УДК 004.056: 004.056.5 **DOI:10.30837/rt.2022.2.209.14**

В.І. ЄСІН, д-р техн. наук, В.В. ВІЛІГУРА

ДОСЛІДЖЕННЯ ОСНОВНИХ МЕТОДІВ І СХЕМ ШИФРУВАННЯ З МОЖЛИВІСТЮ ПОШУКУ

Вступ

За останнє десятиріччя системи баз даних, що пропонуються у вигляді хмарних сервісів, широко використовуються та демонструють вибухове зростання [1]. Модель бази даних як послуга забезпечує користувачів можливістю створювати, зберігати, модифікувати та отримувати дані з віддаленого джерела, маючи доступ до інтернету. Однак, у міру того, як ми продовжуємо агрегувати дані, ключовою проблемою стає балансування дотримання безпеки та приватності (privacy) даних з їх аналітичним використанням для підтримки прийняття рішень. При переміщенні своїх даних у загальнодоступну хмару користувачі турбуються про їхню безпеку та приватність. А оскільки все більша кількість даних переноситься в хмарні сервіси зберігання, потрібні гарантії безпеки цих даних, у тому числі такі, що передбачаються міжнародними законами та стандартами, такими як: Загальний регламент захисту персональних даних Європейського Союзу (General Data Protection Regulation – GDPR) [2], Стандарт безпеки даних індустрії платіжних карт (PCI DSS) [3], Закон про переносимість і підзвітність медичного страхування (Health Insurance Portability and Accountability Act – HIPAA) [4, 5] та деякими іншими. Це стимулювало дослідження в галузі безпечного управління даними та підвищило їх актуальність [6].

Для безпечного зберігання конфіденційних даних на ненадійному віддаленому сервері останні мають бути зашифровані. Шифрування унеможливлює доступ до даних без ключів як для внутрішніх, так і для сторонніх осіб, але в той же час позбавляє власника даних всіх можливостей пошуку за цією інформацією.

Одним з простих рішень цієї проблеми є завантаження всієї бази даних, з подальшим її локальним розшифруванням і пошуком бажаних результатів отриманих розшифрованих даних. Проте для більшості застосунків такий підхід буде недоцільним.

Інший метод дозволяє серверу розшифровувати дані, виконувати запит на стороні сервера і надсилати користувачеві лише результати. Але в цьому випадку знижується рівень безпеки, оскільки дані, що захищені шифруванням, розкриваються серверу. Тому бажано підтримувати максимально повну функціональність пошуку на стороні з найменшою можливою втратою конфіденційності даних. Зокрема, захищена система пошуку повинна бути спрямована на те, щоб сервер нічого не дізнався про дані, що зберігаються в захищеній базі даних, або про запити, а той, хто запитує (querier) відповідні дані, нічого не дізнався, крім результатів запиту [7].

Одним із таких підходів, відомим із досить великої кількості робіт у світі, є підхід, що спирається на шифрування з можливістю пошуку (searchable encryption – SE). Ця технологія ґрунтується на можливостях сучасних криптографічних методів в умовах поділу ролей надання, адміністрування та доступу до даних. Її завдання – забезпечити необхідну функціональність із прийнятною втратою продуктивності.

Однак, незважаючи на досягнутий прогрес у цій галузі, комплексний підхід до підтримки конфіденційних хмарних обчислень поки що відсутній, і, таким чином, цей напрямок залишається плідною областю досліджень [1].

Основні методи шифрування з можливістю пошуку

Шифрування з можливістю пошуку (SE) – це технологія, яка дозволяє виконувати операції пошуку зашифрованих даних без розкриття будь-якої інформації про те, що шукається. Шифрування з можливістю пошуку діє як метод керування даними, який дозволяє власникам

даних зберігати свої дані та керувати ними на сторонньому, ненадійному віддаленому (у тому числі хмарному) сервері, а також дозволяє користувачеві даних делегувати функції пошуку хмарному серверу для отримання цих даних. Таким чином, шифрування з можливістю пошуку забезпечує безпечне зберігання та отримання даних при оптимізованих витратах. Шифрування з можливістю пошуку застосовується у сценаріях, де потрібна як конфіденційність, так і доступність відповідних даних [8].

Схема SE дозволяє серверу виконувати пошук у зашифрованих даних від імені клієнта без отримання інформації про відкриті дані. Схеми шифрування з можливістю пошуку зазвичай поділяються на два класи. Деякі схеми безпосередньо шифрують дані відкритого тексту спеціальним чином так, що шифртекст може бути запитаний (наприклад, за ключовими словами). Це призводить до того, що час пошуку лінійно залежить від довжини даних, що зберігаються на сервері. Наприклад, використання n документів із w ключовими словами дає складність, лінійну за кількістю ключових слів у документі $O(nw)$, оскільки кожне ключове слово має бути перевірено на відповідність.

В інших схемах для прискорення процесу пошуку в базах даних використовується індекс, що створюється на основі відкритих текстових даних. Введення індексу дозволяє значно знизити складність пошуку та, таким чином, збільшити продуктивність схеми пошуку. Підвищення продуктивності пошуку досягається з допомогою етапу попередньої обробки. Але при цьому слід пам'ятати, що оскільки індекс будується на даних відкритого тексту, створення індексу не завжди можливе і залежить від даних, які необхідно зашифрувати [6].

Основними методами доказово безпечного шифрування з можливістю пошуку є симетричне шифрування з можливістю пошуку (Symmetric Searchable Encryption – SSE) та шифрування з відкритим ключем із пошуком за ключовими словами (Public Key Encryption with Keyword Search – PEKS). Хоча існують і деякі інші, такі як предикативне шифрування (Predicate Encryption – PE), шифрування скалярного/внутрішнього добутку (Inner Product Encryption – IPE), анонімне шифрування на основі ідентичності (Anonymous Identity-Based Encryption – AIBE), приховане векторне шифрування (Hidden Vector Encryption – HVE), шифрування з ранговим пошуком за множиною ключових слів (Multi-keyword Rank Searchable Encryption – MRSE) [9], гомоморфне шифрування (Homomorphic Encryption – HE).

Дамо коротку характеристику деяких із цих методів.

Шифрування з відкритим ключем із пошуком за ключовими словами (PEKS)

Цей метод вперше був представлений авторами роботи [10] у 2004 р. Ідея їхньої схеми PEKS полягає у використанні шифрування на основі ідентичності (Identity-Based Encryption – IBE), в якому ключове слово діє як посвідчення ідентичності (identity). Завдяки використанню PKE кожний користувач дозволяє створювати доступний для пошуку контент за допомогою відкритого ключа одержувача. Тільки власник закритого ключа може створити лазівку (Trapdoor) для пошуку всередині зашифрованих даних.

Щоб створити зашифрований текст із можливістю пошуку, відправник шифрує своє повідомлення M (загалом, під повідомленнями можна мати на увазі дані у відкритому вигляді, такі як файли, документи або записи в реляційній базі даних) за допомогою стандартної системи відкритого ключа і додає PEKS кожного ключового слова (тобто загальновідомий рядок, зашифрований за допомогою відкритого ключа (K_{pub}), пов'язаного з ключовим словом (w_j) як посвідчення справжності):

$$E_{K_{pub}}(M) \| C_1 = \text{PECS}(K_{pub}, w_1) \| ... \| C_m = \text{PECS}(K_{pub}, w_m) \quad (1)$$

Схематично цей процес можна подати так (рис. 1).

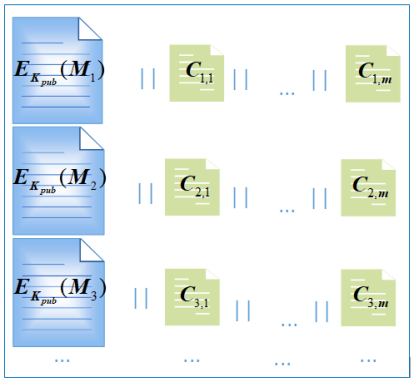


Рис. 1. Процес шифрування повідомлення з можливістю пошуку відповідно до схеми PEKS

Схема PEKS складається з наступних імовірнісних алгоритмів із поліноміальним часом реалізації:

– KeyGen(s). При параметрі безпеки s, що задається, генерується пара відкритий/закритий ключ K_{pub}, K_{priv};

– $c_j \leftarrow \text{PEKS}(K_{pub}, w_j)$. Для відкритого ключа K_{pub} та ключового слова w_j виконується шифрування w_j з можливістю пошуку;

– $T_j \leftarrow \text{Trapdoor}(K_{priv}, w_j)$. При заданому ключовому слові w_j та закритому ключі K_{priv} створюється лазівка T_j;

– $\text{Search}(K_{pub}, C, T_j) \rightarrow \{0,1\}$. Даний алгоритм пошуку/перевірки (Test), враховуючи відкритий ключ K_{pub}, виконане шифрування з можливістю пошуку $c_j \leftarrow \text{PEKS}(K_{pub}, w_j)$ та лазівку T_j, видає значення 1 (істинно), якщо $w_j = w'_j$, (тобто вкладене зашифроване повідомлення містить задане ключове слово), або 0 (хибно) в іншому випадку.

Основні особливості даної схеми [6, 8]:

– Основними сценаріями подібних схем є отримання електронних листів або документів з сервера та дозвіл серверу перенаправляти/маршрутизувати електронні листи. Відсутність взаємодії між відправником та одержувачем.

– Підтримується декілька власників даних (відправників) та один користувач даних (одержувач). Така схема широко відома як архітектура M/S (кілька письменників та один читач).

– Підходить для простору ключових слів поліноміального розміру.

– PEKS захищений від адаптивної атаки за вибраним ключовим словом (PK-CKA2) відповідно до білінійного припущення Діффі – Хеллмана (Bilinear Diffie-Hellman – BDH) у моделі Random Oracle.

– Поштовий сервер вважається чесним та допитливим, тобто правильно виконує всі алгоритми, але може спробувати дізнатися про якусь корисну інформацію.

Обмеження PEKS [6, 8]:

– Потрібен безпечний канал для запобігання прослуховування. Потрібний захищений канал для передачі лазівок, щоб зловмисник не зміг заволодіти лазівкою.

– Для підтримки кількох користувачів даних надсилається те саме повідомлення, зашифроване відкритим ключем кожного передбачуваного користувача, що призводить до надмірності.

– Оскільки простір ключових слів невеликий, PEKS та всі схеми шифрування з можливістю пошуку з відкритим ключем страждають від атак із підбором ключових слів.

– PEKS було розроблено для одноразового використання. Сервер може запам'ятати лазівку та використовувати її для отримання інформації про майбутні електронні листи.

– Не підтримує пошук за кількома ключовими словами.

Предикативне шифрування (PE)

Поняття предикатного шифрування (Predicate Encryption – PE) було вперше представлено у роботі [11]. PE забезпечує детальне керування доступом до зашифрованих даних. Предикатне шифрування – це нова парадигма шифрування з відкритим ключем, яке узагальнює шифрування, що охоплює різні криптографічні примітиви, такі як шифрування на основі ідентичності (IBE) [12 – 15], шифрування з прихованим вектором (HVE) [16, 17] і шифрування на основі атрибутів (ABE) [18]. PE націлений на більш потужні запити, але складність запиту призводить до вищих обчислювальних витрат. У PE секретні/закриті ключі пов'язані з предикатами, а зашифровані тексти пов'язані з атрибутами (I). Користувач може розшифрувати зашифрований текст, якщо предикат закритого (private) ключа набуває значення 1 при застосуванні до атрибуту зашифрованого тексту. PE поставляється у двох версіях: 1) із загальнодоступним індексом; 2) із прихованими атрибутами. Схеми (1) непридатні для шифрування з можливістю пошуку, тому що їм не вистачає властивості анонімності через виток набору атрибутів, за допомогою яких дані зашифровані. Схеми (2) можна використовувати для SE, але вони часто засновані на білінійних парах і тому менш ефективні ніж схеми, засновані на більш простих примітивах.

Схема предикативного шифрування для класу предикатів над множиною атрибутів складається з чотирьох алгоритмів Setup, GenKey, Enc, Dec:

$\mathrm{Setup}(1^n)$ приймає як вхідні дані параметр безпеки 1^n і виводить (головний – master) відкритий ключ *PK* і (головний) секретний ключ *SK*.

$SK_f \leftarrow \mathrm{GenKey}_{SK}(f)$ приймає на вхід головний секретний ключ *SK* та опис предикату $f \in F$; виводить ключ SK_f.

$C \leftarrow \mathrm{Enc}_{PK}(I, M)$ приймає як вхідні дані відкритий ключ *PK*, атрибут $I \in \Sigma$ і повідомлення M в деякому асоційованому просторі повідомлень; повертає зашифрований текст C.

$\begin{cases} M, f(I)=1 \\ \perp, f(I)=1 \end{cases} \leftarrow \mathrm{Dec}_{SK_f}(C)$ приймає як вхідні дані секретний ключ SK_f і зашифрований текст C; повертає або повідомлення M (тільки у тому випадку, коли предикат $f \in F$ і атрибут I пов'язані між собою), або виділений символ $\perp$ (з дуже малою ймовірністю).

Шифрування внутрішнього добутку (IPE)

Криптографічне IPE (Inner Product Encryption) або відоме як обчислення внутрішнього/скалярного добутку найчастіше використовуються в PE, IBE, AIBE та HVE [19]. IPE, представлене в роботах [11, 20], відомо, як криптографічний механізм, що дозволяє більш детально контролювати доступ до даних, що шифруються (механізм, що дозволяє полегшити користувачеві доступ до даних, які задовольняють потреби і вимоги поставленого завдання). В IPE предикати та атрибути представлені у вигляді векторів. Якщо внутрішній/скалярний добуток цих двох векторів дорівнює 0, то предикат дорівнює 1 (наприклад, атрибути відповідають вектору – $\vec{x}$, кожен предикат $f_{\vec{v}}$ відповідає вектору – $\vec{y}$, де $f_{\vec{v}}(\vec{x}) = 1$ тоді і тільки тоді, коли – $\vec{x} \cdot \vec{y} = 0$). Скалярний добуток дозволяє більш складні обчислення диз'юнкцій, багаточленів та формул КНФ (кон'юнктивна нормальна форма)/ДНФ (диз'юнктивна нормальна форма).

У роботах [11, 20] автори запропонували систему над $\square_N$ (для деякого великого цілого числа *N*). Автори робіт [19, 21] надали функції над F_p. Потім у роботах [22 – 25] було

запропоновано досконаліші схеми. Авторам роботи [11] вдалося побудувати схеми приховування атрибутів, які обробляють диз'юнкції предикатів з поліноміальним часом, відмінні від приховування корисного навантаження. Приховування корисного навантаження (payload-hiding) – це поняття безпеки для досягнення високих гарантій рівня безпеки, де зашифрований текст пов'язаний з атрибутом, який приховує всю інформацію, доки не буде отримано секретного ключа для розшифрування. Корисне навантаження та приховування атрибутів трохи відрізняються способом приховування зашифрованого тексту від відкритого тексту. Для приховування атрибута пов'язаний параметр повинен бути прихований разом із зашифрованим текстом, тоді як приховування корисного навантаження потрібно лише приховування відкритого тексту разом із зашифрованим текстом [24].

Анонімне шифрування на основі ідентичності (AIBE)

Анонімне шифрування на основі ідентичності (Anonymous Identity-Based Encryption – AIBE) у своїй стандартній формі може підтримувати лише перевірки на рівність та працює у сценарії/архітектурі M/S. Автори [10] були першими, хто розглянув шифрування з можливістю пошуку в налаштуваннях асиметричного ключа. Вони відзначають, що схема PEKS має тісний зв'язок із AIBE. Згодом їхня схема PEKS була вдосконалена авторами робіт [26, 27]. Автори [28] формалізували AIBE і представили загальну конструкцію SE, перетворивши схему шифрування на основі анонімної ідентифікації на схему шифрування з можливістю пошуку. Більш досконалі схеми IBE, що використовуються для шифрування з можливістю пошуку, було запропоновано авторами робіт [16, 29 – 31]. Щоб дозволити делегування, було введено ієрархічне шифрування на основі ідентичності (HIBE) [32 – 34], у якому закриті ключі та зашифровані тексти пов'язані з упорядкованими списками посвідчень (identities). Пізніше було запропоновано анонімні схеми HIBE (AHIBE) [16, 17, 35, 36]. Автори [28] також запропонували перетворення AHIBE-to-IBE з пошуком за ключовими словами (IBEKS) (hibe-2-ibeks).

Приховане векторне шифрування (HVE)

Приховане векторне шифрування (Hidden Vector Encryption – HVE) – це схема шифрування з відкритим ключем, яка підтримує знаки підстановки всередині ключа. Це дозволяє використовувати різні сценарії застосування. У роботі [37] автори у 2007 р. запропонували першу схему HVE для пошуку у зашифрованих даних. Їх схема допускає кон'юнктивні запити, запити підмножин та діапазонів. У роботі [11] автори розширили список диз'юнкцій, поліноміальних рівнянь та скалярних добутків. HVE можна як узагальнення AIBE [37]. Якщо HVE приховує ключове слово, перетворений PEKS не пропускає жодної інформації про ключове слово, що використовується в алгоритмі шифрування.

Гомоморфне шифрування (HE)

Гомоморфне шифрування (Homomorphic encryption – HE) – це особливий тип шифрування, який дозволяє виконувати операції алгебри над зашифрованими текстами, не розшифровуючи їх. Це робить HE цікавим інструментом пошуку за зашифрованими даними, оскільки над зашифрованими даними можна виконувати осмислені обчислення. Розрізняють криптосистеми частково гомоморфні та повністю гомоморфні. Частково гомоморфна криптосистема дозволяє робити тільки одну з операцій – або додавання, або множення. Повністю гомоморфна криптосистема підтримує виконання обох операцій, тобто, у ній виконуються властивості гомоморфізму як щодо множення, і щодо додавання. Тобто криптосистема є повністю гомоморфною (має і мультиплікативні, і адитивні гомоморфні властивості), якщо:

$$D(E(m_1)\otimes E(m_2)) = m_1 \square m_2;\ D(E(m_1)\oplus E(m_2)) = m_1 + m_2 \qquad (2)$$

де $E()$ – функція шифрування; $D()$ – функція розшифрування; m_1 та m_2 – відкриті тексти;

символи $\otimes$ та $\oplus$ позначають операції множення та додавання над шифртекстами, що відповідають операціям множення та додавання над відкритими текстами.

Більшість схем HE підтримують або додавання [38], або множення [39] за шифротекстом. Схема HE на основі пар, запропонована авторами [40], може виконувати довільну кількість додавань та одне множення. Повністю гомоморфне шифрування (Fully Homomorphic Encryption – FHE), яке може обчислювати довільні функції над зашифрованими даними, запропоновано авторами робіт [41 – 43]. Зазвичай вважається, що FHE може вирішити проблему запиту зашифрованих даних, оскільки будь-які значущі обчислення можуть бути виконані із зашифрованими даними. Однак однією з проблем із FHE є продуктивність, оскільки поточні схеми вимагають великих обчислювальних ресурсів та великих накладних витрат на зберігання. Починаючи з першої схеми FHE, дослідники намагалися зробити схеми ефективнішими, але досі не було запропоновано жодної практичної конструкції [44]. Для застосування можуть використовуватися так звані дещо (певною мірою) гомоморфні схеми шифрування (somewhat homomorphic encryption schemes). Ці схеми більш ефективні, ніж FHE, але допускають лише певну кількість додавань та множень [42, 45]. Основна проблема при частковому або повному використанні HE у тому, що підсумкові схеми пошуку вимагають часу пошуку лінійного за довжиною набору даних. Це занадто повільно для практичного застосування.

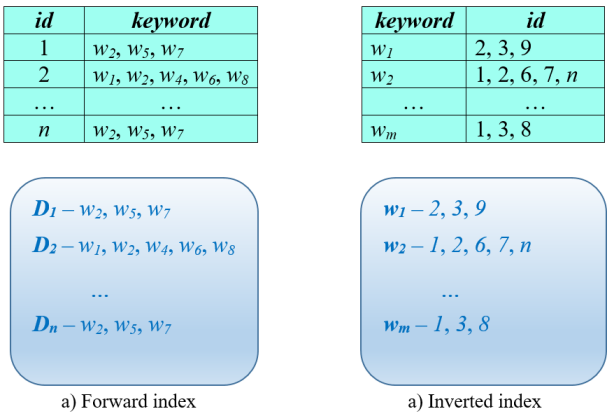
Симетричне шифрування із можливістю пошуку (SSE)

На методі симетричного шифрування з можливістю пошуку (Symmetric Searchable Encryption – SSE) зупинимося докладніше.

Припустимо, що $DB=(D_1,...,D_n)$ – це набір даних деякої БД. Під даними в даному випадку маємо на увазі деякий тип/категорію даних у відкритому вигляді, наприклад, такий як файли, документи, записи/значення атрибутів в реляційній базі даних і т. д. Є деякі похідні елементи даних: $W=(w_1,...,w_m)$, званими ключовими словами w_j. Між усіма ключовими словами W з DB і відповідними ідентифікаторами відповідного документа/запису D_i з DB існує певна відповідність (тобто є відповідності між усіма записами/документами що містять ключове слово w_j, де $j=1,...,m$).

Щоб створити доступний для пошуку зашифрований індекс I для даних, що розглядаються, вилучені з D_i ключові слова w_j зашифровуються (можливо таким способом, який не допускає розшифрування, – наприклад, за допомогою геш-функції) за допомогою секретного ключа K. Застосовується так званий алгоритм індексу збірки BuildIndex. У різних застосовуваних схемах SSE цей алгоритм має особливості реалізації.

Для побудови індексу, як правило, існує два підходи: створення прямого індексу (forward index) та створення зворотного/інвертованого індексу (inverted index) – рис. 2.



a) Forward index a) Inverted index

Рис. 2. Приклади незашифрованого прямого та інвертованого індексу

При першому підході індекс (прямий) будується на основі набору даних $(D_1,...,D_n)$, з кожним з яких пов'язані деякі ключові слова. При другому підході індекс (інвертований) будується за ключовими словами $(w_1,...,w_m)$. Він індексує кожне ключове слово, пов'язане з відповідним(и) D_i.

Прямий індекс є індексом для кожного D_i (рис. 2, *а*) і, природно, скорочує час пошуку до кількості таких D_i, тобто $O(n)$. Це пов'язано з тим, що під час запиту має оброблятися один індекс для кожного D_i.

В даний час переважним методом досягнення сублінійного часу пошуку є використання інвертованого/зворотного індексу, який є індексом за ключовим словом у базі даних (рис. 2, *б*). Інвертований індекс є індексом для кожного окремого слова в базі даних, а не для кожного D_i. Залежно від того, скільки інформації ми готові видати, складність пошуку можна скоротити до $O(\log w')$ (наприклад, за допомогою геш-дерева) або $O(|D(w)|)$ в оптимальному випадку, де $D(w)$ кількість D_i, що містить ключове слово w_j. Ідея використання інвертованого індексу скорочує час пошуку до кількості D_i, що містить ключове слово. Це не тільки сублінійно, а й оптимально [6].

Самі дані (у згаданому вище контексті) зашифровуються алгоритмом Enc за допомогою ключа K' (досить часто $K \neq K'$). Як Enc може використовуватися один із симетричних шифрів.

Зашифрований індекс (I) та зашифровані дані (C) зберігаються на сервері:

$$I = \text{BuildIndex}_K(DB = (D_1,...,D_n), W = (w_1,...,w_m)); \; C = \text{Enc}_{K'}(D_1,...,D_n). \qquad (3)$$

Сервер зазвичай вважається чесним, але допитливим (honest-but-curious), тобто йому можна довіряти у дотриманні протоколів зберігання та запитів, але слід враховувати, що він намагається отримати якнайбільше інформації. «Чесність» сервера виявляється в тому, що він не видаляє і не псує збережені дані, чесно діє за наперед визначеним протоколом, тобто виконує операцію пошуку зашифрованих даних за заданим ключовим словом/індексом і відправляє відповідні дані, пов'язані з запитами. Його «допитливість» полягає в тому, що решту інформації сервер намагається дізнатися з пошукових запитів та індексу.

Для пошуку необхідних зашифрованих даних створюється так звана лазівка/люк (Trapdoor), яку іноді називають токеном [8, 46] пошуку для ключового слова $T = \text{Trapdoor}_K(f)$, де f є предикатом на множині W ($f(w_1,...,w_m)$), що дозволяє серверу перевірити, чи є зашифроване ключове слово в результаті пошуку. За допомогою T сервер може шукати індекс, використовуючи алгоритм пошуку (Search), і дивитися, чи задовольняють зашифровані ключові слова предикату f, і якщо задовольняють, то алгоритм повертає відповідні зашифровані дані (див. рис. 3). Наприклад, f може визначити, чи міститься конкретне ключове слово в індексі, а більш складний може визначити, чи дорівнює 0 внутрішній добуток ключових слів в індексі і цільовому наборі ключових слів [47].

Загальна модель схеми шифрування з можливістю пошуку з урахуванням індексу представлено рис. 3.

Слід зазначити, що можуть бути невеликі відхилення. Наприклад, деякі схеми не вимагають повного списку ключових слів для побудови індексу [6].

Загалом можна виділити такі основні елементи (алгоритми з поліноміальним часом виконання), описаної вище схеми SSE:

$K \leftarrow \text{Keygen}\left(1^k\right)$: алгоритм генерації ключів, що запускається власником даних (користувачем). Він приймає параметр безпеки (security parameter) k як вхідні дані і видає секрет-

ний ключ K. Якщо для шифрування використовується ще другий ключ K', такий що $K \neq K'$, то для нього аналогічно, але з використанням параметра k' генерується ключ K';

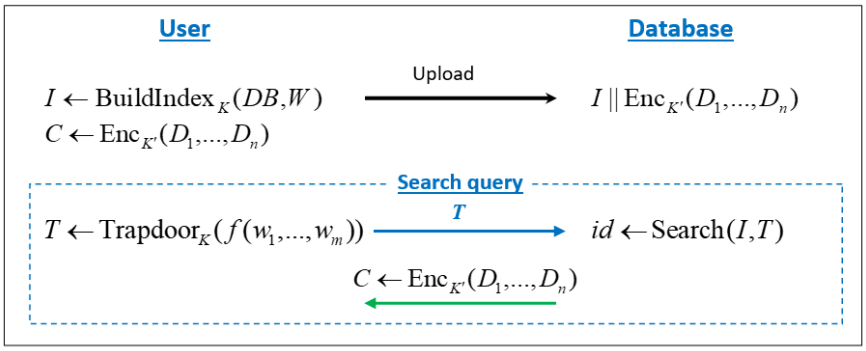


Рис. 3. Загальна модель схеми шифрування з можливістю пошуку на основі індексу

$I \leftarrow \text{BuildIndex}_K(DB,W)$: алгоритм, який запускає власник даних (користувач). Він приймає як вхідні дані секретний ключ K і набір даних $DB = (D_1,...,D_n)$, $W = (w_1,...,w_m)$, а видає безпечний індекс I;

$C \leftarrow \text{Enc}_{K'}(D_1,...,D_n)$: алгоритм, який запускає власник даних (користувач). Він приймає як вхідні дані секретний ключ K' і набір даних $DB = (D_1,...,D_n)$, а видає послідовність зашифрованих даних $C = (c_1,...,c_n) = Enc_{K'}(D_1,...,D_n)$;

$T \leftarrow \text{Trapdoor}_K(f(w_1,...,w_m))$: алгоритм, що запускається користувачем для створення лазівки для заданого ключового слова (внутрішнього добутку ключових слів). Він приймає як вхідні дані секретний ключ K і ключове(і) слово(а) w_j і видає лазівку T;

$id \leftarrow \text{Search}(I,T)$: алгоритм, який запускається сервером для пошуку D_i в DB, що містять ключове слово w_j. Він приймає як вхідні дані зашифрований індекс I для набору даних DB і лазівку T і виводить набір ідентифікаторів id відповідного D_i;

$D_i \leftarrow \text{Dec}_K(c_i)$: алгоритм, який запускається клієнтом для відновлення D_i. Він приймає як вхідні дані секретний ключ K і зашифрований текст c_i, а видає розшифрований D_i.

Моделі схем SE

Для розгляду принципів існуючих захищених пошукових систем слід враховувати деякі особливості сценаріїв їхнього функціонування. Так, наприклад, у роботі [7] виділяють два такі сценарії. А саме двосторонній сценарій, в якому один користувач діє як постачальник/власник даних та запитувач (клієнт). Такий сценарій моделює програму аутсорсингу хмарного сховища, в якому клієнт завантажує файли у хмару, які він може пізніше запросити. На рис. 4 зображено такий сценарій.

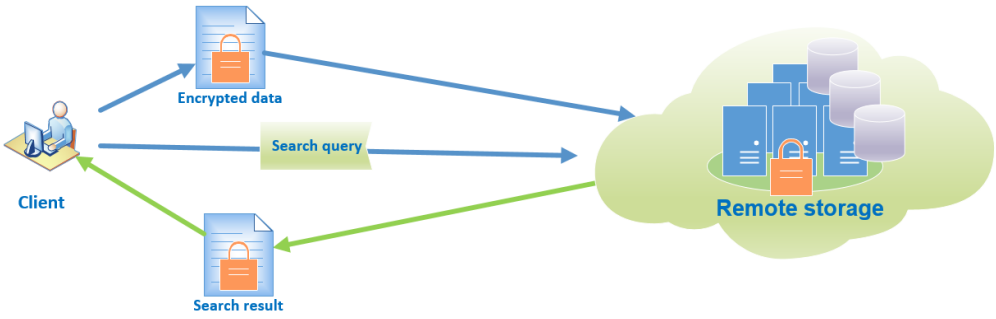


Рис. 4. Модель схеми SE із двома учасниками

У моделі схеми SE із двома учасниками клієнт має право знати всю інформацію у базі даних, тому необхідно враховувати лише захист від допитливого сервера.

Тристоронній сценарій (модель), передбачає наявність трьох учасників: постачальника (довіреного власника даних – trusted data owner), групу користувачів, яким дозволено пошук (які запитують), та напівдовірений (semi-trusted)/чесний, але допитливий сервер. Роль кожного з них така:

– Власник даних (Data owner). Власник даних хотів би передати на аутсорсинг набір даних $DB = (D_1, ..., D_n)$ разом із деякими ключовими словами $W = (w_1, ..., w_m)$. При цьому він має особливим чином зашифрувати ці дані та ключові слова, щоб згодом легко їх шукати, а потім надіслати зашифровані дані на сервер;

– Користувач даних (Data user). Авторизований користувач виконує пошук у наборі даних $DB = (D_1, ..., D_n)$, що містять певне ключове слово, для чого він відправляє лазівку T цього ключового слова на сервер. Після пошуку сервер повертає користувачеві відповідні дані ($id \leftarrow \text{Search}(I, T)$), що містять вказане ключове слово;

– Сервер (Server) Сервер здійснює пошук. Коли сервер отримує лазівку ключового слова запиту від користувача, він шукає зашифровані тексти, а потім повертає відповідні дані користувачеві. Передбачається, що сервер є чесним, але допитливим.

Слід пам'ятати, що безпечна система пошуку для одного сценарію не поширюється автоматично на інший сценарій.

На рис. 5 показано модель схеми SSE із трьома учасниками.

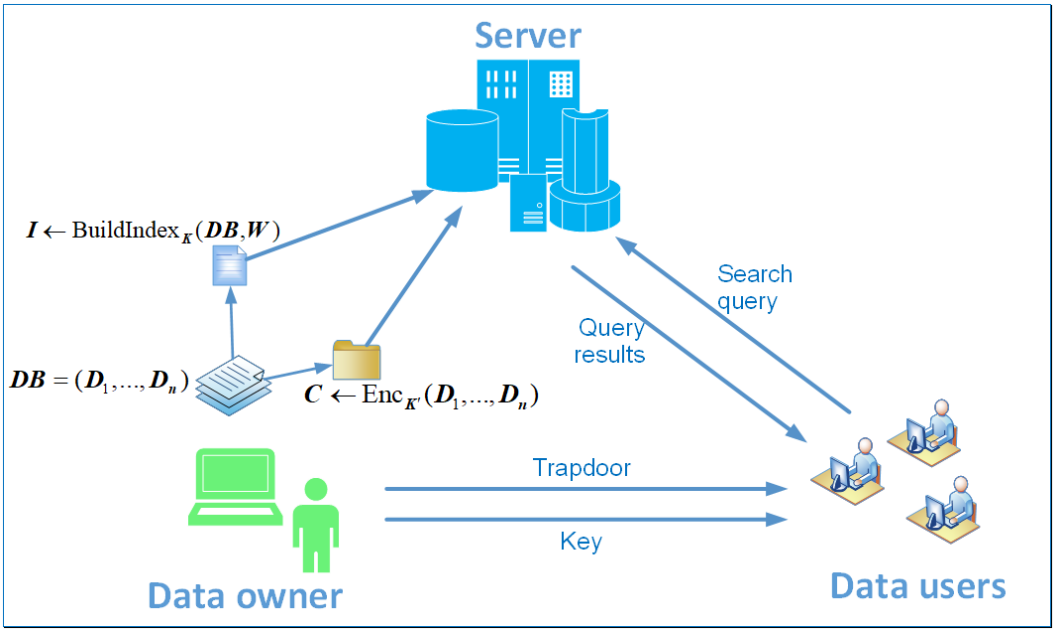


Рис. 5. Модель схеми SSE із трьома учасниками

Архітектури схем SE

Як зазначалося раніше, схеми SE будуються на основі клієнт-серверної моделі. Сервер зберігає зашифровані дані одного чи кількох клієнтів (так званих письменників). Щоб запросити контент із сервера, один або кілька клієнтів (читачів) можуть створювати лазівки для сервера, який потім шукає від імені відповідного клієнта. Як результат можуть мати місце наступні чотири архітектури SE [6]:

– один письменник / один читач (S/S);
– багато письменників / один читач(M/S);
– один письменник / багато читачів (S/M);
– багато письменників / багато читачів (M/M).

В залежності від архітектури схема SE підходить або для аутсорсингу даних (S/S), або для спільного використання даних (M/S, S/M, M/M).

Схеми симетричного шифрування з можливістю пошуку дозволяють одному користувачеві читати та записувати дані (S/S), тобто дозволяють власнику секретного ключа створювати доступні для пошуку зашифровані тексти та лазівки [8]. Перша схема S/S була запропонована у роботі [48].

У схемі шифрування з відкритим ключем (PKE) секретний ключ розшифровує всі повідомлення, які зашифровані відповідним відкритим ключем. Таким чином, PKE допускає розрахований на багато користувачів запис, але тільки власник секретного ключа може виконувати пошук. Для цього потрібна архітектура M/S. Перша схема M/S належить авторам роботи [10], які запропонували схему шифрування з відкритим ключем та пошуком за ключовими словами (PEKS). Серед доступних криптографічних примітивів з відкритим ключем вони використовували шифрування на основі ідентичності (IBE) як базову схему. Вони використовували відкритий ключ, відповідний особі користувача, для шифрування файлу даних та ключових слів, що містяться у файлі даних. Схема була розроблена для ефективного отримання зашифрованих листів з поштового сервера. Власник даних або, точніше, відправник хоче надіслати електронний лист, зашифрований за допомогою відкритого ключа одержувача, разом із ключовими словами, зашифрованими за допомогою того ж ключа. Надіслана електронна пошта зберігається на поштовому сервері одержувача, і одержувачу потрібен механізм для ефективного вилучення з них лише потрібних електронних листів. Цей ефективний механізм є методом шифрування з можливістю пошуку, без якого одержувач спочатку повинен завантажити всі електронні листи, а потім розшифрувати їх локально, використовуючи свій закритий ключ. Але за допомогою методу шифрування з можливістю пошуку одержувач може витягувати електронні листи відповідно до своїх поточних вимог, і йому необхідно розшифрувати лише вибрані електронні листи. Весь цей механізм підходить для тих сценаріїв, де мається один передбачуваний одержувач, однак обмежень на кількість відправників немає. PEKS використовується як назва класу схем M/S [6].

Деякі схеми SE розширюють налаштування */S шляхом дозволу розрахованого на багато користувачів читання (*/M). Це розширення призводить до необхідності розповсюдження (розподілу) секретного ключа, щоб дозволити кільком користувачам здійснювати пошук в зашифрованих даних. Деякі схеми SE використовують спільне використання ключів. Інші схеми використовують розподіл ключів, повторне шифрування проксі (proxy re-encryption) або інші методи вирішення проблеми.

При цьому важливою вимогою, що пред'являється до схем з безліччю читачів, є відкликання (анулювання) користувача. У роботі [49] автори, наприклад, розширюють свою однокористувацьку схему з широкомовним шифруванням (broadcast encryption – BE [50]) до розрахованої на багато користувачів схеми з безліччю читачів (S/M). Але оскільки тільки один ключ є спільним для всіх користувачів, кожне відкликання вимагає розповсюдження нового ключа серед користувачів, що викликає великі накладні витрати на відкликання. В інших схемах у кожного користувача може бути свій ключ, що спрощує відкликання користувача і робить його ефективнішим.

В цілому ж, проєктування захищеної системи пошуку – це баланс/компроміс між безпекою, функціональністю, ефективністю/продуктивністю та зручністю використання [6, 7].

Оскільки безпека ніколи не буває безкоштовною, завжди існує компроміс між безпекою (security) з одного боку та ефективністю (efficiency) та виразністю запитів (query expressiveness) з іншого. Схеми шифрування з можливістю пошуку, в яких використовується модель безпеки з сильнішим противником (зловмисником, порушником), ймовірно, матимуть більш високу складність. Коли у схемі покращується один із аспектів, зазвичай це призводить до погіршення одного або відразу кількох інших аспектів.

Безпека зазвичай визначається інформацією про дані користувача, яка стає відома атакуючому в процесі роботи схеми. Існує два типи сутностей, які можуть становити загрозу

безпеці бази даних: дійсний/законний користувач (valid user), відомий як інсайдер (insider), що виконує одну або кілька ролей, та сторонній (outsider), який може відслідковувати та потенційно змінювати мережеву взаємодію між дійсними користувачами. При цьому є зловмисники (порушники), які є напівчесними (або чесними, але допитливими), тобто вони наслідують запропоновані протоколи, але можуть пасивно намагатися дізнатися додаткову інформацію з повідомлень, які вони спостерігають. І є зловмисні порушники, тобто ті, хто активно бажає виконувати будь-які дії, необхідні для отримання додаткової інформації або впливу на роботу системи. Слід зазначити, більшість активних досліджень у сфері технології захищеного пошуку розглядає напівчесний захист від постійного внутрішнього противника [7].

Виразність схеми запитів визначає, які пошукові запити підтримуються, якого типу, як вони виражаються (зазвичай за допомогою стандартних мов, наприклад, SQL). У сучасних підходах часто буває так, що більш виразні запити призводять до меншої ефективності, або до меншої безпеки.

Ефективність вимірюється обчислювальною та комунікаційною складністю схеми. Вона поряд з обчислювальними та комунікаційними витратами залежить від структури даних та механізмів індексування у базі даних.

Розглянемо ці аспекти.

Проблеми конфіденційності у схемах SE

Для схем шифрування з можливістю пошуку характерне власне поняття безпеки, оскільки всі найпотужніші криптографічні атаки на такі системи будуються на основі експлуатації витоків логічного виводу (inference), до яких схильна більшою чи меншою мірою майже будь-яка практична SE-схема. У процесі роботи SE-схеми відбувається постійна взаємодія між клієнтом та сервером. При спостереженні та статистичному аналізі запитів і відповідей сервера, що надходять від клієнта, зловмисник може отримувати істотну кількість непрямих даних (іноді званих метаданими) про зашифровану інформацію користувачів. Припустимо, що зловмисник має доступ до зашифрованих даних на сервері та може спостерігати запити користувачів. Також він знає, скільки записів повертається у відповідь на кожен запит. Якщо ця кількість унікальна, то запит також унікальний і його можна розрізнити серед безлічі всіляких запитів.

Безпека зазвичай пов'язується з інформацією, яка в процесі роботи схеми розкривається або просочується зловмиснику, який має доступ до сервера бази даних. У схемі шифрування з можливістю пошуку має бути гарантована безпека набору даних та ключових слів, що зберігаються на сервері, а також має бути забезпечена безпека ключових слів запиту.

У схемах SE може мати місце витік інформації, яку можна розділити на три групи [6]:

– Інформація індексу (Index information) відноситься до інформації про ключові слова, що містяться в індексі. Інформація про індекс просочується із збереженого шифртексту (ciphertext)/індексу. Ця інформація може включати кількість ключових слів на документ/базу даних, кількість документів, довжину документів, ідентифікатори документів і/або подібність документів;

– Шаблон пошуку (Search pattern) відноситься до інформації, яка може бути отримана в наступному сенсі: за умови, що два пошуки повертають одні й ті самі результати, потрібно визначити, чи використовують ці два пошуки одне й те саме ключове слово/предикат. Або іншими словами шаблон пошуку визначається як будь-яка інформація, яку можна отримати, знаючи, чи відносяться два результати пошуку до одного й того самого ключового слова. Використання детермінованих лазівок безпосередньо призводить до витоку шаблону пошуку. Доступ до шаблону пошуку дозволяє серверу використовувати статистичний аналіз та (можливо) визначати інформацію про ключові слова запиту або самі ключові слова;

– Шаблон доступу (Access pattern) відноситься до інформації, яка передбачається результатами запиту. Наприклад, один запит може повернути D_i, а інший запит може повернути D_i і ще 5, 10 і т. д. інших $D_l \in DB$. Це означає, що предикат, який використовується

у першому запиті, є суворішим, ніж предикат в інших запитах. Шаблон доступу: визначається як послідовність результатів пошуку $(DB(w_1),...,DB(w_m))$, де $DB(w_j)$ – результати пошуку w_j. Інакше кажучи, $DB(w_j)$ – це набір даних у DB, що містить ключове слово w_j.

Слід зазначити, що у багатьох схемах [6] є витік як мінімум шаблону пошуку та шаблону доступу. Хоча більшість схем дотримуються визначення безпеки, що використовується у традиційному шифруванні з можливістю пошуку. А саме, в них виконується вимога того, щоб з віддалених даних і індексів нічого не просочувалося, крім результату і шаблону пошукових запитів. Схеми SE не повинні пропускати ключові слова відкритого тексту ні в лазівку, ні в індекс.

Щоб формально визначити безпеку схеми, було запропоновано багато різних моделей безпеки.

Коли автори роботи [48] запропоновували першу схему SE, не було формальних визначень безпеки для конкретних SE. Проте вони довели, що їхня схема є безпечною з погляду нерозрізненості для атак на основі підібраного відкритого тексту (indistinguishability under chosen plaintext attack – IND-CPA), інакше кажучи, є IND-CPA надійною [46]. Неформально схема шифрування є IND-CPA надійною, якщо зловмисник A не може розрізнити шифрування двох довільних повідомлень (вибраних самим A), навіть якщо може адаптивно посилати запити провісника (oracle) шифрування. Інтуїтивно це означає, що схема безпечна з погляду IND-CPA, якщо результуючі зашифровані тексти не містять навіть часткової інформації про відкриті тексти. Це визначення IND-CPA гарантує, що зашифровані тексти не допускають витоку інформації. Однак у SE основний витік інформації походить з лазівки/запиту, що не враховується у безпеці IND-CPA. Таким чином, безпека IND-CPA не співвідноситься з поняттям безпеки SE.

Перше поняття безпеки в контексті SE було введено автором роботи [51], який визначив безпеку індексів як семантичну безпеку (нерізницю) від адаптивних атак за вибраним ключовим словом (semantic security (indistinguishability) against adaptive chosen keyword attacks – IND1-CKA). IND1-CKA гарантує, що A не може відтворити вміст документа з його індексу. Схема безпеки IND1-CKA створює індекси, що містять однакову кількість слів для документів однакового розміру (на відміну документів різного розміру). Це означає, що за наявності двох зашифрованих документів рівного розміру та індексу, A не може вирішити, який із документів закодований в індексі. IND1-CKA був запропонований для «захищених індексів», захищеної структури даних з безліччю застосувань поряд із SSE. При цьому в роботі [51] зазначається, що IND1-CKA не вимагає, щоб лазівки були безпечними, оскільки це необхідно не для всіх застосувань безпечних індексів.

Автори роботи [52] представили нове визначення IND-CKA, засноване на моделюванні, яке є суворішою версією IND1-CKA у тому сенсі, що зловмисник не може відрізнити навіть індекси двох документів з різним розміром. Для цього потрібно, щоб документи різного розміру мали індекси, що містять однакову кількість слів. Пізніше було представлено визначення безпеки IND2-CKA, яке захищає розмір документа, як і визначення, дане в роботі [52], але все ж таки не забезпечує безпеку для лазівок. Обидва визначення безпеки IND1/2-CKA вважаються слабкими в контексті SE, оскільки вони не гарантують безпеку лазівок, тобто вони не гарантують, що сервер не зможе відновити інформацію про запитані слова з лазівки або самі слова. У роботі [49] переглянуто існуючі визначення безпеки. Її автори вказали, що попередні визначення не підходять для SSE і що безпека індексів та безпека лазівок нерозривно пов'язані. Вони представили дві нові змагальні моделі для шифрування з можливістю пошуку: неадаптивну (IND1-CKA) та адаптивну (IND2-CKA), які на сьогодні широко використовуються як стандартні визначення для симетричного шифрування з можливістю пошуку. Інтуїтивно зрозуміло, що визначення вимагають, щоб з файлів та індексів, що віддалено зберігаються, нічого не просочувалося, крім результату і шаблону пошукових запитів. Визначення безпеки IND-CKA1/2 включають безпеку для лазівок і гарантують, що лазівки не пропуска-

ють інформацію про ключові слова (за винятком тієї, яку можна вивести з шаблонів пошуку та доступу). Неадаптивні визначення гарантують безпеку схеми лише у тому випадку, якщо клієнт генерує всі запити одночасно. Це може бути нездійсненним для певних (практичних) сценаріїв. Адаптивне визначення дозволяє зловмиснику A вибирати свої запити залежно від раніше отриманих лазівок та результатів пошуку. Таким чином, IND-CKA2 вважається надійним визначенням безпеки SSE.

В асиметричному варіанті (з відкритим ключем) автори схеми [10] не гарантують безпеку лазівок, оскільки зазвичай лазівки генеруються з використанням відкритого ключа. Визначення в цьому контексті гарантує, що жодна інформація про ключове слово не буде отримана тільки в тому випадку, якщо для цього слова не буде доступна лазівка. Зловмисник не повинен мати можливості розрізняти шифрування двох вибраних ним ключових слів виклику (challenge keywords), навіть якщо дозволено отримувати лазівки для будь-яких ключових слів (крім ключових слів виклику). Дотримуючись попереднього поняття, для позначення невідмінності від адаптивних атак за вибраним ключовим словом схем з відкритим ключем, використовується позначення PK-CKA2 (public key CKA2).

Інші визначення безпеки були введені та/або адаптовані для SE наступним чином:

– Універсальна компоновність (universal composability – UC) – це модель загального призначення, в якій йдеться про те, що протоколи залишаються безпечними, навіть якщо вони довільно складені з інших екземплярів того ж чи інших протоколів.

– Вибірково безпечна (selectively secure – SEL-CKA) – ця модель схожа на PK-CKA2, але зловмисник A повинен зафіксувати ключові слова пошуку на початку забезпечення безпеки, а не після першої фази запиту.

– Цілком безпечна (fully secure – FS) – це визначення безпеки в контексті SSE, введене в роботі [47], яке не дозволяє витекти нічому, крім шаблону доступу.

Ефективність схем SE

Ефективність вимірюється обчислювальною та комунікаційною складністю схеми, а саме, даний аспект фокусується на обчислювальній складності алгоритмів шифрування/генерації індексу (фаза завантаження) та алгоритмів пошуку/тестування (фаза запиту). Для об'єктивного порівняння схем використовуються такі метрики, як кількість необхідних операцій, складність оновлення або інтерактивності (кількість раундів) та деякі інші. На продуктивність та зручність використання впливають структури даних бази даних та механізми індексування, а також необхідні обчислювальні та мережеві витрати.

З точки зору SSE, складність пошуку в деяких схемах лінійна за кількістю документів, що зберігаються на сервері. Хоча в деяких схемах досягається сублінійний час пошуку, при якому складність пошуку знаходиться в логарифмічній залежності кількості ключових слів у всіх документах. При цьому необхідно пам'ятати та враховувати, що документи/набір даних слід акуратно динамічно оновлювати, оскільки індекс пошуку прив'язаний до ключових слів. Отже, питання, як побудувати ефективну динамічну схему SSE, є відкритим.

З точки зору PKES велика кількість схем заснована на спарюванні (pairing)/білінійних відображеннях (bilinear maps). У результаті ці схеми є неефективними, тому що неефективними є алгоритми білінійних відображень.

Схеми симетричного шифрування з можливістю пошуку (SSE) швидше, ніж схеми шифрування з відкритим ключем та пошуком за ключовими словами (PEKS) [8]. Тому деякі автори [7] рекомендують по-можливості уникати повільніших операцій з відкритим ключем або мінімізувати їх на користь швидших примітивів із симетричним ключем. Однак, слід пам'ятати, що розподіл секретних ключів між усіма користувачами, враховуючи необхідність періодичного відкликання користувача і розповсюдження нового ключа серед користувачів, є досить складною процедурою, що викликає великі накладні витрати.

Захищений пошук швидко розвивається з 2000 р., переходячи від лінійних запитів на рівність за статичними даними до складного пошуку за динамічними даними. Сьогодні накладні витрати становлять від 30 до 500 % порівняно із стандартним SQL [7].

Порівняння кількох класичних схем SSE з погляду найгіршого часу паралельного пошуку за ключовим словом [53, 54] наведено у табл. 1.

Таблиця 1

Схема	Часова складність алгоритму пошуку	Часова складність алгоритму побудови індексу	Безпека
Song, et al.[48]	$O(n/p)$	N/A	IND-CPA
Goh[51]	$O(n/p)$	$O(n)$	IND-CKA1
Chang, et al. [52]	$O(n/p)$	$O(mn)$	IND-CKA1
Curtmola, et al.[49](SSE-1)	$O(r)$	$O(m+n)$	IND-CKA1
Curtmola, et al.[49](SSE-2)	$O(r)$	$O(mn)$	IND-CKA2
Liesdonk, et al. [55]	$O(n)$	$O(mn)$	IND-CKA2
Kurosawa, et al. [56]	$O(n)$	$O(mn)$	UC
Kamara, et al. [46]	$O(r)$	$O(m+n)$	IND-CKA2
Kamara, et al. [54]	$O((r/p)\log n)$	$O(mn)$	IND-CKA2

Де IND-CPA – невиразність для атак на основі підібраного відкритого тексту; IND1-CKA – семантична безпека (невідмінність) від адаптивних атак за вибраним ключовим словом; IND-CKA1 – неадаптивна модель, у якій зловмисник не враховує лазівки та результати попередніх пошуків, коли вибирає складні пошукові запити; IND-CKA2 – це адаптивна модель, в якій зловмисник вибирає свої пошукові запити, знаючи раніше отримані лазівки та результати пошуку; n – потужність множини DB, r – кількість D_i (документів), що містять ключове слово запиту w_j, m – розмір простору ключових слів, p – кількість ядер, N/A (not available) – немає відомостей (дані відсутні).

У той же час, наприклад, ефективність схеми PEKS [10] характеризується такими значеннями: шифрування вимагає від сервера виконання одного обчислення симетричної пари простого порядку p, двох зведень у ступінь e та застосування двох геш функцій h для кожного ключового слова; складність пошуку – лінійна (одне білінійне відображення, одна геш функція) за кількістю ключових слів у документі.

Виразність запитів у схемах SE

Проведено дослідження щодо розширення виразності запитів. Щоб зробити схеми більш практичними, підтримуються не тільки точний пошук за одним ключовим словом, але також нечіткий пошук за ключовими словами, пошук за діапазоном та пошук за підмножиною. При цьому результати запиту можуть також оптимізуватися.

Наприклад, ранжований пошук за ключовими словами знаходить найближчі результати, а пошук, що перевіряється за ключовими словами, перевіряє правильність і повноту результатів. Однак багато схем покращують виразність запитів за рахунок ефективності чи безпеки. Тому в майбутніх дослідженнях доцільно звернути увагу на компроміс між виразністю запитів та ефективністю чи безпекою. У схемі SE є такі варіанти компромісів [6]: безпека та виразність запитів, ефективність та виразність запитів.

Вирішення проблеми захищеного пошуку вимагає тісної взаємодії між фахівцями в галузі криптографії, розробниками захищеного пошуку та експертами з баз даних [7].

Висновки

Широке поширення конфіденційних даних у відкритих інформаційних та комунікаційних інфраструктурах стимулювало дослідження в галузі безпечного управління даними та підвищило їх актуальність. На підставі проведеного аналізу було виявлено, що методи SSE та PEKS є найбільш популярними методами SE, які використовуються серед інших методів шифрування з можливістю пошуку.

З моменту запропонування перших схем SSE та PEKS область досліджень шифрування з можливістю пошуку привернула значну увагу. Прогресу було досягнуто в наступних трьох основних напрямках:

1. *Виразність запитів*. Було проведено багато досліджень щодо розширення виразності запитів. Щоб зробити схеми більш практичними, підтримується не тільки точний пошук за одним ключовим словом, але також нечіткий пошук за ключовими словами, пошук за діапазоном та пошук за підмножиною. Однак багато схем покращують виразність запитів за рахунок ефективності чи безпеки. Тому майбутні дослідження повинні звернути увагу на компроміс між виразністю запитів та ефективністю чи безпекою.

2. *Ефективність.* З точки зору SSE, складність пошуку в деяких схемах лінійна за кількістю документів, що зберігаються на сервері. Однак у деяких схемах досягається сублінійний час пошуку, в якому складність пошуку логарифмічна за кількістю ключових слів у всіх документах. З настанням ери великих даних великі обсяги даних тепер потрібно зберігати на серверах. В результаті виникає закономірне питання про те, як ефективно працювати з великомасштабними даними, а отже, цей напрямок також вважається перспективним для подальшої роботи. Більше того, документи не можна гнучко оновлювати, оскільки пошуковий індекс прив'язаний до ключових слів. Отже, питання про те, як побудувати ефективну динамічну схему SSE, є ще одним напрямком майбутньої роботи.

З погляду PKES, велика кількість схем ґрунтується на парних відображеннях. У результаті ці схеми є неефективними, тому що парні (білінійні) відображення є неефективними алгоритмами. Таким чином, побудова практичних схем PKES є напрямом майбутніх робіт.

3. *Безпека.* Хоча практично всі схеми SE забезпечують безпеку, вони не використовують загальну модель безпеки. Тобто різні схеми використовують різні моделі безпеки при різних припущеннях. Тому завжди складно порівнювати їхню захищеність. Тому розробка деякої стандартної моделі безпеки для схем SE є перспективним напрямом майбутніх досліджень. Крім того, більшість схем компрометують шаблон пошуку та шаблон доступу. Таким чином, побудова ефективної схеми, що не допускає витоку шаблону пошуку та шаблону доступу, є ще одним перспективним напрямом досліджень.

Список літератури:

1. Abadi D., Ailamaki A., Andersen D., Bailis P., Balazinska M., Bernstein P., Boncz P., Chaudhuri S., et al. The Seattle Report on Database Research. ACM SIGMOD Record. 2019. 48. P. 44–53.
2. General Data Protection Regulation GDPR. URL: https://gdpr-info.eu/ (дата звернення: 12.06.2022).
3. Payment Card Industry (PCI) Data Security Standard. Requirements and Testing Procedures Version 4.0. 2022. URL: https://www.pcisecuritystandards.org/documents/PCI-DSS-v4_0.pdf (дата звернення: 12.06.2022).
4. Atchinson B. K., Fox D. M. From the field: the politics of the health insurance portability and accountability act // Health affairs. 1997. 16(3). P. 146-150.
5. Scholl M., Stine K., Hash J., Bowen P., Johnson A., et al. NIST Special Publication 800-66 Revision 1. An Introductory Resource Guide for Implementing the Health Insurance Portability and Accountability Act (HIPAA) Security Rule. 2008. URL: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-66r1.pdf (дата звернення: 12.06.2022).
6. Bösch, C., Hartel, P., Jonker, W., Peter, A. A survey of provably secure searchable encryption // ACM Computing Surveys (CSUR). 2014. 47(2). P. 1–51.
7. Fuller B., Varia M., Yerukhimovich A., Shen E., Hamlin A., Gadepally V., Shay R., Mitchell J. D., Cunningham R. K. Sok: Cryptographically protected database search // 2017 IEEE Symposium on Security and Privacy (SP), 2017. P. 172–191. https://doi.org/10.1109/SP.2017.10.
8. Gupta B., Mamta Secure Searchable Encryption and Data Management (1st ed.). CRC Press. 2021. 116 p. https://doi.org/10.1201/9781003107316.
9. Li R., Xu Z., Kang W., Yow K. C., Xu C. Z. Efficient multi-keyword ranked query over encrypted data in cloud computing // Future Generation Computer Systems. 2014. 30. P. 179–190.
10. Boneh D., Crescenzo G. D., Ostrovsky R., Persiano G. Public Key Encryption with Keyword Search // Cachin, C., Camenisch, J.L. (eds) Advances in Cryptology – EUROCRYPT 2004. EUROCRYPT 2004. Lecture Notes in Computer Science, 2004. Vol 3027. P. 506–522. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-24676-3_30/

11. Katz J., Sahai A., Waters B. Predicate encryption supporting disjunctions, polynomial equations, and inner products // Smart, N. (eds) Advances in Cryptology – EUROCRYPT 2008. EUROCRYPT 2008. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 2008. Vol. 4965. P. 146–162. https://doi.org/10.1007/978-3-540-78967-3_9.

12. Shamir A. Identity-Based Cryptosystems and Signature Schemes // Blakley, G.R., Chaum, D. (eds) Advances in Cryptology. CRYPTO 1984. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg. 1985. Vol. 196. P. 47–53. https://doi.org/10.1007/3-540-39568-7_5.

13. Boneh D., Franklin M. Identity-Based Encryption from the Weil Pairing // Kilian, J. (eds) Advances in Cryptology – CRYPTO 2001. CRYPTO 2001. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg. 2001. Vol 2139. P. 213–229. https://doi.org/10.1007/3-540-44647-8_13.

14. Boneh D., Franklin M. Identity-based encryption from the Weil pairing // SIAM journal on computing. 2003. 32(3). P. 586–615.

15. Cocks C. An Identity Based Encryption Scheme Based on Quadratic Residues // Honary, B. (eds) Cryptography and Coding. Cryptography and Coding 2001. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg. 2001. Vol 2260. P. 360–363. https://doi.org/10.1007/3-540-45325-3_32.

16. Boyen X., Waters B. Anonymous Hierarchical Identity-Based Encryption (Without Random Oracles) // Dwork, C. (eds) Advances in Cryptology – CRYPTO 2006. CRYPTO 2006. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg. 2006. Vol 4117. P. 290–307. https://doi.org/10.1007/11818175_17.

17. Shi E., Bethencourt J., Chan T. H., Song D., Perrig A. Multi-dimensional range query over encrypted data. // 2007 IEEE Symposium on Security and Privacy (SP'07). IEEE, 2007. P. 350–364. https://doi.org/10.1109/SP.2007.29.

18. Sahai A., Waters B. Fuzzy Identity-Based Encryption // Cramer, R. (eds) Advances in Cryptology – EUROCRYPT 2005. EUROCRYPT 2005. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg. 2005. Vol 3494. P. 457–473. https://doi.org/10.1007/11426639_27.

19. Lewko A., Okamoto T., Sahai A., Takashima K., Waters, B. Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption // Gilbert, H. (eds) Advances in Cryptology – EUROCRYPT 2010. Lecture Notes in Computer Science, 2010. Vol 6110. P. 62-91. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-13190-5_4.

20. Katz J., Sahai A., Waters B. Predicate encryption supporting disjunctions, polynomial equations, and inner products // Journal of cryptology. 2013. 26(2). P. 191-224. https://doi.org/10.1007/s00145-012-9119-4.

21. Okamoto T., Takashima K. Hierarchical Predicate Encryption for Inner-Products // Matsui, M. (eds) Advances in Cryptology. ASIACRYPT 2009. Lecture Notes in Computer Science, 2009, Vol 5912. P. 214-231. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-10366-7_13.

22. Okamoto T., Takashima K. Fully Secure Functional Encryption with General Relations from the Decisional Linear Assumption // Rabin, T. (eds) Advances in Cryptology – CRYPTO 2010. CRYPTO 2010. Lecture Notes in Computer Science, 2010. Vol 6223. P. 191–208. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-14623-7_11.

23. Okamoto, T., Takashima, K. Adaptively Attribute-Hiding (Hierarchical) Inner Product Encryption // Pointcheval, D., Johansson, T. (eds) Advances in Cryptology – EUROCRYPT 2012. EUROCRYPT 2012. Lecture Notes in Computer Science, 2012. Vol 7237. P. 591–608. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-29011-4_35.

24. Okamoto T., Takashima K. Adaptively attribute-hiding (hierarchical) inner product encryption // IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences. 2016. 99(1). P. 92-117.

25. Park J. H. Inner-product encryption under standard assumptions // Designs, Codes and Cryptography, 2011. 58(3), 235–257. https://doi.org/10.1007/s10623-010-9405-9.

26. Baek J., Safavi-Naini, R., Susilo W. Public Key Encryption with Keyword Search Revisited // Gervasi, O., Murgante, B., Laganà, A., Taniar, D., Mun, Y., Gavrilova, M.L. (eds) Computational Science and Its Applications – ICCSA 2008. ICCSA 2008. Lecture Notes in Computer Science, 2008. Vol 5072. P. 1249-1259. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-69839-5_96.

27. Rhee H. S. et al. Improved searchable public key encryption with designated tester // Proceedings of the 4th International Symposium on Information, Computer, and Communications Security. ASIACCS '09. ACM. 2009. P. 376-379. https://doi.org/10.1145/1533057.1533108.

28. Abdalla M., Bellare M., Catalano D., Kiltz E., Kohno T., Lange T., Shi H. Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions // Journal of cryptology, 2008. 21(3). P. 350–391. https://doi.org/10.1007/s00145-007-9006-6.

29. Gentry C. Practical Identity-Based Encryption Without Random Oracles // Vaudenay, S. (eds) Advances in Cryptology – EUROCRYPT 2006. EUROCRYPT 2006. Lecture Notes in Computer Science, 2006. Vol 4004. P. 445–464. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11761679_27.

30. Kiltz E. From selective-ID to full security: The case of the inversion-based Boneh-Boyen IBE scheme. Cryptology ePrint Archive. 2007. URL: https://eprint.iacr.org/2007/033.pdf.

31. Nishide T., Yoneyama K., Ohta K. Attribute-Based Encryption with Partially Hidden Encryptor-Specified Access Structures // Bellovin, S.M., Gennaro, R., Keromytis, A., Yung, M. (eds) Applied Cryptography and Network Security. ACNS 2008. Lecture Notes in Computer Science, 2008. Vol 5037. P. 111-129. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-68914-0_7.

32. Horwitz J., Lynn B. Toward Hierarchical Identity-Based Encryption // Knudsen, L.R. (eds) Advances in Cryptology – EUROCRYPT 2002. EUROCRYPT 2002. Lecture Notes in Computer Science, 2002. Vol 2332. P. 466–481 Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-46035-7_31.

33. Gentry C., Silverberg A. Hierarchical ID-Based Cryptography // Zheng, Y. (eds) Advances in Cryptology — ASIACRYPT 2002. ASIACRYPT 2002. Lecture Notes in Computer Science, 2002, Vol 2501. P. 548–566. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-36178-2_34.

34. Boneh D., Boyen X., Goh EJ. Hierarchical Identity Based Encryption with Constant Size Ciphertext // Cramer R. (eds) Advances in Cryptology – EUROCRYPT 2005. EUROCRYPT 2005. Lecture Notes in Computer Science, 2005. Vol 3494. P. 440–456. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11426639_26.

35. Shi E., Waters B. Delegating Capabilities in Predicate Encryption Systems // Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds) Automata, Languages and Programming. ICALP 2008. Lecture Notes in Computer Science, 2008. Vol 5126. P. 560–578. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-70583-3_46.

36. Lee K. S., Lee D. H. New techniques for anonymous hibe with short ciphertexts in prime order groups // KSII Transactions on Internet and Information Systems (TIIS). 2010. 4(5). P. 968–988.

37. Boneh D., Waters B. Conjunctive, Subset, and Range Queries on Encrypted Data // Vadhan, S.P. (eds) Theory of Cryptography. TCC 2007. Lecture Notes in Computer Science, 2007. Vol 4392. P. 535–554. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-70936-7_29.

38. Paillier P. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes // Stern J. (eds) Advances in Cryptology – EUROCRYPT '99. EUROCRYPT 1999. Lecture Notes in Computer Science, 1999. Vol 1592. P. 223–238. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-48910-X_16.

39. ElGamal T. A public key cryptosystem and a signature scheme based on discrete logarithms // IEEE transactions on information theory. 1985. 31(4). P. 469-472. https://doi.org/10.1109/TIT.1985.1057074.

40. Boneh D., Goh EJ., Nissim K. Evaluating 2-DNF Formulas on Ciphertexts // Kilian J. (eds) Theory of Cryptography. TCC 2005. Lecture Notes in Computer Science, 2005. Vol 3378. P. 325–341. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-30576-7_18.

41. Gentry C. Fully homomorphic encryption using ideal lattices // Proceedings of the forty-first annual ACM symposium on Theory of computing. 2009. P. 169-178. https://doi.org/10.1145/1536414.1536440.

42. Gentry C. Computing arbitrary functions of encrypted data //Communications of the ACM. 2010. 53(3). P. 97-105. https://doi.org/10.1145/1666420.1666444.

43. van Dijk M., Gentry C., Halevi S., Vaikuntanatha, V. Fully Homomorphic Encryption over the Integers // Gilbert, H. (eds) Advances in Cryptology – EUROCRYPT 2010. EUROCRYPT 2010. Lecture Notes in Computer Science, 2010. Vol 6110. P. 24–43. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-13190-5_2.

44. Naehrig M., Lauter K., Vaikuntanathan V. Can homomorphic encryption be practical? // Proceedings of the 3rd ACM workshop on Cloud computing security workshop. 2011. P. 113-124. https://doi.org/10.1145/2046660.2046682.

45. Brakerski Z., Vaikuntanathan V. Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages // Rogaway P. (eds) Advances in Cryptology – CRYPTO 2011. CRYPTO 2011. Lecture Notes in Computer Science, 2011. Vol 6841. P. 505–524. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-22792-9_29.

46. Kamara S., Papamanthou C., Roeder T. Dynamic searchable symmetric encryption // Proceedings of the 2012 ACM conference on Computer and communications security. 2012. P. 965–976.

47. Shen E., Shi E., Waters B. Predicate Privacy in Encryption Systems // Reingold O. (eds) Theory of Cryptography. TCC 2009. Lecture Notes in Computer Science, 2009. Vol 5444. P. 457–473. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-00457-5_27.

48. Song D. X., Wagner D., Perrig A. Practical techniques for searches on encrypted data // Proceeding 2000 IEEE symposium on security and privacy. S&P 2000. IEEE, 2000. P. 44–55. https://doi.org/10.1109/SECPRI.2000.848445.

49. Curtmola R., Garay J., Kamara S., Ostrovsky R. Searchable symmetric encryption: improved definitions and efficient constructions // Journal of Computer Security, 2011. 19(5). P. 895–934.

50. Fiat A., Naor M. Broadcast encryption // Annual International Cryptology Conference. Springer, Berlin, Heidelberg, 1993. P. 480–491.

51. Goh E. J. Secure Indexes. Cryptology ePrint Archive, Report 2003/216. 2003. URL: http://eprint.iacr.org/2003/216/

52. Chang Y. C., Mitzenmacher M. Privacy preserving keyword searches on remote encrypted data // International conference on applied cryptography and network security. Springer, Berlin, Heidelberg, 2005. P. 442–455.

53. Wang Y., Wang J., Chen X. Secure searchable encryption: a survey // Journal of communications and information networks. 2016. 1(4). P. 52–65. https://doi.org/10.11959/j.issn.2096-1081.2016.043.

54. Kamara S., Papamanthou C. Parallel and dynamic searchable symmetric encryption // International conference on financial cryptography and data security. LNCS 7859. Springer, Berlin, Heidelberg, 2013. P. 258–274. https://doi.org/10.1007/978-3-642-39884-1_22.

55. van Liesdonk P., Sedghi S., Doumen J., Hartel P., Jonker W. Computationally Efficient Searchable Symmetric Encryption // Jonker, W., Petković, M. (eds) Secure Data Management. SDM 2010. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg. 2010. Vol 6358. P. 87–100. https://doi.org/10.1007/978-3-642-15546-8_7.

56. Kurosawa K., Ohtaki Y. UC-secure searchable symmetric encryption // International conference on financial cryptography and data security. LNCS. Springer, Berlin, Heidelberg. 2012. Vol. 7397. P. 285–298. https://doi.org/10.1007/978-3-642-32946-3_21..

Надійшла до редколегії 11.05.2022

Відомості про авторів:

Есін Віталій Іванович – д-р техн. наук, професор кафедри безпеки інформаційних систем і технологій, факультет комп'ютерних наук; Харківський національний університет імені В.Н. Каразіна, Україна; e-mail: v.i.yesin@karazin.ua; ORCID: https://orcid.org/0000-0003-1977-7269

Вілігура Владислав Вікторович – аспірант кафедри безпеки інформаційних систем і технологій, факультет комп'ютерних наук; Харківський національний університет імені В.Н. Каразіна, Україна; e-mail: viligura93@gmail.com; ORCID: https://orcid.org/0000-0002-1137-2382