

МЕТОДИ ПОШУКУ ІНФОРМАЦІЇ ПО ЗАШИФРОВАНИХ ДАНИХ

О. А. Орехов¹, М. С. Ходацька¹

¹Національний технічний університет України «Київський політехнічний інститут»

Анотація

У роботі розглянуто питання безпечного пошуку по зашифрованих даних, існуючі методи шифрування та пошуку по зашифрованих даних. Також представлено схему, яка надає можливість пошуку входжень символічних рядків в зашифрований файл без попереднього його розшифрування.

Ключові слова: хмарні сховища, методи шифрування, пошук по зашифрованих даних, пошук входжень

Вступ

З розвитком інформаційних технологій постало питання зберігання великих об'ємів інформації. Для вирішення цієї проблеми використовують хмарні сховища. Хмарні сховища даних – це спосіб зберігання інформації в мережі Інтернет, доступ до якої можна отримати з будь-якої точки планети за допомогою пристроїв, що мають доступ до інтернету. Даний спосіб зберігання інформації стає все популярнішим з кожним днем.

Сервіси надають місце для зберігання даних, але ніхто не може надати стовідсоткову гарантію безпеки даних. Загрозу можуть становити як зловмисники, так і адміністратори сховища, що використовується. Тому постає питання шифрування інформації перед її відправкою на хмару, щоб забезпечити кращий захист. І для того, щоб користувач міг отримати файли які його цікавлять, не виконуючи зайвих дій, можна використовувати схеми шифрування, що дозволяють проводити пошук по даних, не розшифровуючи їх.

1. Постановка задачі

Нехай користувач Аліса має набір документів, які розміщує на недовіреному сервері, власником якого є Боб. Наприклад, Аліса – користувач мобільної системи і зберігає свої повідомлення на поштовому сервері. Оскільки вона не довіряє Бобу, тому хоче зашифрувати свої повідомлення і передати Бобу тільки шифртекст. Кожен документ може бути поділений на «слова». «Словом» може бути будь-який знак; це може бути 64-х бітний блок, слово англійської мови, речення чи будь-яка інша атомарна одиниця залежно від області використання. Для простоти вважатимемо, що «слова» мають однакову довжину. Можливо, Алісі потрібні тільки ті документи, в яких є слово «W», тому для досягнення цієї мети потрібно розробити алгоритм, після виконання якого Боб визначить потрібні документи, але не буде знати їхній вміст.

2. Існуючі рішення

У 2000 році опубліковано статтю Сонгра «Practical Techniques for Searches on Encrypted Data»[1]

В статті представлено набір алгоритмів, що дозволяють проводити пошук в зашифрованих даних. Складність пошуку цих алгоритмів є лінійною $O(n)$ для кожного зашифрованого документа, де n - кількість «слів» на які розбивається файл. Алгоритм шифрування полягає в наступному:

- На вхід подається документ, який розбивається на слова за попередньо визначеним принципом (зазвичай, це розділ по відстані між словами).
- З одного майстер-ключа, наданого користувачем (МК), створюються три підключа $KDF(MK) = \langle k_1, k_2, k_3 \rangle$, що використовуються в різних частинах алгоритму і не дозволяють серверу розшифрувати дані.
- На наступному кроці кожне слово шифрується стандартним блочним алгоритмом $E_{k_2}(W_i)$, і розбивається на дві нерівні частини $\langle L_i, R_i \rangle$ (залежно від параметра користувача), додатково обробляються $S_i = G(k_3)$, $F_{k_1} = (S_i)$, де $k_i = f_{k_i}(L_i)$ і отримані значення додаються між собою (виконується операція додавання за модулем 2), створюючи тим самим шифртекст $C = \langle L_i, R_i \rangle \oplus \langle S_i, F_{k_1}(S_i) \rangle$.

Алгоритм пошуку:

- Для операції пошуку необхідно зашифрувати ключове слово згідно алгоритму шифрування і передати на сторону сервера $\langle E_{k_2}(W), f_{k_1}(L) \rangle$.
- Після цього сервер починає обробляти кожен збережений зашифрований запис $C_i \oplus E_{k_2}(W_i)$, отримуючи на виході пару значень $\langle S_i, F_{k_1}(S_i) \rangle$.
- Оскільки значення довжини початкового поточного шифру відома, а саме x , з отриманої пари можна отримати рядок S_i .
- В результаті потрібно порівняти значення функції з бітами, що залишились $F_{k_1}(S_i)$.

Схема представляє собою ймовірнісний пошук: пошук певного слова повертає всі можливі позиції, де може знаходитись це слово у відкритому тексті. Але можуть видаватися і хибні позиції. Можна перед-

бачити кількість хибних позицій, а перевірити їх користувач зможе вже в розшифрованому тексті.

Обчислювальна складність наведеного алгоритму детермінованого пошуку ключового слова в зашифрованих даних зростає експоненційно при збільшенні вхідних даних, що пояснює, чому його не використовують на практиці.

3. Аналіз методів

Для використання в реальних умовах потрібні не тільки швидкі і якісні методи, але й безпечні. Адже вони спрямовані на захист конфіденційної інформації.

Представлена техніка у статті Сонга є безпечною, оскільки недовірений сервер не отримує ніякої інформації, окрім шифртексту. Пошук в такій схемі може проводитись лише авторизованими користувачами, тому сервер не зможе самостійно робити запити на пошук. Коли клієнт робить запит, то надає серверу тільки зашифроване слово для пошуку. В результаті сервер отримує лише результати пошуку.

Як наслідок, з'явилося багато рішень, які використовують різні техніки та зменшують обчислювальну складність запропонованих схем. Але в результаті задача пошуку зупинилась на можливості визначення входжень ключового слова в зашифрованій множині даних.

Універсальні рішення будуть приводити до надлишковості, тобто крім шифрування основного тексту потрібно проводити його модифікацію (привести до нижнього регістру) і також зашифрувати. Цей спосіб збільшить об'єм пам'яті, потрібної для зберігання інформації, а також збільшить швидкість пошуку потрібного ключового слова як мінімум в 2 рази.

Схеми, що розроблялись надалі, надають можливість пошуку по ключових словах при повному співпаданні. Для прикладу, в схемі Гоха [2], яка базується на понятті безпечних індексів, складність пошуку залежить від параметра d (кількості слів у словнику). В таблиці 1 наведено особливості таких рішень і порівняльна характеристика.

Табл. 1. Ключові особливості існуючих схем пошуку по зашифрованих даних

| Схема | Складність пошуку | Тип пошуку | Пошук входжень |
|-------------------|-------------------|-----------------------------------|----------------|
| Song[1] | $O(n)$ | Лінійний | Ні |
| Goh[2] | $O(d)$ | Використання попередніх обчислень | Можливо |
| Improved Index[3] | $O(1)$ | Використання попередніх обчислень | Можливо |
| Peks[4] | $O(n)$ | Лінійний | Можливо |

4. Пропонований підхід для забезпечення пошуку по зашифрованих даних

Для забезпечення надійного і швидкого пошуку по зашифрованих даних пропонується використати наступний підхід, що реалізується клієнт-серверним застосунком.

Клієнт завантажує файл, застосунок генерує необхідні дані для гешування та шифрування, а саме: випадкові дані (криптографічну сіль) для гешування рядків, ключ та вектор ініціалізації для шифрування файла. Рядки в файлі поділяються на дві категорії:

- ті, які повторюються щонайменше двічі в файлі;
- рядки, що зустрічаються тільки один раз в файлі.

Далі клієнт, використовуючи функцію SHA-1, гешує рядки файлу за наступною схемою: рядки першого типу гешуються з використанням солі; для рядків другого типу також зберігаються індекси їх розміщення в файлі. Перед відправкою на сервер всі дані зберігаються в gZIP архіві та шифруються, для безпечної передачі. Файл, що відправляється на сервер, зашифровується AES алгоритмом. Для цього файл розділяється на блоки довжини 100, кожен блок шифрується з додаванням геш-функції MD-5 та вектором ініціалізації. Такі операції виконуються для отримання різних шифртекстів для однакових блоків відкритого тексту. Для визначення розміщення рядків в файлі можна виконувати бінарний пошук по геш-таблиці.

Головною перевагою такої схеми є те, що всі операції виконує застосунок на боці клієнта.

Висновки

В даній роботі розглянуто питання безпечного пошуку по зашифрованих даних. Наведено існуючі методи шифрування та пошуку по зашифрованих даних. Також представлено схему, яка надає можливість пошуку входжень символічних рядків в зашифрований файл, без попереднього його розшифрування.

Перелік використаних джерел

1. Song D.X. Practical Techniques for Searches on Encrypted Data. — 2000. — P. 44–55.
2. Goh Eu-Jin. Secure indexes. — 2004. — P. 71. — URL: <http://crypto.stanford.edu/~eujin/papers/secureindex/secureindex.pdf>.
3. Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions / R. Curtmola, J. Garay, S. Kamara, R. Ostrovsky. — 2006. — P. 79–88.
4. Public Key Encryption with Keyword Search / D. Boneh, G.D. Crescenzo, R. Ostrovsky, G. Persiano. — 2004. — P. 506–522.