XÂY DỰNG CHỈ MỤC HỖ TRỢ TÌM KIẾM CHÍNH XÁC CHUỖI CON TRÊN CƠ SỞ DỮ LIỆU QUAN HỆ MÃ HOÁ

Hoàng Ngọc Cảnh[*], Ngô Đức Thiện[#]
[*] Trường Đại học Thương mại, 79 Hồ Tùng Mậu, Cầu Giấy, Hà Nội
[#] Học viện Công nghệ Bưu chính Viễn Thông, Hà Nội

Tóm tắt - Nghiên cứu này phát triển một kỹ thuật mới cho phép tìm kiếm chính xác và trực tiếp sự tồn tại của một chuỗi con bất kỳ trên các bản ghi mã hoá trong cơ sở dữ liệu quan hệ. Để giải quyết ý tưởng trên, bài báo đề xuất với mỗi dữ liệu ký tự rõ nhạy cảm cần mã hoá trên cơ sở dữ liệu quan hệ sẽ được biến đổi thành hai phiên bản mã hoá với chức năng khác nhau. Phiên bản thứ nhất là kết quả của việc mã hóa dữ liệu rõ bởi các thuật toán mã hóa tiêu chuẩn như AES, DES,… Phiên bản còn lại là một giá trị chỉ mục đặc biệt ($StringIndex$) được xây dựng trên cấu trúc mới mà chúng tôi đề xuất là "**danh sách mảng nhị phân biểu diễn vị trí ký tự đã gây nhiễu của chuỗi rõ"**. Quá trình tìm kiếm chính xác chuỗi con qua cú pháp $LIKE$ sẽ được thực hiện trên từng chỉ mục bằng thuật toán tối ưu $SearchOn\_StringIndex$ có hiệu năng cao và độ dư thừa dữ liệu của tập kết quả sau truy vấn trên cơ sở dữ liệu thuê ngoài rất thấp. Bên cạnh đó, chỉ mục $StringIndex$ được chứng minh có tính bảo mật tốt và chống được rò rỉ dữ liệu rõ trước các kiểu tấn công phổ biến hiện nay.

Từ khóa - cơ sở dữ liệu thuê ngoài, chỉ mục tìm kiếm, vị trí ký tự gây nhiễu, tìm kiếm chuỗi con

I. GIỚI THIỆU

Ngày nay, sự phát triển của các mô hình điện toán đám mây đã thúc đẩy các doanh nghiệp triển khai các dịch vụ, hệ thống thông tin quản lý và cơ sở dữ liệu quan hệ lên các nền tảng này nhằm tận dụng được lợi thế về chi phí, nhân lực quản trị và tính sẵn sàng của hạ tầng thuê ngoài. Tuy nhiên dù đã có những ràng buộc nhất định trong hợp đồng với các nhà cung cấp đám mây thì người sở hữu dữ liệu vẫn luôn phải đối mặt với tình trạng rò rỉ thông tin hoặc mất an toàn, kiểm soát với cơ sở dữ liệu của mình bởi nhà cung cấp luôn có thể là một đối tượng "trung thực nhưng tò mò".

Giải pháp phù hợp nhất để đảm bảo sự an tâm cho người sở hữu dữ liệu là mã hoá toàn bộ cơ sở dữ liệu quan hệ trước khi triển khai lên đám mây. Tuy nhiên quá trình mã hoá làm dữ liệu mất đi các tính chất vốn có của nó (so sánh, sắp xếp, tổng hợp,…), điều này tương đương với việc gây cản trở quá trình truy vấn trên dữ liệu mã, gây ảnh hưởng nghiêm trọng tới khả năng khai thác các cơ sở dữ liệu quan hệ mã theo các cách thức truyền thống (ví dụ thực thi các bộ lệnh T-SQL, P/L – SQL).

Một giải pháp đã được nghiên cứu trong nhiều năm qua là xây dựng các lược đồ hỗ trợ tìm kiếm trên mã (SE – Searchable Encryption) [1 - 11]. Các lược đồ SE chia làm hai nhóm chính: nhóm thứ nhất xây dựng một phương pháp mã và giải mã mới cho phép truy vấn trực tiếp trên chính dữ liệu mã hóa [2]; nhóm thứ hai thiết kế một giá trị chỉ mục riêng (Blind Index) độc lập với dữ liệu mã hóa, cho phép truy vấn trên chỉ mục với nhiều lợi thế về hiệu năng và tính khả dụng trong triển khai thực tiễn. Trong mỗi nhóm trên đều có hai cơ chế mã hóa là đối xứng (lược đồ SSE) [1-7] và bất đối xứng (lược đồ PKSE) [8,9], tuy nhiên hầu hết các lược đồ trên đều tập trung vào kỹ thuật tìm kiếm theo từ khóa.

Tuy nhiên trong thực tế, tìm kiếm theo từ khóa chỉ là một trường hợp đặc biệt của tìm kiếm chuỗi con và không phù hợp đối với các yêu cầu truy vấn đa dạng từ người dùng cuối trên môi trường đám mây, ví dụ như tìm kiếm dữ liệu trên các công cụ tìm kiếm, tìm kiếm email, tra cứu nội dung văn bản,... Trong nghiên cứu này sẽ tập trung xây dựng một kỹ thuật truy vấn mới cho phép tìm kiếm chuỗi con chính xác trên trường dữ liệu ký tự đã mã hóa trong cơ sở dữ liệu quan hệ với hiệu năng, tính bảo mật cao và độ dư thừa dữ liệu (các bản ghi dương tính giả) thấp.

II. CÁC NGHIÊN CỨU LIÊN QUAN

Song và cộng sự [2] là những người đi đầu khi đề xuất một lược đồ có tính bảo mật được chứng minh rõ ràng nhưng thời gian tìm kiếm lại tuyến tính độ dài văn bản, hơn nữa việc coi mỗi văn bản là một chuỗi các từ khóa tìm kiếm có độ dài bằng nhau sẽ làm cho dạng thức của từ khoá không giống thực tế.

Cho tới nay, các công trình nghiên cứu về tìm kiếm chuỗi con khá là ít bởi sự phức tạp trong thiết kế cũng như độ phức tạp về không gian lưu trữ, chi phí thời gian tìm kiếm lớn. Dựa trên giải pháp SSE hỗ trợ tìm kiếm theo từ khóa được đề xuất bởi Curtmola và cộng sự [3], chúng ta có thể ứng dụng cho phép tìm kiếm chuỗi con trên dữ liệu được mã hóa. Tuy nhiên chỉ một số từ khóa được phân tích từ chuỗi con tồn tại trong bộ sưu tập từ

Tác giả liên hệ: Ngô Đức Thiện,
Email: thienptit@gmail.com
Đến tòa soạn: 8/2022, chỉnh sửa: 10/2022, chấp nhận đăng: 11/2022.

khóa, kéo theo số lượng bản ghi dương tính giả trả về rất lớn khi chỉ sử dụng một số từ khóa đã nói để đại diện cho chuỗi con khi tìm kiếm trên các văn bản mã hóa. Trong thực tế người dùng sẽ yêu cầu tìm kiếm một chuỗi con bất kỳ trên tập văn bản mã hóa, vì vậy độ phức tạp lưu trữ và tính toán sẽ trở lên rất lớn nếu áp dụng giải pháp này.

Chase và Shen [10] chỉ ra cách thực hiện các truy vấn chuỗi con dựa trên cấu trúc cây hậu tố với chi phí lưu trữ $O(\lambda.n)$, độ phức tạp tìm kiếm $O(\lambda.m + k)$ với λ là tham số bảo mật, m là chiều dài chuỗi con, k là số lần xuất hiện của chuỗi con trong văn bản có độ dài n, số vòng giao tiếp dữ liệu là ba nhưng thực tế đã được chỉ ra mất bốn vòng giao tiếp như trong tài liệu [11]. Phương pháp của Chase và Shen cũng tồn tại nhược điểm rò rỉ một số thông tin liên quan tới cấu trúc trong của cây sau mã hóa như: số lá, số con và số nhánh chạm đôi, để giải quyết vấn đề này tác giả đề xuất sử dụng thêm các nút giả để tăng độ nhiễu giúp ẩn các thông tin thực của cây, điều này làm tăng chi phí lưu trữ và tính toán. Đặc biệt khi chỉnh sửa cập nhật dữ liệu dù một phần nhỏ cũng dẫn đến việc phải khởi tạo lại cả cây hậu tố. Vì vậy việc ứng dụng lược đồ này vào cơ sở dữ liệu thực tiễn là một thách thức lớn.

III. QUY ƯỚC VÀ KÝ HIỆU

Ký hiệu một quan hệ trong cơ sở dữ liệu với các trường dữ liệu $ID, X_1, X_2, \ldots X_t, \ldots X_n$ là $R(ID, X_1, X_2, \ldots X_t, \ldots X_n)$ [12, 17], trong đó X_t là trường dữ liệu nhạy cảm kiểu chuỗi ký tự. Khi đó R sẽ được cấu trúc lại thành R^E khi lưu trên cơ sở dữ liệu quan hệ mã hoá: $R^E(ID, X_1, X_2, \ldots X_t^E, \ldots X_n, SIX_t)$, trong đó X_t^E là trường mã hóa của trường X_t, SIX_t là trường chỉ mục hỗ trợ tìm kiếm ($StringIndex$) tương ứng với trường X_t.

Cho A_s và A_s' là hai chuỗi ký tự, A_s && A_s' là chuỗi ghép nối của hai chuỗi đã cho. A_s' được gọi là chuỗi con của A_s nếu tồn tại các chuỗi X, Y sao cho $A_s = X$ && A_s' && Y.

Ký hiệu $len(A_s)$ là số lượng ký tự của chuỗi A_s, tương tự nếu cho tập hợp AS có v phần tử thì $v = len(AS)$. Giá trị tuyệt đối của một số V được ký hiệu là $|V|$.

Cho B_i và B_j là hai chuỗi bít cùng kích cỡ, khi đó phép và (AND) nhị phân của hai chuỗi được ký hiệu là B_i & B_j, phép hoặc (OR) nhị phân của hai chuỗi được ký hiệu là $B_i || B_j$. Phép dịch vòng phải k bits của chuỗi B_i được ký hiệu là $B_i^{\gg k}$, phép dịch vòng trái k bits của chuỗi B_i được ký hiệu là $B_i^{\ll k}$.

IV. GIẢI PHÁP TRUY VẤN CHUỖI CON TRÊN DỮ LIỆU MÃ HÓA

A. Đề xuất mô hình truy vấn

Dựa trên mô hình DAS [12], chúng tôi đề xuất mô hình Proxy gồm ba phần: Client Site, Proxy, DSP (hay còn gọi là Server Site). Trong đó Client Site dành cho người dùng cuối, thông qua các ứng dụng để truy xuất vào hệ thống với dữ liệu truy vấn đầu vào là dữ liệu rõ. Proxy là lớp trung gian tin cậy được quản lý bởi người sở hữu dữ liệu, tại đây lưu trữ các $MetaData$ bí mật, cho phép thực hiện các tác vụ mã và giải mã, tạo chỉ mục, tiếp nhận yêu cầu truy vấn từ Client Site và biến đổi thành câu lệnh truy vấn trên dữ liệu mã để gửi tới DSP, cuối cùng Proxy nhận dữ liệu mã trả về từ DSP để thực hiện giải mã và truy vấn lại trước khi trả về kết quả cho Client Site. DSP là nơi lưu trữ cơ sở dữ liệu mã hóa, tiếp nhận câu lệnh truy vấn từ Proxy và sau đó thực hiện truy vấn trực tiếp trên cơ sở dữ liệu mã hóa và trả dữ liệu mã về Proxy.

Trong đó chúng tôi nhấn mạnh rằng Proxy là khu vực an toàn được cả người dùng và người sở hữu dữ liệu ủy nhiệm thực hiện các thuật toán $GenSecretMetadata$, $BuildStringIndex$, $TranQuery$ trong lược đồ SE. Các $MetaData$ lưu trữ an toàn tại Proxy bao gồm:

q là một số nguyên tố lớn; $\boldsymbol{\alpha}$ là tham số gây nhiễu;

Khóa bí mật $sKey$: dùng để mã và giải mã dữ liệu;

Tập giá trị bí mật $\boldsymbol{VA} = \{\boldsymbol{Va_1}, \boldsymbol{Va_2}, \ldots, \boldsymbol{Va_z}\}$: là tập giá trị bí mật đại diện cho các ký tự dựa trên q.

B. Xây dựng chỉ mục StringIndex

Định nghĩa 1. Cho $A = \{a_1, a_2, \ldots, a_z\}$ là tập z các ký tự của bảng chữ cái anphabet, và $A_s = "c_1 c_2 \ldots c_v"$ là một chuỗi được xây dựng dựa trên các ký tự trong tập A ($z \geq 1, v \geq 1$; $\forall i\ 1 \leq i \leq v, c_i \in A$), tập $C = \{a_1, a_2 \ldots, a_g\}$ là tập các ký tự phân biệt của A_s ($1 \leq g \leq z, g \leq v, len(C) = g$). Gọi $PA_s = \{Pa_1, Pa_2, \ldots, Pa_g\}$ là **danh sách mảng nhị phân biểu diễn vị trí ký tự của chuỗi $\boldsymbol{A_s}$** nếu:

- Pa_j là mảng $v\ bits$ biểu diễn các vị trí của ký tự a_j trong chuỗi A_s ($1 \leq j \leq g$)
- $Pa_j[i] = 1$ khi và chỉ khi $a_j = c_{v-i+1}$ ($1 \leq i \leq v$)

Ví dụ 1: Xây dựng danh sách mảng nhị phân biểu diễn vị trí ký tự của chuỗi $A_s = "aabcdadb"$. Khi đó xác định được $v = 8, C = \{a, b, c, d\}, g = 4, 1 \leq j \leq 4, a_1 = 'a', a_2 = 'b', a_3 = 'c', a_4 = 'd'$, $PA_s = \{Pa, Pb, Pc, Pd\}$ với $Pa = 00100011; Pb = 10000100; Pc = 00001000; Pd = 01010000$.

Định nghĩa 2. Cho chuỗi $A_s = "c_1 c_2 \ldots c_v"$, $PA_s = \{Pa_1, Pa_2, \ldots, Pa_g\}$ là **danh sách mảng nhị phân biểu diễn vị trí ký tự của chuỗi $\boldsymbol{A_s}$** ($1 \leq g \leq v$), gọi $PA_s^N = \{Pa_1^N, Pa_2^N, \ldots, Pa_g^N\}$ là **danh sách mảng nhị phân biểu diễn vị trí ký tự đã gây nhiễu của chuỗi $\boldsymbol{A_s}$** nếu $Pa_i^N = Pa_i \,||\, (P_\alpha^N)_i$ trong đó P_α^N là mảng bit ngẫu nhiên có tối đa là α bit 1 (α *là tham số gây nhiễu tùy chọn* $0 \leq \alpha \leq v, len((P_\alpha^N)_i) = len(Pa_i),\ 1 \leq i \leq g$).

Ví dụ 2. Cho PA_s là danh sách mảng nhị phân biểu diễn vị trí ký tự của chuỗi $\boldsymbol{A_s}$ như trong **ví dụ 1**, xác định **danh sách mảng nhị phân biểu diễn vị trí ký tự đã gây nhiễu của chuỗi $\boldsymbol{A_s}$** như sau:

$PA_s^N = \{Pa^N, Pb^N, Pc^N, Pd^N\}$ *với* $\alpha = 2$ thì mỗi chuỗi nhị phân P_α^N có tối đa 2 bít 1.

$Pa = 00100011;\quad P_\alpha^N = 00101000 \rightarrow \quad Pa^N = 00100011 \,||\, 00101000 = 00101011$

$Pb = 10000100;\quad P_b^N = 01100000 \rightarrow\quad Pb^N = 10000100 \,||\, 01100000 = 11100100$

$Pc = 00001000;\quad P_c^N = 10000000 \rightarrow\quad Pc^N = 00001000 \,||\, 10000000 = 10001000$

$Pd = 01010000;\quad P_d^N = 00000011 \rightarrow\quad Pd^N = 01010000 \,||\, 00000011 = 01011011$

Định lý 1. Cho chuỗi ký tự $A'_s = "c'_1c'_2 \dots c'_t"$ và chuỗi ký tự $A_s = "c_1c_2 \dots c_v"$ $(t \le v)$, $C = \{a_1, a_2 \dots, a_g\}$ là tập các ký tự phân biệt của A_s, $PA_s^N = \{Pa_1^N,\ Pa_2^N,\ \dots, Pa_g^N\}$ là **danh sách mảng nhị phân biểu diễn vị trí ký tự đã gây nhiễu của chuỗi A'_s**. Gọi $(Pa_i^N)^{\gg j}$ là kết quả của phép dịch vòng phải j bits của Pa_i^N $(1 \le i \le g)$, khi đó với $(1 \le j \le t,\ c'_j \subset A'_s,\ c'_j \in C,\ Pc_j'^N \in PA_s^N)$ sẽ có các kết luận sau về mối quan hệ giữa A'_s và A_s:

- A'_s chắc chắn không phải là chuỗi con của A_s nếu và chỉ nếu: $(Pc_1'^N)^{\gg 1}$ & $(Pc_2'^N)^{\gg 2}$ & ... & $\left(Pc_j'^N\right)^{\gg j}$ & ... & $(Pc_t'^N)^{\gg t} = 0$
- A'_s là chuỗi con của A_s với một tỷ lệ dương tính giả nếu: $\left(Pc_1'^N\right)^{\gg 1}$ & $(Pc_2'^N)^{\gg 2}$ & ... & $\left(Pc_j'^N\right)^{\gg j}$ & ... & $(Pc_t'^N)^{\gg t} \ne 0$

Chứng minh Định lý 1 như sau:

Cho chuỗi ký tự $A_s = "c_1c_2 \dots c_v"$, tập $PA_s = \{Pa_1,\ Pa_2,\ \dots, Pa_g\}$ là **danh sách mảng nhị phân biểu diễn vị trí ký tự của chuỗi A_s**. Hiển nhiên $Pa_1 \,\&\, Pa_2 \,\&\, \dots \,\&\, Pa_g = 0$ do mỗi chuỗi của tập PA_s chứa các bit 1 biểu diễn các vị trí đảo ngược từ phải sang trái của một ký tự phân biệt trên A_s (nếu bit 1 nằm tại vị trí thứ i trên chuỗi $Pa_j \in A_s$ thì ký tự a_j sẽ nằm tại vị trí thứ $v - i + 1$ trên chuỗi A_s). Nếu tại vị trí thứ i trên chuỗi Pa_j bằng 1 thì tất cả vị trí thứ i của các chuỗi bít còn lại đều bằng 0.

Xét chuỗi ký tự $A'_s = "c'_1c'_2 \dots c'_t"$ $(t \le v)$ nếu c'_j $(1 \le j \le t)$ là một ký tự trong A_s thì sẽ tồn tại một chuỗi bit $Pc'_j \in PA_s$. Ta chứng minh điều kiện cần và đủ để A'_s là chuỗi A_s như sau:

Không mất tính tổng quát, xét hai ký tự kề nhau bất kỳ c'_j và c'_{j+1} trong A'_s và A_s, chúng ta rút ra kết luận:

Nếu $c'_j = c'_{j+1}$ khi đó Pc'_j giống hệt Pc'_{j+1} và trên mỗi chuỗi Pc'_j hay Pc'_{j+1} đều chắc chắn có tối thiểu 2 bit 1 kề nhau đại diện cho các vị trí đảo ngược của c'_j và c'_{j+1} vì vậy $(Pc'_j)^{\gg j}$ & $\left(Pc'_{j+1}\right)^{\gg (j+1)} \ne 0$.

Nếu $c'_j \ne c'_{j+1}$ khi đó Pc'_j khác Pc'_{j+1} và trên chuỗi Pc'_j sẽ tồn tại ít nhất một bit 1 nằm lệch một đơn vị về mặt vị trí so với ít nhất một bit 1 trên chuỗi Pc'_{j+1} vì vậy $(Pc'_j)^{\gg j}$ & $\left(Pc'_{j+1}\right)^{\gg (j+1)} \ne 0$.

Hai phân tích nêu trên chứng tỏ nếu chuỗi "$c'_jc'_{j+1}$" là chuỗi con của A_s thì $(Pc'_j)^{\gg j}$ & $\left(Pc'_{j+1}\right)^{\gg (j+1)} \ne 0$.

Ở chiều ngược lại nếu hai ký tự bất kỳ c'_i và c'_j trong A'_s mà có Pc'_i & $\left(Pc'_j\right)^{\gg 1} \ne 0$ thì có thể xảy ra các trường hợp:

Nếu $Pc'_i \,\&\, Pc'_j \ne 0$ thì chứng tỏ $c'_i = c'_j$, $Pc'_i = Pc'_j$ và trên Pc'_i có tối thiểu 2 bit 1 kề nhau, nói cách khác c'_i nằm kề nhau tối thiểu 2 lần trên A'_s và A_s.

Nếu $Pc'_i \,\&\, Pc'_j = 0$ thì chứng tỏ $c'_i \ne c'_j$ và tồn tại ít nhất một vị trí của c'_i nằm lệch một đơn vị so với vị trí của c'_j trên A'_s và A_s.

Hai phân tích ở trên cho thấy nếu Pc'_i & $\left(Pc'_j\right)^{\gg 1} \ne 0$ thì "$c'_ic'_j$" là chuỗi con của A_s.

Từ các kết luận ở trên để kiểm tra A'_s có phải là chuỗi con của A_s hay không, chúng ta chỉ cần kiểm tra $(Pc'_1)^{\gg 1}$ & $(Pc'_2)^{\gg 2}$ & ... & $\left(Pc'_j\right)^{\gg j}$ & ... & $(Pc'_t)^{\gg t} \ne 0$.

Tuy nhiên trong **Định lý 1** lại sử dụng tập $PA_s^N = \{Pa_1^N,\ Pa_2^N,\ \dots, Pa_g^N\}$ là **danh sách mảng nhị phân biểu diễn vị trí ký tự đã gây nhiễu của chuỗi A'_s** thay vì sử dụng tập $PA_s = \{Pa_1,\ Pa_2,\ \dots, Pa_g\}$. Theo **Định nghĩa 2**, mỗi phần tử $Pa_i^N = Pa_i \,||\, (P_\alpha^N)_i$ trong đó $(P_\alpha^N)_i$ là mảng bit ngẫu nhiên có tối đa là α bit 1. Vì vậy công thức sau sẽ được biến đổi:

$\left(Pc_1'^N\right)^{\gg 1}$ & $(Pc_2'^N)^{\gg 2}$ & ... & $\left(Pc_j'^N\right)^{\gg j}$ & ... & $(Pc_t'^N)^{\gg t} \ne 0 \Leftrightarrow$

$(Pc'_1||(P_\alpha^N)_1)^{\gg 1}$ & $(Pc'_2||(P_\alpha^N)_2)^{\gg 2}$ & ... & $\left(Pc'_j||(P_\alpha^N)_j\right)^{\gg j}$ & ... & $(Pc'_t||(P_\alpha^N)_t)^{\gg t} \ne 0 \Leftrightarrow$

$((Pc'_1)^{\gg 1}$ & $(Pc'_2)^{\gg 2}$ & ... & $\left(Pc'_j\right)^{\gg j}$ & ... & $(Pc'_t)^{\gg t})$
$||((P_\alpha^N)_1{}^{\gg 1}$ & $(P_\alpha^N)_2{}^{\gg 2}$ & ... & $(P_\alpha^N)_j{}^{\gg j}$ & ... & $(P_\alpha^N)_t{}^{\gg t}) \ne 0$

$(P_\alpha^N)_1{}^{\gg 1}$ & $(P_\alpha^N)_2{}^{\gg 2}$ & ... & $(P_\alpha^N)_j{}^{\gg j}$ & ... & $(P_\alpha^N)_t{}^{\gg t}$ là biểu thức tác động sai số. Khi kết quả biểu thức này là một chuỗi bit khác 1 có thể dẫn đến việc kiểm tra A'_s là chuỗi con của A_s sẽ có sai số theo một tỷ lệ dương tính giả phụ thuộc vào độ dài v của chuỗi A_s, độ dài t của chuỗi con A'_s và tham số α. Bài toán xác định sai số chuyển về dạng tính xác suất để ma trận nhị phân cỡ txv với tối đa α bit 1 mỗi dòng có tối thiểu 1 cột toàn bít 1. Tuy nhiên, ngay cả khi biểu thức nêu trên trả về chuỗi bit khác 0 thì cũng chưa hẳn gây ra dương tính giả nếu trước đó biểu thức sau trả về chuỗi bit khác 0:

$(Pc'_1)^{\gg 1}$ & $(Pc'_2)^{\gg 2}$ & ... & $\left(Pc'_j\right)^{\gg j}$ & ...
& $(Pc'_t)^{\gg t} \ne 0$

Tương tự, chúng ta có thể chứng minh nhận xét: A'_s chắc chắn không phải là chuỗi con của A_s nếu và chỉ nếu: $(Pc_1'^N)^{\gg 1}$ & $(Pc_2'^N)^{\gg 2}$ & ... & $\left(Pc_j'^N\right)^{\gg j}$ & ...
& $(Pc_t'^N)^{\gg t} = 0$.

Ví dụ 2. Kiểm tra chuỗi ký tự $A'_s = "abc"$ có phải là chuỗi con của $A_s = "aabcdadb"$ không?

Theo **ví dụ 1**, chúng ta có $C = \{a, b, c, d\}$, $PA_s^N = \{Pa^N, Pb^N, Pc^N, Pd^N\}$ $với\ Pa^N = 00101011, Pb^N = 11100100,\ Pc^N = 10001000, Pd^N = 01011011.$

Dựa vào **định lý 1**, tính $(Pa^N)^{\gg 1}, (Pb^N)^{\gg 2}, (Pc^N)^{\gg 3}$ sau đó so sánh:

$$(Pa^N)^{\gg 1}\ \&\ (Pb^N)^{\gg 2}\ \&\ (Pc^N)^{\gg 3}\ với\ 0\ .$$

$(Pa^N)^{\gg 1} = 10010101, \quad (Pb^N)^{\gg 2} = 00111001,$

$(Pc^N)^{\gg 3} = 01101011$

$(Pa^N)^{\gg 1}\ \&\ (Pb^N)^{\gg 2}\ \&\ (Pc^N)^{\gg 3} = 00000001 \neq 0$. Vậy kết luận A'_s có thể là chuỗi con của A_s.

Nhận xét: Sử dụng **danh sách mảng nhị phân biểu diễn vị trí ký tự đã gây nhiễu của chuỗi A_s** giúp che giấu đi mối liên quan giữa các vị trí của cùng một ký tự cũng như tần suất xuất hiện của một ký tự trong chuỗi A_s.

Định nghĩa 3. Cho $A = \{a_1, a_2, \dots, a_z\}$ là tập z các ký tự của bảng chữ cái anphabet, cho q là một số nguyên tố lớn, tập $VA = \{Va_1, Va_2, \dots, Va_z\}$ được gọi là **tập giá trị bí mật đại diện cho các ký tự của tập A dựa trên q** với Va_i đại diện cho a_i và Va_j đại diện cho a_j nếu:

$$\begin{cases} Va_i > q, Va_j > q\ \forall\ i,j\ (1 \le i,j \le z), Va_i\ và\ Va_j\ là\ các\ số\ nguyên \\ 1 \le |Va_i - Va_j| \le {}^{q}/_{2} \end{cases}$$

Để sinh được các phần tử của tập VA thỏa mãn **định nghĩa 3**, ta có thể lựa chọn ngẫu nhiên một số cơ sở $V > q$, sau đó tính $Va_i = V + \varepsilon_{a_i}$ với $1 \le \varepsilon_{a_i} \le {}^{q}/_{2}$ sao cho $Va_i \neq Va_j, \forall\ i,j\ (1 \le i,j \le z)$

Định nghĩa 4. Cho chuỗi ký tự $A_s = "c_1 c_2 \dots c_v"$ và $PA_s^N = \{Pa_1^N, Pa_2^N, \dots, Pa_g^N\}\ (1 \le g \le v)$ là **danh sách mảng nhị phân biểu diễn vị trí ký tự đã gây nhiễu của chuỗi A_s**, cho tập ký tự $A = \{a_1, a_2, \dots, a_z\}$, cho q là số nguyên tố lớn và $VA = \{Va_1, Va_2, \dots, Va_z\}$ là **tập giá trị bí mật đại diện cho các ký tự của tập A dựa trên q**. Chỉ mục $Index2$ của chuỗi A_s là tập $IP_{A_s} = \left\{\left(I_{a_1}, IPa_1^N\right), \left(I_{a_2}, IPa_2^N\right), \dots, \left(I_{a_g}, IPa_g^N\right)\right\}$ $với\ 1 \le i \le g$, và mỗi phần tử là một tuble (I_{a_i}, IPa_i^N). Giá trị của một tuble (I_{a_i}, IPa_i^N) được xác định như sau:

$I_{a_i} = Va_i + k_i * q$ $với\ Va_i \in VA$,

k_i là số nguyên ngẫu nhiên riêng biệt cho mỗi I_{a_i}

$IPa_i^N = (Pa_i^N)^{\gg (k_{A_s} + Va_i)}$

Với k_{A_s} là số nguyên ngẫu nhiên dùng chung cho mọi IPa_i^N

Nhận xét: Cách thức xây dựng và sử dụng chỉ mục này tương tự như mô hình của ổ khóa kết hợp (combination lock) trong thực tế nhưng độ phức tạp để mở khóa sẽ cao hơn, bởi số dãy mã cần lăn sẽ lớn hơn và số lượng giá trị số để lựa chọn trong một dãy sẽ nhiều hơn.

Giải thuật xây dựng StringIndex như sau:

Algorithm: BuildStringIndex

Input

$A_s = "c_1 c_2 \dots c_v"$: chuỗi ký tự rõ nhạy cảm cần xây dựng $StringIndex$ tương ứng;

q: số nguyên tố lớn;

$VA = \{Va_1, Va_2, \dots, Va_z\}$: tập giá trị bí mật đại diện cho z ký tự của bảng chữ cái dựa trên q;

α: là tham số gây nhiễu tùy chọn;

Output

Tập IP_{A_s}: đại diện cho chỉ mục $StringIndex$ của chuỗi A_s;

Các bước thực hiện

(1) Xác định tập $C = \{a_1, a_2 \dots, a_g\}$ là tập các ký tự phân biệt của $A_s, 1 \le g \le v$;

(2) Xây dựng $PA_s = \{Pa_1, Pa_2, \dots, Pa_g\}$ là **danh sách mảng nhị phân biểu diễn vị trí ký tự của chuỗi A_s**;

(3) Xây dựng $PA_s^N = \{Pa_1^N, Pa_2^N, \dots, Pa_g^N\}$ là **danh sách mảng nhị phân biểu diễn vị trí ký tự đã gây nhiễu của chuỗi A_s** với tham số α;

(4) Xây dựng tập $IP_{A_s} = \left\{\left(I_{a_1}, IPa_1^N\right), \left(I_{a_2}, IPa_2^N\right), \dots, \left(I_{a_g}, IPa_g^N\right)\right\}$

$với\ 1 \le i \le g$;

$Sinh\ số\ nguyên\ ngẫu\ nhiên\ k_{A_s}$;

For $i = 1$ ***to*** g

Begin

$Sinh\ số\ nguyên\ ngẫu\ nhiên\ k_i$;

$với\ \lambda\ là\ tham\ số\ bảo\ mật$;

$Va_i \leftarrow f^v(a_i, VA)$
$với\ f^v\ là\ hàm\ trả\ về\ phần\ tử\ Va_i\ trong\ VA\ mà\ đại\ diện\ cho\ ký\ tự\ a_i$;

$I_{a_i} = Va_i + k_i * q$;

$IPa_i^N = (Pa_i^N)^{\gg (k_{A_s} + Va_i)}$;

End

(5) Return IP_{A_s};

Bảng 1. Mô tả cấu trúc của $StringIndex$

Plaintext	**Characters**	$\boldsymbol{StringIndex}$	
		$\boldsymbol{I_{a_i}}$	$\boldsymbol{IPa_i^N}$
A_s = "aabcdadb"	a	00101010	01011001
	b	00011011	10011100
	c	01010011	00010001
	d	10000110	01101101

C. Truy vấn trên chỉ mục $\boldsymbol{StringIndex}$

Theo **định nghĩa 4**, chỉ mục $StringIndex$ của chuỗi rõ nhạy cảm $\boldsymbol{A_s}$ là tập IP_{A_s} với mỗi phần tử là một bộ (I_{a_i}, IPa_i^N) trong đó $I_{a_i} = Va_i + k_i * q$ (với $Va_i \in VA$ và k_i là số nguyên ngẫu nhiên riêng biệt cho mỗi I_{a_i}). Trên cơ sở này, chuỗi con A'_s tại Proxy sẽ được biến đổi thành một tập giá trị đại diện có bản chất tương tự như I_{a_i} để có thể truy vấn được trên $StringIndex$. Gọi $\boldsymbol{TranQuery}$ là thuật toán biến đổi chuỗi con $A'_s = c'_1 c'_2 \dots c'_t$ thành tập $I = \{I_{c'_1}, I_{c'_2}, \dots, I_{c'_t}\}$ (với $t = len(A'_s)$) để có thể truy vấn được trên chỉ mục $StringIndex$. Khi đó thuật toán $\boldsymbol{SearchOn\_StringIndex}$ được mô tả như sau:

Algorithm: $\boldsymbol{SearchOn\_StringIndex}$

Input

$I = \{I_{c'_1}, I_{c'_2}, \dots, I_{c'_t}\}$: kết quả trả về của hàm $\boldsymbol{TransQuery}(.)$ với tham số đầu vào là chuỗi con A'_s;

$IP_{A_s} = \left\{\left(I_{a_1}, IPa_1^N\right), \left(I_{a_2}, IPa_2^N\right), \dots, \left(I_{a_g}, IPa_g^N\right)\right\}$: giá trị $StringIndex$ đại diện cho chuỗi ký tự rõ nhạy cảm $A_s = c_1 c_2 \dots c_v$;

Output

$True$: nếu A'_s là chuỗi con của A_s với một tỷ lệ dương tính giả xác định;

$False$: nếu A'_s không là chuỗi con của A_s;

Các bước thực hiện

(1) Xây dựng tập $SB = \{(I_{c'_1}, IPc_1'^N), (I_{c'_2}, IPc_2'^N), \dots, (I_{c'_t}, IPc_t'^N)\}$ với mỗi phần tử $\left(I_{c'_i}, IPc_i'^N\right)$ tương ứng với một phần tử $\left(I_{a_j}, IPa_j^N\right) \in IP_{A_s}$ $(1 \le i \le t, 1 \le j \le v)$. Mã giả xây dựng SB như sau:

```
SB = {∅};
For i = 1 to t
Begin
    For j = 1 to g
    Begin
        if (I_{c'_i} − I_{a_j}) mod q = 0 then
        Begin
            I_{c'_i} = I_{a_j};
            IPc'_i^N = IPa_j^N;
            SB = SB.Add((I_{c'_i}, IPc'_i^N));
            Break;
        End
    End
End
```

(2) Biến đổi tập SB bằng cách giữ cố định $IPc_1'^N$ và với mỗi $IPc_i'^N$ thực quay vòng bít trái hoặc quay vòng bít phải một lượng $bias$, $(2 \le i \le t)$. Mã giả như sau:

```
For i = 2 to t
Begin
    θ = (I_{c'_1} − I_{c'_i}) mod q
    if q/2 ≤ θ < q then
    Begin
        bias = −1 * (θ − q/2)
        IPc'_i^N = (IPc'_i^N)^{≪|bias|}
    End
Else if 0 ≤ θ < q/2
    Begin
        bias = θ
        IPc'_i^N = (IPc'_i^N)^{≫ bias}
    End
End
```

(3) Kiểm tra A'_s có là chuỗi con của A_s hay không:

If $IPc_1'^{N \gg 1} \wedge IPc_2'^{N \gg 2} \wedge \dots \wedge IPc_i'^{N \gg i} \wedge \dots \wedge IPc_t'^{N \gg t} \neq 0$ **then**

```
    Result = True
```

Else

$Result = False$

(4) Return $Result$

Nhận xét: hàm $\boldsymbol{SearchOn\_StringIndex}$ có tốc độ thực thi cao do chỉ cần thực hiện các phép "**And Bitwise**" đơn giản để trả về kết quả.

V. PHÂN TÍCH BẢO MẬT CỦA CHỈ MỤC STRINGINDEX

Giả sử kẻ tấn công cũng biết trước tập chuỗi ký tự rõ P. Kẻ tấn công biết tập mẫu PI (chứa các giá trị $StringIndex$) tương ứng với một tập mẫu P' nào đó là con của P (kẻ tấn công không biết P'). Giả sử tập P có 10000 chuỗi rõ, tập PI có 100 phần tử tương ứng với tập $P' \subset P$. Kẻ tấn công thống kê số lần xuất hiện của mỗi ký tự phân biệt trong từng chuỗi ký tự rõ của tập P, sau đó thống kê tiếp số lần xuất hiện của các bit có giá trị 1 trong từng chuỗi IPa_i^N trong mỗi chỉ mục $StringIndex$ thuộc tập PI. Khi đó với mỗi tập số liệu thống kê của từng phần tử trong tập PI, kẻ tấn công sẽ so sánh lần lượt với tập số liệu thống kê của từng chuỗi ký tự rõ trong tập P để tìm sự tương đồng, thông qua đó suy luận được chuỗi ký tự rõ từ $StringIndex$.

Gọi $N_1, N_2, \dots, N_{10000}$ lần lượt là các tập hợp mô phỏng số lượng các ký tự trong mỗi phần tử của tập P, với $N_j = \{num_1, num_2, \dots, num_{g'}\}$ và num_i là số lượng của ký tự a_i trong phần tử thứ j của tập P $(1 \le j \le 10000, 1 \le i \le g')$. Gọi $N'_1, N'_2, \dots, N'_{100}$ lần lượt là các tập mô phỏng số lượng bit giá trị 1 trong từng chuỗi IPa_i^N của mỗi chỉ mục $StringIndex$ thuộc tập PI_2, $N'_j = \{num'_1, num'_2, \dots, num'_g\}$ với num'_i là số lượng bít 1 trong chuỗi IPa_i^N của chỉ mục $StringIndex$ thứ j của tập PI $(1 \le j \le 100, 1 \le i \le g)$.

Với mỗi tập N'_j kẻ tấn công sẽ so sánh với từng tập N_i để tìm ra các điểm tương đồng và suy luận chuỗi ký tự rõ tương ứng với $StringIndex$. Tuy nhiên, trong thuật toán $BuildStringIndex$ sử dụng hằng số gây nhiễu bí mật $\alpha \neq 0$ để gán thêm các chuỗi bít ngẫu nhiên có số lượng bit giá trị 1 bằng α vào các chuỗi IPa_i^N, điều này làm số lượng ký tự a_i trong bản rõ sẽ khác với số lượng bit 1 trong chuỗi IPa_i^N của $StringIndex$ tương ứng, bên cạnh đó sự tương quan về số lượng bit 1 giữa hai chuỗi IPa_i^N và IPa_j^N bất kỳ của cùng một chỉ mục $StringIndex$ cũng sẽ bị che mờ (ví dụ: một chuỗi A_s có x ký tự a_i và y ký tự a_j, khi đó trong $StringIndex$ sẽ thống kê được x' bit 1 trong chuỗi IPa_i^N và y' bit 1 trong chuỗi IPa_j^N. Kẻ tấn công không tìm thấy mối liên quan nào giữa cặp (x, y) và (x', y')).

VI. ĐÁNH GIÁ HIỆU QUẢ TRUY VẤN TRÊN CHỈ MỤC STRINGINDEX

Để đánh giá tính hiệu quả của mô hình đề xuất trong bài báo, trong phần này chúng tôi thực hiện một số phân tích về độ phức tạp thuật toán tìm kiếm $\boldsymbol{SearchOn\_StringIndex}$ cũng như thực nghiệm trên dữ liệu thực để kiểm tra tính hiệu quả dựa trên các tiêu chí về: tỷ lệ lọc và tỷ lệ lỗi của tập dữ liệu mã sau mỗi lần truy vấn trên $StringIndex$ so với truy vấn trên chỉ mục hỗ trợ tìm kiếm được xây dựng dựa trên bộ lọc Bloom [13-16]

A. Phân tích hiệu quả thuật toán dựa trên thực nghiệm

Theo tài liệu [17], chúng tôi sử dụng các tham số về tỷ lệ lọc (FIR) và tỷ lệ lỗi (FAR) để đánh giá hiệu quả của các lệnh truy vấn trên $StringIndex$ theo công thức:

$$FIR = \frac{N_R - N_{Index1}}{N_R - N_{Result}}$$

$$FAR = \frac{N_{Index1} - N_{Result}}{N_{Index1}}$$

Trong đó N_R là số lượng bản ghi của quan hệ R, N_{Result} là số lượng bản ghi trả về khi truy vấn dữ liệu rõ trên R, N_{Index} là số bản ghi mã trả về sau khi truy vấn chỉ mục $StringIndex$ trên R^E.

Dữ liệu thử nghiệm: 1 triệu bản ghi bảng LINEITEM trong cơ sở dữ liệu TPC-H [28] làm đối tượng thực hiện các thử nghiệm và sử dụng cột L_COMMENT để tạo các chỉ mục $StringIndex$ và $BloomIndex$.

Gọi FIR_1 và FAR_1 lần lượt là tỷ lệ lọc và tỷ lệ lỗi khi truy vấn trên chỉ mục $BloomIndex$. Tương tự gọi FIR_2 và FAR_2 lần lượt là tỷ lệ lọc và tỷ lệ lỗi khi truy vấn trên chỉ mục $StringIndex$.

Trong mỗi kịch bản thử nghiệm, chúng tôi sử dụng một số điều kiện truy vấn chuỗi con như bảng sau để thử nghiệm và đánh giá, với S_1, S_2, S_3 là các chuỗi con có số ký tự tương ứng là 2, 8, 16.

Bảng 2. Các điều kiện truy vấn sử dụng trong thử nghiệm

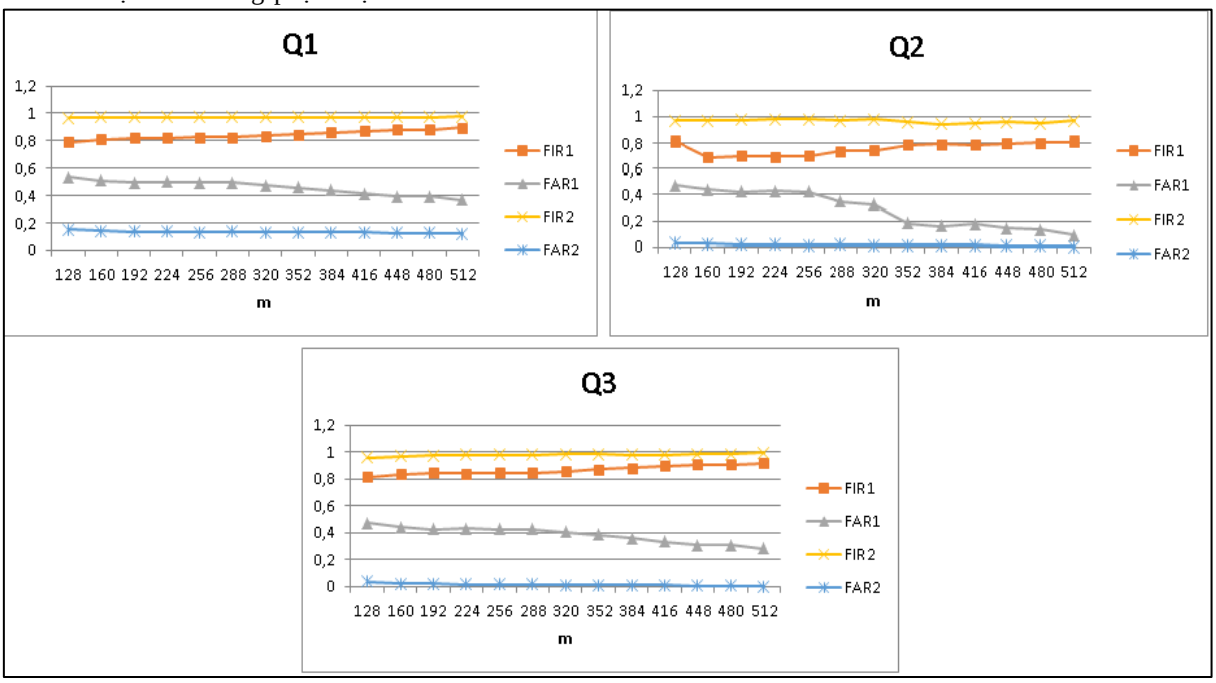
Query	Condition
Q1	L_COMMENT like '%S_1%'
Q2	L_COMMENT like '%S_2%'
Q3	L_COMMENT like '%S_3%'

Ứng với mỗi truy vấn Q1, Q2, Q3 chúng tôi sử dụng 30 mẫu chuỗi con S_1, S_2, S_3 để thử nghiệm, các giá trị $FIR_1, FAR_1, FIR_2, FAR_2$ tại mỗi kịch bản truy vấn sẽ được tính bằng trung bình cộng các kết quả trả về. Chúng tôi sử dụng thêm một tham số m đại diện cho chiều dài của $BloomIndex$ để làm đại lượng so sánh sự biến động của tỷ lệ lọc, tỷ lệ lỗi khi truy vấn.

Chọn $N_R = 1000000$, với m tăng dần từ 128 đến 512 lần lượt thực hiện các truy vấn Q1, Q2, Q3 và tính toán FIR_1, FAR_1, FIR_2, FAR_2. Kết quả được tổng hợp dưới biểu đồ trong Hình 1.

Từ biểu đồ có thể thấy giá trị FIR2 luôn có xu hướng tiệm cận giá trị 1, FAR2 luôn xu hướng tiệm cận giá trị 0, điều này khẳng định hiệu quả lọc để loại bỏ các bản ghi không thỏa mãn truy vấn cũng như tỷ lệ lỗi của mô hình sau khi truy vấn trên chỉ mục $StringIndex$ là rất tốt và vượt trội so với truy vấn trên chỉ mục $BloomIndex$.

Hình 1 cũng thể hiện được sự tác động độ dài m lên hiệu quả truy vấn trên $BloomIndex$, khi m tăng thì các đường FIR_1 tiệm cận dần về giá trị 1 hơn và đường FAR_1 tiệm cận dần về giá trị 0 hơn. Trong khi đó FIR_2, FAR_2 vẫn luôn ổn định và không phụ thuộc vào chiều dài của chỉ mục StringIndex, nói cách khác hiệu quả lọc của mô hình truy vấn trên $StringIndex$ không phụ thuộc vào độ dài dữ liệu rõ ban đầu.



Hình 1. FIR1, FAR1, FIR2, FAR2 với truy vấn Q1, Q2, Q3 trong trường hợp $m \in [128, 512]$

B. Phân tích độ phức tạp thuật toán SearchOn_StringIndex

Thuật toán ***SearchOn_StringIndex*** sử dụng các phép toán số học cơ bản trên các chuỗi bit trong 2 vòng lặp có số bước lặp là các giá trị g và t. Trong đó g là số lượng ký tự phân biệt trong mỗi chuỗi rõ trước khi được mã hoá (giá trị này nhỏ hơn số lượng ký tự trong bảng chữ cái latinh) và t là chiều dài chuỗi con cần tìm kiếm. Khi đó với n bản ghi trên quan hệ R^E thì độ phức tạp của thuật toán ước chừng $O(kn)$ với k là hằng số. Dưới đây là bảng so sánh độ phức tạp của thuật toán ***SearchOn_StringIndex*** so với một số thuật toán truy vấn trong các nghiên cứu trước đó.

Bảng 3. So sánh độ phức tạp thuật toán ***SearchOn_StringIndex*** *với các công trình trước đó.*

Công trình	Độ phức tạp tìm kiếm	Số vòng truy vấn
[18]	$O(mn)$	1
[19]	$O((m+occ)n)$	3
[20]	$O(\lambda m + occ)$	3
[21]	$O(nm)$	2
[22]	$O(m+occ)n$	>=1
SearchOn StringIndex	$O(kn)$	1

Trong bảng 3, m là chiều dài của chuỗi con truy vấn, n là kích cỡ của văn bản hoặc số bản ghi, z là kích cỡ từ điển, occ là số lần xuất hiện của chuỗi con trong mỗi bản rõ, λ là tham số bảo mật.

VII. KẾT LUẬN

Bài báo này nghiên cứu một kỹ thuật mới cho phép xây dựng chỉ mục hỗ trợ tìm kiếm cho dữ liệu kiểu ký tự được mã hóa trong các cơ sở dữ liệu thuê ngoài. Sử dụng mô hình Proxy, chúng tôi đã trình bày đầy đủ các bước từ $GenSecretMetadata$, $BuildStringIndex$, $TranQuery$ cho đến $QueringOn\_StringIndex$ để làm cơ sở thực hiện các truy vấn chuỗi con trên trường dữ liệu ký tự mã hóa. Bên cạnh đó nghiên cứu đã chỉ ra khả năng bảo mật của chỉ mục trước các tấn công phân tích thống kê tần suất, tấn công biết trước bản rõ,… cũng như hiệu quả truy vấn dựa trên các kết quả thực nghiệm đo đạc tỷ lệ lọc và tỷ lệ lỗi của tập kết quả thu được sau trên ServerSite. Các ý tưởng trong bài báo mở ra một cách tiếp cận mới trong bài toán SE nói chung và trong kỹ thuật truy vấn chuỗi con trên dữ liệu mã nói riêng.

TÀI LIỆU THAM KHẢO

[1]. Song, D.X.; Wagner, D.; Perrig, A. Practical Techniques for Searches on Encrypted Data. In Proceedings of the 2000 IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 14–17 May 2000.

[2]. Goh, E.J. Secure Indexes. Cryptology ePrint Archive, Report 2003/216, 2003. Available online: http://eprint.iacr.org/2003/216/ (accessed on 10 January 2016).

[3]. R. Curtmola, J.A. Garay, S. Kamara and R. Ostrovsky, Searchable symmetric encryption: Improved definitions and efficient constructions, in:Proceedings of the 13th ACM Conference on Computer and Communications Security, Alexandria,Virginia, USA, 2006.

[4]. D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M. Rosu and M. Steiner, Highly-scalable searchable symmetric encryption with support for Boolean queries, in:Advances in Cryptology – CRYPTO'13, Springer, 2013.

[5]. M. Chase and S. Kamara, Structured encryption and controlled disclosure, in:Advances in Cryptology – ASIACRYPT'10, Lecture Notes in Computer Science, Vol. 6477, Springer, 2010, pp. 577–594.

[6]. Moataz, T.; Shikfa, A. Boolean Symmetric Searchable Encryption. In Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security, Hangzhou, China , 8–10 May 2013.

[7]. Chang, Y.C.; Mitzenmacher, M. Privacy Preserving Keyword Searches on Remote Encrypted Data. In Proceedings of the 3rd International Conference on Applied Cryptography and Network Security,New York, NY, USA, 7–10 June 2005.

[8]. Boneh, D.; Crescenzo, G.D.; Ostrovsky, R.; Persiano, G. Public Key Encryption with Keyword Search. In Proceedings of the EUROCRYPT 2004, Jeju Island, Korea, 5–9 December 2004

[9]. Boneh, D.; Waters, B. Conjunctive, Subset, and Range Queries on Encrypted Data. In Proceedings of the 4th IACR Theory of Cryptography Conference, Amsterdam, The Netherlands, 21–24 February 2007.

[10]. M. Chase and E. Shen. Substring-searchable symmetric encryption. Proceedings on Privacy Enhancing Technologies, 2015

[11]. D. Cash, J. Jaeger, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Rosu, and M. Steiner. Dynamic searchable encryption in very large databases: Data structures and implementation. In Proc. of NDSS, volume 14, 2014

[12]. Hacigümüs, Hakan & Iyer, Balakrishna & Li, Chen & Mehrotra, Sharad. (2002). Executing SQL over Encrypted Data in the Database-Service-Provider Model. Proceedings of the ACM SIGMOD International Conference on Management of Data. 216-227. 10.1145/564691.564717.

[13]. Liu, Lianzhong & Gai, Jingfen. (2009). Bloom Filter Based Index for Query over Encrypted Character Strings in Database. 1. 303-307. 10.1109/CSIE.2009.979.

[14]. Suga, T., Nishide, T., & Sakurai, K. (2012). Secure Keyword Search Using Bloom Filter with Specified Character Positions. Lecture Notes in Computer Science, 235–252. doi:10.1007/978-3-642-33272-2_15

[15]. Warren, Jr, Claude. (2020). Indexing Encrypted Data Using Bloom Filters. 10.13140/RG.2.2.25976.39681.

[16]. XIAN, He-qun & ZHANG, Shu-guang & ZHANG, Man & XIANG, Jun-zheng & LI, Min. (2017). Efficient Keyword Query in Encrypted Cloud Databases. DEStech Transactions on Engineering and Technology Research. 10.12783/dtetr/eeta2017/7761.

[17]. Wu, Z., Xu, G., Yu, Z., Yi, X., Chen, E., & Zhang, Y. (2012). Executing SQL queries over encrypted character strings in the Database-As-Service model. Knowledge-Based Systems, 35, 332–348. doi:10.1016/j.knosys.2012.05.009.

[18]. Wang, B., Song, W., Lou, W., & Hou, Y. T. (2017). Privacy-preserving pattern matching over encrypted genetic data in cloud computing. IEEE INFOCOM 2017 - IEEE Conference on Computer Communications.

[19]. Kana Shimizu, Koji Nuida, and Gunnar Rätsch. 2016. Efficient privacypreserving string search and an application in genomics. Bioinformatics 32, 11 (2016). https://doi.org/10.1093/bioinformatics/btw050.

[20]. Boldyreva, A.; Chenette, N. Efficient Fuzzy Search on Encrypted Data. In Proceedings of the 21st International Workshop on Fast Software Encryption, London, UK, 3–5 March 2014.

[21]. Baron, J., El Defrawy, K., Minkovich, K., Ostrovsky, R., & Tressler, E. (2012). 5PM: Secure Pattern Matching. Security and Cryptography for Networks, 222–240. doi:10.1007/978-3-642-32928-9_13.

[22]. Hahn, F., Loza, N., & Kerschbaum, F. (2018). Practical and Secure Substring Search. Proceedings of the 2018 International Conference on Management of Data - SIGMOD '18.

BUILD AN INDEX SUPPORTING EXACT SEARCH OF SUBSTRING ON ENCRYPTED RELATION DATABASE

Abstract **-** This research develops a new technique that allows to accurately and directly search for the existence of any substring on encrypted records in an relation database. To solve the above idea, the paper proposes that each sensitive character data that needs to be encrypted on relation database will be transformed into two encrypted versions. The first version is the result of plaintext data encryption by standard encryption algorithms such as AES, DES, etc. The other version is a special index value ($StringIndex$) built on the new structure we propose as "a binary arraylist representing the noisy character positions of an explicit string". The exact search for substrings through $LIKE$ syntax will be performed on each index using the $SearchOn\_StringIndex$ optimization algorithm with high performance and data redundancy of the result set after querying on an outsourced database is very low. In addition, the $StringIndex$ is proven to have good security and data leakage resistance against today's common attack types.

Keywords **–** outsourced database, searching index, noisy character position, substring search

Hoàng Ngọc Cảnh, Nhận học vị Thạc sĩ năm 2012, đang là nghiên cứu sinh tại Học viện Công nghệ Bưu chính Viễn thông khóa 2020. Hiện công tác tại Đại học Thương mại.

Lĩnh vực nghiên cứu: Cơ sở dữ liệu, lý thuyết mã hóa và mật mã.

Ngô Đức Thiện, Nhận học vị Tiến sĩ năm 2010. Hiện công tác tại Học viện Công nghệ Bưu chính Viễn thông.

Lĩnh vực nghiên cứu: Lý thuyết thông tin, lý thuyết mã hóa và mật mã.