному дереві [18]. Ця ОПВФ була побудована у [17] та отримала назву найкраще покриття діапазону або *Best Range Cover (BRC)*. Розглянемо принцип її роботи. Нехай мається деякий генератор псевдовипадкових чисел $G: \{0,1\}^{\lambda} \to \{0,1\}^{2\lambda}$, тоді позначимо результат від деякого випадкового початкового значення k як $G(k)$, а першу та другу його частину як $G_0(k)$ та $G_1(k)$ відповідно. Тоді ПВФ GGM від деякого цілого числа x довжиною n біт буде визначатись як:

$$F_K(x) = G_{x_0}\left(\dots\left(G_{x_n-1}(K)\right)\right).$$

Таким чином, листя дерева у *GGM* – це результат функції *F*, вона може бути зіставлена з вихідним значенням x, а також частковим обчисленням цієї функції (*тобто, з* $l < n$ *біт числа* x). Наведемо простий приклад дерева ПВФ з [17].

З рисунку 6 зрозуміло, яким чином отримуються значення у листках та взагалі у вузлах: записуються значення від кореня до вузла. Якщо потрібно пройти до листа з позначкою 3, то обчислюємо $G_1\left(G_1\left(G_0(G_0(k))\right)\right)$. Також можемо проводити асоціювання з частковими значеннями ПВФ: $G_1\left(G_1(f_k(00))\right)$ або $G_1(f_k(001))$. При цьому для деякого діапазону, наприклад, [2-7], як на рисунку 6, існує множина таких часткових значень, які являють собою піддерева. Для обраного діапазону, це $f_k(001)$ з глибиною 1, та відповідно $f_k(01)$, з глибиною 2 (*глибина до листя*). Задача *BRC* як раз і полягає в тому, щоб знайти мінімальну кількість таких часткових значень.

У методі *Diana* треба обмежити ПВФ на проміжку $[0, c]$, для цього будуть генеруватися такі вузли, що покривають цей діапазон, причому не містять шляху до листя, що в заданий діапазон не входять. В іншому, все йде згідно алгоритму *FS-RCPRF*.

В описаному вище методі можливе лише додавання. Уможливити видалення можна подібно до *Σοφος*, тобто створити два екземпляри SE-схеми з підтримкою прямої секретності, одна з яких буде для додавання, а інша – для видалення. Результат отримується як різниця результатів двох схем, але, якщо ці операції буде проводити сервер, то він буде отримувати інформацію про видалені записи. Пропонується інший підхід: при додаванні додаються записи виду (w, ind) до першого екземпляру. Поряд з цим передається пара значень $(F'(K_w, (w, ind)), Enc_{K'}(c))$, де F' це деяка ПВФ, а Enc – деяка CPA-надійна схема шифрування. Сервер приймає цю пару та заносить до спеціального відображення. При видаленні додається до другого екземпляру пара значень $(w, F'(K_w, (w, ind)))$.

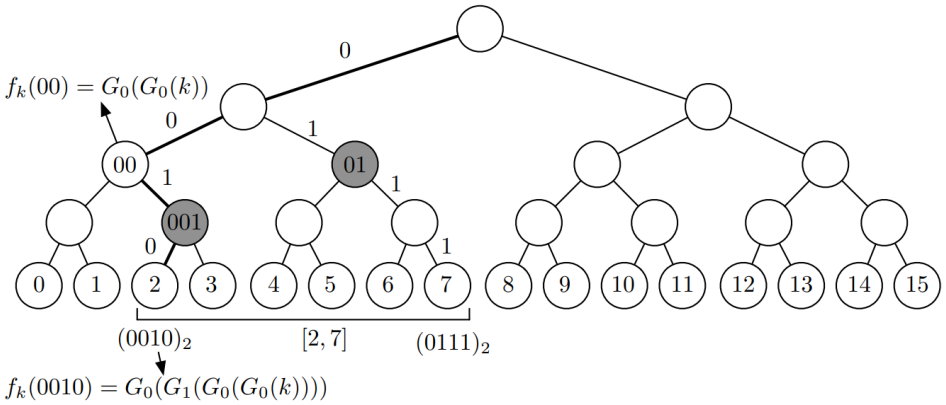

Рис. 6 – Приклад деревовидної ПВФ *GGM*

При пошуку даних спочатку робиться пошук у другому екземплярі, сервер витягує відповідні ключовому слову теги $F'(K_w, (w, ind))$, що відповідають видаленим записам. Далі він застосовує ці теги для витягнення $Enc_{K'}(c)$ з відображення, та відправляє ці зашифровані ідентифікатори клієнту. Клієнт розшифровує ідентифікатори та дізнається про те, які записи були видалені. Маючи цю інформацію, клієнт з початкового діапазону $[0, c]$ генерує декілька діапазонів, які не включають видалені записи. Якщо маємо видалені ідентифікатори $x_1, \dots, x_n$, то діапазони мають вигляд $[0, x_1 - 1], [x_1 + 1, x_2 - 1], \dots, [x_n + 1, c]$. Варто зауважити, що цей метод припускає, що один і той же індекс не додається двічі. Якщо його було видалено, то його не можна додати знов.

4. Аналіз методів шифрування з можливістю пошуку

Для зручності сприйняття проведеного аналізу зведемо всі дані про методи до табл. 2 за показниками часової (*обчислювальної*), комунікаційної та просторової складностей, а загальну характеристику схем зведемо до табл. 3. Позначки таблиць мають наступні значення:

- m – загальна кількість ключових слів;
- n – загальна кількість записів/документів;
- p – кількість процесорів;
- $DB(w)$ – кількість записів, що містять ключове слово w;
- d_w – кількість видалених записів, що містять ключове слово w;
- q – це кількість ключових слів у логічному запиті;
- $co(w)$ – множина ключових слів, що наявні у документах разом з w;
- $query$ – логічний запит;
- l – кількість диз'юнкцій у запиті;
- $M = max_{i \in [q]} |DB(w_i)|$ – це потужність максимальної множини документів для деякого ключового слова w_i;

Таблиця 2 – Порівняння розглянутих методів шифрування з можливістю пошуку за складністю

НАЗВА	ЧАСОВА СКЛАДНІСТЬ		КОМУНІКАЦІЙНА СКЛАДНІСТЬ		ПРОСТОРОВА СКЛАДНІСТЬ	
	Пошук	***Оновлення***	***Пошук***	***Оновлення***	***Клієнт***	***Сервер***
PDSSE	$O(\frac{\lvert DB(w)\rvert}{p}\log n)$	$O(\frac{m}{p}\log n)$	$O(\lvert DB(w)\rvert)$	$O(\lvert W_{id}\rvert + m\log n)$	$O(1)$	$O(m*n)$
Dyn2Lev	$O(\frac{\lvert DB(w)\rvert + d_w}{p})$	$O(1)$	$O(\lvert DB(w)\rvert)$	$O(\lvert W_{id}\rvert + m\log n)$	$O(1)$	$O(n)$
BIEX-ZMF	$O\left(q^2(M + l\lvert DB(\Delta_1)\rvert)\right)$	–	$O(\lvert DB(query)\rvert + lq)$	–	$O(1)$	$O\left(\sum_w(\lvert DB(w)\rvert + int_1)\right)$
BIEX-2Lev	$O\left(q^2(M + l\lvert DB(\Delta_1)\rvert)\right)$	–	$O(\lvert DB(query)\rvert + lq)$	–	$O(1)$	$O\left(\sum_w(\lvert DB(w)\rvert + int_2)\right)$
Σοφος	$O(\lvert DB(w)\rvert)$	$O(1)$	$O(\lvert DB(w)\rvert)$	$O(1)$	$O(m*\log n)$	$O(\sum_w(\lvert DB(w)\rvert + d_w))$
Fides	$O(\lvert DB(w)\rvert)$	$O(1)$	$O(\lvert DB(w)\rvert + d_w)$	$O(1)$	$O(m*\log n)$	$O(\sum_w(\lvert DB(w)\rvert))$
Diana$_{del}$	$O(\lvert DB(w)\rvert)$	$O(\log\lvert DB(w)\rvert)$	$O(\lvert DB(w)\rvert + d_w\log\lvert DB(w)\rvert)$	$O(1)$	$O(m*\log n)$	$O(\sum_w(\lvert DB(w)\rvert + d_w))$

Таблиця 3 – Порівняння розглянутих методів шифрування з можливістю пошуку

НАЗВА	Базис	Криптопримітиви	Безпека	FP	BP	Витік
PDSSE	Червоно-чорне дерево	ПВФ, геш-таблиця, геш-функція, CPA-надійна схема шифрування	IND2-CKA^{RO}	–	–	Шаблон пошуку, шаблон доступу, кількість ключових слів, розмір БД, ідентифікатори документу, розмір документів
Dyn2Lev	Словник, кортеж	ПВФ, CPA-надійна схема шифрування	IND2-CKA^{RO}	–	–	Шаблон пошуку, шаблон доступу, шаблон оновлення, розмір БД, розмір зовнішнього масиву, розмір блоків
BIEX-ZMF	Базується на *ZMF*, фільтр-матрьошка, мультивідображення, словник	ПВФ, ПВП, фільтр Блума, онлайн-шифр, CPA-надійна схема шифрування	IND2-CKA^{RO}	–	–	Шаблон пошуку, шаблон доступу, розмір БД, загальний обсяг всіх ЛМВ у словнику, перетини множин
BIEX-2Lev	Базується на *2Lev*, мультивідображення, словник, кортеж	ПВФ, ПВП, CPA-надійна схема шифрування	IND2-CKA^{RO}	–	–	Шаблон пошуку, шаблон доступу, розмір БД, загальний обсяг всіх ЛМВ у словнику, перетини множин + виток з *2Lev*
Σοφος	Відображення	ПВФ, RSA TDP, геш-функція, CPA-надійна схема шифрування	IND2-CKA^{RO}	+	–	Шаблон пошуку, вся історія взаємодій з ключовим словом: тип операції, час та ідентифікатор документу
Fides	Базується на *Σοφος*, відображення	ПВФ, RSA TDP, геш-функція, CPA-надійна схема шифрування	IND2-CKA^{RO}	+	II	Шаблон пошуку, кількість оновлень за ключовим словом та час, коли вони були виконані
Diana$_{del}$	Відображення	ПВФ, ОПВФ BRC, геш-функція, CPA-надійна схема шифрування	IND2-CKA^{RO}	+	III	Шаблон пошуку, кількість оновлень, їх час, ідентифікатор документу, історія оновлень цього документу, яке саме видалення скасувало додавання запису

- W_{id} – множина ключових слів, що передається при оновленні у методах *PDSSE* та *Dyn2Lev*;
- $int_1 = \max_{v \in co(w)} |DB(w) \cap DB(v)|$ – потужність найбільшої множини у локальному МВ;
- $int_2 = \sum_{v \in co(w)} |DB(w) \cap DB(v)|$ – загальна кількість локальних МВ.

5. Висновки

В роботі було розглянуто процес шифрування з можливістю пошуку, проведено його класифікацію та виділено характерні особливості. Здійснено розгляд та аналіз методів шифрування з можливістю пошуку.

На підставі проведеного аналізу методу *PDSSE* була визначена його відносно висока складність серед більш сучасних методів ШМП, яка підвищує складність пошуку для великих баз даних, навіть при незмінній кількості записів, що містять шукане ключове слово. Проте цей недолік можна компенсувати можливістю паралельного обчислення. Іншим, більш серйозним недоліком є його просторова складність: хоча клієнт зберігає тільки секретні ключи, сервер на кожний елемент дерева зберігає *m*-бітний вектор, тобто по біту на кожне ключове слово. Навіть середні за розміром бази даних можуть мати досить багато ключових слів, через що *PDSSE* займе дуже багато місця на диску. Цих 2-х недоліків достатньо, щоб говорити про неефективність практичного застосування цього методу.

Метод *Dyn2Lev*, завдяки використанню словника, здатен проводити операції оновлення за постійний час. Складність пошуку та займаний простір теж мають досить ефективні значення. Серед недоліків схеми можна відмітити зростаючу кількість видалених записів, які не видаляються насправді, з чого випливає необхідність в періодичному перешифруванні всієї БД. При цьому, метод використовує досить багато покажчиків, тому для отримання ідентифікаторів також витрачається зайвий час, тому реальна швидкість методу, хоча й краща за *PDSSE* та є непоганою на практиці, все ще поступається більш сучасним методам.

Схема *IEX*, як і її логічна та динамічна модифікації, відрізняється від всіх інших розглянутих схем, оскільки являє собою надбудову над іншими схемами для досягнення більшої виразності запитів. Її ефективність та безпека здебільшого базуються на підлеглій схемі, що використовується у якості мультивідображення, тобто нічого не заважає реалізувати вказані схеми над якоюсь більш ефективнішою та безпечнішою схемою.

Серед запропонованих авторами схем *BIEX-ZMF* та *BIEX-2Lev* вибір йде або убік компактності отриманої зашифрованої бази даних (*ZMF*), або убік швидкості роботи та комунікаційної складності (*2Lev*). Отже, рішення про використання *IEX* залежить від необхідності виконувати логічні запити до БД у конкретному проекті, враховуючи на збільшення часу пошуку, комунікаційної та просторової складності.

Метод *Σοφος*, порівняно з раніше розглянутими методами, характеризується кращою часовою та комунікаційною складністю. Серед його недоліків можна виділити необхідність клієнта зберігати значення лічильників для кожного ключового слова. При цьому, незважаючи на часову складність, реальний час пошуку схеми «страждає» через використання перестановки лазівки *RSA*, але він все одно значно кращий чим той же *Dyn2Lev*.

За допомогою фреймворка *FS-RCPRF* на базі *Σοφος* було визначено метод *Fides*, який відповідає не тільки поняттю прямої секретності, але й зворотної другого рівня, при цьому має майже однакові показники складності. Тому використання *Σοφος* не є доцільним, тому що *Fides* має більш кращі показники безпеки.

Модифікація схеми *Diana*, що дозволяє проводити видалення, базується повністю на симетричних криптопримітивах, що у поєднанні з оптимальною часовою складністю робить її найшвидшою з розглянутих схем. При цьому вона відповідає поняттям прямої секретності

та зворотної секретності третього рівня, а що головне – працює значно швидше всіх інших розглянутих методів. Хоч вона і розкриває більше інформації, ніж *Fides*, але швидкість пошуку розрізняється в десятки разів.

Отже, серед розглянутих в межах цієї роботи, методів ШМП, кращим та більш ефективнішим є метод *Diana*. Якщо безпека для проекту є надто важливою, то слід звернути увагу на *Fides*, поступившись швидкістю роботи.

Слід враховувати що, розглянуті методи перш за все підходять для *NoSQL* баз даних, оскільки мають дуже «слабку» виразність запитів. Виразність запитів можна покращити реалізацією *IEX* або *BIEX* «поверх» основної схеми для створювання логічних запитів, але це погіршить всі інші характеристики використовуваного методу.

Список літератури

[1] Paverd, A., Martin, A., & Brown, I. (2014). Modelling and automatically analysing privacy properties for honest-but-curious adversaries. *Tech. Rep*.

[2] Stefanov, E., Papamanthou, C., & Shi, E. (2013). Practical dynamic searchable encryption with small leakage. *Cryptology ePrint Archive*.

[3] Bost, R., Minaud, B., & Ohrimenko, O. (2017, October). Forward and backward private searchable encryption from constrained cryptographic primitives. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (pp. 1465-1482).

[4] Liu, C., Zhu, L., Wang, M., & Tan, Y. A. (2014). Search pattern leakage in searchable encryption: Attacks and new construction. *Information Sciences*, 265, 176-188.

[5] Islam, M. S., Kuzu, M., & Kantarcioglu, M. (2012, February). Access pattern disclosure on searchable encryption: ramification, attack and mitigation. In *Ndss* (Vol. 20, p. 12).

[6] Curtmola, R., Garay, J., Kamara, S., & Ostrovsky, R. (2006, October). Searchable symmetric encryption: improved definitions and efficient constructions. In *Proceedings of the 13th ACM conference on Computer and communications security* (pp. 79-88).

[7] Boneh, D., Crescenzo, G. D., Ostrovsky, R., & Persiano, G. (2004, May). Public key encryption with keyword search. In *International conference on the theory and applications of cryptographic techniques* (pp. 506-522). Springer, Berlin, Heidelberg.

[8] Bösch, C., Hartel, P., Jonker, W., & Peter, A. (2014). A survey of provably secure searchable encryption. *ACM Computing Surveys (CSUR), 47*(2), 1-51.

[9] Kamara, S., & Papamanthou, C. (2013, April). Parallel and dynamic searchable symmetric encryption. In International *conference on financial cryptography and data security* (pp. 258-274). Springer, Berlin, Heidelberg.

[10] Cash, D., Jaeger, J., Jarecki, S., Jutla, C., Krawczyk, H., Roşu, M. C., & Steiner, M. (2014). Dynamic searchable encryption in very-large databases: Data structures and implementation. *Cryptology ePrint Archive*.

[11] Kamara, S., & Moataz, T. (2017, April). Boolean searchable symmetric encryption with worst-case sub-linear complexity. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (pp. 94-124). Springer, Cham.

[12] Bloom, B. H. (1970). Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, *13*(7), 422-426.

[13] Bellare, M., Boldyreva, A., Knudsen, L., & Namprempre, C. (2001, August). Online ciphers and the hash-CBC construction. In *Annual International Cryptology Conference* (pp. 292-309). Springer, Berlin, Heidelberg.

[14] Bost, R. (2016, October). ∑οφος: Forward secure searchable encryption. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (pp. 1143-1154).

[15] Bost, R., Minaud, B., & Ohrimenko, O. (2017, October). Forward and backward private searchable encryption from constrained cryptographic primitives. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (pp. 1465-1482).

[16] Boneh, D., & Waters, B. (2013, December). Constrained pseudorandom functions and their applications. In *International conference on the theory and application of cryptology and information security* (pp. 280-300). Springer, Berlin, Heidelberg.

[17] Kiayias, A., Papadopoulos, S., Triandopoulos, N., & Zacharias, T. (2013, November). Delegatable pseudorandom functions and applications. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security* (pp. 669-684).

[18] Goldreich, O., Goldwasser, S., & Micali, S. (1986). How to construct random functions. *Journal of the ACM (JACM), 33*(4), 792-807.

Received: August 2022. Accepted: October 2022.

Authors:
Teimur Makhmudov, CSD Student, V.N. Karazin Kharkiv National University, Ukraine.
E-mail: timkin552364964@gmail.com
Vitalii Yesin, Doctor of Engineering Sciences, Professor, Department of Security of Information Systems and Technologies, V.N. Karazin National University, Kharkiv, Ukraine.
ORCID ID https://orcid.org/0000-0003-1977-7269

E-mail: v.i.yesin@karazin.ua

Analysis of data search methods in cryptographically protected databases.

Abstract. The issue of compliance with data security, namely their confidentiality and integrity, generally being resolved through the use of appropriate cryptographic primitives, taking into account the development of computing power (and/or computational complexity of algorithms). But, in connection with the specific method of storage (in the cloud), the question arises of the effectiveness of the search for the necessary information. The problem considered in this paper is that encryption makes it impossible for an attacker to access data without access key, but deprives the legal owner of the data, of the ability to search for this information.

The article reviewe several encryption methods with searchable. For each of them algorithms given, examples of the use of these methods, explanatory figures and tables were provided. The considered methods are symmetric and dynamic, due to which they are effective and have a relatively high level of security, but low query expressiveness, which is why they are most used in NoSQL databases. The analysis was conducted to evaluate the complexity and level of security of the methods, and the performance of practical implementations was also considered. It was concluded that, conclusions were made about the feasibility of using one or another searchable encryption method in practice, and recommendations were made regarding the combination of the described methods to obtain expected results.

Keywords*:* searchable encryption, keyword, index, dynamic symmetric searchable encryption, search token, trapdoor.