BỘ KHOA HỌC VÀ CÔNG NGHỆ
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

HOÀNG NGỌC CẢNH

PHÁT TRIỂN MỘT SỐ PHƯƠNG PHÁP TRUY VẤN HIỆU QUẢ TRÊN CƠ SỞ DỮ LIỆU QUAN HỆ MÃ HOÁ

Ngành: **Hệ thống thông tin**
Mã số: **9.48.01.04**

LUẬN ÁN TIẾN SĨ KỸ THUẬT

HÀ NỘI – NĂM 2025

BỘ KHOA HỌC VÀ CÔNG NGHỆ
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

HOÀNG NGỌC CẢNH

PHÁT TRIỂN MỘT SỐ PHƯƠNG PHÁP TRUY VẤN HIỆU QUẢ TRÊN CƠ SỞ DỮ LIỆU QUAN HỆ MÃ HOÁ

Ngành: **Hệ thống thông tin**
Mã số: **9.48.01.04**

LUẬN ÁN TIẾN SĨ KỸ THUẬT

NGƯỜI HƯỚNG DẪN KHOA HỌC:
1. GS.TS. Nguyễn Hiếu Minh
2. TS. Ngô Đức Thiện

HÀ NỘI – NĂM 2025

LỜI CAM ĐOAN

Tôi xin cam đoan các kết quả trình bày trong luận án Tiến sĩ ***"Phát triển một số phương pháp truy vấn hiệu quả trên cơ sở dữ liệu quan hệ mã hóa"*** là các công trình nghiên cứu của tôi dưới sự hướng dẫn của GS. TS. Nguyễn Hiếu Minh và TS. Ngô Đức Thiện, trừ những kiến thức tham khảo từ các tài liệu đã được tham chiếu rõ ràng.

Các kết quả nghiên cứu trong luận án là trung thực, một phần đã được công bố trên các Tạp chí, Hội thảo khoa học (danh sách các công trình được liệt kê tại cuối Luận án), phần còn lại chưa được công bố trong bất kỳ công trình nào khác.

Mọi nội dung/dữ liệu được tham khảo trong luận án đều được trích dẫn đầy đủ và đúng quy định.

Hà Nội, ngày .... tháng .... năm 2025

Tác giả luận án

Hoàng Ngọc Cảnh

LỜI CẢM ƠN

Suốt quá trình chuẩn bị, học tập, nghiên cứu và hoàn thành luận án, Nghiên cứu sinh luôn nhận được sự giúp đỡ, hướng dẫn và nhiều góp ý quý báu cùng những lời động viên khích lệ từ các Thầy Cô, các Nhà khoa học, các Cộng sự, Đồng nghiệp và Gia đình. Với sự biết ơn sâu sắc, Nghiên cứu sinh trân trọng gửi lời cảm ơn tới:

Ban Giám đốc Học viện Công nghệ Bưu chính Viễn thông, Khoa Sau đại học và các Thầy cô, Cán bộ quản lý của Học viện đã luôn ân cần, quan tâm tạo điều kiện thuận lợi để nghiên cứu sinh hoàn thành nhiệm vụ học tập, nghiên cứu.

Nghiên cứu sinh xin gửi lời cảm ơn đặc biệt tới GS.TS. Nguyễn Hiếu Minh và TS. Ngô Đức Thiện. Các Thầy đã luôn tận tâm chỉ bảo, theo sát và động viên tôi trong suốt thời gian học tập, nghiên cứu và hoàn thành luận án.

Tôi xin chân thành cảm ơn sự hỗ trợ tuyệt vời của các nhà khoa học, các đồng nghiệp và các cộng sự trong các công trình nghiên cứu đã công bố.

Tôi xin cảm ơn Đại học Thương mại, cơ quan tôi đang công tác, đã luôn tạo điều kiện tối đa để tôi hoàn thành quá trình học tập nghiên cứu của mình.

Cuối cùng tôi xin dành lời tri ân sâu sắc tới Gia đình, Bố mẹ, Vợ con cùng những người thân đã luôn quan tâm, đồng hành và chia sẻ để tôi vững tâm thực hiện nhiệm vụ học tập, nghiên cứu được trọn vẹn.

Mặc dù đã dành sự tập trung và cố gắng để thực hiện luận án nhưng Nghiên cứu sinh tự nhận thấy còn nhiều hạn chế về chuyên môn, về tính logic trong các nội dung luận án và khả năng trình bày văn bản. Vì vậy, Nghiên cứu sinh kính mong nhận được nhiều góp ý quý báu từ các Thầy Cô và các Nhà khoa học để luận án được hoàn thiện hơn.

Xin chân thành biết ơn!

NCS. Hoàng Ngọc Cảnh

MỤC LỤC

DANH MỤC CÁC THUẬT NGỮ, CHỮ VIẾT TẮT

Thuật ngữ	Diễn giải tiếng Anh	Diễn giải tiếng Việt
AES	Advanced Encryption Standard	Chuẩn mã hóa nâng cao
CSDL	Database	Cơ sở dữ liệu
CSDLQH	Database Relationships	Cơ sở dữ liệu quan hệ
CS	Cloud Server	Máy chủ đám mây
DAS	Database As a Service	Cơ sở dữ liệu như một dịch vụ
DBMS	Database Management System	Hệ quản trị cơ sở dữ liệu
DES	Data Encryption Standard	Chuẩn mã hóa dữ liệu
DIQ-SSE	Double Indexes Query - Searchable Symmetric Encryption	Lược đồ SSE hỗ trợ truy vấn hiệu quả dữ liệu ký tự mã hóa qua cặp chỉ mục
DO	Data Owner	Người sở hữu dữ liệu
DSP	Database Service Provider	Nhà cung cấp dịch vụ cơ sở dữ liệu thuê ngoài/đám mây
ERD	Entity Relationship Diagram	Lược đồ liên kết thực thể
ERM	Entity Relationship Model	Mô hình liên kết thực thể
ESIT-SSE	Efficient search index tree - Searchable Symmetric Encryption	Lược đồ SSE hỗ trợ truy vấn hiệu quả dữ liệu số mã hóa qua cây chỉ mục
EU	End User	Người dùng cuối
FHE	Fully Homomorphic encryption	Mã hóa đồng hình hoàn toàn
FHOPE	Frequency-hiding order preserving encryption	Mã hóa bảo toàn thứ tự ẩn tần số
IND-CPA	Indistinguishability against chosen plaintext attacks	Khả năng không phân biệt được đối với các cuộc tấn công văn bản gốc được chọn
IND-CKA (IND1-CKA, IND2-CKA, IND-CKA1, IND-CKA2)	Indistinguishability against adaptive chosen keyword attacks. IND-CKA1 (Nonadaptive) IND-CKA2 (Adaptive)	Khả năng không phân biệt được đối với các cuộc tấn công từ khóa được chọn (thích ứng/không thích ứng)
HE	Homomorphic encryption	Mã hóa đồng hình
IBSE	Index-based Searchable Encryption	Mã hoá có thể tìm kiếm dựa trên chỉ mục

IHV	Information hiding vector	Vector giấu tin
NCS		Nghiên cứu sinh
OLAP	Online analytical processing	Phân tích trực tuyến
OLTP	Online Transactional Processing	Xử lý giao dịch trực tuyến
OPE / OPES	Order Preserving Encryption Scheme	Lược đồ mã hóa bảo toàn thứ tự
OPESS	Order preserving encryption with splitting and scaling	Mã hóa bảo toàn thứ tự với kỹ thuật chia tách và nhân rộng
PDV	Probability Distribution Deviation	Độ lệch phân phối xác suất
PHE	Partially Homomorphic encryption	Mã hóa đồng hình một phần
PEKS	Public key encryption with keyword search	Mã hóa công khai với tìm kiếm bằng từ khóa
PKSE	Public key searchable encryption	Mã hoá công khai có thể tìm kiếm
PPT	Probabilistic Polynomial-Time	Xác suất đa thức thời gian
PROXY	Proxy Server	Máy chủ trung gian an toàn, tin cậy được quản lý hoặc uỷ quyền bởi DO.
PRF	Pseudorandom Function	Hàm giả ngẫu nhiên.
PRG	Pseudorandom Generator	Bộ tạo giả ngẫu nhiên.
RDB	Relation Database	Cơ sở dữ liệu quan hệ
RDM	Relational Data Model	Mô hình liên kết dữ liệu
SE	Searchable Encryption	Mã hoá có thể tìm kiếm
SSE	Searchable Symmetric Encryption	Mã hoá đối xứng có thể tìm kiếm

DANH MỤC CÁC KÝ HIỆU

Ký hiệu	Ý nghĩa của ký hiệu
f^{Enc}	Là hàm mã hóa.
f^{Dec}	Là hàm giải mã.
$h_i(.)$	Là hàm băm.
$\|\|$	Phép nối hai chuỗi
$a \leftarrow AT(.)$	$a \leftarrow AT(.)$ là kết quả trả về khi thực thi thuật toán AT với $(.)$ là các tham số.
$\|A\|$	Phép lấy số ký tự của một chuỗi hoặc số phần tử của một tập giá trị.
$\overline{v_i}$	Phép lấy giá trị tuyệt đối của số v_i.
$\cup$	Phép hợp hai tập giá trị.
$\subset$	Ký hiệu tập con.
$\wedge$	Phép và (And Bitwise) nhị phân.
$\vee$	Phép hoặc (Or Bitwise) nhị phân.
$B_i^{\gg k}$	Phép dịch vòng phải k bit của chuỗi B_i.
$B_i^{\ll k}$	Phép dịch vòng trái k bit của chuỗi B_i.
M	Ma trận khả nghịch.
M^{-1}	Ma trận nghịch đảo của M.
M^T	Ma trận chuyển vị của M.
M x M'	Phép nhân hai ma trận.
$\overrightarrow{U_b} \perp \overrightarrow{U_c}$	Hai vector trực giao.
$\overrightarrow{U_b} . \overrightarrow{U_c}$	Tích vô hướng hai vec tor.
R	Là một quan hệ rõ (là một bảng) trong CSDL quan hệ.
R^E	Là quan hệ mã tương ứng với R.
$\overrightarrow{ıhv_b(b)}$	Là vector giấu tin cho mỗi giá trị b trong các bản ghi lưu trên CS.
$\overrightarrow{ıhv_c(c)}$	Là vector giấu tin cho giá trị c của điều kiện truy vấn được gửi từ Proxy.

DANH MỤC CÁC HÌNH ẢNH

DANH MỤC CÁC BẢNG

DANH MỤC CÁC THUẬT TOÁN

DANH MỤC CÁC NHẬN XÉT

MỞ ĐẦU

1. Đặt vấn đề

Trong thời đại chuyển đổi số, việc ứng dụng các hệ thống thông tin quản lý tại các tổ chức, doanh nghiệp đã trở thành xu hướng tất yếu. Trái tim của các hệ thống này chính là cơ sở dữ liệu (CSDL), trong đó cơ sở dữ liệu quan hệ (CSDLQH) là loại được ứng dụng rộng rãi đã được chứng minh chặt chẽ về toán học, nơi mà người chủ sở hữu dữ liệu (DO) có thể dễ dàng lưu trữ, quản lý và phân phối thông tin cho các hoạt động của tổ chức, doanh nghiệp. Có hai hình thức triển khai CSDL là: trong hạ tầng của nội bộ tổ chức và trên nền tảng hạ tầng thuê ngoài của các nhà cung cấp dịch vụ (DSP). Trong đó hình thức triển khai CSDL trên nền tảng hạ tầng thuê ngoài (gọi chung là CSDL thuê ngoài) ngày càng phổ biến bởi các lợi ích như: DO không phải quan tâm tới vấn đề đầu tư phần cứng, phần mềm, đường truyền và đội ngũ nhân viên quản trị; chi phí rẻ do ngày càng có nhiều DSP cạnh tranh cung cấp các dịch vụ cho thuê hạ tầng (gồm việc tích hợp sẵn dịch vụ CSDL) …trên nền tảng công nghệ điện toán đám mây (ví dụ một số DSP trong nước: VNPT, Mắt bão, Viettel, FPT, … và trên thế giới: Amazon EC2 và S3, Rackspace, Linode, MS Azure, Oracle, …).

Trong thực tế sẽ luôn có các hợp đồng ràng buộc giữa DO và DSP để đảm bảo sự an toàn bảo mật của CSDL thuê ngoài, nhưng điều đó chưa đủ để DO tin tưởng các quản trị viên của DSP, bởi họ có thể trung thực nhưng tò mò với dữ liệu của người dùng hoặc một kịch bản xấu hơn là tin tặc chiếm quyền root và có toàn quyền truy cập vào máy chủ để dễ dàng khai thác dữ liệu. Vì vậy, một vấn đề lớn được đặt ra là làm sao để CSDL thuê ngoài đảm bảo được an toàn, ngăn cấm việc khai thác thông tin bất hợp pháp của các tổ chức/cá nhân không có thẩm quyền, gồm cả nhà cung cấp dịch vụ. Rõ ràng ngoài việc DO cần có thêm các điều khoản hợp đồng chặt chẽ hơn về an ninh dữ liệu với DSP và tăng cường các lớp bảo mật cho đường truyền, máy chủ, hệ điều hành, hệ quản trị CSDL thì việc xây dựng các biện pháp chủ động che giấu nội dung dữ liệu nhạy cảm (bằng phương pháp mã hóa) là vô cùng quan trọng. Có thể nói mã hóa dữ liệu trên CSDL thuê ngoài là trạng thái bảo mật cao nhất nhằm ngăn chặn sự tò mò của DSP cũng như các cuộc tấn công của kẻ gian khi họ đã vượt qua các lớp bảo mật hoặc lợi dụng truy cập trái phép nhằm tiếp cận dữ liệu nhạy cảm.

Khi dữ liệu mã hóa được lưu trên CSDL thuê ngoài, đồng nghĩa CSDL này sẽ không được tồn tại các khóa hoặc metadata bí mật để phục vụ quá trình mã hóa, giải

mã hoặc truy xuất thông tin. Do đó mọi yêu cầu truy vấn trên dữ liệu rõ từ người dùng phải được chuyển đổi thành các lệnh truy vấn trên dữ liệu mã hóa và gửi tới CSDL thuê ngoài để thực thi. Tuy nhiên dữ liệu nhạy cảm (dạng số, ký tự, logic, …) sau khi mã hóa bởi các thuật toán mật mã tiêu chuẩn như AES [37], DES [13], Blowfish [98]… sẽ không còn giữ được các tính chất vốn có ban đầu như: thứ tự, đối sánh, tính toán số học, thống kê,… Vì vậy bài toán ***truy vấn hiệu quả trên dữ liệu mã (đặc biệt là truy vấn trên CSDLQH mã hoá thuê ngoài) có tính cấp thiết*** và được quan tâm đặc biệt trong những năm gần đây. Tính hiệu quả được hiểu là khả năng cân bằng được càng nhiều yếu tố sau càng tốt, gồm: bảo mật, hiệu năng, đa dạng truy vấn và khả thi triển khai (trong đó có khả năng tuỳ chỉnh các tham số để nâng cao một trong các yếu tố trên cho phù hợp với nhu cầu triển khai thực tiễn).

2. Lý do chọn đề tài

Trong những năm qua, nhiều công trình khoa học đã tập trung giải quyết các khía cạnh liên quan của lĩnh vực "mã hóa có thể tìm kiếm" nói chung và bài toán "xử lý truy vấn trên cơ sở dữ liệu quan hệ mã hoá" nói riêng. Các nghiên cứu đã giới thiệu hàng loạt lược đồ mã hoá có thể tìm kiếm (SE) dựa vào các kỹ thuật mã hoá [123], [47], [100], [19], [78], [8], [71] và phương thức tìm kiếm [43], [114], [69], [40], [32], [58], [30], [5], [41], [120], [31] cho phép thực hiện được một số kiểu truy vấn trên dữ liệu mã hoá tương ứng với truy vấn trên dữ liệu rõ. Tuy nhiên các lược đồ đã đề xuất đều chưa giải quyết trọn vẹn các khía cạnh về tính hiệu quả truy vấn, trong đó việc đánh đổi giữa hiệu năng và bảo mật là một tồn tại điển hình. Theo đó, các giải pháp hỗ trợ truy vấn trên dữ liệu ký tự mã hoá tập trung quá nhiều vào vấn đề tìm kiếm từ khoá, chỉ phù hợp trong các ứng dụng đã xác định được tập từ khoá cố định trên giao diện tìm kiếm. Còn đối với giải pháp truy vấn trên dữ liệu số mã hoá, việc nâng cao hiệu năng truy vấn khoảng lại gây ra nhiều nguy cơ rò rỉ thông tin bản rõ qua tính chất bảo toàn thứ tự bản mã. Rõ ràng còn rất nhiều vấn đề cần tiếp tục được nâng cấp và giải quyết nhằm đưa lời giải bài toán truy vấn trên dữ liệu mã trở lên hiệu quả hơn, khả thi triển khai trong thực tế hơn. Ví dụ như người dùng cần một hệ thống có khả năng tích hợp các chức năng cho phép tìm kiếm hiệu quả trên hai kiểu dữ liệu phổ biến là ký tự và số mã hoá với các phương pháp tìm kiếm phổ rộng, gần gũi thực tiễn như truy vấn chuỗi con, truy vấn khoảng giá trị số bất kỳ trong CSDLQH.

Từ thực trạng và nhận định nêu trên, NCS lựa chọn đề tài luận án ***"Phát triển***

một số phương pháp truy vấn hiệu quả trên cơ sở dữ liệu quan hệ mã hóa" nhằm nghiên cứu, xây dựng các mô hình/lược đồ tìm kiếm hỗ trợ hiệu quả một số dạng thức truy vấn phổ biến trên các dữ liệu kiểu số và ký tự trong CSDLQH mã hóa thuê ngoài. Kết quả nghiên cứu của luận án có tính cấp thiết, ý nghĩa khoa học và thực tiễn cao nhằm tăng cường bảo mật và khai thác dữ liệu hiệu quả trong các hệ thống thông tin của các doanh nghiệp/cơ quan trước xu thế chuyển đổi số, thuê dịch vụ CSDLQH thuê ngoài ngày càng lớn hiện nay.

3. Đối tượng, phạm vi nghiên cứu của luận án

Đối tượng nghiên cứu:

- Các lược đồ mã hoá có thể tìm kiếm.

- Mô hình cơ sở dữ liệu như một dịch vụ (DAS), khu vực trung gian (Proxy) trong triển khai lược đồ mã hoá có thể tìm kiếm.

- Các yếu tố liên quan đến tính hiệu quả của các lược đồ mã hoá có thể tìm kiếm.

- Phương pháp hỗ trợ truy vấn hiệu quả trên các kiểu dữ liệu phổ biến trong CSDLQH mã hóa thuê ngoài.

Phạm vi nghiên cứu

1) Các dạng thức truy vấn trên CSDLQH mã hoá thuê ngoài: Các yêu cầu truy vấn trong CSDLQH rõ luôn đa dạng, phong phú và được hỗ trợ linh hoạt bởi tập lệnh SQL tiêu chuẩn. Tuy nhiên trong CSDLQH mã hoá, việc xử lý một câu lệnh truy vấn cơ bản (như truy vấn bằng) cũng trở lên rất phức tạp và khó khăn do tính chất của dữ liệu mã. Vì vậy, luận án sẽ ***giới hạn việc truy vấn trên dữ liệu mã với các dạng thức lệnh truy vấn phổ biến và cơ bản nhất***, là cơ sở quan trọng để mở rộng nghiên cứu cho phép thực thi các loại truy vấn đa dạng sau này. Cụ thể ***phạm vi các dạng thức truy vấn*** trong luận án được giới hạn như sau:

- Việc ưu tiên đa dạng truy vấn có phải đánh đổi bởi các yếu tố bảo mật, hiệu năng và khả thi (**Hình 1.10**). Thực tế là nhiều nghiên cứu và ứng dụng truy vấn trên dữ liệu mã [140], [141], [55] được giới thiệu gần đây cũng chỉ tập trung vào giải quyết một số dạng thức và điều kiện truy vấn phổ biến (như Microsoft Always Encrypted giới hạn hỗ trợ điều kiện truy vấn bằng). Theo xu hướng này, luận án cũng lựa chọn các truy vấn đơn giản (như dạng thức lệnh Select trên một quan hệ với điều kiện truy vấn phổ dụng trên một thuộc tính cụ thể) để tập trung hơn cho các đề xuất và kỹ thuật xử lý nhằm cải thiện tính hiệu quả khi truy vấn tương ứng trên dữ liệu mã. Đây là

những dạng truy vấn cốt lõi, là cơ sở để mở rộng ra các câu lệnh truy vấn phức tạp hơn trong các nghiên cứu tiếp theo của lĩnh vực SE cũng như hướng phát triển luận án. Mục tiêu này của luận án phù hợp ứng dụng vào các cơ sở dữ liệu trong hệ thống OLAP khi tập trung chính vào vấn đề truy xuất và phân tích dữ liệu. Còn đối với các hệ thống OLTP ưu tiên cho việc xử lý các giao dịch, thì các kết quả nghiên cứu của luận án cũng khả thi khi áp dụng vào quá trình mã hóa dữ liệu khi Insert hay thực thi điều kiện truy vấn của mệnh đề Where khi Update/Delete trên dữ liệu mã. Tuy nhiên để nội dung nghiên cứu được tập trung hơn vào những tồn tại hiện nay trong lĩnh vực SE và giải quyết hiệu quả các câu lệnh truy vấn "lõi" với điều kiện truy vấn phổ biến thì vấn đề mở rộng các dạng thức truy vấn khác (như truy vấn chéo trên nhiều bảng, truy vấn lồng, kết hợp hàm thống kê, đa điều kiện truy vấn và nhóm lệnh thay đổi dữ liệu Insert, Delete, Update …) sẽ không nằm trong phạm vi của luận án.

- Luận án tập trung giải quyết các lệnh truy vấn đơn giản nhưng phổ dụng trên các trường dữ liệu dạng số và ký tự, đây là hai kiểu dữ liệu được dùng nhiều nhất trong các CSDLQH. Cụ thể luận án lựa chọn mục tiêu xử lý hiệu quả các truy vấn được người dùng thường xuyên yêu cầu là: truy vấn chuỗi con theo chuỗi điều kiện "$LIKE\ '\%\ substring\ \%'$" trên dữ liệu ký tự và truy vấn khoảng theo chuỗi điều kiện "$BETWEEN\ l\ and\ h$" trên dữ liệu số.

2) Các lược đồ mã hóa đối xứng có thể tìm kiếm (SSE) dựa trên chỉ mục mã hoá hỗ trợ tìm kiếm (hay còn gọi là chỉ mục mù). Các phạm vi nghiên cứu cụ thể khi thiết kế lược đồ gồm:

- Nghiên cứu cấu trúc các lược đồ SSE tương ứng với các kiểu dữ liệu và điều kiện truy vấn đã đề xuất.

-Phương pháp và cấu trúc dữ liệu hỗ trợ xây dựng các chỉ mục mù, quy trình và các thuật toán hỗ trợ truy vấn trên chỉ mục mù.

- Đánh giá hiệu quả của lược đồ đề xuất. Tính hiệu quả được diễn giải trên các tiêu chí về: ***bảo mật, hiệu năng, khả thi triển khai***. Trong đó bảo mật tập trung vào khả năng chống rò rỉ thông tin chỉ mục và mẫu truy vấn; hiệu năng tập trung vào độ phức tạp tìm kiếm và thời gian thực thi đo đạc trên dữ liệu thử nghiệm; khả thi triển khai tập trung vào việc dễ dàng tích hợp các thành phần, quy trình vận hành của các lược đồ thống nhất trên một mô hình triển khai, cũng như hỗ trợ tuỳ chỉnh tham số phù hợp vận hành thực tiễn.

3) Triển khai lược đồ tìm kiếm SSE đề xuất trên mô hình DAS-PROXY. Các

vấn đề mở rộng sau ***không nằm trong phạm vi luận án***, gồm: đánh giá năng lực Proxy; cân bằng tải; xử lý song song; đa người sở hữu dữ liệu/đa máy chủ; xác thực - toàn vẹn dữ liệu; quản lý khoá; phân quyền sử dụng; cập nhật thay đổi chỉ mục, …

4. Mục tiêu của luận án

Mục tiêu chung

Mục tiêu nghiên cứu của luận án là tổng quan về các lược đồ mã hoá có thể tìm kiếm và các tiêu chí phản ánh tính hiệu quả của lược đồ. Đồng thời chỉ ra các thực trạng đối với truy vấn trên dữ liệu ký tự, dữ liệu số mã hoá nói chung và trong CSDLQH mã hoá nói riêng. Từ đó đề xuất các mô hình, lược đồ và thuật toán truy vấn hiệu quả với hai dạng thức dữ liệu phổ biến nêu trên theo các chuỗi điều kiện truy vấn hay được sử dụng trong CSDLQH mã hoá thuê ngoài.

Mục tiêu cụ thể

- Tổng quan được tình hình nghiên cứu trong lĩnh vực mã hoá có thể tìm kiếm. Phân tích và chỉ ra tồn tại của các công trình nghiên cứu liên quan tới truy vấn dữ liệu dạng số và ký tự đã mã hóa.

- Nghiên cứu áp dụng mô hình DAS-PROXY trong triển khai các lược đồ mã hoá đối xứng có thể tìm kiếm (SSE) dựa trên chỉ mục mù.

- Nghiên cứu các kiến thức liên quan về CSDL quan hệ, cấu trúc dữ liệu B^+Tree, bộ lọc Bloom, … phục vụ xây dựng các chỉ mục mù và thuật toán truy vấn hiệu quả.

- Đề xuất các lược đồ SSE mới dựa vào chỉ mục mù cho phép truy vấn hiệu quả dữ liệu dạng số và ký tự trên CSDLQH mã hóa thuê ngoài theo các điều kiện truy vấn thường được người dùng yêu cầu.

- Thực nghiệm và phân tích tính hiệu quả của các lược đồ SSE đã đề xuất, trong đó tập trung chính vào các khía cạnh là ***đánh giá bảo mật*** và ***hiệu năng thực thi*** truy vấn của các lược đồ cũng như ***tính khả thi triển khai*** lược đồ truy vấn.

- So sánh hiệu quả với các công trình nghiên cứu liên quan nổi bật trước đó.

5. Phương pháp nghiên cứu trong luận án

Một số phương pháp và cách thức tiếp cận nghiên cứu trong luận án như sau:

Phương pháp lý luận chung: Phân tích vấn đề an toàn dữ liệu và khả năng đáp ứng của các giải pháp bảo mật dữ liệu truyền thống khi triển khai CSDL thuê ngoài. Từ đó chỉ ra các vấn đề tồn tại và tính cấp thiết, bước đầu xác định được định hướng nghiên cứu: "*Tìm kiếm trên dữ liệu mã hoá trong CSDL thuê ngoài*". Tiếp theo, hệ

thống danh mục công trình nghiên cứu tiêu biểu liên quan đến hướng nghiên cứu nói chung và đề tài nói riêng. Từ đó phân tích chỉ ra các tồn tại nhằm xác định được đối tượng, phạm vi chi tiết, mục tiêu và nội dung nghiên cứu cụ thể của luận án.

Phát triển các phương pháp giải quyết bài toán dựa trên các đề xuất cấu trúc dữ liệu mới kết hợp với lựa chọn áp dụng các điểm mạnh từ những nghiên cứu trước đó: Xây dựng cơ sở dẫn luận và ý tưởng giải quyết bài toán, sau đó hiện thực theo thứ tự: lựa chọn mô hình triển khai lược đồ tìm kiếm, đề xuất thiết kế và thành phần lược đồ, xây dựng chỉ mục mù dựa trên các cấu trúc dữ liệu và phương pháp biến đổi, xây dựng quy trình và thuật toán truy vấn trên chỉ mục.

Đánh giá hiệu quả của các phương pháp đề xuất: Lựa chọn CSDL và các kịch bản, chỉ số đo lường phù hợp để thực nghiệm và phân tích đánh giá hiệu quả của lược đồ đề xuất. Cuối cùng là so sánh sự hiệu quả với các nghiên cứu liên quan trước đó.

6. Ý nghĩa khoa học và đóng góp của luận án

Ý nghĩa khoa học của luận án:

Ngày nay người dùng, tổ chức ngày càng ưu tiên sử dụng các dịch vụ CSDL thuê ngoài và đang phải đối mặt với nhiều rủi ro về an ninh, an toàn dữ liệu. Bối cảnh này cho thấy đề tài luận án có tính thời sự, tính mới và ý nghĩa khoa học. Luận án đóng góp vào bức tranh nghiên cứu sôi động trong những năm gần đây của lĩnh vực mã hoá có thể tìm kiếm nói chung và truy vấn trên CSDLQH mã hoá nói riêng.

Đóng góp của luận án:

Luận án có *hai đóng góp chính* là *đề xuất hai lược đồ SSE* tương ứng hỗ trợ truy vấn hiệu quả chuỗi con trên dữ liệu ký tự và truy vấn khoảng trên dữ liệu số trong CSDLQH mã hoá. Đồng thời áp dụng nhất quán mô hình DAS-PROXY [135], [111], [6], [91], [9], [125], [137], [92] trong triển khai hai lược đồ đã đề xuất, mang lại tính khả thi và tiềm năng cao trong ứng dụng thực tiễn. Tính hiệu quả thể hiện qua khả năng đảm bảo các tiêu chí về bảo mật, hiệu năng và tính khả thi triển khai của các lược đồ. Tính bảo mật thể hiện ở khả năng chống rò rỉ thông tin chỉ mục và mẫu truy vấn. Hiệu năng thể hiện qua: sử dụng một vòng giao tiếp dữ liệu, chi phí truy vấn tập trung tại CS, thời gian thực thi truy vấn tốt, độ phức tạp tìm kiếm đạt ưu thế so với các công trình tiêu biểu liên quan. Tính khả thi triển khai thể hiện thông qua quy trình triển khai thống nhất và cho phép điều chỉnh các tham số tác động đến tính bảo mật, hiệu năng trong ứng dụng thực tiễn. Ngoài ra hai dạng thức lệnh truy vấn được đề

xuất trong luận án được đánh giá là phổ biến, gần gũi với thói quen tìm kiếm của người dùng trên các ứng dụng kết nối CSDLQH.

Cụ thể hai đóng góp như sau:

1) Đề xuất *xây dựng lược đồ DIQ-SSE* dựa trên chỉ mục mù hỗ trợ truy vấn chuỗi con hiệu quả trên dữ liệu ký tự trong CSDLQH mã hoá khi người dùng yêu cầu chuỗi điều kiện "$LIKE\ '\%\ substring\ \%'$". Điểm nổi bật của đóng góp này là *xây dựng quy trình truy vấn tuần tự trên hai chỉ mục mù* $Index1, Index2$ và *sử dụng một số cấu trúc dữ liệu mới trong xây dựng* $Index1, Index2$. Với cơ chế tìm kiếm của chỉ mục $Index1$ dựa vào từ khoá, có ưu điểm tốc độ thực thi nhanh, tỷ lệ lọc và tính bảo mật cao. Trong khi đó chỉ mục $Index2$ lại cho phép tìm kiếm chính xác theo chuỗi con, cùng ưu điểm bảo mật tốt, không trả về kết quả dương tính giả. Mặc dù truy vấn trên Index2 có tốc độ truy vấn chậm hơn so với trên $Index1$, nhưng lại có lợi thế chỉ thực hiện trên tập kết quả rút gọn nhỏ hơn được trả về bởi quá trình tìm kiếm trên $Index1$.

2) Đề xuất *xây dựng lược đồ ESIT-SSE* dựa trên chỉ mục mù hỗ trợ truy vấn khoảng hiệu quả trên dữ liệu số trong CSDLQH mã hoá khi được người dùng yêu cầu chuỗi điều kiện "$BETWEEN\ l\ and\ h$". Điểm nổi bật của đóng góp này là *đưa ra quy trình xây dựng chỉ mục mù qua hai bước và xây dựng một cấu trúc dữ liệu đặc biệt hỗ trợ truy vấn khoảng hiệu quả trên chỉ mục mù.* Cụ thể: *Bước 1.* Xây dựng chỉ mục NewBucketIndex dựa trên cải tiến kỹ thuật phân khoảng dữ liệu; *Bước 2.* Đề xuất sử dụng các vector giấu tin IHV cho phép che giấu thông tin thứ tự của các NewBucketIndex; *Bước 3.* Xây dựng cấu trúc dữ liệu IHV_B$^+$Tree từ các giá trị IHV và các thuật toán truy vấn khoảng hiệu quả trên cấu trúc này.

7. Bố cục của luận án

Luận án bao gồm các phần: mở đầu, 3 chương, kết luận và hướng phát triển, các công trình khoa học đã công bố và danh mục tài liệu tham khảo. Nội dung chính 3 chương như sau:

Chương 1. Tổng quan về mã hoá có thể tìm kiếm và vấn đề nghiên cứu

Trong chương 1, luận án tập trung giới thiệu về lược đồ mã hoá có thể tìm kiếm, phân loại các lược đồ, các kỹ thuật tổ chức mã hoá - phương thức tìm kiếm trong xây dựng lược đồ, mô hình DAS-PROXY và tình hình nghiên cứu liên quan tới đề tài luận án. Từ đó nêu các nhận xét, bình luận, chỉ ra các khoảng trống nghiên cứu và kết

luận các vấn đề cần giải quyết trong luận án. Trong chương này cũng trình bày một số kiến thức cơ sở phục vụ cho các nội dung sẽ trình bày tại chương 2, 3.

Chương 2. Phát triển phương pháp truy vấn chuỗi con hiệu quả trên dữ liệu ký tự trong CSDLQH mã hoá

Trong chương 2, luận án tập trung đề xuất xây dựng lược đồ DIQ-SSE và triển khai trên mô hình DAS-PROXY cho phép truy vấn chuỗi con hiệu quả trong CSDLQH mã hoá thuê ngoài thông qua chuỗi điều kiện "$LIKE\ '\%\ substring\ \%'$". Trong đó điểm nhấn là xây dựng cặp chỉ mục mù đặc biệt $Index1, Index2$, các thuật toán liên quan cùng quy trình truy vấn hiệu quả cho lược đồ. Chương 2 cũng dành một phần lớn nội dung để phân tích bảo mật, thực nghiệm đánh giá hiệu năng, độ phức tạp tìm kiếm để khẳng định tính hiệu quả của lược đồ DIQ-SSE.

Chương 3. Phát triển phương pháp truy vấn khoảng hiệu quả trên dữ liệu số trong CSDLQH mã hoá

Trong chương 3, luận án tập trung đề xuất xây dựng lược đồ ESIT-SSE và triển khai trên mô hình DAS-PROXY cho phép truy vấn khoảng hiệu quả trong CSDLQH mã hoá thuê ngoài thông qua chuỗi điều kiện "$BETWEEN\ l\ and\ h$". Trọng tâm của quá trình này là kỹ thuật phân khoảng dữ liệu số và các phương pháp biến đổi cho phép tạo ra một chỉ mục đặc biệt (còn gọi là vector giấu tin IHV) cung cấp khả năng so sánh an toàn nhưng không tiết lộ nội dung và thứ tự chỉ mục. Tiếp theo đề xuất cấu trúc dữ liệu IHV_B$^+$Tree và thuật toán tìm kiếm cho phép tăng tốc truy vấn khoảng. Cuối chương tập trung làm rõ tính hiệu quả của lược đồ ESIT-SSE thông qua các thực nghiệm đánh giá hiệu năng, độ phức tạp tìm kiếm và phân tích bảo mật.

CHƯƠNG 1.
TỔNG QUAN VỀ MÃ HOÁ CÓ THỂ TÌM KIẾM VÀ VẤN ĐỀ NGHIÊN CỨU

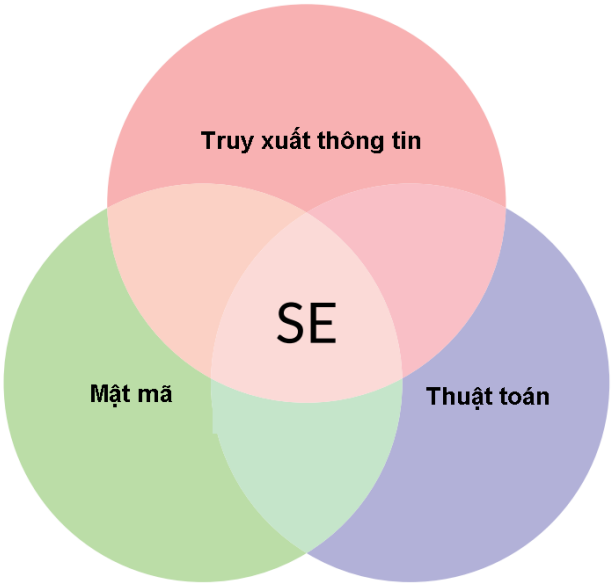
Phần đầu chương 1 trình bày tổng quan về mã hoá có thể tìm kiếm, cụ thể: các lược đồ mã hoá có thể tìm kiếm và triển khai chúng trên mô hình DAS-PROXY; phân loại và giới thiệu một số kỹ thuật tổ chức mã hoá, phương thức tìm kiếm tiêu biểu; các tiêu chí về tính hiệu quả của lược đồ, … Phần tiếp theo tập trung liệt kê, phân tích và chỉ ra các tồn tại trong các công trình về tìm kiếm trên dữ liệu số và ký tự mã hoá, từ đó làm căn cứ để xác định hướng nghiên cứu đề tài luận án. Cuối cùng là kết luận chương và mở ra các mục tiêu cần giải quyết cho các chương tiếp theo.

1.1. Tổng quan về mã hoá có thể tìm kiếm (SE)

Với các thách thức của bài toán mã hoá có thể tìm kiếm khi được triển khai lưu trữ, quản lý tại các hạ tầng thuê ngoài, các nhà khoa học tạo ra một lĩnh vực nghiên cứu mới là "Mã hóa có thể tìm kiếm (sau đây gọi là SE)", là sự kết hợp đặc biệt giữa các nghiên cứu về bảo mật, truy xuất thông tin và giải thuật tìm kiếm (xem **Hình 1.1**) nhằm giải quyết hiệu quả bài toán đặt ra. Cụm từ "Truy vấn" sử dụng trong luận án liên quan tới việc tìm kiếm và truy xuất thông tin trong CSDL nói chung và CSDLQH nói riêng. Vì vậy, vấn đề "truy vấn dữ liệu trong CSDLQH mã hóa thuê ngoài" cũng chỉ là một trường hợp trong lĩnh vực SE và vẫn mang đầy đủ các đặc tính của bài toán "mã hoá có thể tìm kiếm" sẽ được trình bày trong chương 1.

Dựa vào đối tượng và phạm vi nghiên cứu của đề tài, nhằm tạo sự nhất quán trong các nội dung trình bày, luận án sử dụng cụm từ *"**CSDLQH mã hóa**"* để thay cho các cụm từ ***"CSDL mã hóa thuê ngoài"***, ***"CSDL quan hệ mã hoá"*** hay ***"CSDL quan hệ mã hoá thuê ngoài"***.

Trong thực tế, cách thức tìm kiếm của người dùng trên dữ liệu rõ phong phú về ý tưởng và điều kiện truy xuất, vì vậy khi chuyển sang tìm kiếm trên dữ liệu mã hóa thì các nhà nghiên cứu cũng cố gắng đề xuất những giải pháp tương ứng nhằm đưa SE gần với ứng dụng thực hơn. Mặc dù **Bảng 1.1** chỉ thống kê một số hướng và công trình nghiên cứu tiêu biểu trong lĩnh vực SE theo dòng thời gian, nhưng qua đó cho thấy sự đa dạng về kỹ thuật tổ chức mã hoá và phương thức truy vấn trên dữ liệu mã.



Hình 1.1. Các lĩnh vực liên quan trong SE [51]

Bảng 1.1. Một số hướng nghiên cứu trong SE theo thời gian

Năm	**Hướng nghiên cứu tiêu biểu trong SE**
2000	Mã hóa đối xứng có thể tìm kiếm [100]
2003	Chỉ mục mã hóa có thể tìm kiếm [43]
2004	Mã hóa công khai có thể tìm kiếm [19]
2008	Tìm kiếm đa từ khóa và tìm kiếm liên hợp các từ khóa [114]
2010	Tìm kiếm từ khóa đơn có xếp hạng [109] Tìm kiếm mờ [69]
2011	Tìm kiếm đa từ khóa có xếp hạng [24]
2013	Tìm kiếm dựa trên từ khóa đồng nghĩa [40] Tìm kiếm dựa trên ngữ nghĩa của từ khóa [32]
2014	Giải pháp SE dựa trên phần cứng [97]
2015-2023	Chỉ mục tìm kiếm thích ứng [58] Tìm kiếm chuỗi con [30] Truy vấn dữ liệu mã dựa trên chia sẻ thông tin bí mật [48] Mã hóa đồng hình [78] Tìm kiếm từ khóa dựa trên lược đồ mã hóa thuộc tính [117] Thuật toán xử lý truy vấn kNN trên CSDLQH mã hóa [61] Bảo vệ quyền riêng tư trong mã hoá có thể tìm kiếm [63]

Trong giai đoạn từ năm 2000 đến 2014 được đánh dấu là khoảng thời gian sôi nổi của cộng đồng nghiên cứu dành cho lĩnh vực SE, các công trình đã giúp phân loại, đặt vấn đề và hướng giải quyết cho các bài toán cần nghiên cứu. Kết quả nghiên cứu trong giai đoạn này là tập trung chủ yếu vào hướng tìm kiếm từ khóa trên tài liệu mã hóa. Do sự bùng nổ của điện toán đám mây, giai đoạn 2015 đến nay tiếp tục cho thấy sự quan tâm hơn nữa của các nhà nghiên cứu đối với SE, nhiều hướng tiếp cận mới được thực hiện bao gồm: nâng cấp/mở rộng các kết quả của giai đoạn trước, giải

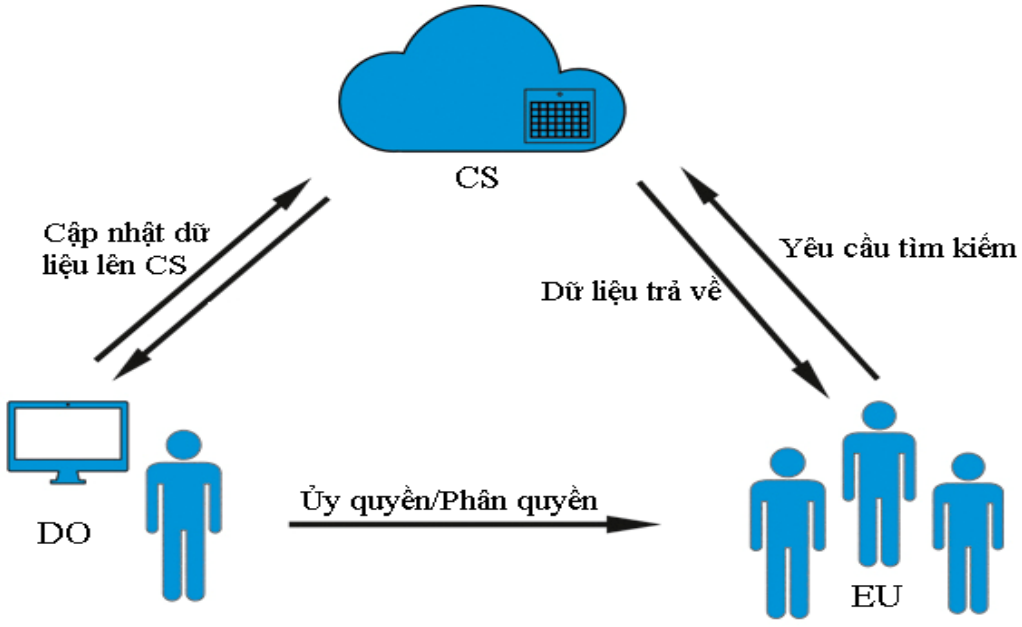
quyết bài toán truy vấn trên dữ liệu số và ký tự trên các tài liệu mã hoá hoặc CSDLQH mã hoá, nghiên cứu về hiệu năng và bảo mật của SE,... **Bảng 1.1** cũng cho thấy lĩnh vực SE nói chung và nghiên cứu truy vấn dữ liệu trên CSDLQH mã hóa nói riêng đã và đang được cộng đồng khoa học quan tâm và dành nhiều nguồn lực nghiên cứu.

Trong các phần tiếp theo, luận án sẽ dùng cụm từ *"lược đồ SE"* hoặc *"lược đồ mã hoá có thể tìm kiếm"* để mô tả các quy trình tổ chức xử lý dữ liệu và quy trình truy vấn của các giải pháp tìm kiếm trong lĩnh vực SE.

1.2. Triển khai lược đồ SE trên mô hình DAS

1.2.1. Mô hình DAS

Bên cạnh việc thiết kế, tổ chức các thành phần, nghiệp vụ trong lược đồ SE thì việc triển khai lược đồ trên các mô hình dịch vụ cũng rất quan trọng, cho thấy tính khả thi trong ứng dụng thực tiễn. Trong đó DAS [92], [47] là một mô hình cung cấp CSDL như một dịch vụ, mang tới khả năng quản lý dữ liệu từ xa đối với CSDL nói chung và cho phép triển khai hiệu quả các lược đồ SE trên CSDLQH mã hóa nói riêng. Về cơ bản DAS gồm 3 đối tượng là DO, EU và CS. **Hình 1.2** mô tả mối quan hệ vận hành giữa ba đối tượng này khi triển khai tìm kiếm dữ liệu rõ trên mô hình DAS.



Hình 1.2. Mô hình DAS [133]

Khi sử dụng DAS để triển khai vận hành lược đồ SE trên CSDLQH mã hóa thì vai trò, nhiệm vụ của các đối tượng DO, EU và CS được bổ sung thêm nhằm giải quyết các vấn đề liên quan tới bảo mật, truy vấn, truyền tải và lưu trữ dữ liệu. Cụ thể:

- DO: là người sở hữu dữ liệu, thực hiện thuê khoán hoặc giao quyền quản trị CSDLQH mã hóa cho các nhà cung cấp dịch vụ (DSP). DO cũng quyết định việc phân quyền truy cập dữ liệu cho EU và lưu trữ các MetaData bí mật phục vụ các thao tác mã hoá, giải mã dữ liệu hoặc thủ tục tính toán bí mật, do đó DO hoạt động tác nghiệp tại những khu vực tin cậy, an toàn. Mọi trao đổi thông tin giữa EU và CSDLQH mã hóa đều phải đi qua các khu vực được DO kiểm soát.

- EU: là người dùng cuối được DO phân quyền cho phép truy cập khai thác CSDLQH mã hóa. Quy trình cơ bản khi EU đề nghị truy vấn là: từ câu lệnh truy vấn rõ của EU, DO sẽ tiếp nhận và cung cấp chức năng kiểm soát, biến đổi, mã hoá câu truy vấn về dạng thức có thể thực thi được trên CSDLQH mã hóa tại CS, DO cũng tiếp nhận kết quả truy vấn trả về từ CS và tiến hành giải mã để trả về EU. Trong mô hình đơn người dùng thì DO cũng có thể đóng vai trò của EU.

- CS: là máy chủ đám mây được cung cấp bởi DSP, là nơi triển khai CSDL và thực hiện hầu hết các chức năng quản trị, khai thác, truy vấn, tính toán trên dữ liệu. Trong bài toán truy vấn dữ liệu trên CSDLQH mã hóa, CS hoàn toàn làm việc trên các dữ liệu mã hoá được chia sẻ bởi DO. Khi đó DSP hoặc bất cứ "kẻ gian" nào cũng không thể phân biệt được các bản ghi dữ liệu đã mã hoá với nhau và cũng không thể biết được yêu cầu truy vấn rõ của EU là gì. Mọi khoá phục vụ mã, giải mã và các thông tin bí mật khác đều không được tiết lộ tại CS dưới bất kỳ hình thức nào.

Trong luận án có sử dụng tới cụm từ ***"kẻ gian"*** nhằm chỉ đến các đối tượng (con người, phần mềm, dịch vụ, ...) có các hành vi khai thác thông tin hoặc gây tổn hại đến CSDL. Kẻ gian bao gồm các đối tượng bên trong (các "nhà cung cấp dịch vụ - DSP" trung thực nhưng tò mò) và đối tượng bên ngoài (các kẻ tấn công chủ đích).

1.2.2. Áp dụng mô hình DAS vào triển khai lược đồ SE

Hình 1.3 mô tả quy trình vận hành lược đồ SE trên mô hình DAS. Để đảm bảo nhiệm vụ của các đối tượng trong mô hình DAS, lược đồ SE cần được thiết kế từ 5 thành phần/thuật toán là: $KeyGen,\ BuildIndex,\ Trapdoor,\ Search,\ Decrypt$.

- $\boldsymbol{KeyGen}$**:** là thuật toán được sử dụng để khởi tạo các thông số ban đầu cho lược đồ SE, bao gồm các khoá để thực hiện mã/giải mã dữ liệu và các tham số hoặc dữ liệu bí mật phục vụ xây dựng chỉ mục mù hoặc các cấu trúc dữ liệu liên quan. $KeyGen$ thường được quản lý và vận hành tại vùng an toàn do DO kiểm soát.

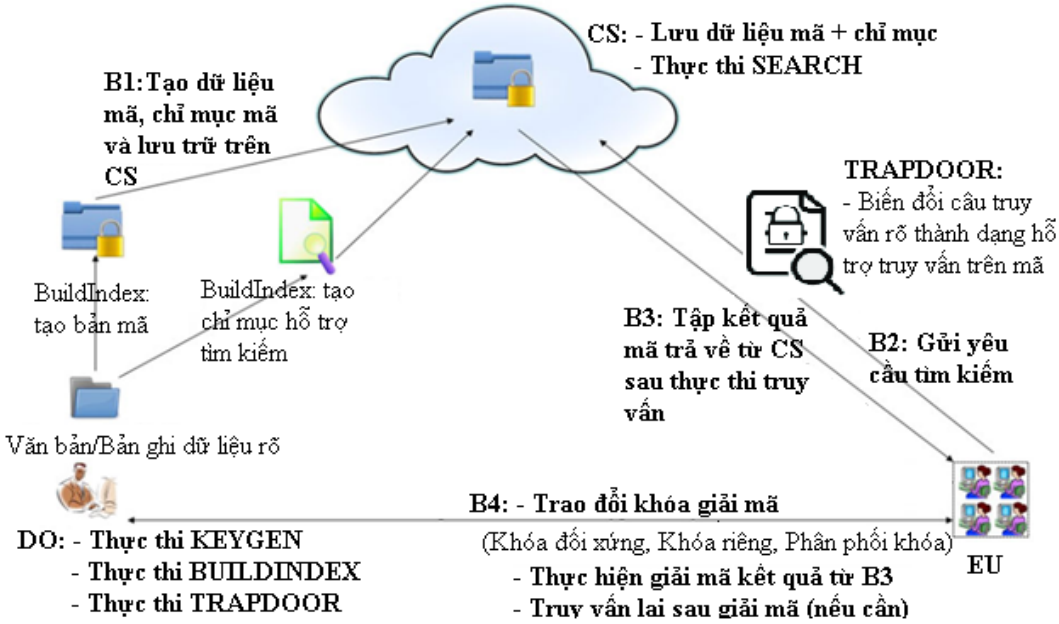
- $\boldsymbol{BuildIndex}$**:** là thuật toán sinh ra các bản mã và chỉ mục mù tương ứng với

các dữ liệu rõ đầu vào. $BuildIndex$ được sử dụng trong biến đổi lần đầu tập dữ liệu rõ thành chỉ mục mù để lưu trữ tại CSDLQH mã hóa trên CS hoặc trong các lần cập nhật lại chỉ mục khi có biến động dữ liệu (Insert/Update/Delete). Tương tự $KeyGen$, $BuildIndex$ cũng được quản lý và vận hành tại vùng an toàn do DO kiểm soát.

- ***Trapdoor*:** hay còn gọi là cửa sập, là một thuật toán tiếp nhận các yêu cầu truy vấn rõ từ EU và sử dụng các tham số bí mật được sinh ra bởi $KeyGen$ để biến đổi câu truy vấn rõ thành dạng thức cho phép truy vấn được trên chỉ mục mù hoặc dữ liệu mã. $Trapdoor$ có thể bao hàm cả chức năng xác thực, kiểm tra toàn vẹn của dữ liệu. $Trapdoor$ cũng hoạt động tại vùng an toàn do DO hoặc EU kiểm soát.

- ***Search*:** thuật toán $Search$ thực thi câu lệnh truy vấn được trả bởi $Trapdoor$ trên CSDLQH mã hóa tại CS. Chú ý với mỗi lược đồ SE khác nhau thì cách thức vận hành của thuật toán $Search$ cũng khác nhau, hỗ trợ các phương pháp tìm kiếm khác nhau theo từng kiểu dữ liệu (như so sánh bằng, lớn hơn, nhỏ hơn, tìm kiếm theo từ khoá đơn, tìm kiếm liên hợp nhiều từ khóa, tìm theo chuỗi con, tìm kiếm khoảng, tìm kiếm mờ, tìm kiếm đồng nghĩa, tìm theo ngữ nghĩa…) và hỗ trợ các kỹ thuật duyệt dữ liệu đa dạng (quét toàn bộ bảng, tìm kiếm theo cây chỉ mục,…). Kết quả trả về của thuật toán $Search$ có thể là tập kết quả chính xác nhưng cũng có thể dư thừa.

- ***Decrypt***: DO hoặc EU tiếp nhận kết quả trả về bởi $Search$, yêu cầu khóa giải mã, thực hiện giải mã và cuối cùng là truy vấn lại (nếu có dư thừa).



Hình 1.3. Triển khai lược đồ SE trên mô hình DAS [51]

1.3. Triển khai lược đồ SE trên mô hình DAS-PROXY

Mô hình DAS [92], [47] có ưu điểm lớn khi phân chia rõ ràng vai trò, nhiệm vụ và mối quan hệ tương tác của các đối tượng EU, DO, CS trong triển khai 5 thành phần $KeyGen,\ BuildIndex,\ Trapdoor,\ Search,\ Decrypt$ cùng các quy trình của lược đồ SE. Quá trình triển khai SE trên DAS thường đề cập tới thuật ngữ "*khu vực an toàn do DO kiểm soá*t", tuy nhiên khu vực này không được thể hiện rõ ràng trong mô tả tại **Hình 1.3**. Theo đó các thành phần $KeyGen, BuildIndex$ được DO sắp xếp thực hiện duy nhất trong một phiên tiền xử lý dữ liệu tại một khu vực tin cậy (như cơ chế sandbox), nhằm mục đích hạn chế tối đa rò rỉ thông tin về: quy trình sinh mã/chỉ mục mù hay các giá trị metadata/khoá. Còn với $Trapdoor, Decrypt$ là các thành phần bắt buộc vận hành thường xuyên khi có yêu cầu truy vấn nên không thể dùng cơ chế thực hiện một lần. $Trapdoor$ và $Decrypt$ cũng có xu hướng được tích hợp vào phía ứng dụng cài riêng cho EU thay vì một khu vực độc lập an toàn do DO kiểm soát. Rõ ràng mô hình DAS cũng còn một số tồn tại như:

- Gặp khó khăn khi DO có nhu cầu đổi khoá và dữ liệu bí mật sau một thời gian hoạt động hoặc khi có sự thay đổi dữ liệu (thêm/sửa/xoá) đối với CSDLQH mã hóa.

- Các tham số bí mật được sinh một lần và sử dụng chung cho các thuật toán mã hoá, thuật toán tạo chỉ mục mù và thuật toán biến đổi lệnh truy vấn, khiến đầu ra của chúng ở trạng thái xác định (cùng một dữ liệu đầu vào thì thuật toán cho ra kết quả giống nhau), điều này làm giảm bảo mật của lược đồ SE.

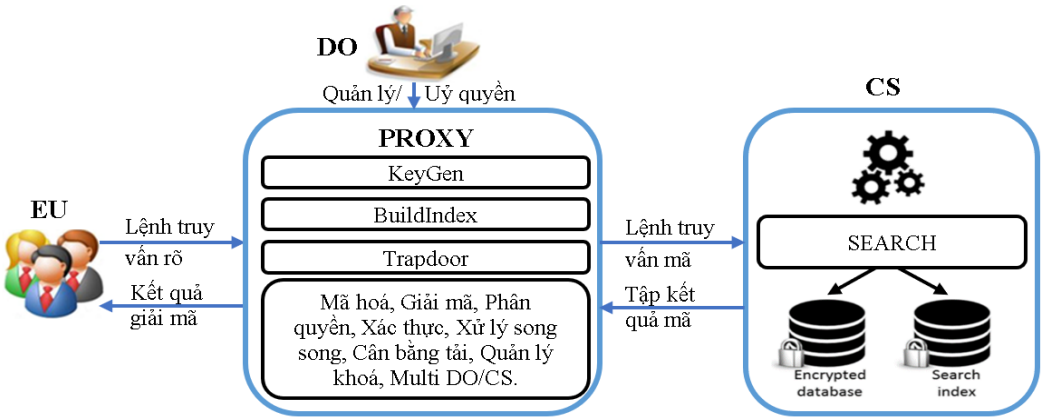
- $Trapdoor$ và $Decrypt$ được tích hợp tại EU gây áp lực cho các thiết bị cuối của người dùng, đồng thời làm giảm mức độ an toàn của lược đồ cũng như phải giới hạn tập người dùng được phép sử dụng hệ thống. Bên cạnh đó quá trình chia sẻ dữ liệu bí mật giữa $KeyGen, BuildIndex$ (vận hành bởi DO) với $Trapdoor, Decrypt$ (vận hành bởi EU) cũng là một vấn đề nhạy cảm và cần được đảm bảo.

Để khắc phục các tồn tại của mô hình DAS, ý tưởng sử dụng một máy chủ trung gian tin cậy Proxy để quản lý tất cả dữ liệu bí mật và thực thi các nghiệp vụ mà DO uỷ quyền đã được đề xuất trong nhiều nghiên cứu [135], [111], [6], [91], [9], [125], [137], [92]. Ở đây, ta cần làm rõ vai trò của Proxy và CS trước câu hỏi *"Proxy có thể thay thế được CS hay không?"*. Rõ ràng CS đại diện cho các máy chủ với khả năng triển khai dữ liệu quy mô lớn, tiện dụng, chi phí rẻ và đáp ứng tính chất luôn sẵn sàng trên môi trường điện toán đám mây (không gây sự cố gián đoạn dịch vụ của người dùng). Ngoài năng lực lưu trữ, khả năng chịu tải tốt và khả năng thích ứng mở rộng

theo nhu cầu thì CS còn tích hợp sẵn các hệ quản trị CSDLQH, dịch vụ CSDL cho phép truy vấn và tính toán mạnh mẽ. Trong khi đó Proxy chỉ là một máy chủ có cấu hình phù hợp phục vụ các nhiệm vụ không đòi hỏi nghiêm ngặt về chi phí thời gian như tạo bản mã/chỉ mục lần đầu cho CSDL hoặc các nhiệm vụ phát sinh thường xuyên nhưng chi phí thực hiện thấp như sinh tham số bí mật ($KeyGen$), biến đổi lệnh truy vấn ($Trapdoor$). Khi sử dụng lược đồ SE tìm kiếm trên một CSDLQH mã hóa với số lượng bản ghi lớn, CS sẽ thể hiện được vai trò xử lý, truy vấn và thích ứng mở rộng mạnh mẽ của nó, trong khi đó Proxy gần như không đòi hỏi thay đổi về năng lực cấu hình. Vì vậy, Proxy không thể thay thế cho CS và máy chủ Proxy là một thành phần bổ sung cần thiết giúp mô hình DAS khắc phục tốt những nhược điểm đã nêu.

Trong luận án này, máy chủ Proxy cũng được vận dụng đưa vào trong mô hình DAS (gọi là mô hình **DAS-PROXY**) để triển khai thống nhất cho các lược đồ SE đề xuất tại chương 2 và 3. Luận án tập trung vào xây dựng các lược đồ hỗ trợ truy vấn trên các kiểu dữ liệu số, ký tự mã hoá và tích hợp vận hành lược đồ lên mô hình DAS-PROXY sao cho hiệu quả. Vì vậy các vấn đề về đánh giá năng lực, lựa chọn cấu hình hay sử dụng Proxy để giải quyết một số yêu cầu mở rộng như: đa người sở hữu dữ liệu/đa máy chủ; xử lý song song; cân bằng tải; xác thực - toàn vẹn dữ liệu; quản lý khoá; phân quyền sử dụng;… đều không nằm trong phạm vi của luận án, mặc dù máy chủ Proxy có tiềm năng để thực hiện tốt các công việc này.

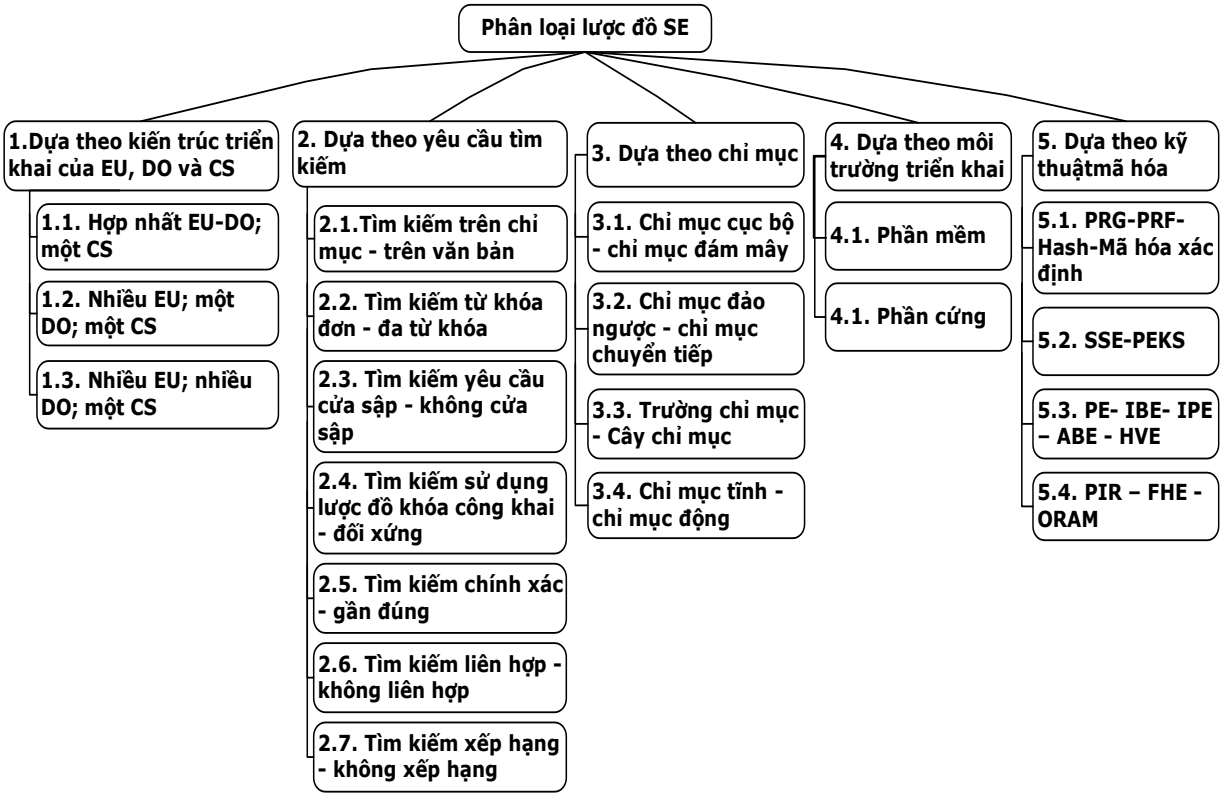
Theo **Hình 1.4**, PROXY thể hiện vai trò lớp trung gian tin cậy giữa EU và CS. Mọi công việc cần đến tính riêng tư như: sinh khóa, quản lý khóa, tạo chỉ mục, biến đổi truy vấn, mã hóa, giải mã hay phân quyền, xác thực người dùng đều được tích hợp tại PROXY. Toàn bộ dữ liệu trên đường truyền và quy trình truy vấn trên CS đều là mù với kẻ gian. Thậm chí PROXY có thể đảm nhiệm cả chức năng cân tải, xử lý song song và kết nối đa CS khi cần thiết.



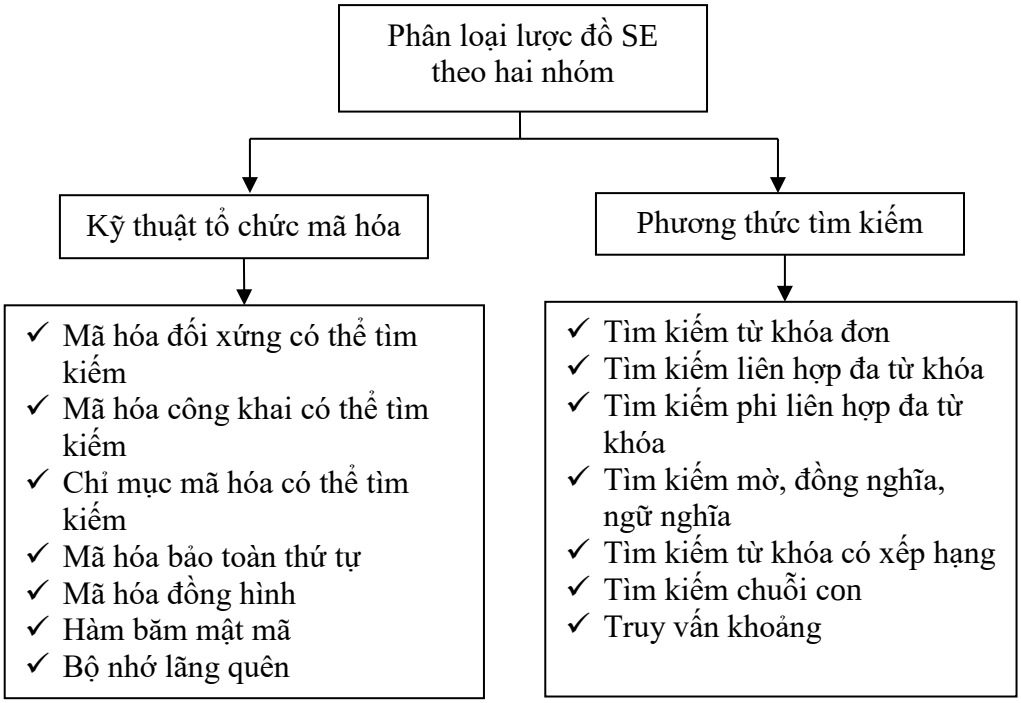
Hình 1.4. Triển khai lược đồ SE trên mô hình DAS-PROXY

1.4. Phân loại lược đồ SE

Có nhiều cách tiếp cận khác nhau trong việc phân loại các lược đồ SE như: dựa trên số lượng EU và DO; dựa trên phương thức tìm kiếm; dựa trên cấu trúc chỉ mục mù; dựa trên kết quả truy xuất; dựa trên phương pháp mã hoá/biến đổi,... **Hình 1.5** phân loại lược đồ SE dựa trên các nghiên cứu tổng quan về mã hoá có thể tìm kiếm gần đây [51], [68], cho thấy sự đa dạng trong các hướng nghiên cứu. Trong khi đó **Hình 1.6** [107] tổng hợp và trình bày phân loại các lược đồ SE đơn giản hơn theo góc nhìn về "kỹ thuật tổ chức mã hóa" và "phương thức tìm kiếm". Các "kỹ thuật tổ chức mã hóa" bao hàm nội dung rộng hơn, chúng đặt ra các nguyên tắc thiết kế lược đồ SE và cách thức tạo ra các dạng mã/chỉ mục hỗ trợ tìm kiếm, là nền tảng để triển khai các chức năng, phương thức tìm kiếm cụ thể trong từng ngữ cảnh. Giả sử cùng là kỹ thuật "Mã hoá đối xứng có thể tìm kiếm" nhưng có thể hỗ trợ phương thức tìm kiếm từ khoá đơn (Single Keyword Search), tìm kiếm đa từ khoá (Multi Keyword Search) hoặc tìm kiếm liên hợp từ khóa (Conjunctive Keyword Search),… trên dữ liệu mã. Ngược lại, bất cứ chức năng, hay phương thức tìm kiếm trên dữ liệu mã nào được yêu cầu bởi EU thì cũng phải được giải quyết bằng một trong các "kỹ thuật tổ chức mã hóa" đã được phân loại trong **Hình 1.6**. Trong thực tế, một lược đồ SE có thể kết hợp đồng thời nhiều "kỹ thuật tổ chức mã hóa" và "phương thức tìm kiếm".



Hình 1.5. Tổng hợp phân loại lược đồ SE [51]



Hình 1.6. Phân loại lược đồ SE theo kỹ thuật tổ chức mã hoá và theo phương thức tìm kiếm [107]

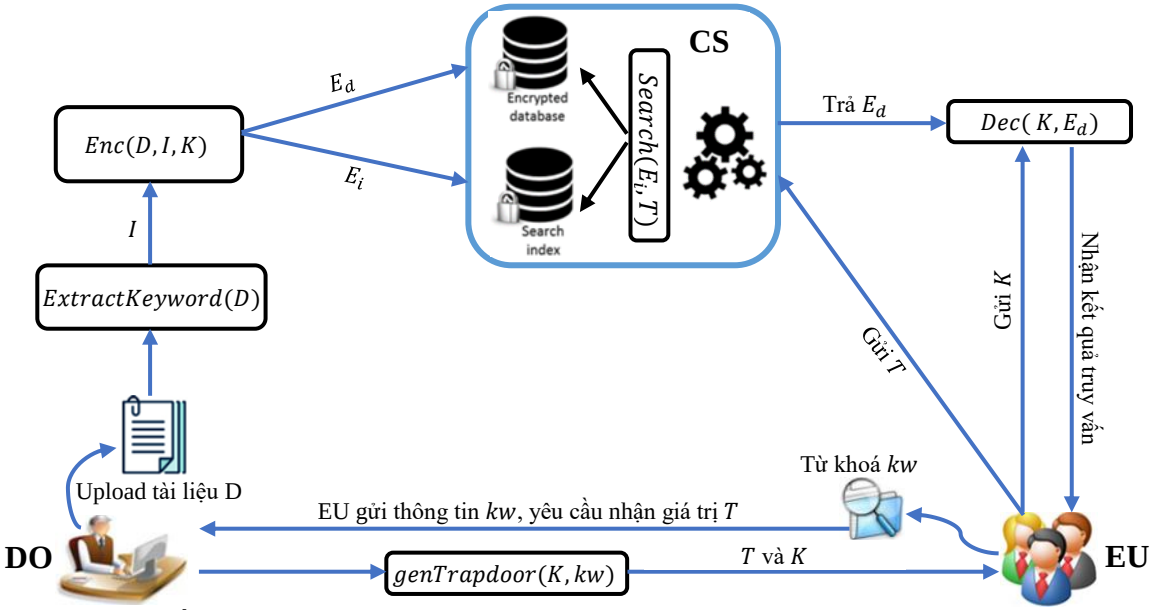
1.5. Một số kỹ thuật tổ chức mã hóa trong thiết kế lược đồ SE

Về kỹ thuật tổ chức mã hoá, có hai cơ chế thực hiện phổ biến là mã hoá đối xứng và mã hoá công khai. Với lĩnh vực SE, các lược đồ SSE [100], [62], [33], [95] hay PKSE [19], [44], [134], [119] chính là các đại diện điển hình sử dụng các cơ chế đã nêu nhằm tạo ra các bản mã có thể tìm kiếm. Một biến thể phổ biến của hai loại lược đồ nêu trên là IBSE [43], [124], [108], [8], [115], [75], [89] cho phép tạo ra các chỉ mục mù một chiều hỗ trợ tìm kiếm với hiệu quả cao. Trong khi đó các lược đồ mã hoá đồng hình, hàm băm mật mã hay bộ nhớ lãng quên mặc dù cũng sử dụng những cơ chế tổ chức mã hoá nêu trên, nhưng sự phổ biến của chúng lại thấp hơn do một số nhược điểm về tính khả thi triển khai thực tiễn khi chi phí xây dựng bản mã và truy vấn quá cao. Do đó, mục này của luận án sẽ tập trung giới hạn trình bày về một số lược đồ SE điển hình [107], [51], [116] như: mã hoá đối xứng có thể tìm kiếm (SSE); mã hoá công khai có thể tìm kiếm (PKSE); mã hoá có thể tìm kiếm dựa trên chỉ mục (IBSE).

1.5.1. Lược đồ mã hoá đối xứng có thể tìm kiếm (SSE)

Trong lược đồ mã hóa đối xứng có thể tìm kiếm (hay còn gọi là SSE) [68], các tài liệu/bản ghi dữ liệu được mã hóa bằng cách sử dụng các khóa đối xứng riêng tư giữa DO và EU (xem **Hình 1.7**). Sử dụng khóa đối xứng kết hợp cùng các thuật toán

đặc biệt, DO mã hóa tài liệu cùng với chỉ mục và gửi chúng đến CS, sau đó EU có thể truy cập dữ liệu được mã hóa bằng cách tạo cửa sập (Trapdoor) cho máy chủ đám mây. CS thực hiện tìm kiếm trên dữ liệu được mã hóa dựa trên thông tin từ cửa sập và gửi kết quả tới EU, sau đó, kết quả thu được có thể được giải mã bằng cách sử dụng khóa đối xứng.



Hình 1.7. Mô hình lược đồ SSE [107]

Tương tự lược đồ SE (**Mục 1.2.2**), lược đồ SSE [100], [62], [33], [95] gồm các thành phần/thuật toán: $KeyGen, ExtractKeyword, Enc, genTrapdoor, Search$ và *Dec*, được bố trí vận hành trong ba giai đoạn sau:

Giai đoạn 1: Mã hóa

- $KeyGen(s) \rightarrow K$: Là thuật toán đầu tiên trong SSE, và nó được khởi tạo bởi DO. Nó lấy tham số bảo mật s làm đầu vào và xuất ra khóa riêng K.
- $ExtractKeyword\ (D) \rightarrow I$: Thuật toán này được điều hành bởi DO. Nó phân tích và liệt kê ra tập các từ khóa W (trong đó $W = W_1, W_2, \dots, W_n$) từ tập tài liệu đầu vào D, đầu ra là chỉ mục I chứa thông tin của tập từ khóa W.
- $Enc(D, I, K) \rightarrow E_d, E_i$: Thuật toán này được bắt đầu bởi DO. Nó nhận tập tài liệu D trong đó $D = d_1, d_2, \dots, d_k$ (với d_i là tài liệu thứ i), chỉ mục I, khóa K làm đầu vào và tạo ra chỉ mục mã hóa E_i và tập các tài liệu mã hóa $E_d = \{E_{d1}, E_{d2}, \dots, E_{dk}\}$ làm đầu ra.

Giai đoạn 2: Tìm kiếm

- $genTrapdoor(K, kw) \rightarrow T$: Thuật toán này được điều hành bởi EU hoặc DO tùy thuộc vào mô hình triển khai. Nó nhận khóa bí mật K và tham số kw (là

một hoặc nhiều từ khoá trong chuỗi điều kiện của lệnh truy vấn) làm đầu vào. Sau đó, nó xuất ra cửa sập T bằng cách mã hóa kw bởi khóa K.

- $Search(E_i, T) \rightarrow E_d$: Thuật toán này do CS điều hành. Nó nhận cửa sập T và chỉ mục được mã hóa E_i làm đầu vào, sau đó thực hiện thao tác tìm kiếm với cửa sập trên chỉ mục được mã hóa E_i và trả về tài liệu E_d thoả mãn truy vấn.

Giai đoạn 3: Giải mã

- $Dec(K, E_d) \rightarrow D$: Thuật toán này cũng được thực thi bởi DO hoặc EU. Nó giải mã tập mã E_d bằng khóa bí mật K và trả về tập kết quả D.

1.5.2. Lược đồ mã hoá công khai có thể tìm kiếm (PKSE)

Các lược đồ SSE yêu cầu một kênh bảo mật bổ sung để chia sẻ khóa đối xứng riêng tư giữa DO và EU, nhưng rõ ràng không thể đảm bảo rằng kênh bảo mật đó không thể bị xâm phạm [116], [96]. Các lược đồ mã hóa công khai có thể tìm kiếm [19] (PKSE) sử dụng cặp khóa công khai và khóa riêng để mã hóa và giải mã. Như **Hình 1.8**, DO mã hóa tài liệu cùng với chỉ mục tài liệu bằng khóa công khai của EU và lưu trữ lên CS. Khi đó EU có thể sử dụng khoá bí mật để tạo cửa sập (Trapdoor) và yêu cầu truy vấn trên CS, sau đó giải mã kết quả trả về bởi CS.

Tương tự lược đồ SE (**Mục 1.2.2**), lược đồ PKSE [19], [44], [134], [119] gồm các thành phần/thuật toán: $KeyGen, ExtractKeyword, Enc, Trapdoor, Test$ và *Dec*, được bố trí vận hành trong ba giai đoạn sau:

Giai đoạn 1: Mã hóa

- $KeyGen(s) \rightarrow (PK, SK)$: Là thuật toán đầu tiên trong quy trình PKSE và nó được DO khởi xướng. Nó lấy tham số bảo mật s làm đầu vào, xuất ra cặp khóa công khai PK và khóa riêng SK phân phối cho EU.
- $ExtractKeyword(D) \rightarrow I$: Thuật toán này được điều hành bởi DO. Đầu vào là tài liệu D và đầu ra là tập từ khóa I (với $I = \{I_1, I_2, \dots, I_n\}$) của D.
- $Enc(PK, D) \rightarrow E_d$: Thuật toán này được bắt đầu bởi DO. Nó nhận tài liệu D, khóa PK làm đầu vào và tạo đầu ra là tài liệu mã hóa E_d.
- $PEKS(PK, I) \rightarrow E_i$: Thuật toán này được bắt đầu bởi DO. Nó nhận tập từ khoá I, khóa công khai PK làm đầu vào và tạo ra tập chỉ mục mã hóa $E_i = \{E_{i_1}, E_{i_2}, \dots, E_{i_n}\}$ làm đầu ra.

Giai đoạn 2: Tìm kiếm

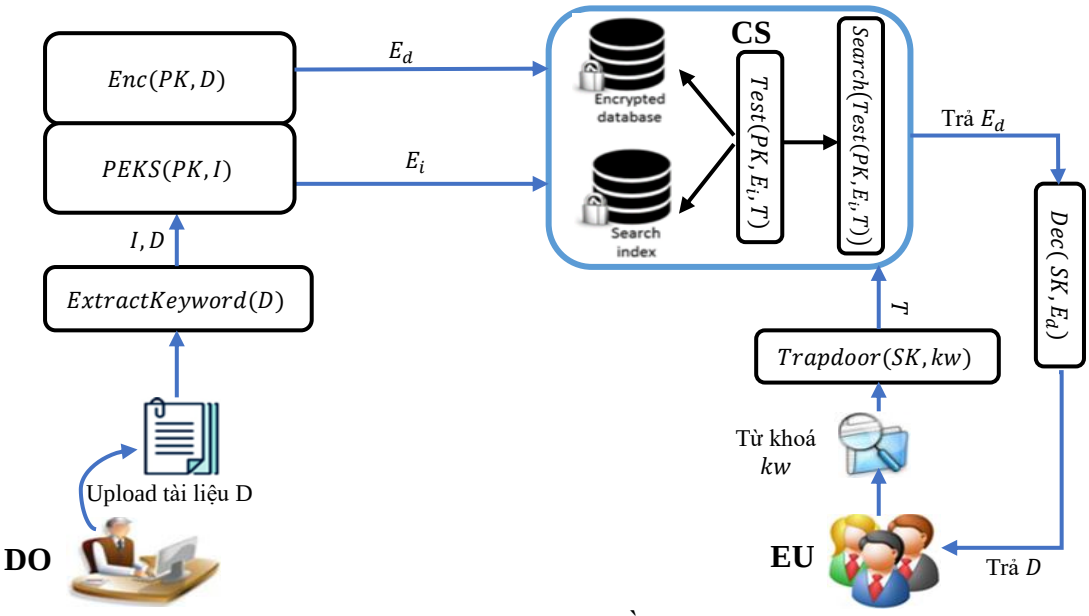
- $Trapdoor(SK, kw) \rightarrow T$: Thuật toán này được điều hành bởi EU, sử dụng

khóa riêng SK và từ khóa truy vấn kw làm đầu vào, sau đó xuất ra cửa sập T.

- $Test(PK, E_i, T) \rightarrow \{0,1\}$: Thuật toán này do CS điều hành, sử dụng khoá công khai PK, giá trị cửa sập T và chỉ mục mã hóa E_i làm tham số đầu vào. Nếu $kw \in I$ thì hàm $Test$ trả về giá trị 1 và ngược lại.
- $Search\big(Test(PK, E_i, T)\big) \rightarrow Result$: Thuật toán này chạy trên CS, nếu $Test = 1$ thì $Result = E_d$ tương ứng với E_i, ngược lại $Result = Null$.

Giai đoạn 3: Giải mã

- $Dec(SK, E_d) \rightarrow D$: Thuật toán này được thực thi bởi EU. Lấy kết quả tìm kiếm E_d và khóa riêng SK làm đầu vào và cung cấp phiên bản tài liệu gốc D tương ứng bằng cách giải mã E_d.



Hình 1.8. Mô hình lược đồ PKSE [134]

1.5.3. Lược đồ mã hoá có thể tìm kiếm dựa trên chỉ mục

Các lược đồ SSE [100] và PKSE [19] cho thấy một số hạn chế về tính khả thi khi ứng dụng thực tế [107], [21], [27], như: chưa hỗ trợ tốt với CSDLQH mã hóa; độ dài bản mã hỗ trợ tìm kiếm tuyến tính với chiều dài bản rõ, chỉ hỗ trợ tìm kiếm theo từ khóa với chiều dài cố định, không phù hợp với các phương pháp tìm kiếm đa dạng trong thực tế; chi phí tìm kiếm lớn, đặc biệt đối với lược đồ PKSE,…

Một trong các hướng tiếp cận hiệu quả với các lược đồ SSE và PKSE là giải pháp thay thế bản mã hỗ trợ tìm kiếm bằng chỉ mục mã hoá hỗ trợ tìm kiếm (còn gọi là chỉ mục mù) [43], [124], [108], [8], [115], [75], [89]. Chỉ mục mù có ưu điểm không thể giải mã, có tính bảo mật cao và cho phép kết hợp với các cấu trúc dữ liệu nhằm tăng tốc độ cũng như hỗ trợ đa dạng các phương thức truy vấn từ người dùng.

Theo đó với mỗi bản rõ sẽ được sinh ra thành một cặp là ***<Bản mã; Chỉ mục mù>***, với ***"Bản mã"*** được tạo ra bởi các thuật toán mã hóa tiêu chuẩn như AES, DES, Blowfish … [37], [13], [98] và ***"Chỉ mục mù"*** được tạo ra bởi các thuật toán cho phép thực hiện truy vấn trực tiếp hoặc qua các hàm/toán tử đặc biệt. Các lược đồ SE xây dựng trên chỉ mục mù được coi là một trường hợp của các lược đồ SSE hoặc PKSE.

Có nhiều kỹ thuật trong xây dựng chỉ mục mã hóa như: Chỉ mục dựa trên nhóm (Bucket Index) [92], [47], [58], [53], [113], [4]; chỉ mục dựa trên bộ lọc Bloom (Bloom Index) [100], [43], [101], [14], [123], [132], [76]; chỉ mục dựa trên mã hóa bảo toàn thứ tự (OPE) [71], [5], [66], [18], [17], [56]; chỉ mục dựa trên mã hóa đồng hình (HE) [78], [41], [22], [7]; chỉ mục dựa trên hàm băm mật mã (Hash Index) [8], [120], [122],…

Nhận xét 1.1: Một số nhận xét từ các nội dung trình bày ở trên như sau (xem thêm **Bảng 1.2**):

- Lược đồ SSE có ưu điểm đơn giản, dễ triển khai trên mô hình DAS và đặc biệt có hiệu năng thực thi cao do sử dụng các giải thuật mã hóa đối xứng. Tuy nhiên, lại rủi ro trong việc trao đổi khóa bí mật giữa DO và EU trên kênh truyền.

- Lược đồ PKSE giải quyết được các tồn tại của lược đồ SSE nhưng các thuật toán, quy trình triển khai phức tạp, tốn kém chi phí tính toán, gây ảnh hưởng đến hiệu năng truy vấn và tính khả thi triển khai thực tiễn.

- Lược đồ mã hóa có thể tìm kiếm dựa trên chỉ mục là các lược đồ SSE hoặc PKSE sử dụng chỉ mục mù với kích thước lưu trữ nhỏ hơn và thực hiện tìm kiếm nhanh hơn so với các bản mã hỗ trợ tìm kiếm truyền thống. Đặc biệt chỉ mục mù phù hợp cho triển khai truy vấn trong CSDLQH mã hóa, cho phép kết hợp với các cấu trúc dữ liệu tăng tốc truy vấn trên các hệ quản trị CSDL hiện nay.

- Chọn lược đồ SSE xây dựng trên chỉ mục mù và triển khai lược đồ trên mô hình DAS-PROXY là một giải pháp tốt để giải quyết bài toán mã hoá có thể tìm kiếm nói chung và đề tài luận án *"Phát triển một số phương pháp truy vấn hiệu quả trên CSDL quan hệ mã hóa"* nói riêng.

Bảng 1.2. So sánh giữa SSE, PKSE và Tìm kiếm dựa vào chỉ mục mù triển khai trên mô hình DAS-PROXY [51], [119], [21]

Lược đồ	**Ưu điểm**	**Nhược điểm**
SSE [100], [62], [33], [95]	- Tìm kiếm trực tiếp trên dữ liệu mã; - Hiệu suất cao do sử dụng mã hóa đối xứng; - Dễ triển khai; - Tiết kiệm lưu trữ tại CS.	- Bảo mật phụ thuộc vào quản lý khóa bí mật dùng chung, rủi ro chia sẻ khóa và khó mở rộng khi nhiều người dùng; - Độ dài bản mã tuyến tính với bản rõ, tiềm ẩn rủi ro lộ thông tin; - Khả năng tìm kiếm hạn chế, thường hỗ trợ tìm kiếm chính xác. Đối với giải pháp [100], từ khóa tìm kiếm chiều dài cố định không phù hợp với các ngữ cảnh thực tế; - Chủ yếu hỗ trợ tìm kiếm trên dữ liệu dạng ký tự.
PKSE [19], [44], [134], [119]	- Tìm kiếm trực tiếp trên dữ liệu mã; - Bảo mật cao, chia sẻ khóa an toàn; - Tiết kiệm lưu trữ tại CS.	- Hiệu suất thấp hơn so với SSE; - Triển khai phức tạp; - Chủ yếu hỗ trợ tìm kiếm theo từ khóa; - Chủ yếu hỗ trợ tìm kiếm trên dữ liệu dạng ký tự.
Tìm kiếm dựa vào chỉ mục mù [123], [47], [100], [8] triển khai trên mô hình DAS-PROXY [135], [111], [6], [91], [9], [125], [137], [92]	- Tách biệt chỉ mục mù và bản mã, thuận lợi cho thiết kế chỉ mục cũng như tăng cường tính bảo mật với tính chất một chiều không cần giải mã; - Dễ tổ chức lưu trữ chỉ mục trên các cấu trúc hỗ trợ tìm kiếm tốc độ cao như BTree hoặc B^+Tree; - Có thể hỗ trợ nhiều loại tìm kiếm; - PROXY hỗ trợ quản lý các dữ liệu bí mật tập trung và thuận tiện; - PROXY giúp giảm tải cho thiết bị người dùng, có thể mở rộng để phục vụ nhiều mục đích khác như mô tả trong mục 1.3.	- Tốn dung lượng lưu trữ cho chỉ mục; - Bổ sung thêm máy chủ PROXY; - Đảm bảo quản lý PROXY.

1.6. Một số phương thức tìm kiếm trong các lược đồ SE

1.6.1. Tìm kiếm từ khóa đơn chính xác

Lược đồ SE hỗ trợ tìm kiếm từ khóa đơn [100], [43], [109], [33], [95], [106] cho phép EU chỉ đưa một từ khóa vào để truy vấn và trả về các tài liệu có chứa từ khóa này. Nếu EU có nhu cầu tìm các tài liệu chứa đồng thời nhiều từ khoá thì quá

trình tìm kiếm sẽ được chuyển thành thực thi lần lượt các truy vấn với mỗi từ khóa đơn. Tiếp theo đó thực hiện phép giao giữa các tập kết quả trả về từ các truy vấn để tìm được kết quả thoả mãn. Các lược đồ thường sử dụng danh sách các từ khóa hay được yêu cầu truy vấn để xây dựng thành các bảng chỉ mục hỗ trợ tìm kiếm (sử dụng cấu trúc chỉ mục chuyển tiếp [43] hoặc đảo ngược [109] như **Hình 1.9**).

Từ khóa	ID các tài liệu/ bản ghi
Hà Nội	1, 2, 6, …
Hải Phòng	1, 2, 4, …
Nam Định	1, 2, 4, 5, 6, …
Thái Bình	2, …
Hà Nam	1, 3, 4, 5, 6, …
Nghệ An	1, 3, 4, 5, …
…	…

(A)

ID tài liệu/ bản ghi	Các từ khóa
1	Hà Nội, Hải Phòng, Nam Định, Hà Nam, Nghệ An, …
2	Hà Nội, Hải Phòng, Nam Định, Thái Bình, …
3	Hà Nam, Nghệ An, …
4	Hải Phòng, Nam Định, Hà Nam, Nghệ An, …
5	Nam Định, Hà Nam, Nghệ An, …
6	Hà Nội, Nam Định, Hà Nam, …
…	…

(B)

Hình 1.9. Cấu trúc chỉ mục đảo ngược (A) và chỉ mục chuyển tiếp (B)

1.6.2. Tìm kiếm đa từ khóa chính xác

- Tìm kiếm liên hợp đa từ khóa

Lược đồ SE hỗ trợ tìm kiếm liên hợp đa từ khóa [114], [32], [104], [86], [87] cho phép EU đưa ra yêu cầu tìm kiếm nhiều từ khoá đồng thời, khi đó điều kiện truy vấn được biểu thị qua chuỗi liên hợp của các từ khóa $Q = \{kw_1\ and\ kw_2\ and\ \cdots\ and\ kw_n\}$ và kết quả trả về là những tài liệu/bản ghi có chứa các từ khóa trong chuỗi này. Tính chất tìm kiếm liên hợp thể hiện ở việc không chấp nhận tài liệu/bản ghi trả về thiếu một trong bất kỳ từ khóa truy vấn nào và chỉ cần chạy một lần truy vấn với chuỗi liên hợp là trả về tập kết quả thoả mãn.

- Tìm kiếm phi liên hợp đa từ khóa

Lược đồ SE hỗ trợ tìm kiếm phi liên hợp đa từ khoá [31], [59], [36] cho phép truy xuất các tài liệu chứa tất cả hoặc một tập hợp con các từ khoá tìm kiếm. Nói cách khác các lược đồ này cho phép thực hiện chuỗi điều kiện truy vấn $Q = \{kw_1\ or\ kw_2\ or\ \cdots or\ kw_n\}$.

1.6.3. Tìm kiếm từ khóa gần đúng

- Tìm kiếm mờ

Trong các công cụ tìm kiếm trên internet, có thể người dùng nhập một truy vấn

không chính xác (một truy vấn có lỗi chính tả nhỏ), nhưng họ vẫn nhận được danh sách các tài liệu liên quan để truy xuất. Cách tiếp cận chung để tìm kiếm từ khóa mờ trên dữ liệu mã [69], [70], [128] như sau: 1) truy xuất các tài liệu khớp chính xác với truy vấn tìm kiếm và 2) truy xuất các tài liệu gần nhất có thể nếu có sự mâu thuẫn nhỏ trong truy vấn tìm kiếm.

- Tìm kiếm dựa vào từ đồng nghĩa

Một số lược đồ SE mà danh mục từ khóa không được chia sẻ, dẫn đến EU có thể nhập từ đồng nghĩa của các từ khóa có trong từ điển. Ví dụ: đối với từ khóa chuẩn là "Chăm chỉ", nhưng EU lại nhập cụm từ tìm kiếm là "Siêng năng" hoặc "Cần cù" hoặc "Chuyên cần",… và mong muốn chương trình vẫn có thể tìm thấy các tài liệu liên quan. Fu và cộng sự [40] đã giới thiệu lược đồ SE hỗ trợ tìm kiếm dựa trên từ đồng nghĩa bằng cách sử dụng phần mở rộng từ khóa, bao gồm việc liệt kê tất cả các từ đồng nghĩa có thể có của các từ khóa được trích xuất trong từ điển.

- Tìm kiếm dựa vào ngữ nghĩa

Các lược đồ SE hỗ trợ tìm kiếm dựa trên ngữ nghĩa mang lại sự mở rộng cho kết quả tìm kiếm từ khóa bằng cách không chỉ cung cấp cho người dùng cuối các tài liệu chứa các thuật ngữ chính xác mà còn cung cấp các tài liệu được liên kết về mặt ngữ nghĩa với từ khoá tìm kiếm [11]. Chen và cộng sự [32] đã đề xuất một lược đồ SE tìm kiếm ngữ nghĩa dựa trên cách tiếp cận kNN, các từ khóa có liên quan với nhau sẽ được nhóm lại. Xingming và cộng sự [127] đề xuất sử dụng thư viện mối quan hệ ngữ nghĩa, được xây dựng bằng cách tính toán giá trị xảy ra đồng thời ($Co-occurrence(k1,k2) = \frac{p(k_1,k_2)}{p(k_1).p(k_2)}$ với $p(k_1,k_2)$ biểu thị xác suất mà hai thuật ngữ xuất hiện cùng nhau, và $p(k_1), p(k_2)$ biểu thị xác suất mà chúng xuất hiện độc lập).

1.6.4. Tìm kiếm chuỗi con

Tìm kiếm các tài liệu/bản ghi có chứa một chuỗi ký tự bất kỳ là nhu cầu thực tế và có tính linh động cao trong tra cứu dữ liệu của EU, vì vậy các hệ quản trị CSDLQH hiện nay hỗ trợ phổ biến chức năng này trên dữ liệu ký tự rõ qua cú pháp "Like / Contains / FreeText". Tuy nhiên, tìm kiếm chuỗi con trên CSDLQH mã hóa lại là một vấn đề phức tạp và khó thực hiện hơn và ít nghiên cứu hơn so với tìm kiếm từ khoá do chuỗi con được EU yêu cầu có nội dung hoặc chiều dài thay đổi tuỳ ý. Có hai hướng tiếp cận để giải quyết bài toán này là: phân tích chuỗi con thành tập các từ khoá và tận dụng các lược đồ SE tìm kiếm theo từ khoá để tìm kiếm [33] hoặc thiết

kế các lược đồ SE với chức năng tìm kiếm chuỗi con chuyên biệt để thực hiện [30], [67], [50], [129], [83].

1.6.5. Tìm kiếm khoảng

Tìm kiếm khoảng cũng là yêu cầu phổ biến của EU đối với truy xuất thông tin trong CSDLQH mã hóa vì đây là dạng thức truy vấn tổng quát, bao hàm đầy đủ so sánh bằng, lớn hơn, nhỏ hơn trên dữ liệu số. Về cơ bản các lược đồ SE hỗ trợ truy vấn khoảng trên dữ liệu số mã hóa đều sử dụng chỉ mục mù. Các chỉ mục có thể được thiết kế dựa trên kỹ thuật phân nhóm/khoảng dữ liệu [92], [47], [58], [53], [113], [4] hoặc hàm băm [8], [120], [122] hoặc cơ chế bảo toàn thứ tự [71], [5], [66], [18], [17], [56] hay mã hóa đồng hình [78], [41], [22], [7].

1.7. Các yêu cầu đối với lược đồ mã hoá có thể tìm kiếm

Yêu cầu để có một lược đồ SE hiệu quả là sự cân bằng của 4 yếu tố: "Bảo mật", "Hiệu năng", "Khả thi triển khai" và "Đa dạng khả năng truy vấn". Các lược đồ SE đều biến đổi dữ liệu rõ thành các bản mã hoặc chỉ mục mù hỗ trợ tìm kiếm, vì vậy chúng cần phải chứa đựng một số thông tin bản rõ và đây chính là điểm yếu để kẻ gian có thể tìm cách khai thác/tấn công. Trong thực tế, hoàn toàn ta có thể tăng khả năng bảo mật lên cao tuy nhiên nó lại làm giảm hiệu năng cũng như khả năng đa dạng thực thi các lệnh truy vấn và ngược lại (xem **Hình 1.10**).

Hình 1.10. Các yếu tố tác động đến sự hiệu quả của lược đồ SE

Bốn yếu tố của SE được chi tiết qua các tiêu chí tại **Bảng 1.3** như sau:

Bảng 1.3. Các yêu cầu với lược đồ SE [51]

Các yêu cầu	Các tiêu chí cụ thể	Mô tả
Bảo mật	Tính riêng tư của tài liệu/bản ghi rõ	Các dữ liệu rõ có hai phiên bản được lưu trên CS là "Chỉ mục mã hoá" và "Bản mã dữ liệu". Trong đó "Bản mã dữ liệu" phải tuyệt đối bảo mật, được mã hoá bởi những thuật toán tiêu chuẩn như AES, DES, Blowfish [37], [13], [98] với khoá đủ lớn cùng các chế độ phi tất định để kẻ gian không có cơ hội khai thác.
	Tính riêng tư của chỉ mục	Chỉ mục mù lưu trữ trên CS không được tiết lộ bất cứ thông tin nào về bản rõ sinh ra nó như: nội dung, từ khoá, tần suất, thứ tự, chiều dài và các đặc trưng khác… Kẻ gian không thể suy diễn bản rõ tương ứng từ chỉ mục mù và đặc biệt không thể tìm được mối liên quan giữa các chỉ mục mù với nhau dựa trên biết trước một số thông tin tập bản rõ.
	Tính riêng tư của mẫu truy vấn	Là khả năng chống kẻ gian kết luận hai hay nhiều truy vấn có cùng điều kiện tìm kiếm hay không. Nói cách khác lộ mẫu truy vấn chính là cho kẻ gian có thể biết thông tin về các từ khoá hoặc chuỗi điều kiện tìm kiếm được EU yêu cầu. Như vậy lộ mẫu truy vấn có thể hiểu là rò rỉ nội dung từ khoá trực tiếp từ mẫu truy vấn mã hoá hoặc thông qua quan sát, thống kê và phân tích tập các mẫu truy vấn mã hoá tại CS. Mẫu truy vấn (gồm các từ khóa, toán tử) cần được mã hóa ở trạng thái phi tất định.
	Tính riêng tư của mẫu truy cập	Là khả năng chống lộ thông tin các tập kết quả truy vấn trả về trên CS. Vậy mẫu truy cập cho kẻ gian nhận thức được những tài liệu/bản ghi nào được truy cập ứng với một mẫu truy vấn bất kỳ. Qua đó, kẻ gian có thể đánh giá hai tập kết quả trả về giống nhau có thể đến từ hai yêu cầu truy vấn với mẫu truy vấn tương tự nhau.
	Tính riêng tư của cửa sập	- Chỉ duy nhất người sở hữu dữ liệu được vận hành và cũng chỉ duy nhất người dùng có thẩm quyền mới được yêu cầu cửa sập phục vụ. - Không kẻ gian nào có thể mô phỏng hoặc học hỏi cách thức hoạt động của cửa sập thông qua cơ chế thử nghiệm, học hỏi thích ứng từ các lần yêu cầu truy vấn trước đó.
Hiệu năng	Chi phí mã hoá và xây dựng chỉ mục	Bao gồm chi phí: tổ chức và cập nhật các cấu trúc dữ liệu trung gian; mã hoá dữ liệu lần đầu; sinh chỉ mục mù lần đầu; cập nhật dữ liệu mã hoá và chỉ mục đối với lược đồ SE động.
	Chi phí truy vấn	Bao gồm chi phí: biến đổi lệnh truy vấn trên dữ liệu rõ sang truy vấn trên mã (chi phí thực thi cửa sập tại Proxy); thực thi các thuật toán tìm kiếm (trên CS/PROXY/EU); chi phí giải mã và truy vấn lại đối với các tập kết quả trả về từ CS có chứa bản ghi âm tính giả hoặc dương tính giả.
	Chi phí truyền dữ liệu	Bao gồm chi phí: đóng gói và truyền tải dữ liệu giữa EU, PROXY, CS; số vòng giao tiếp dữ liệu (được hiểu là đối với mỗi yêu cầu truy vấn, EU/PROXY hỏi thì CS sẽ xử lý trả lời mà không tốn thêm một lần hỏi đáp bất kỳ nào nữa).

	Chi phí lưu trữ	Bao gồm chi phí: lưu trữ các cấu trúc chỉ mục, bản mã và cấu trúc dữ liệu hỗ trợ khác trên CS; lưu trữ các Metadata, từ điển tại PROXY hoặc EU.
Đa dạng truy vấn	Điều kiện truy vấn	Hỗ trợ đa dạng kiểu truy vấn trên nhiều kiểu dữ liệu khác nhau như: truy vấn trên dữ liệu số, ký tự, chuỗi con, đơn/đa từ khoá, xếp hạng, tìm kiếm mờ, tìm kiếm vị trí xuất hiện, toán tử logic, hàm thống kê,…
Tính khả thi triển khai	Khả năng triển khai thuận lợi quy trình, thành phần của các lược đồ SE	- Hỗ trợ triển khai trên CSDLQH. - Hỗ trợ triển khai đa người dùng cuối, đa người sở hữu dữ liệu, … - Mô hình thiết kế của lược đồ SE cho phép thiết lập vùng an toàn và các dịch vụ liên quan thuận lợi. Cho phép thiết lập các chính sách quản lý, phân phối khóa, phân quyền, xác thực truy cập, mã và giải mã … dễ dàng. - Tận dụng năng lực tối đa của CS cho lưu trữ và các thuật toán tìm kiếm, tính toán, … - Các lược đồ SE nên cung cấp các dịch vụ mà EU có thể truy cập dễ dàng thông qua các kiến trúc nền tảng phổ biến (website, webservices, API, mobile app, …) mà không yêu cầu cấu hình đặc biệt của thiết bị cuối.

1.8. Các mô hình bảo mật của lược đồ SE

Bảng 1.4 liệt kê bốn yếu tố cần có để một lược đồ SE hoạt động hiệu. Trong đó bảo mật là vấn đề được ưu tiên hàng đầu và là mục tiêu nguyên thủy mà lĩnh vực SE hướng đến nhằm giải quyết các vấn đề về an toàn thông tin khi triển khai dữ liệu trên hạ tầng thuê ngoài. Các mô hình bảo mật được liệt kê trong **Bảng 1.4.**

Hình 1.10 chỉ ra sự đánh đổi giữa các tiêu chí liên quan tới tính hiệu quả của SE, khi xét trên phương diện bảo mật chỉ có giải pháp ORAM [45] đảm bảo được các yêu cầu chống rò rỉ thông tin như **Bảng 1.4** mô tả, tuy nhiên lược đồ này có chi phí xử lý quá cao và không phù hợp trong triển khai thực tế với quy mô dữ liệu lớn.

Với lược đồ SE đầu tiên được đề xuất bởi Song và cộng sự [100], không có một định nghĩa bảo mật chính thức nào được đưa ra. Tuy nhiên, các tác giả đã chứng minh rằng sơ đồ của họ dựa trên một bộ sinh số giả ngẫu nhiên cho phép an toàn trước kiểu tấn công lựa chọn bản rõ [57]. Nói cách khác các văn bản mã hóa trong lược đồ không rò rỉ thông tin về các văn bản gốc, đạt được bảo mật theo mô hình IND-CPA. Tuy nhiên, trong SE, sự rò rỉ thông tin lại thường đến từ cửa sập hoặc mẫu truy vấn, điều này không được tính đến trong IND-CPA.

Khái niệm bảo mật đầu tiên trong ngữ cảnh của SE được giới thiệu bởi Goh [43], cho phép tạo ra các chỉ mục tìm kiếm chống lại các cuộc tấn công từ khóa được chọn thích ứng (IND1-CKA). Mô hình đảm bảo rằng kẻ gian không thể suy luận nội dung của tài liệu từ chỉ mục của nó và các chỉ mục có chiều dài tương đương nhau bất kể bản rõ. Tuy nhiên IND1-CKA lại không giải quyết vấn đề an toàn cửa sập. Để

giải quyết vấn đề này, Chang và Mitzenmacher [28] đã giới thiệu một định nghĩa IND-CKA dựa trên mô phỏng mới, đây là một phiên bản mạnh hơn của IND1-CKA nhằm cố gắng bảo vệ các cửa hậu bằng định nghĩa bảo mật của họ. Tuy nhiên Curtmola và cộng sự [33] chỉ ra các định nghĩa này không chính xác.

Sau đó, Goh đã giới thiệu tiếp định nghĩa bảo mật IND2-CKA, nhưng cả hai định nghĩa bảo mật IND1/2-CKA đều được coi là yếu trong ngữ cảnh của SE vì chúng không đảm bảo bảo mật của các cửa sập. Chúng không đảm bảo kẻ gian không thể phân tích nội dung từ khóa từ kết quả thu được của cửa sập.

Curtmola và cộng sự [33] đã xem xét lại các định nghĩa bảo mật hiện có và chỉ ra rằng bảo mật của chỉ mục và bảo mật của cửa sập vốn liên kết với nhau. Họ giới thiệu hai mô hình bảo mật mới cho SE, một mô hình không thích ứng (IND-CKA1) và một mô hình thích ứng (IND-CKA2), được sử dụng rộng rãi như các định nghĩa tiêu chuẩn cho SSE cho đến nay. Trong đó, IND-CKA2 mạnh mẽ hơn, cho phép kẻ gian điều chỉnh chiến lược lựa chọn các truy vấn dựa trên kết quả tìm kiếm trước đó.

Bảng 1.4. Các mô hình bảo mật cho lược đồ SE [51], [21]

Mô hình bảo mật	**Mô tả**
Known cipher-text model [24]	Trong mô hình này, kẻ gian (bao gồm cả CS) có quyền truy cập vào các bản ghi/tài liệu được mã hóa và chỉ mục mù tương ứng. Kẻ gian cố gắng thu thập thông tin bản rõ bằng các phân tích tần suất, thống kê đặc điểm của tập dữ liệu có được. Đây cũng là một kịch bản phổ biến trong tấn công các CSDL mã trên CS.
IND-CPA [100]	Mô hình này cho phép: ngay cả khi kẻ gian biết thuật toán mã hóa, văn bản mã hóa tương ứng với một tin nhắn M và độ dài Len(M) của tin nhắn thì cũng không thể thu được bất kỳ thông tin nào khác về M.
IND1-CKA [86]	Theo IND-CKA, chỉ mục mù cần đảm bảo bảo mật đối với nội dung của tài liệu, tức là không có thông tin nào về tài liệu bị tiết lộ cho kẻ gian từ chỉ mục được mã hóa. Hơn nữa, hai tài liệu có kích thước bằng nhau nên có các chỉ mục tương tự nhau, tức là đối với một kẻ gian, dường như hai chỉ mục chứa cùng số lượng từ khóa.
IND2-CKA [28]	Mô hình đảm bảo kẻ gian không thể phân biệt được chỉ mục của hai bản rõ có chiều dài, đặc tính khác nhau. Nói cách khác, các chỉ mục có chung độ dài và không có sự khác biệt đặc điểm khi quan sát, thống kê.
Nonadaptive IND-CKA1 [109]	Trong mô hình này, người ta giả định rằng các truy vấn của kẻ gian độc lập với các truy vấn trước đó, tức là kẻ gian đưa ra tất cả các truy vấn cùng một lúc và nhận được các kết quả mã hóa mong muốn. Kẻ gian sử dụng các truy vấn và kết quả trả về từ các quy trình, giai đoạn để thu thập thông tin, khai thác lỗ hổng của lược đồ SE như: bản mã, chỉ mục mù, mẫu truy vấn, hoạt động của cửa sập.
Adaptive IND-CKA2 [33], [95],	Đây là một biến thể của IND-CKA1, trong đó kẻ gian có thể sử dụng thông tin của các truy vấn trước đó (tức là kết quả và cửa sập) để tạo ra các truy vấn mới để có được cái nhìn sâu sắc về các tài liệu và chỉ mục được mã hóa (mà không có giả định

[57], [103], [29]	rằng tất cả các truy vấn được thực hiện cùng một lúc). Đây được coi là mô hình bảo mật tốt nhất áp dụng cho các lược đồ SE tới nay.

Các mô hình bảo mật trong **Bảng 1.4** đều được định nghĩa và chứng minh bảo đảm toán học đầy đủ trong các tài liệu tham chiếu liên quan. Theo **Nhận xét 1.1**, lược đồ SSE dựa trên chỉ mục mù được đánh giá phù hợp cho hướng nghiên cứu của đề tài luận án, vì vậy luận án sẽ trình bày thêm các định nghĩa bảo mật cho phép kiểm chứng lược đồ SSE đảm bảo mô hình IND-CK2. Trong mật mã nói chung và lĩnh vực SE nói riêng, người ta thường sử dụng một trò chơi với hai kịch bản cho kẻ tấn gian và người thách thức (được hiểu là DO) là trong mô hình thực (Real) và trong mô hình mô phỏng lý tưởng (Ideal). Mục tiêu cuối cùng là kẻ gian không thể phân biệt được các kết quả nhận được từ hai mô hình này, mức độ khác biệt được đặc tả qua các giá trị nằm dưới một ngưỡng e đủ nhỏ. Các định nghĩa bảo mật liên quan được luận án trình bày trong **Phụ lục 1. Các định nghĩa bảo mật cho lược đồ SSE**.

1.9. Tình hình nghiên cứu liên quan tới đề tài luận án

Các mục trên đã mang đến một bức tranh tổng quan về lĩnh vực mã hoá có thể tìm kiếm. Với đề tài "***Phát triển một số phương pháp truy vấn hiệu quả trên cơ sở dữ liệu quan hệ mã hóa***" luận án sẽ tiếp tục liệt kê và phân tích rõ hơn các công trình có liên quan. Qua đó thấy được các tồn tại, khoảng trống nghiên cứu, là cơ sở khẳng định được tính thời sự và xác định được phạm vi, đối tượng nghiên cứu của đề tài.

Trong quá trình nghiên cứu và tìm kiếm các tài liệu liên quan trong nước, một số đề tài luận án tiêu biểu được xác định có nội dung về vấn đề bảo mật trong CSDL, tuy nhiên không tập trung vào vấn đề truy vấn trên dữ liệu mã hoá theo các tiêu chuẩn của lược đồ SE. Cụ thể:

Trong luận án của tác giả Nguyễn Anh Tuấn (2011) [1] cho phép thực hiện truy vấn đa dạng trên CSDL mã hoá nhưng phù hợp với giả định không gian máy chủ quản lý CSDL an toàn. Theo đó, dữ liệu được mã hoá bởi các thuật toán mật mã tiêu chuẩn, khi truy vấn các dữ liệu mã được đẩy vào hệ thống các bảng ảo và thực hiện giải mã trước, truy vấn sau. Giải pháp có nhiều ưu điểm trong tận dụng các thế mạnh và kỹ thuật trên hệ quản trị CSDL, tuy nhiên việc mã và giải mã vẫn diễn ra trên máy chủ nên chưa phù hợp với các tiêu chí của lược đồ SE.

Nghiên cứu của tác giả Phạm Thị Bạch Huệ (2013) [2] tập trung vào bảo vệ tính riêng tư của dữ liệu và người dùng, vấn đề xác thực và ghi nhật ký. Vấn đề truy vấn

được phản ánh thông qua cơ chế tạo nhiễu và gây dư thừa bản ghi nhằm hạn chế việc rò rỉ thông tin qua phương pháp thống kê kết quả trả về. Rõ ràng, mục tiêu chính của truy vấn hướng tới khía cạnh bảo vệ tính riêng tư dữ liệu, chưa quan tâm đến các vấn đề khác như hiệu quả, khả thi hay đa dạng câu lệnh.

Một luận án khác của tác giả Hồ Kim Giàu (2021) [3] lại hướng tới giải quyết vấn đề xử lý song song trong truy vấn trên dữ liệu mã, xác thực kết quả trả về và quản lý thay đổi mã khoá. Các giải pháp về truy vấn hiệu quả trên dữ liệu mã qua các câu lệnh phổ biến cũng không phải là mục tiêu trong nghiên cứu này.

Vì vậy, sau đây tác giả sẽ tập trung vào các nghiên cứu nổi bật trong cộng đồng quốc tế để làm rõ về thực trạng liên quan tới mục tiêu của luận án.

1.9.1. Tình hình nghiên cứu về truy vấn trên dữ liệu ký tự mã hóa

Lĩnh vực tìm SE đã được đề cập và nghiên cứu hơn một thập kỷ gần đây, truy vấn dữ liệu ký tự mã hóa qua từ khóa và qua một số dạng thức khác được trình bày nhiều trong các lược đồ SSE [100], [43] cũng như các lược đồ PKSE [19], [20].

Trong các lược đồ SSE, Song và cộng sự [100] 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 tài liệu. Hơn nữa thiết kế của lược đồ này coi mỗi tài liệu là một chuỗi các từ khóa tìm kiếm có độ dài bằng nhau dẫn đến dạng thức của các từ khoá không phù hợp thực tế. Goh và cộng sự [43] đã giới thiệu các định nghĩa bảo mật chính thức của mã hóa đối xứng có thể tìm kiếm và đề xuất một lược đồ dựa trên bộ lọc Bloom [14] với thời gian tìm kiếm tuyến tính độ dài chỉ mục và chấp nhận một tỷ lệ dương tính giả có thể tối ưu được trên tập kết quả trả về. Một số lược đồ khác [62], [33], [95], [12], [124] tiếp tục cải tiến về tính hiệu quả tìm kiếm từ khóa thông qua triển khai các bảng chỉ mục đảo ngược. Các công trình gần đây cũng đã nghiên cứu một số trường hợp mở rộng của các lược đồ SSE nhằm hỗ trợ tìm kiếm các liên từ trong thời gian dưới tuyến tính [26], hỗ trợ truy vấn Boolean [84] và một số loại truy vấn khác.

Với các lược đồ PKSE, Boneh và cộng sự [19] là những người đầu tiên đề xuất mã hóa có thể tìm kiếm được bằng cách sử dụng mật mã bất đối xứng. Các tác giả đã phát triển một lược đồ mà bất kỳ ai có khóa công khai đều có thể ghi vào dữ liệu được lưu trữ trên máy chủ từ xa, nhưng chỉ những người dùng được ủy quyền với khóa riêng mới có thể tìm kiếm. Để hỗ trợ các truy vấn phức tạp hơn, tìm kiếm từ khóa liên

hợp, truy vấn tập hợp con và truy vấn khoảng trên dữ liệu được mã hóa cũng đã được đề xuất trong [20], [65], [99], [46], [54] trên cơ sở khai thác các tính chất trong toán học như đồng hình hoặc ghép cặp song tuyến. Tuy nhiên, các lược đồ PKSE vẫn là các lựa chọn có hiệu năng kém hơn các lược đồ SSE và tính khả thi thấp trong triển khai thực tế.

Các lược đồ SE nêu trên chủ yếu tập trung vào hỗ trợ tìm kiếm theo từ khóa chính xác. Li và cộng sự [69], [70] là người đầu tiên đề xuất lược đồ tìm kiếm từ khóa mờ trên dữ liệu được mã hóa. Họ đã đưa ra giải pháp xây dựng các tập từ khóa mờ dựa trên việc thu thập tài liệu và sau đó sử dụng khoảng chỉnh sửa để đo mức độ giống nhau giữa truy vấn từ khóa và truy vấn tập khóa mờ. Wang và cộng sự [110] cải tiến lược đồ tìm kiếm trước đó để đạt được độ phức tạp về thời gian tìm kiếm là một hằng số. Sau đó, Boldyreva và cộng sự [16] đã đưa ra một lược đồ tìm kiếm mờ hiệu quả để định vị các bản ghi tương tự, tuy nhiên kích cỡ bản mã và thời gian tính toán quá lớn nên giải pháp này cũng không phù hợp trong triển khai thực tiễn.

Cho tới nay, các công trình nghiên cứu về tìm kiếm chuỗi con trên dữ liệu mã (xem mục **1.6.4**) chưa được phong phú như các giải pháp tìm kiếm theo từ khóa 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 có chi phí tìm kiếm thấp trên một bản ghi ($O(1)$) được đề xuất bởi Curtmola và cộng sự [33], ta có thể ứng dụng sang giải quyết tìm kiếm chuỗi con trên dữ liệu được mã hóa. Cụ thể với một chuỗi con cần truy vấn, ta phân tích thành tập các từ khóa liên quan tới chuỗi con và kiểm tra sự tồn tại của mỗi phần tử thuộc tập đó trong bộ sưu tập các từ khóa tương ứng của từng tài liệu/bản ghi mã hóa. Việc kết luận tài liệu/bản ghi có chứa chuỗi con hay không có thể lựa chọn theo một trong hai cách sau:

- *Cách thứ nhất:* Nếu tồn tại ít nhất một từ khoá của chuỗi con nằm trong tập từ khoá của tài liệu/bản ghi thì coi tài liệu/bản ghi đó chứa chuỗi con. Kết quả so sánh này có thể dẫn đến việc chỉ một số từ khóa được phân tích từ chuỗi con tồn tại trong bộ sưu tập từ khóa của từng tài liệu/bản ghi, kéo theo số lượng tài liệu/bản ghi dương tính giả trả về rất lớn. Một trong các nguyên nhân khác nữa là việc chỉ sử dụng một số từ khóa sẽ không thể để đại diện hết mọi đặc tính cho chuỗi con (ví dụ các đặc tính: số loại ký tự, vị trí tương đối giữa các ký tự hay độ dài chuỗi, …) khi tìm kiếm trên các tài liệu/bản ghi mã hóa. Trong thực tế người dùng sẽ yêu cầu tìm kiếm một chuỗi con bất kỳ trên tập tài liệu/bản ghi mã hóa, vì vậy độ phức tạp lưu trữ và tính

toán sẽ trở lên rất lớn nếu áp dụng giải pháp này.

- *Cách thứ hai:* Nếu tất cả các từ khoá của chuỗi con đều nằm trong tập từ khoá của tài liệu/bản ghi thì kết luận tài liệu/bản ghi đó chứa chuỗi con. Ta có thể thấy các nhược điểm của giải pháp này khi xét ví dụ sau: yêu cầu tìm tất cả bản ghi có chứa chuỗi con *"searchable dat"*. Rõ ràng là không thể tìm được kết quả thoả mãn bởi không tồn tại từ khóa "*dat"* được định nghĩa từ trước trong trong bộ sưu tập từ khoá của CSDL (thực tế chỉ có tồn tại từ khoá *"data"*). Để cố gắng giải quyết yêu cầu này, ta cần định nghĩa tất cả các khả năng kết hợp khác nhau giữa các nhóm ký tự liền kề của các chuỗi dữ liệu nhạy cảm trong mỗi tài liệu/bản ghi nhằm tạo ra tập từ khóa đầy đủ, sau đó xây dựng chỉ mục tìm kiếm theo tập từ khóa này. Giả sử CSDL có N bản ghi, chuỗi ký tự nhạy cảm trên mỗi bản ghi có chiều dài trung bình là l_{str} thì số lượng từ khóa là $N * (1 + l_{str}) * l_{str}/2 \approx N * {l_{str}}^2$, điều này gây tăng chi phí xây dựng và lưu trữ chỉ mục tìm kiếm.

Với các nhược điểm được chỉ ra ở trên khi tận dụng giải pháp SSE hỗ trợ tìm kiếm theo từ khóa để truy vấn chuỗi con, một số lược đồ SSE được thiết kế riêng chức năng tìm kiếm chuỗi con đã được đề xuất trong những năm qua (xem **Bảng 2.4**). Các giải pháp này chủ yếu dựa vào một số cấu trúc dữ liệu (ví dụ: cây hậu tố, cây vun đống, $k - gram$, …) và các kỹ thuật tổ chức mã hoá, biến đổi dữ liệu để cho phép tìm kiếm chuỗi con trên tài liệu/bản ghi mã hoá tại CS. Ý tưởng thực hiện và tồn tại trên các lược đồ này được tóm tắt như sau:

- Chase và Shen (2015) [30] 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 trên một tài liệu/bản ghi mã là $O(\lambda.l_{substr} + occ)$ với λ là tham số bảo mật, l_{substr} là chiều dài chuỗi con, occ là số lần xuất hiện của chuỗi con trong tài liệu có độ dài l_{str}. Theo thông tin trong nghiên cứu này thì số vòng giao tiếp dữ liệu là ba, nhưng thực tế trong tài liệu [25] đã được chỉ ra cần mất bốn vòng. Phương pháp của Chase và Shen cũng tồn tại nhược điểm rò rỉ một số thông tin liên quan tới cấu trúc bên trong của cây sau mã hóa như: số lá, số con và số nhánh chạm đôi. Để giải quyết vấn đề này, nghiên cứu đề xuất sử dụng thêm các nút giả để tăng độ nhiễu giúp ẩn các thông tin thực của cây, nhưng 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 truy vấn chuỗi con trên CSDLQH mã hóa trong thực tiễn còn nhiều thách thức.

- Một lược đồ khác được trình bày bởi Faber và cộng sự (2015) [38] là một sự mở rộng của lược đồ SSE hỗ trợ các truy vấn liên hợp [25]. Ý tưởng với mỗi tài liệu rõ, xây dựng một chỉ mục mã hoá chứa đựng các $k-gram$ chồng chéo nhau cùng vị trí liên quan của chúng. Quá trình tìm kiếm chuỗi con trên chỉ mục mã hoá này chính là thực hiện truy vấn liên hợp các từ khoá của chuỗi con (chuỗi con cũng được phân tích thành tập từ khoá) với tất cả các $k-gram$ trong mỗi chỉ mục. Giải pháp này đòi hỏi chi phí tìm kiếm và tính toán cao cỡ $O((l_{substr}/kp).l_{str})$ và yêu cầu không gian lưu trữ lớn $O(N.((l_{str}-kp)+3)$, trong đó l_{str} là chiều dài trung bình chuỗi dữ liệu rõ.

- Strizhov và Ray (2016) [102] đề xuất lược đồ SSE hỗ trợ tìm kiếm chuỗi con dựa trên cây vun đống vị trí cho chuỗi dữ liệu rõ. Quá trình tạo cây vun đống vị trí sẽ được thực hiện trước với chuỗi dữ liệu rõ, sau đó thực hiện mã hóa từng nút của cây. Cách làm này sẽ tiết lộ một số thông tin về dữ liệu rõ thông qua khai thác mối quan hệ vị trí giữa các nút trong cây vun đống mặc dù nội dung trong mỗi nút đã được mã hoá. Hơn nữa lược đồ này yêu cầu duyệt từng ký tự của tài liệu để xác định các vị trí xuất hiện của chuỗi con trong tài liệu, do đó phải thiết kê thêm chỉ mục mã hóa cho từng ký tự để hỗ trợ quá trình tìm kiếm. Chỉ mục này sẽ làm lộ tần suất xuất hiện của các ký tự, đồng thời làm gia tăng chi phí tìm kiếm lên $O({l_{substr}}^2+occ)$ với occ là số lần xuất hiện của chuỗi con trên mỗi tài liệu/bản ghi mã và phải thực hiện tới ba vòng giao tiếp dữ liệu.

- Moataz và cộng sự (2017) [83] đề xuất cấu trúc SED là sự cải tiến các lược đồ SSE hỗ trợ tìm kiếm từ khoá đơn bằng cách cho phép tìm kiếm bất kỳ phần nào của các từ khóa được mã hóa mà không yêu cầu phải lưu trữ tất cả các kết hợp có thể có của các chuỗi con từ một từ điển nhất định. Kỹ thuật được đề xuất dựa trên ý tưởng về trực giao hóa chữ cái cho phép kiểm tra sự tồn tại chính xác của chuỗi con bằng cách thực hiện các tích vô hướng. Tuy nhiên thời gian truy vấn lớn cỡ $O(l_d+occ)$ với l_d là độ lớn của từ điển chứa tất cả các từ khóa của CSDL.

- Leontiadis và Li (2018) [67] đã trình bày một lược đồ SSE tìm kiếm chuỗi con bằng cách triển khai một cấu trúc dữ liệu được gọi là $FM-index$, là sự kết hợp của biến đổi Burrows-Wheeler Transform (BWT) và một mảng hậu tố, để giảm kích thước chỉ mục của lược đồ do Chase và Shen đề xuất. Chi phí lưu trữ lúc này sẽ là $O(4.N)$. Để xây dựng được $FM-index$ cần chia tài liệu thành $k-gram$, coi mỗi

$k - gram$ (còn gọi là một nhóm hoặc một bucket) đại diện bởi một ký tự và chèn thêm các nhóm giả vào tài liệu ban đầu để gây nhiễu, giúp ẩn tần suất thực của dữ liệu. Khi đó chi phí tìm kiếm trên một tài liệu/bản ghi mã trong trường hợp xấu nhất sẽ tuyến tính với chiều dài của chuỗi con $O(l_{substr} + occ + C_{dummy})$. Ngoài ra lược đồ này cũng không hỗ trợ truy vấn được các chuỗi con có độ dài nhỏ hơn k và mất tới ba vòng giao tiếp dữ liệu.

- Hahn và cộng sự (2018) [50] đề xuất chuyển đổi tìm kiếm chuỗi con thành các truy vấn phạm vi hiệu quả và sử dụng một cấu trúc đặc biệt là mã hóa bảo toàn thứ tự ẩn tần số (FHOPE). Tuy nhiên việc tốn tới hai vòng giao tiếp máy trạm và máy chủ, cũng như lựa chọn tham số k sao cho đảm bảo hiệu năng, bảo mật là một vấn đề cần nhiều đánh giá khi ứng dụng trong một CSDL có sẵn.

Mainardi và cộng sự (2019) [79] đề xuất giao thức PPSS hỗ trợ đa người dùng truy vấn mà không phát sinh thêm các giao tiếp xác thực giữa các người dùng với chủ sở hữu dữ liệu, đồng thời tối thiểu hóa chi phí giao tiếp dữ liệu (sử dụng một vòng giao tiếp với chi phí truyền tải cỡ $O((l_{substr} + occ).log^2(N))$). Giải pháp sử dụng kỹ thuật truy xuất thông tin riêng tư (PIR) [73] trên cơ sở mã hóa đồng hình kết hợp cùng Burrows Wheeler Transform (BWT) [23] để bảo vệ mẫu truy cập và xác định được chính xác sự tồn tại của chuỗi con. Tuy nhiên giao thức đề xuất gây ra không gian lưu trữ chỉ mục lớn $O(z.N)$ với z là số ký tự phân biệt trong các bản rõ và chi phí truy vấn cỡ $O(l_{substr} + occ)$ trên một tài liệu/bản ghi mã.

Mainardi và cộng sự (2021) [80] tiếp tục cải tiến giải pháp [79], ngoài việc cung cấp tính riêng tư của mẫu tìm kiếm và mẫu truy cập thì giao thức PPSS hỗ trợ cả tìm kiếm chuỗi chính xác và tìm kiếm mẫu ký tự bằng ký tự đại diện với chi phí truyền tải dữ liệu tiếp tục được cải thiện $O(l_{substr}.log^2(N) + occ.log(N))$. Tương tự, độ phức tạp tính toán cho quá trình tìm kiếm trên mỗi tài liệu/bản ghi tại CS cũng được cải thiện cỡ $O(l_{substr})$.

Một đề xuất liên quan tiếp theo của Yin và cộng sự (2021) [131] hướng tới tìm kiếm chuỗi con hiệu quả trên các từ khóa được đề xuất truy vấn. Mục tiêu hỗ trợ người dùng nhận được gợi ý phù hợp khi chỉ nhập vào một phần (chuỗi con) của từ khóa. Bản chất giải pháp của Yin là xây dựng một lược đồ SSE hỗ trợ tìm kiếm theo từ khóa, nhấn mạnh khả năng hỗ trợ người dùng lọc từ khóa phù hợp thông qua khả năng tìm kiếm chuỗi con trên danh sách tập từ khóa mã hóa. Trên CS sẽ lưu cặp chỉ

mục là I_W, I_F, trong đó I_W được xây dựng dựa trên cây vun đống vị trí các ký tự trong danh mục các từ khóa được phép tìm kiếm, nó hỗ trợ cơ chế chuỗi con do người dùng nhập vào để tìm được các từ khóa phù hợp. Chỉ mục I_F được sử dụng để truy vấn các văn bản/bản ghi có chứa các từ khóa nhận được sau quá trình tìm kiếm trên I_W. Ngoài việc tăng không gian lưu trữ cho hai chỉ mục, quá trình truy vấn trên CS cũng phải thực hiện qua hai pha với độ phức tạp của pha thứ nhất phụ thuộc vào độ lớn của chuỗi con, độ phức tạp của pha thứ hai phụ thuộc vào chiều dài và số lượng các văn bản/bản ghi trong CSDL.

Gần đây, Yamamoto cùng cộng sự (2022) [130] đã đề xuất lược đồ SSE hỗ trợ truy vấn chuỗi con có chiều dài bất kỳ dựa trên cải tiến biểu đồ DAWG [15], một cấu trúc cho phép dễ dàng tìm kiếm chuỗi con trên bản rõ. Theo đó cấu trúc chỉ mục tìm kiếm là một bộ gồm 3 chỉ mục con với không gian lưu trữ tổng thể tuyến tính với chiều dài chuỗi rõ và số lượng bản ghi. Ngoài ra hiệu quả tìm kiếm của giải pháp cũng chưa được đánh giá cao khi có độ phức tạp cỡ $O({l_{substr}}^2 + occ)$.

Nhận xét 1.2: Phần lớn các nghiên cứu liên quan tới truy vấn trên dữ liệu ký tự mã hóa tập trung chính vào kỹ thuật tìm kiếm theo từ khóa. Tuy nhiên, tìm kiếm theo từ khóa chỉ là một trường hợp đặc biệt của tìm kiếm chuỗi con và thực tế ít được sử dụng trong các yêu cầu truy vấn đa dạng của EU trên CSDLQH được triển khai tại CS (người dùng có xu hướng: gõ chuỗi con đầu vào bất kỳ để tìm kiếm dữ liệu trên các hộp tìm kiếm, email hay tra cứu nội dung tài liệu, ...). Tìm kiếm chuỗi con bao gồm một số dạng thức phổ biến: tìm kiếm tiền tố, hậu tố và tìm kiếm chuỗi con độ dài bất kỳ trên dữ liệu mã. Các công trình tiêu biểu về tìm kiếm chuỗi con trên dữ liệu mã được trích dẫn và phân tích trong luận án đã chỉ ra một số tồn tại, khó khăn trong vấn đề chi phí lưu trữ cao (hầu hết cỡ $O(N)$), chưa có sự cân bằng giữa hiệu năng tìm kiếm (chi phí cỡ $O(l_{substr} + occ)$ đến $O({l_{substr}}^2 + occ)$ trên mỗi văn bản/bản ghi) và bảo mật thông tin (rò rỉ thông tin từ chỉ mục, mẫu truy vấn, ...). Các nghiên cứu phơi bày vấn đề đánh đổi giữa hiệu năng và bảo mật, tạo ra thách thức về tính khả thi trong triển khai thực tiễn. Vì vậy, một trong các nội dung quan trọng được tập trung nghiên cứu và trình bày trong luận án này là đề xuất phương pháp truy vấn chuỗi con hiệu quả trên dữ liệu ký tự mã hóa trong CSDLQH mã, tính hiệu quả được tập trung thể hiện thông qua các tiêu chí là: mô hình triển khai khả thi, đảm bảo cân bằng hiệu năng truy

vấn và bảo mật thông tin của lược đồ tìm kiếm. Trong đó, hiệu năng truy vấn được đánh giá thông qua độ phức tạp thuật toán tìm kiếm khi thực thi trên toàn bộ tập bản ghi của CSDL tại CS cùng thời gian thực thi một quy trình truy vấn với một số kịch bản thử nghiệm trên dữ liệu mẫu và kết quả tìm kiếm tại CS không tồn tại giá trị dương tính giả. Bảo mật thông tin được đánh giá thông qua khả năng chống rò rỉ thông tin của chỉ mục mã hóa và mẫu truy vấn trước các tấn công phổ biến.

1.9.2. Tình hình nghiên cứu về truy vấn trên dữ liệu số mã hóa

Với tính chất đặc trưng của dữ liệu số, hầu hết các giải pháp truy vấn trên dữ liệu số mã hóa đều hỗ dạng thức truy vấn khoảng, bởi điều kiện truy vấn này bao trùm hết các toán tử so sánh bằng, lớn hơn và nhỏ hơn. Có hai hướng chính được sử dụng để giải quyết bài toán truy vấn khoảng trong lĩnh vực SE gồm:

Hướng thứ nhất: đưa bài toán truy vấn khoảng trên dữ liệu số về dạng truy xuất thông tin qua các so sánh bằng. Một ví dụ điển hình của ý tưởng này là điều kiện truy vấn "$X_t\ between\ l\ and\ h$" sẽ được biến đổi thành dạng "$X_t\ in\ (v_1, v_2, \dots, v_m)$" $\forall\ l \leq v_i \leq h$. Một cách tổng quát, việc truy vấn trên khoảng $[l, h]$ được liệt kê thành một loạt các truy vấn bằng trên tất cả các giá trị hoặc các khoảng con thuộc $[l, h]$. Bằng cách tạo các ánh xạ giữa giá trị số với từ khoá và kết hợp biểu diễn trên các cấu trúc dữ liệu đặc biệt, khi đó việc truy xuất thông tin sẽ được thực hiện thông qua các lược đồ SE hỗ trợ tìm kiếm theo từ khoá chính xác. Một số công trình tiêu biểu liên quan được nhận xét trong tài liệu tổng quan [68] như sau:

- Bằng cách kế thừa và mở rộng lược đồ của Cash [26], Faber và các cộng sự [38] xây dựng một cấu trúc cây nhị phân đầy đủ và ánh xạ các giá trị số lưu trong các bản ghi vào các lá của cây nhị phân này. Nội dung trong các lá được thể hiện qua các chuỗi bít và các được sắp xếp tăng dần. Mỗi truy vấn khoảng $[l, h]$ sẽ được xác định tương đương với một cây con bao gồm tập các nút bao phủ toàn bộ khoảng truy vấn, tiếp sau đó là việc so sánh cụ thể trên từng nút để tìm được chính xác tập nút lá thoả mãn yêu cầu. Lược đồ của Faber có chi phí tìm kiếm cỡ $O(\log(N) + N_r)$ và chi phí lưu trữ tuyến tính với số bản ghi của CSDL. Tương tự Pappas và cộng sự [88] đã phát triển một cấu trúc chỉ mục cho tìm kiếm boolean bằng cách sử dụng cây b-ary cân bằng dựa trên bộ lọc Bloom. Sau đó sử dụng phương pháp của Raykova [93] để chuyển điều kiện truy vấn khoảng thành các yêu cầu tìm kiếm boolean rời rạc.

- Demertzis và cộng sự [35] đã giới thiệu một lược đồ cho phép thực hiện truy vấn khoảng thông qua tìm kiếm dựa trên từ khóa. Phương pháp cơ bản của họ bao gồm việc liệt kê tất cả các khoảng truy vấn con có thể có từ khoảng truy vấn được yêu cầu và chỉ định một từ khóa riêng biệt đại diện cho mỗi khoảng con này. Do đó, mỗi tài liệu/bản ghi được liên kết với một tập hợp các từ khóa đại diện cho các khoảng truy vấn con mà có chứa giá trị trường dữ liệu số của tài liệu/bản ghi. Cuối cùng, tập dữ liệu từ khóa - tài liệu/bản ghi đã chuyển đổi này có thể được sử dụng để thực hiện truy vấn khoảng thông qua tìm kiếm từ khoá bằng các lược đồ SSE hiện có. Lược đồ tìm kiếm của Demertzis có ba tuỳ chọn Quadratic, Constant và Logarithmic. Trong đó phiên bản Quadratic có chi phí tìm kiếm tốt nhất cỡ $O(N_r)$, tuy nhiên chi phí lưu trữ cao và rò rỉ thông tin bản mã, thông tin mẫu truy vấn.

- Ngoài ra, Li và cộng sự [72] đã sử dụng lược đồ mã hóa tiền tố [74] để chuyển đổi các giá trị thành các họ tiền tố và tạo ra một cây nhị phân đầy đủ dựa trên bộ lọc Bloom làm cấu trúc chỉ mục. Sau đó, truy vấn khoảng $[l, h]$ được chuyển đổi thành một tập hợp tối thiểu các tiền tố. Lúc này, truy vấn khoảng được đơn giản hóa thành việc kiểm tra xem có các phần tử chung giữa hai tập hợp hay không. Giải pháp của Li có chi phí tìm kiếm phía CS cỡ $O(occ.N_r.logN)$, chi phí lưu trữ $O(N.v.\log 1/\epsilon)$ và tồn tại vấn đề rò rỉ thông tin.

- Một số nghiên cứu tương tự [34], [52], [121] cũng sử dụng cấu trúc B^+ Tree để tổ chức lưu trữ chỉ mục tìm kiếm và tạo ra một khái niệm mới là chỉ mục dựa trên B^+ Tree. Cụ thể cây B^+ Tree được xây dựng trên dữ liệu rõ, sau đó mã hóa từng nút của cây thay vì mã hóa toàn bộ cây. Cây sau khi mã hóa sẽ được lưu trữ trên CS, quá trình truy vấn sẽ được thực hiện thông qua việc duyệt cây B^+ Tree mã hóa. Cụ thể, bắt đầu từ nút gốc, toàn bộ dữ liệu mã của nút gốc sẽ được gửi về Client để thực hiện giải mã và so sánh với giá trị cần truy vấn để xác định nút trong tiếp theo sẽ được duyệt trên cây B^+ Tree tại CS, quá trình này được lặp lại đến khi xác định được nút lá. Như vậy, giải pháp chỉ mục dựa trên B^+ Tree cho phép hỗ trợ tìm kiếm bằng, truy vấn khoảng với tốc độ cao và tập dữ liệu trả về không bị dư thừa. Nhược điểm của giải pháp này nằm ở số vòng giao tiếp dữ liệu giữa người dùng và CSDLQH mã hóa tại CS. Khi số bản ghi lớn, số phần tử trong mỗi Node tăng và chiều cao của cây cũng được điều chỉnh, dẫn đến số vòng trao đổi dữ liệu tăng theo, chi phí truyền tải và giải mã cao gây áp lực lên hệ thống mạng và hiệu năng tại thiết bị người dùng cuối.

Hướng thứ hai: bảo tồn khả năng thực hiện các phép so sánh $>, \geq, <, \leq$ trong

quá trình truy vấn khoảng dựa trên các kỹ thuật mã hoá như OPE hay Bucket Index hoặc HE. Dưới đây là thực trạng các giải pháp hỗ trợ truy vấn khoảng xây dựng theo hướng này:

- Hacigumus và cộng sự [47] đề xuất lược đồ SE sử dụng các chỉ mục dựa trên nhóm/khoảng (còn gọi là Bucket). Quá trình truy vấn trên CS hoàn toàn thực hiện trực tiếp trên các BucketID và trả về các bản ghi chứa các giá trị số thuộc các Bucket mà có BucketID tương ứng thỏa mãn điều kiện truy vấn. Phương pháp chia Bucket có thể là nạp chồng hoặc không nạp chồng miền giá trị giữa các Bucket, bảo toàn hoặc không bảo toàn thứ tự các giá trị BucketID tương ứng với mối quan hệ thứ tự giữa các giá trị rõ thuộc các Bucket đó. Với phương pháp nạp chồng dữ liệu và không bảo toàn thứ tự, giá trị BucketID sẽ giúp tăng sự an toàn của chỉ mục nhưng lại giảm hiệu năng và sự đa dạng các câu lệnh truy vấn do gặp khó khăn trong sự biến đổi câu lệnh truy vấn trên dữ liệu rõ thành câu truy vấn trên dữ liệu mã tại các cửa sập. Ngược lại, với phương pháp chia các Bucket không nạp chồng và bảo toàn thứ tự lại đem tới khả năng dễ dàng biến đổi các câu lệnh truy vấn, tốc độ thực thi cao trên CSDLQH mã hóa nhưng lại gây rò rỉ thông tin bản rõ qua các tấn công về thống kê tần suất và khai thác thứ tự BucketID. Ngoài các ưu và nhược điểm nói trên, thì tỷ lệ dương tính giả của tập kết quả trả về lớn sau quá trình truy vấn dẫn đến tăng áp lực truyền tải dữ liệu và giải mã phía EU cũng là một điểm yếu khác của giải pháp này.

- Đối với giải pháp xây dựng chỉ mục hỗ trợ truy vấn trên dữ liệu số mã hóa dựa vào phương pháp chia Bucket, Hore và cộng sự [53] tìm ra các đại lượng là Varience và Entropy, mà sự thay đổi giá trị của chúng có liên quan đến các vấn đề về tối thiểu số lượng kết quả dư thừa trả về sau truy vấn trên các BucketID và tăng cường bảo mật cho các Bucket. Nghiên cứu chỉ ra Varience đại diện cho độ phân tán dữ liệu trong mỗi Bucket, giá trị này càng lớn thì khả năng chống lại các tấn công rò rỉ thông tin động càng tốt, còn Entropy đại diện cho khả năng phân bố dữ liệu giữa các Bucket, giá trị này càng nhỏ thì khả năng chống lại các tấn công rò rỉ thông tin tĩnh càng hiệu quả. Tuy nhiên, các giải pháp trong nghiên cứu của Hore khá riêng biệt và thiếu tính liên kết để chỉ ra làm thế nào chia Bucket để đồng thời Varience tăng và Entropy nhỏ nhưng vẫn đảm bảo tối thiểu hoá các kết quả dương tính giả trả về sau truy vấn.

- Wang và cộng sự [113] tiếp tục chỉ ra các vấn đề tồn tại của Varience. Theo đó Varience tác động đến khả năng bảo mật nội bộ trong từng Bucket, độ lớn giá trị Varicence có liên quan mật thiết tới độ lớn Bucket (độ lớn ở đây là kích thước miền

giá trị trong Bucket), đồng thời Varicence tăng có xu hướng làm mất đi sự phân bố đều của dữ liệu trong mỗi Bucket. Nghiên cứu cũng khẳng định, với hai Bucket có độ lớn bằng nhau thì Bucket nào phân phối đều hơn sẽ bảo mật hơn. Chính vì vậy, phương sai lớn là chưa đủ, các tác giả đã chỉ ra một đại lượng mới là "Độ lệch phân phối xác suất" (PDV) để thể hiện độ lệch giữa các giá trị thực với giá trị phân phối đều của Bucket, giá trị PDV càng nhỏ thì Bucket càng bảo mật. Nghiên cứu này chỉ đưa ra một số trường hợp mà Varience cao nhưng vẫn gây rò rỉ dữ liệu trong Buket, sau đó sử dụng PDV như một thước đo bổ sung để xử lý vấn đề cục bộ này mà chưa có giải pháp toàn diện hơn để người dùng có thể thiết kế ra các Bucket vừa nạp chồng, vừa bảo toàn thứ tự để đồng thời đảm bảo cả khía cạnh bảo mật và hiệu quả truy vấn.

- Bên cạnh chỉ mục tìm kiếm dựa trên Bucket thì chỉ mục tìm kiếm dựa trên hàm băm cũng được Damiani và cộng sự [34] đề xuất, nó được thiết kế dựa trên hàm băm mã hóa một chiều. Theo đó mỗi giá trị số bản rõ sẽ được ánh xạ qua một hàm băm một chiều để tạo ra một chỉ mục đại diện. Tuy nhiên do tính chất của hàm băm, các giá trị chỉ mục là tất định nên chỉ hỗ trợ được phép so sánh bằng và dễ bị tấn công qua các phương pháp thống kê. Ngoài ra sử dụng hàm băm tồn tại nguy cơ xung đột giá trị băm khi áp dụng trên một số lượng bản ghi lớn. Để hỗ trợ truy vấn khoảng đối với chỉ mục thiết kế dựa trên hàm băm, Damiani và cộng sự đã tổ chức lưu trữ các giá trị chỉ mục trên cấu trúc dữ liệu B^+ Tree để bảo tồn được thứ tự giữa các giá trị băm.

- Để hỗ trợ truy vấn khoảng và truy vấn bằng trên dữ liệu mã hoá mà không dùng chỉ mục dựa trên B^+ Tree, lược đồ mã hóa bảo toàn thứ tự (OPES) đã được giới thiệu bởi Agrawal và cộng sự [5] với ba giai đoạn "Model", "Flatten" và "Transform" để tạo ra chỉ mục OPES. Bằng cách sử dụng kỹ thuật mới trong chia Bucket dựa trên hai pha là "Growth phase" và "Prune phase" cùng các hàm ánh xạ đặc biệt và tham số "Scale Factor" đã tạo ra các chỉ mục bảo mật hơn so với các BucketIndex mà Hacigumus và cộng sự đề xuất [47]. Tuy nhiên, việc bảo toàn thứ tự các chỉ mục OPES mặc định tiết lộ thứ tự dữ liệu rõ và một số nội dung nếu kẻ gian biết trước một số thông tin của tập bản rõ hoặc miền giá trị của các trường dữ liệu rõ.

- Một giải pháp nâng cấp cho OPES là mã hóa bảo toàn thứ tự với kỹ thuật chia tách và nhân rộng (OPESS) [118] được Wang và cộng sự đề xuất giúp giải quyết vấn đề chống tấn công suy luận thứ tự khi biết trước một số thông tin của tập bản rõ. Kỹ thuật chia tách giúp làm phẳng tần suất xuất hiện của các chỉ mục để chống các tấn

công suy diễn dựa trên đặc trưng về tần suất xuất hiện của dữ liệu rõ (ví dụ một số bản rõ xuất hiện quá nhiều, một số xuất hiện quá ít), trong khi đó kỹ thuật mở rộng giúp tiếp tục tăng cường khả năng bảo mật chống lại các kiểu tấn công suy diễn dựa trên việc đã biết thông tin về cả tần suất và thứ tự giữa các giá trị rõ.

- Ưu điểm lớn nhất của giải pháp OPES và OPESS là cho phép truy vấn bằng, truy vấn khoảng trực tiếp trên dữ liệu mã hoá mà không trả về dữ liệu dương tính giả, có thể kết hợp với các cấu trúc dữ liệu BTree/B^+ Tree được sử dụng bởi các DBMS quan hệ để lập chỉ mục tăng tốc tìm kiếm dữ liệu mức vật lý. Mặc dù OPESS là một giải pháp nâng cấp cho OPES nhưng vẫn còn một số hạn chế, đặc biệt là việc không hỗ trợ cập nhật lại chỉ mục khi người dùng có sự thay đổi về giá trị dữ liệu rõ. Kết luận chung, các giải pháp này ngoài việc tồn tại một số vấn đề về bảo mật thì chi phí truy vấn còn cao do phải thực hiện các ánh xạ nhiều bước giữa bản rõ và chỉ mục.

- Các nghiên cứu đã giới thiệu ở trên đều tập trung vào khả năng hỗ trợ truy vấn thì giải pháp mã hóa đồng hình (HE) [78], [41], [22], [7] lại khai thác các tính chất trên đại số module cho phép tính toán, tổng hợp (thực hiện các toán tử cộng, trừ, nhân; thống kê đếm, tổng, tích, trung bình, lớn nhất, nhỏ nhất, ...) trực tiếp trên các chỉ mục mà không cần giải mã. Mã hóa đồng hình hoàn toàn (FHE) [41] là một nhánh quan trọng trong lĩnh vực này, FHE cho phép thực hiện tùy ý các phép tính trên dữ liệu mã hóa mà vẫn trả về tập kết quả mã chính xác. Tuy nhiên các nghiên cứu [42], [81] cũng chỉ ra FHE đòi hỏi nhiều tài nguyên phục vụ tính toán và kém hiệu quả trong triển khai thực tiễn.

Một số nghiên cứu trong những năm gần đây (xem **Bảng 3.5**) hỗ trợ truy vấn khoảng trên dữ liệu số mã hoá đã tập trung tối ưu hoá các tiêu chí như: chi phí lưu trữ; chi phí tìm kiếm trên CS, EU, Trapdoor/Token; số vòng giao tiếp và chi phí truyền tải dữ liệu; khả năng chống rò rỉ thông tin. Các giải pháp [60], [49], [94], [64] có lợi thế tận dụng tối đa năng lực của CS để đáp ứng các chi phí tìm kiếm nhưng độ phức tạp tìm kiếm trên CS cao [60], [49] hoặc gây rò rỉ thông tin [64] hoặc số vòng giao tiếp lớn [60] hoặc chi phí lưu trữ lớn [94]. Trong khi đó các giải pháp [136], [112], [90], [82], [105] mặc dù hỗ trợ bảo mật cao nhưng lại phát sinh chi phí tìm kiếm trên cả CS, EU hoặc Trapdoor/Token. Giải pháp [77] lại tập trung vào vấn đề xác thực dữ liệu và hỗ trợ tính toán thống kê (SUM, MAX, MIN, COUNT, AVG) nên chi phí tìm kiếm, lưu trữ còn cao và có rò rỉ thông tin.

Nhận xét 1.3: Các phân tích từ các nghiên cứu về truy vấn trên dữ liệu số cho

thấy việc đồng thời đảm bảo tính bảo mật, hiệu năng truy vấn, tính khả thi triển khai và sự đa dạng câu lệnh là một thách thức, điển hình trong đó là mối quan hệ đối nghịch giữa bảo mật và hiệu năng. Các giải pháp thường tiếp cận đẩy mạnh một trong bốn yêu cầu đã nêu, dẫn đến luôn tồn tại vấn đề trong mỗi lược đồ đã đề xuất như: i) để đảm bảo nâng cao tính bảo mật thì lại đánh đổi bởi chi phí tính toán và tìm kiếm quá lớn hoặc ii) chấp nhận rò rỉ quá nhiều thông tin bản rõ để tăng hiệu năng tính toán hoặc iii) cho phép kết quả truy vấn dư thừa quá nhiều dữ liệu nhằm hạn chế lộ thông tin bản rõ. Hai hướng tiếp cận của lĩnh vực SE trong xử lý truy vấn khoảng trên dữ liệu số đã được trình bày cho thấy những ưu nhược điểm của các giải pháp hiện thời với truy vấn khoảng:

- ***Hướng thứ nhất*** có lợi thế tận dụng được các lược đồ tìm kiếm theo từ khoá để phục vụ truy xuất thông tin rời rạc về các phần tử hoặc khoảng con nằm trong khoảng cần truy vấn. Tuy nhiên, việc phân rã khoảng truy vấn thành các đối tượng nhỏ hơn là phức tạp, không mang tính tổng quát và không đáp ứng tốt trong trường hợp xuất hiện các yêu cầu truy vấn khoảng mới chưa từng được thống kê. Hệ quả quan trọng nhất là gây tăng chi phí tìm kiếm do phải thực hiện nhiều lần khi tìm kiếm từ khoá tương ứng với các giá trị số/khoảng con mà nó đại diện. Để cải thiện hiệu năng, các nghiên cứu theo hướng này tích hợp các cấu trúc dữ liệu liên quan (cây nhị phân, bộ lọc Bloom, …) nhằm xây dựng lên các chỉ mục hỗ trợ tìm kiếm hiệu quả, tuy nhiên các cấu trúc này còn tồn tại các vấn đề về rò rỉ thông tin bản rõ.

- ***Hướng thứ hai*** sử dụng các kỹ thuật tổ chức mã hoá cho phép biến đổi trực tiếp các giá trị số về dạng chỉ mục hỗ trợ các phép so sánh. Ưu điểm chung của các kỹ thuật này là tạo ra các chỉ mục hỗ trợ triển khai truy vấn dữ liệu số thuận lợi, tuy nhiên tồn tại một số vấn đề như: bảo toàn thứ tự gây rò rỉ thông tin bản rõ (giải pháp OPE), kết quả trả về tồn tại bản ghi dư thừa (giải pháp Bucket), chi phí xây dựng chỉ mục và tìm kiếm trên chỉ mục cao (giải pháp FHE).

Vì vậy, bên cạnh việc đề xuất phương pháp truy vấn chuỗi con hiệu quả trên dữ liệu ký tự mã hóa, thì luận án này còn tập trung nghiên cứu phương pháp truy vấn khoảng trên dữ liệu số trong CSDLQH mã hóa. Cơ sở thực hiện dựa trên việc lựa chọn một số kỹ thuật, cấu trúc dữ liệu (như: Bucket, OPE, B^+Tree được trình bày trong hai hướng nghiên cứu ở trên) với nhiều ưu điểm về hiệu năng kết hợp cùng các phương pháp giấu tin đặc biệt sẽ tạo ra một chỉ mục hỗ trợ truy vấn khoảng hiệu quả. Tính hiệu quả sẽ được tác giả nhấn mạnh ở khía cạnh mô hình triển khai có tính khả

thi cao, lược đồ tìm kiếm đồng thời đáp ứng hài hòa về bảo mật và hiệu năng truy vấn (gồm độ phức tạp tìm kiếm, thời gian thực thi trên dữ liệu thực, số vòng giao tiếp và tính chính xác của tập dữ liệu trả về sau truy vấn trên CS).

1.10. Kết luận chương 1

Chương 1 đã tổng quan bức tranh mã hoá có thể tìm kiếm (SE), mô hình DAS-PROXY, các kỹ thuật tổ chức mã hoá cùng những phương thức tìm kiếm phổ biến trong SE và tình hình nghiên cứu liên quan tới đề tài luận án. Đó là cơ sở để rút ra các nhận xét quan trọng sau đây nhằm định hướng nội dung nghiên cứu ở các chương tiếp theo, cụ thể:

- *Nhận xét 1.1*: Chọn lược đồ SSE xây dựng trên chỉ mục mù và triển khai lược đồ trên mô hình DAS-PROXY là giải pháp cho đề tài luận án.

- *Nhận xét 1.2*: Đề xuất phương pháp truy vấn chuỗi con hiệu quả trên dữ liệu ký tự mã hóa trong CSDLQH mã hóa là mục tiêu thứ nhất cần giải quyết của luận án.

- *Nhận xét 1.3*: Đề xuất phương pháp truy vấn khoảng hiệu quả trên dữ liệu số mã hóa trong CSDLQH mã hóa là mục tiêu thứ hai cần giải quyết của luận án.

- Tính hiệu quả được nhấn mạnh ở khả năng chống rò rỉ thông tin và hiệu năng thực thi truy vấn của các lược đồ đề xuất.

CHƯƠNG 2.
PHÁT TRIỂN PHƯƠNG PHÁP TRUY VẤN CHUỖI CON HIỆU QUẢ TRÊN DỮ LIỆU KÝ TỰ TRONG CSDLQH MÃ HÓA

Chương 2 tập trung thiết kế lược đồ SSE dựa trên chỉ mục cho phép truy vấn chuỗi con hiệu quả thông qua chuỗi điều kiện "$LIKE\ '\%\ substring\ \%'$" trong CSDLQH mã hóa. Các đóng góp nổi bật gồm: xây dựng lược đồ DIQ-SSE vận hành trên mô hình DAS-PROXY qua 4 giai đoạn; đề xuất phương pháp xây dựng cặp chỉ mục mù $Index1, Index2$; đề xuất quy trình và thuật toán truy vấn hiệu quả chuỗi con tuần tự qua hai chỉ mục theo 4 bước; cuối cùng là xây dựng các kịch bản thử nghiệm và thực hiện đánh giá hiệu năng, phân tích bảo mật và so sánh lược đồ DIQ-SSE với các công trình tiêu biểu liên quan. Các ý tưởng và đề xuất trong chương 2 được NCS công bố tại các công trình liên quan là [CT1], [CT2] và [CT3].

2.1. Đặt bài toán và dẫn luận ý tưởng thực hiện

Đặt bài toán:

Quan hệ $R(ID, X_1, X_2, \ldots X_t, \ldots X_n)$ có số lượng bản ghi lớn với X_t là trường dữ liệu ký tự rõ nhạy cảm, quan hệ mã $R^E(ID, X_1, X_2, \ldots X_t^E, \ldots X_n)$ lưu trên đám mây tương ứng với quan hệ R và X_t^E là trường dữ liệu mã tương ứng của X_t. Cho một chuỗi ký tự con A_s' bất kỳ, yêu cầu:

- Truy vấn trên trường dữ liệu X_t^E của quan hệ R^E để tìm ra ***chính xác*** những bản ghi có chứa A_s'.

- Quá trình truy vấn chuỗi con đạt hiệu quả tốt thông qua các tiêu chí về: ***bảo mật*** (chống rò rỉ thông tin theo mô hình IND-CKA2 (**Bảng 1.4**)); ***hiệu năng*** (thời gian thực thi tốt trên dữ liệu thực nghiệm lớn, quy trình tìm kiếm và độ phức tạp thuật toán tìm kiếm có ưu thế khi so sánh với các nghiên cứu gần đây); ***đa dạng truy vấn*** (hỗ trợ dạng thức truy vấn chuỗi con phổ biến qua các biến thể của cú pháp LIKE trong CSDLQH); ***khả thi triển khai*** (thuận lợi quản lý metadata và thực thi các thuật toán bảo mật, hỗ trợ tùy biến các tham số để điều chỉnh bảo mật và hiệu năng).

Dẫn luận ý tưởng thực hiện:

Về vấn đề chọn lược đồ và mô hình: dựa vào **Nhận xét 1.1**, luận án ưu tiên sử dụng lược đồ SSE thiết kế dựa trên chỉ mục mù [43], [33], [76] và triển khai trên mô hình DAS-PROXY [135], [6], [91], [9]. Theo phân tích trong **mục 1.8**, lược đồ SSE kế thừa thiết kế từ các công trình nêu trên có tính bảo mật được đảm bảo theo mô

hình IND-CKA2. Việc sử dụng chỉ mục mù giúp tăng cường tính khả thi triển khai do dễ kết hợp với các cấu trúc dữ liệu cho phép tăng tốc truy vấn để đạt hiệu năng tốt hơn. Trong khi đó mô hình DAS-PROXY giải quyết được một số tồn tại trong quản lý khoá/metadata/trapdoor của lược đồ SSE, hỗ trợ quá trình mã/giải mã độc lập với EU giúp tăng cường bảo mật do EU chỉ gửi bản rõ mà không nắm đc bản mã/chỉ mục mù tương ứng và các thuật toán bí mật. Ngoài ra DAS-PROXY có thể hỗ trợ các vấn đề khác trong hướng phát triển của luận án: lược đồ SSE động (đáp ứng biến động dữ liệu như thêm, sửa, xoá), đổi khoá định kỳ, cân bằng tải, multi DO/CS...

Ý tưởng truy vấn chung: **Nhận xét 1.2** cho thấy việc tìm kiếm trực tiếp chuỗi con trên dữ liệu mã đòi hỏi độ phức tạp cao. Do đó để khắc phục tồn tại này, bước đầu luận án sẽ sử dụng phương pháp tìm kiếm từ khoá chính xác với hiệu năng cao để lọc bớt dữ liệu mã không thoả mãn điều kiện, sau đó mới thực hiện truy vấn chuỗi con bằng các thuật toán đặc biệt trên phạm vi kết quả nhỏ hơn. Như vậy quá trình truy vấn được chia làm hai giai đoạn: *Giai đoạn 1*. Tìm kiếm từ khoá chính xác trên toàn bộ CSDLQH mã hóa; *Giai đoạn 2*. Truy vấn chuỗi con trên kết quả giai đoạn 1.

Ý tưởng thực hiện Giai đoạn 1: từ ý tưởng phân tích một chuỗi dữ liệu thành tập các cặp ký tự liền kề [120], [122], luận án phát triển một cấu trúc dữ liệu mới gọi là "*Tập các cặp ký tự kết nối phân biệt theo khoảng cách*" [CT1] cho phép mô tả chi tiết hơn các đặc tính của chuỗi rõ. Mỗi phần tử của cấu trúc nói trên được coi như một từ khóa trích xuất từ chuỗi dữ liệu. Dựa vào một số lược đồ SE [100], [43], [101], [132] sử dụng chỉ mục mù dựa trên bộ lọc Bloom hỗ trợ hiệu quả tìm kiếm theo từ khóa, luận án vận dụng bộ lọc này kết hợp các phương pháp gây nhiễu để mã hóa một chiều các phần tử trong cấu trúc đề xuất thành một chuỗi m bit (gọi là $Index1$). Khi EU yêu cầu truy vấn, chuỗi con cũng được biến đổi thành một chuỗi m bit theo cách tương tự. Quá trình truy vấn trên $Index1$ được thực hiện với tốc độ cao qua phép "And Bitwise" hai chuỗi bit này. Các phân tích về bộ lọc Bloom trong **PL2.2** còn cho thấy $Index1$ có chi phí lưu trữ thấp và cho phép kiểm soát tỷ lệ dương tính giả.

Ý tưởng thực hiện Giai đoạn 2: Mục tiêu xây dựng kỹ thuật tìm kiếm chuỗi con cho phép loại bỏ hoàn toàn các bản ghi dương tính giả trong kết quả trả về sau truy vấn trên $Index1$. Luận án tiếp tục duy trì quan điểm thực hiện tìm kiếm dựa trên chỉ mục và đề xuất thiết kế một chỉ mục mù mới gọi là $Index2$ [CT3]. Để tạo chỉ mục hỗ trợ tìm kiếm chuỗi con, các công trình [67], [25] dựa vào giải pháp tạo một $n-gram$ khổng lồ, quá trình tạo và lưu trữ phức tạp, rò rỉ thông tin và có thể trả về kết

quả truy vấn chứa giá trị dương tính giả. Trong khi đó giải pháp [10] hỗ trợ tạo ra chỉ mục hỗ trợ tìm kiếm chuỗi con chính xác, được xây dựng dựa trên cấu trúc dữ liệu cho phép mô tả vị trí các ký tự trong chuỗi rõ, tuy nhiên cấu trúc dữ liệu này gây rò rỉ thông tin bản rõ. Dựa trên lợi thế tìm kiếm chính xác của [10], luận án kế thừa để thiết kế $Index2$ nhưng có cải tiến trong cách xây dựng các dãy bit mô tả vị trí mỗi ký tự phân biệt của chuỗi rõ, kết hợp với phương pháp biến đổi dịch vòng các dãy bít này nhằm tạo ra $Index2$ như một ổ khoá kết hợp có tính bảo mật tốt hơn.

Qua phương pháp tiếp cận và dẫn luận các ý tưởng thực hiện hai giai đoạn truy vấn cho thấy mối liên hệ chặt chẽ giữa chỉ mục $Index1$ và $Index2$. Trong đó $Index1$ đóng vai trò như một phễu lọc hiệu quả, loại bỏ phần lớn dữ liệu không thỏa mãn điều kiện truy vấn trên cơ sở tìm kiếm tốc độ cao thông qua các từ khóa lấy từ "*Tập các cặp ký tự kết nối phân biệt theo khoảng cách*". Còn quá trình thực hiện tìm kiếm chuỗi con phức tạp hơn sẽ được thực hiện qua $Index2$ trên tập dữ liệu rút gọn trả về sau khi được lọc bởi $Index1$. Tính hiệu quả của sự kết hợp 2 chỉ mục đề xuất được phân tích thêm tại **mục 2.6.3** của luận án.

Để hiện thực các ý tưởng trên, cần xây dựng một lược đồ SSE (gọi là DIQ-SSE) dựa trên cặp chỉ mục mù $Index1, Index2$ [CT1] đại diện cho các giá trị trong trường X_t. Các trường chỉ mục SIX_t^1, SIX_t^2 tương ứng sẽ được bổ sung thêm vào quan hệ $R^E(ID, X_1, X_2, \ldots X_t^E, \ldots X_n, SIX_t^1, SIX_t^2)$ để lưu các giá trị của $Index1$ và $Index2$.

2.2. Đề xuất lược đồ và mô hình triển khai truy vấn chuỗi con trên CSDLQH mã hóa

Dựa trên các giai đoạn cơ bản của lược đồ SSE (tại **Mục 1.5.1**) và ý tưởng được dẫn luận trong **Mục 2.1**, luận án đề xuất lược đồ DIQ-SSE hỗ trợ truy vấn chuỗi con hiệu quả trên CSDLQH mã hóa. Lược đồ gồm 5 giai đoạn sau (**Hình 2.1**):

Giai đoạn 1. $\boldsymbol{GenSecretMetadata(\lambda)}$: giai đoạn này gồm các thuật toán sinh tập dữ liệu bí mật $MetaData$ (bao gồm các khóa, tham số bí mật khác,…) do người sở hữu dữ liệu thực thi với tham số đầu vào λ.

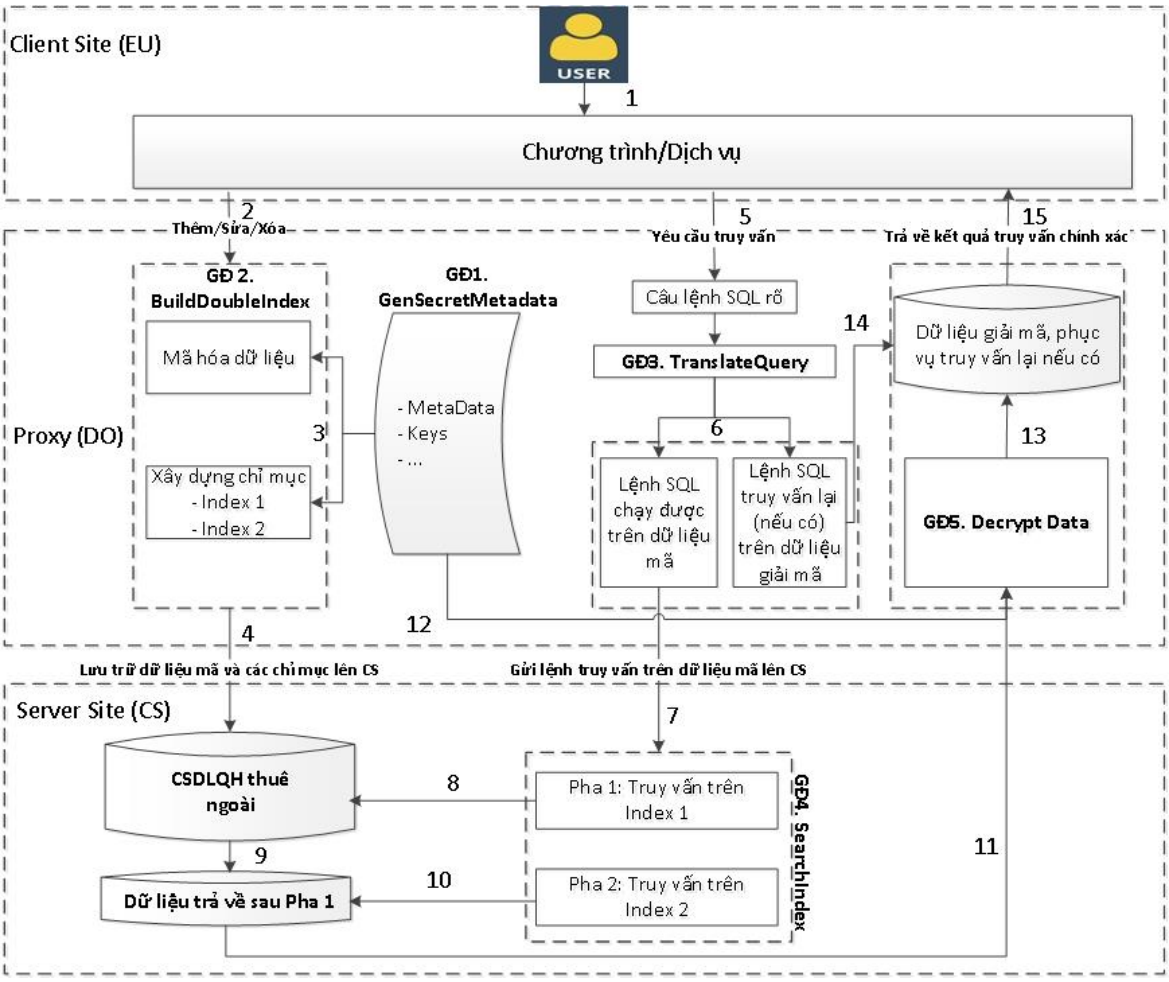
Giai đoạn 2. $\boldsymbol{BuildDoubleIndex}$: giai đoạn này bao gồm các thuật toán mã hóa, thuật toán $BuildIndex1(A_s, MetaData)$ và $BuildIndex2(A_s, MetaData)$ để sinh các bản mã và chỉ mục mù $Index1$, $Index2$ cho chuỗi dữ liệu rõ A_s.

Giai đoạn 3. $\boldsymbol{TranslateQuery}$: giai đoạn này bao gồm hai thuật toán $Tran1(A_s', MetaData)$ và $Tran2(A_s', MetaData)$ để biến đổi chuỗi con truy vấn A_s'

thành các giá trị tương ứng $I1$ và $I2$ cho phép tìm kiếm được trên chỉ mục $Index1$ và $Index2$. $TranslateQuery$ tương đương với cửa sập trong các lược đồ SE khác.

Giai đoạn 4. ***SearchIndex***: giai đoạn này bao gồm hai thuật toán $SearchIndex1(Index1, I1)$ và $SearchIndex2(Index2, I2)$ cho phép xác định được A'_S có phải là chuỗi con của A_S hay không một cách hiệu quả. Trong bước này, $SearchIndex1$ sẽ được thực thi trước trên toàn bộ các bản ghi của quan hệ R^E và trả về tập kết quả mã dư thừa, sau đó thuật toán $SearchIndex2$ tiếp tục được thực thi trên tập bản ghi được trả về bởi $SearchIndex1$ và trả về tập kết quả mã chính xác.

Giai đoạn 5. Decrypt: giải mã tập kết quả của **Giai đoạn 4** và trả về EU.



Hình 2.1. Triển khai lược đồ DIQ-SSE trên mô hình DAS-PROXY

Luận án lựa chọn mô hình DAS-PROXY với ba khu vực: Client Site, Proxy, Server Site để triển khai – vận hành lược đồ DIQ-SSE đã đề xuất (**Hình 2.1**), cụ thể:

- ***Client Site:*** là khu vực dành cho EU đưa ra các yêu cầu truy vấn chuỗi con.

- ***Proxy:*** là khu vực tin cậy được quản lý bởi DO. Các thuật toán $GenSecretMetadata$, $BuildDoubleIndex$, $TranslateQuery$ trong lược đồ DIQ-SSE sẽ được thực hiện tại Proxy. Các $MetaData$ được lưu tại Proxy bao gồm:

1) Tham số k, m, q với: k là số lượng hàm băm $\{h_1, h_2, \dots, h_k\}$; m là chiều dài chuỗi bit đại diện cho $Index1$; q là một số nguyên tố lớn.

2) Khóa bí mật $sKey$: dùng để mã và giải mã dữ liệu, là tham số cho các hàm f^{Enc} và f^{Dec} (trong mô hình này sử dụng thuật toán AES với OFB Mode).

3) Tập khóa bí mật H_KEY: sử dụng trong các hàm băm (trong mô hình này sử dụng các hàm băm SHA-256), cụ thể :

- Tạo tập khóa bí mật H_KEY thông qua các hàm giả ngẫu nhiên theo **Định nghĩa 1** ($H\_KEY = \{hKey_1, hKey_2, \dots, hKey_k\} \mid hKey_i \leftarrow PRF(\lambda)$ với λ là tham số bảo mật). Tập khóa H_KEY được sử dụng trong thuật toán $BuildIndex1$ và $Tran1, Tran2$.

4) Tham số bí mật u, u': sử dụng tạo tập các cặp ký tự kết nối theo khoảng cách.

5) Tập giá trị bí mật $VA = \{Va_1, Va_2, \dots, Va_z\}$: là tập giá trị bí mật dựa trên q, đại diện cho các ký tự phân biệt của dữ liệu rõ.

- ***Server Site*** (trong luận án này hiểu là máy chủ đám mây – CS): được quản lý bởi DSP, là nơi lưu trữ CSDLQH mã hóa cùng các chỉ mục $Index1, Index2$. CS tiếp nhận câu lệnh truy vấn mã từ Proxy và thực thi trên CSDLQH mã hóa thông qua các thuật toán $SearchIndex1$, $SearchIndex2$, cuối cùng trả tập kết quả mã về Proxy.

2.3. Xây dựng cặp chỉ mục hỗ trợ tìm kiếm

Coi **Giai đoạn 1. $\boldsymbol{GenSecretMetadata(\lambda)}$** đã được DO thực hiện tại Proxy. Mục này tập trung mô tả **Giai đoạn 2. $\boldsymbol{BuildDoubleIndex}$** của lược đồ DIQ-SSE.

Cho tập $AS = \{A_{s1}, A_{s2}, \dots, A_{sn}\}$ với mỗi phần tử là một chuỗi ký tự rõ nhạy cảm lưu trong trường X_t tương ứng tại mỗi bản ghi của quan hệ R. Khi đó với mỗi $A_s \in AS$, tại Proxy cần tạo ra bộ ba giá trị là: bản mã A_s^E, cặp chỉ mục $Index1$ và $Index2$. Các giá trị này được lưu tương ứng tại các trường X_t^E, SIX_t^1, SIX_t^2 trong quan hệ R^E trên CS. Với $A_s^E = f^{Enc}(A_s, sKey)$ và $A_s = f^{Dec}(A_s^E, sKey)$.

2.3.1. Xây dựng chỉ mục Index1

Định nghĩa 2.1. Cho $A = \{a_1, a_2, \dots, a_z\}$ là tập các ký tự phân biệt để tạo nên các chuỗi rõ và $A_s = c_1 c_2 \dots c_v$ là một chuỗi được tạo bởi các ký tự trong tập A ($z, v \geq 1$; $c_i \in A$ *với* $1 \leq i \leq v$). Tập A_s^u gọi là *tập các cặp ký tự kết nối theo khoảng cách* u ($0 \leq u \leq v$) *trên chuỗi* A_s được định nghĩa như sau:

$$A_s^u = A_s^{uFirstCharater} \cup A_s^{uSingleCharater} \cup A_s^{uPairCharater} \cup A_s^{uLastCharater}$$

trong đó:

$$\begin{cases} A_s^{uFirstCharater} = \{c_1 || *\} \\ A_s^{uSingleCharater} = \{c_i\} \; với \; 1 \le i \le v \\ A_s^{uPairCharater} = \{(c_i \; || \; j-i-1 \; || \; c_j)\} \; với \; 1 \le i < j \le v \\ và \; 0 \le j-i-1 \le u \\ A_s^{uLastCharater} = \{* \; || c_v\} \end{cases} \quad (2.1)$$

Ví dụ 2.1: Giả sử có chuỗi $A_s = "aabcdadb"$, khi đó $v = 8$ và theo **Định nghĩa 2.1** thì *tập các cặp ký tự kết nối theo khoảng cách* $u = 1$ *trên* A_s như sau:

$A_s^u = \{a*, \; a, \; a, \; b, \; c, \; d, \; a, \; d, \; b, \; a0a, \; a1b, \; a0b, \; a1c, \; b0c, \; b1d, \; c0d, \; c1a, \; d0a, \; d1d, \; a0d, \; a1b, \; d0b, \; *b\}$

Ta có thể chọn $\boldsymbol{u} \in [\mathbf{0}, \mathbf{6}]$ trong **Ví dụ 2.1** tùy thuộc vào nhu cầu mô tả mối quan hệ giữa các ký tự trong chuỗi A_s, nếu $\boldsymbol{u}$ càng lớn thì tập A_s^u càng thể hiện được nhiều đặc tính của chuỗi ký tự A_s.

Định nghĩa 2.2. Cho A_s^u là *tập các cặp ký tự kết nối theo khoảng cách u trên chuỗi ký tự* A_s. Gọi $DA_s^u = \{DW_1, DW_2, \dots, DW_l\}$ là *tập các cặp ký tự kết nối phân biệt theo khoảng cách u trên* A_s nếu DA_s^u chỉ chứa l phần tử phân biệt của A_s^u.

Theo **Định nghĩa 2.2**, xét **Ví dụ 2.1**, khi đó tập DA_s^u tương ứng với tập A_s^u được mô tả như sau: $DA_s^u = \{a*, \; a, \; b, \; c, \; d, \; a0a, \; a1b, \; a0b, \; a1c, \; b0c, \; b1d, \; c0d, \; c1a, \; d0a, \; d1d, \; a0d, \; d0b, \; *b\}$

Công thức (2.1) cho thấy nếu cho $u' \le u$ thì $A_s^{u'} \subset A_s^u$, do đó $DA_s^{u'} \subset DA_s^u$ và nếu A_s' là chuỗi con của A_s thì $DA_s'^{u'} \subset DA_s^u$ với $0 \le u' \le |A_s'| \le u$.

Ta sử dụng DA_s^u để biểu diễn thay cho chuỗi A_s, khi đó tập $AS = \{A_{s1}, A_{s2}, \dots, A_{sn}\}$ sẽ được đại diện bởi tập $D = \{DA_{s1}^u, DA_{s2}^u, \dots, DA_{sn}^u\}$. Lúc này việc xây dựng chỉ mục $Index1$ sẽ được thực hiện trên mỗi phần tử DA_{si}^u thuộc D. Dựa vào ý tưởng của [100], [43], [101], [132], luận án xây dựng $Index1$ bằng cách dùng bộ lọc $Bloom$ để nén DA_{si}^u về chuỗi m bit thông qua k hàm băm. Giá trị $m, \; k$ được chọn dựa trên công thức (**PL2.2**) với: m nguyên, $k \ge 1$ và n là là số lượng *"cặp ký tự kết nối theo khoảng cách* u*"* trung bình trên tất cả các phần tử DA_{si}^u trong D.

Gọi $f^{ElementMax}(D)$ là hàm lấy ra phần tử DA_{si}^u có chiều dài lớn nhất trong tập D, giải thuật tạo $Index1$ được xây dựng như sau:

Thuật toán 2.1: $BuildIndex1$

Input

$A_s = c_1c_2 \dots c_v$: chuỗi ký tự rõ nhạy cảm cần xây dựng chỉ mục $Index1$;

ID: giá trị định danh của bản ghi chứa chuỗi rõ A_s;

m: chiều dài chuỗi bit của $Index1$;

k: số lượng hàm băm;

$H\_KEY = \{hKey_1, hKey_2, \dots, hKey_k\}$: tập khóa sử dụng cho k hàm băm;

u: khoảng cách tối đa giữa các ký tự trong các cặp ký tự kết nối;

$nMax$: chiều dài của $f^{ElementMax}(D)$;

Output

$Index1$: là chuỗi bít BF với chiều dài m;

Các bước thực hiện

(1) Xây dựng tập $DA_s^u = \{DW_1, DW_2, \dots, DW_l\}$ theo **Định nghĩa 2.2**;

(2) Với mỗi từ khoá (một cặp ký tự kết nối) $DW_i \in DA_s^u \mid i \in [1, l]$, tính toán:

- Tập giá trị $\{p_1, p_2, \dots, p_k\}$ như sau:

$\{p_1 = h_1(hKey_1, DW_i), p_2 = h_2(hKey_2, DW_i), \dots, p_k = h_k(hKey_k, DW_i)\}$;

- Tập giá trị $\{q_1, q_2, \dots, q_k\}$ như sau:

$\{q_1 = h_1(p_1, ID), q_2 = h_2(p_2, ID), \dots, q_k = h_k(p_k, ID)\}$;

- Đặt $BF[q_j] = 1$ với $j \in [1, k]$;

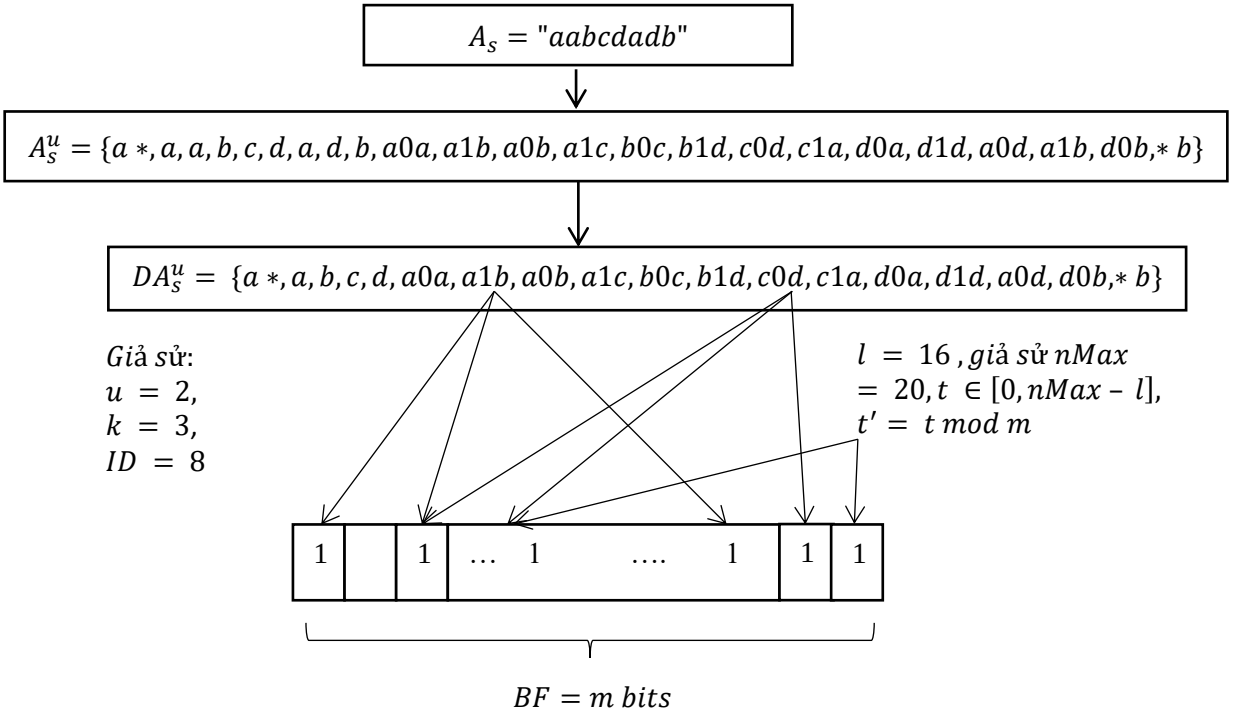
(3) Sinh số ngẫu nhiên t trong khoảng $[0,\ nMax - l]$ thông qua hàm PRF, sau đó thực hiện:

- Tính $t' = t \bmod m$;

- Tính $BF = BF \vee t'$;

(4) Trả về BF;

Ví dụ 2.2: Trong quan hệ R, cho bản ghi có $ID = 8$ có giá trị tại trường X_t là "$aabcdadb$", yêu cầu xây dựng $Index1$ của X_t tại bản ghi này để lưu vào trường SIX_t^1 trên quan hệ R^E. Quá trình tính toán được mô tả như **Hình 2.2.**



Hình 2.2. Mô tả lược đồ tạo ***Index1*** dựa trên bộ lọc Bloom

2.3.2. Xây dựng chỉ mục Index2

Một cách trực quan, để kiểm tra chuỗi A'_s có phải là một chuỗi con của A_s = "c_1c_2 ... c_v" dựa trên hai điều kiện:

- *Điều kiện cần:* Tập ký tự phân biệt của A'_s phải xuất hiện trong A_s và tần suất của mỗi ký tự này trên A'_s phải nhỏ hơn hoặc bằng tần suất của chúng trên A_s.

- *Điều kiện đủ:* Tồn tại ít nhất một vị trí thứ i trên S sao cho "$c_i c_{i+1} \dots c_{i+|A'_s|-1}$" = A'_s.

Điều kiện đủ cho thấy vị trí tương quan giữa các ký tự của A'_s trên A_s đóng vai trò quan trọng trong việc xác định A_s có chứa A'_s hay không. Từ quan sát này, ý tưởng sơ bộ để kiểm tra mối quan hệ giữa A_s và A'_s được đưa ra với ba bước:

- ***Bước 1:*** Xác định tập ký tự phân biệt trên chuỗi A_s.

- ***Bước 2:*** Với mỗi ký tự phân biệt của A_s, sử dụng một chuỗi nhị phân cùng chiều dài với A_s để đánh dấu vị trí xuất hiện của ký tự này. Nếu ký tự xuất hiện tại vị trí thứ i trên A_s thì bit thứ i trên chuỗi nhị phân có giá trị là 1.

- ***Bước 3:*** Để kiểm tra A'_s có thuộc A_s hay không, với mỗi ký tự của A'_s ta xác định chuỗi nhị phân tương ứng đã tạo tại ***Bước 2***, sau đó thực hiện phép OR nhị phân các chuỗi này để xem chuỗi kết quả có chứa một dãy các giá trị bit 1 liền nhau hay không. Nếu chiều dài dãy bit 1 liền nhau bằng hoặc lớn hơn chiều dài của A'_s thì kết luận A_s có chứa A'_s.

Xét **Ví dụ 2.3** để kiểm chứng ý tưởng trên như sau:

Ví dụ 2.3. Cho chuỗi ký tự $A_s = "aabcdadb"$, cho chuỗi con $A'_s =$ "$bcda$", Yêu cầu kiểm tra A_s có chứa A'_s hay không?

- Chuỗi A_s có 4 ký tự phân biệt a, b, c, d. Các chuỗi Pa, Pb, Pc, Pd như sau:

$Pa = 11000100;\ Pb = 00100001;\ Pc = 00010000;\ Pd = 00001010$

- Chuỗi A'_s có 4 ký tự liên tiếp là b, c, d, a nên ta thực hiện phép OR như sau:

$$Result = Pb \vee Pc \vee Pd \vee Pa = 11111111$$

Nhận xét 2.1. Chuỗi $Result$ có chứa số lượng giá trị bit 1 liên tiếp lớn hơn 4 nên kết luận A_s có chứa A'_s. Tuy nhiên chúng ta thấy với bất kỳ hoán vị nào của A'_s như "$bacd$" hoặc "$dabc$" hoặc "$cdba$",… thì vẫn luôn cho kết luận A_s có chứa A'_s và hiển nhiên đây là một kết luận sai. Nguyên nhân là khi thực hiện toán tử $\vee$ thì việc hoán đổi vị trí các chuỗi nhị phân (Pa, Pb, Pc, Pd) không làm thay đổi kết quả $Result$, nói cách khác biểu thức này không mô tả được thứ tự xuất hiện của các ký tự trong A'_s. Để giải quyết vấn đề, dựa trên ý tưởng của Baron và cộng sự [10] luận án cải tiến cách thức để đánh dấu các bit giá trị 1 trên các chuỗi nhị phân biểu diễn vị trí ký tự của A_s như các định nghĩa và nhận xét sau đây.

Định nghĩa 2.3. Cho $A = \{a_1,\ a_2, \dots, a_z\}$ là tập các ký tự phân biệt để tạo nên các chuỗi rõ và $A_s = c_1 c_2 \dots c_v$ là một chuỗi được tạo bởi các ký tự trong tập A $(z, v \geq 1;\ c_i \in A\ với\ 1 \leq i \leq v)$. Tập $C = \{a_1, a_2 \dots, a_g\}$ là tập các ký tự phân biệt của A_s $(1 \leq g \leq z,\ g \leq v)$. Gọi $PA_s = \{Pa_1,\ Pa_2,\ \dots, Pa_g\}$ là *tập các mảng nhị phân biểu diễn vị trí ký tự của chuỗi* A_s nếu thỏa mãn công thức (2.2):

$$\begin{cases} 1 \leq i \leq v, 1 \leq j \leq g; \\ Pa_j \text{ là mảng } v\ bits \text{ biểu diễn các vị trí của ký tự } a_j \text{ trong chuỗi } A_s; \\ Pa_j[i] = 1 \text{ khi } a_j = c_{v-i+1}; \end{cases} \quad (2.2)$$

Ví dụ 2.4: Xây dựng tập các mảng nhị phân biểu diễn vị trí ký tự của chuỗi $A_s = "aabcdadb"$. Theo **Định nghĩa 2.3**, ta có:

$v = 8, \quad C = \{a, \quad b, \quad c, \quad d\}, \quad g = 4,$

$1 \leq i \leq 8,\ 1 \leq j \leq 4, a_1 = 'a',\ a_2 = 'b',\ a_3 = 'c',\ a_4 = 'd',$

$PA_s = \{Pa,\ Pb,\ Pc,\ Pd\}$ với: $Pa = 00100011$; $Pb = 10000100$

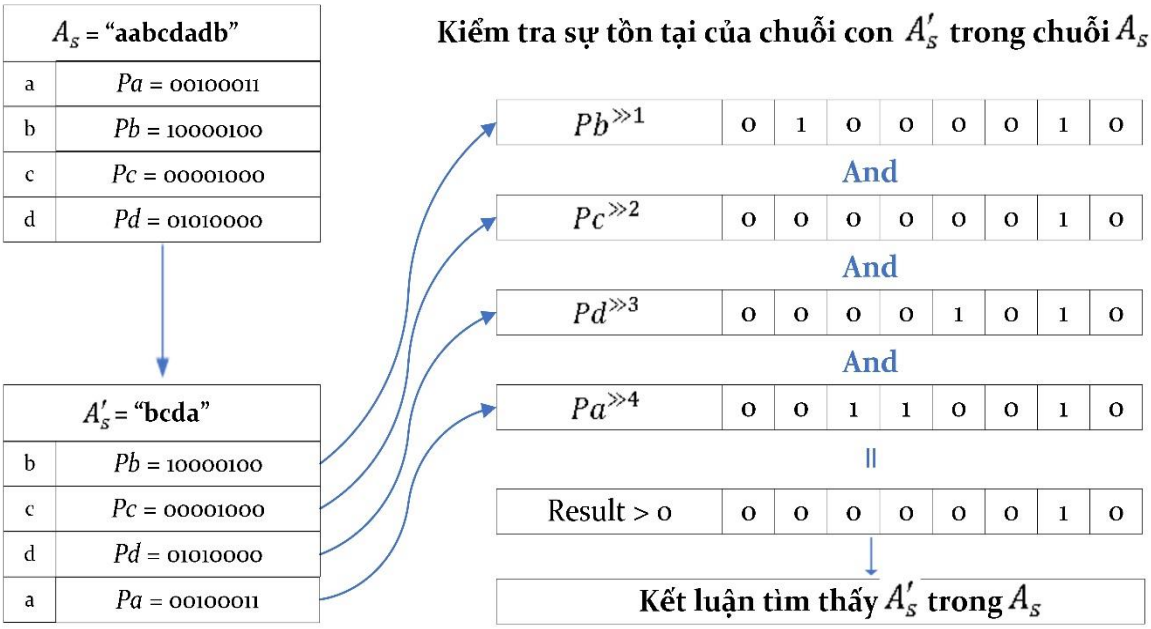
và $Pc = 00001000; Pd = 01010000$

Từ **Định nghĩa 2.3**, cho chuỗi ký tự $A'_s = "c'_1 c'_2 \dots c'_t"$ $(v \geq t \geq 1)$ khi đó với $c'_j \in C$ ta sẽ xác định được $Pc'_j \in PA_s$. Để khắc phục vấn đề trong **Nhận xét 2.1**, ta

cần đảm bảo biểu thức xác định "A'_s *có phải là chuỗi con của* A_s" phải bao hàm được *"thứ tự xuất hiện từ trái qua phải của các ký tự trong chuỗi con"*. Một cách trực quan như **Hình 2.3** cho thấy việc dịch vòng phải mỗi Pc'_j đi một lượng j bit (với j là vị trí xuất hiện của ký tự c'_j trong chuỗi con A'_s) chính là các thể hiện mối quan hệ thứ tự giữa hai ký tự c'_j và c'_{j+1} bất kỳ. Gọi ${Pc'_j}^{\gg j}$ là kết quả phép dịch vòng phải j bit của Pc'_j $(1 \leq j \leq t)$ khi đó A'_s là chuỗi con của A_s nếu thoả mãn biểu thức (2.3):

$$ {Pc'_1}^{N \gg 1} \wedge {Pc'_2}^{N \gg 2} \wedge \ldots \wedge {Pc'_j}^{N \gg j} \wedge \ldots \wedge {Pc'_t}^{N \gg t} \neq 0 \qquad (2.3) $$

Dựa trên **Ví dụ 2.4** và biểu thức (2.3), quá trình kiểm tra A_s có chứa chuỗi con $A'_s =$ "$bcda$" được mô tả như **Hình 2.3**.



Hình 2.3. Mô tả tìm chuỗi con $\boldsymbol{A'_s}$ trên tập các mảng nhị phân biểu diễn vị trí ký tự của chuỗi $\boldsymbol{A_s}$

Nhận xét 2.2: Biểu thức (2.3) cho phép kiểm tra chính xác và nhanh chóng sự tồn tại của chuỗi con trong chuỗi rõ. Theo đó, mỗi chuỗi rõ A_s được tổ chức thành cấu trúc $\{(a_1,\ Pa_1),\ (a_2,\ Pa_2),\ \ldots,\ (a_g,\ Pa_g)\}$, việc này tồn tại hai vấn đề:

- Tập $C = \{a_1, a_2, \ldots, a_g\}$ không được che giấu, kẻ gian hoàn toàn có thể khôi phục lại được A_s khi biết các phần tử $(a_i,\ Pa_i)$.

- Giả sử $\{a_1, a_2, \ldots, a_g\}$ được mã hóa và không thể phân biệt được Pa_i đang biểu diễn vị trí cho ký tự nào. Khi đó kẻ gian vẫn có thể quan sát và phân tích vị trí tương quan giữa các bit 1 trong mỗi Pa_i hoặc giữa các cặp (Pa_i, Pa_j) bất kỳ $(1 \leq i, j \leq g)$ để suy luận ra thông tin của A_s dựa trên kiến thức về vị trí ký tự trên bản rõ.

Vì vậy, luận án cần tiếp tục đưa ra giải pháp mã hóa các phần tử $(a_i,\ Pa_i)$

nhưng vẫn duy trì được các đặc điểm để hỗ trợ sự hoạt động của biểu thức (2.3).

Định nghĩa 2.4. Cho $A = \{a_1,\ a_2, \dots, a_z\}$ là tập các ký tự phân biệt sử dụng tạo nên các chuỗi rõ, cho q là một số nguyên tố lớn, tập $VA = \{Va_1,\ Va_2, \dots, Va_z\}$ được gọi là tập chứa các giá trị bí mật đại diện cho các ký tự của tập A dựa trên q nếu:

$$\begin{cases} \forall\ i,j\ (1 \le i,j \le z), Va_i \text{ và } Va_j \text{ là các số nguyên lớn hơn } q; \\ 1 \le \overline{Va_i - Va_j} \le\ q/2; \end{cases} \quad (2.4)$$

Để sinh được các phần tử của tập VA đáp ứng biểu thức (2.4), ta có thể lựa chọn một số cơ sở $V > q$, sau đó tính $Va_i = V + \varepsilon_{a_i}$ với ε_{a_i} nguyên và có giá trị ngẫu nhiên trong khoảng $[1, q/2]$ sao cho $\forall\ i,j\ (1 \le i,j \le z)$ thì $\varepsilon_{a_i} \ne \varepsilon_{a_j}$.

Định nghĩa 2.5. Cho $PA_s = \{Pa_1,\ Pa_2,\ \dots, Pa_g\}\ (1 \le g \le v)$ là tập các mảng nhị phân biểu diễn vị trí ký tự của chuỗi $A_s = "c_1c_2 \dots c_v"$. Cho $VA = \{Va_1,\ Va_2, \dots, Va_z\}$ là tập chứa các giá trị bí mật đại diện cho các ký tự của tập $A = \{a_1,\ a_2, \dots, a_z\}$ dựa trên số nguyên tố lớn q. Chỉ mục $Index2$ của chuỗi A_s là tập $IP_{A_s} = \{(I_{a_1}, IPa_1), (I_{a_2}, IPa_2), \dots, (I_{a_g}, IPa_g)\}$ với giá trị của một phần tử (I_{a_i}, IPa_i^N) được xác định như sau:

$$\begin{cases} 1 \le i \le g\ ; \\ I_{a_i} = Va_i + k_i * q \text{ với } Va_i\ \in\ VA, k_i \text{ ngẫu nhiên riêng biệt cho mỗi } I_{a_i}; \\ IPa_i =\ Pa_i^{\gg k_{A_s} + Va_i} \text{ với } k_{A_s} \text{ bí mật dùng chung cho mọi } IPa_i; \end{cases} \quad (2.5)$$

Nhận xét 2.3: Xây dựng $Index2$ là quá trình mã hóa lần lượt các cặp giá trị $(a_i,\ Pa_i)$ thành (I_{a_i}, IPa_i) dựa trên số ngẫu nhiên k_i, số bí mật k_{A_s} và tập giá trị bí mật VA. Tập IP_{A_s}có cách hoạt động như một ổ khóa kết hợp với nhiều dãy số IPa_i, giúp giải quyết một số vấn đề bảo mật nêu trong **Nhận xét 2.2**.

Tuy nhiên cách xây dựng $Index2$ theo **Định nghĩa 2.5** vẫn tồn tại một số lỗ hổng bảo mật khác mà kẻ gian có thể tận dụng khai thác như:

- Chiều dài của chuỗi bit IPa_i trong tập IP_{A_s} tuyến tính với chiều dài của chuỗi rõ A_s. Đây là lỗ hổng để kẻ gian đoán nhận được $Index2$ có thể đại diện cho những chuỗi ký tự nào có cùng chiều dài với chỉ mục.

- Quan sát số lượng phần tử của tập IP_{A_s}(số bộ (I_{a_i}, IPa_i)) giúp kẻ gian đoán được số ký tự phân biệt có trong chuỗi rõ A_s.

- Quan sát vị trí và số lần xuất hiện của bit 1 trên mỗi chuỗi IPa_i giúp kẻ gian

thống kê số lượng mỗi loại ký tự a_i (mặc dù kẻ gian không biết chính xác a_i thực tế là ký tự nào) trong chuỗi rõ. Qua đó có thể dự đoán được thông tin liên quan tới chuỗi rõ từ $Index2$ khi kẻ gian biết trước được một số đặc trưng dữ liệu của tập rõ.

Một giải pháp phù hợp được luận án sử dụng nhằm giảm thiểu các tồn tại đã nêu là ***làm nhiễu chuỗi rõ*** $\boldsymbol{A_s}$ trước khi xây dựng chỉ mục $Index2$. Nguyên tắc gây nhiễu là thay đổi số ký tự phân biệt đại diện A_s, thay đổi số lần xuất hiện của mỗi ký tự gốc trong A_s và gián tiếp gây ra sự thay đổi vị trí tương quan của các bit 1 trong mỗi chuỗi IPa_i. Cách thức tiếp cận giải quyết vấn đề như sau :

- Dựa ý tưởng của Baron và cộng sự [10], chuỗi rõ A_s được trích xuất các thông tin về tổ hợp các ký tự cùng vị trí tương đối giữa chúng để phục vụ quá trình chuyển đổi thành các bộ giá trị bộ (I_{a_i}, IPa_i). Vì vậy, quá trình gây nhiễu không được làm thay đổi nội hàm vị trí tương đối giữa các ký tự ban đầu của A_s. Nói cách khác, không chèn các ký tự hoặc chuỗi nhiễu vào trong A_s. Các chuỗi gây nhiễu được gán vào trước hoặc sau hoặc đồng thời hai phía của A_s, khi đó chuỗi mới nhận được dài hơn và đa dạng ký tự hơn, gây khó khăn cho việc xác định A_s dựa trên đếm số ký tự phân biệt. Hơn nữa, quá trình dịch vòng bit như công thức 2.5 sẽ khiến các vị trí bit 1 trong IPa_i thay đổi làm đứt gãy sự liên tục của các bit 1 đại diện cho mỗi ký tự của chuỗi A_s nguyên bản. Khi đó kẻ gian không có cơ sở để suy luận ký tự a_i nào đang được biểu diễn trong IPa_i thông qua so sánh vị trí bit 1 tương quan.

- Chuỗi gây nhiễu là sự kết hợp so le giữa các *ký tự gây nhiễu* hoặc *nhóm ký tự gây nhiễu đặc biệt* (đặc biệt ở đây được hiểu là không nằm trong bất cứ chuỗi con nào được EU yêu cầu truy vấn) với các *ký tự* hoặc *chuỗi con* của A_s. Cách làm này giải quyết được các yêu cầu:

+ Tăng số ký tự phân biệt do sự xuất hiện của các *ký tự gây nhiễu* hoặc *nhóm ký tự gây nhiễu đặc biệt*.

+ Tăng số lần xuất hiện của các ký tự gốc trong A_s do kết hợp thêm các ký tự hoặc chuỗi con của A_s vào chuỗi gây nhiễu. Việc kết hợp này cũng giúp chống lại việc trả về kết quả dương tính giả khi truy vấn một chuỗi con không hợp lệ.

+ Sau gây nhiễu, mọi chuỗi con thuộc chuỗi gốc sẽ vẫn thuộc chuỗi đã gây nhiễu. Đồng thời mọi chuỗi con không thuộc chuỗi gốc cũng sẽ không thuộc chuỗi đã gây nhiễu. Hai chuỗi rõ giống nhau thì kết quả sau gây nhiễu có thể khác nhau.

Mô tả cách thực hiện gây nhiễu chuỗi rõ A_s qua ví dụ sau:

- Xét chuỗi $A_s = "aabcdadb"$ từ các ví dụ đã nêu. Giả sử tập nhóm ký tự đặc biệt là $SpecialNoiseChar = \{*, \$, \#, \&, @, !, \hat{} \dots, \#\$\&, *\%@, \dots\}$, khi đó một phiên bản của chuỗi gây nhiễu có thể được thiết kế như sau: $NoiseString_{previous/after} = "@bcd\$d\#\$\&aabc*\%@b\hat{}d"$. Chuỗi A_s được bổ sung nhiễu qua công thức (2.6):

$$A_s = NoiseString_{previous} || A_s || NoiseString_{after} \quad (2.6)$$

Các bước xây dựng $Index2$ được mô tả trong thuật toán $BuildIndex2$ như sau:

Thuật toán 2.2. $BuildIndex2$

Input

$A_s = "c_1 c_2 \dots c_v"$: chuỗi ký tự rõ đã được gây nhiễu;

q: số nguyên tố lớn;

$VA = \{Va_1, Va_2, \dots, Va_z\}$: tập các giá trị bí mật dựa trên q, đại diện cho các ký tự phân biệt tạo lên mọi chuỗi rõ A_s;

Output

Tập IP_{A_s}: chỉ mục $Index2$ của chuỗi A_s;

Các bước thực hiện

(1) Xác định $C = \{a_1, a_2 \dots, a_g\}$ 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 *tập các mảng nhị phân biểu diễn vị trí ký tự của chuỗi* A_s;

(3) Xây dựng tập $IP_{A_s} = \{(I_{a_1}, IPa_1), (I_{a_2}, IPa_2), \dots, (I_{a_g}, IPa_g)\}$ *với* $1 \le i \le g$

$k_{A_s} \leftarrow PRF(\lambda)$ *với* λ *là tham số bảo mật*;

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

Begin

$k_i \leftarrow PRF(\lambda)$ *với* λ *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

(4) Return IP_{A_s};

Ví dụ 2.5. Xét chuỗi rõ $A_s = "aabcdadb"$ và $PA_s = \{Pa, Pb, Pc, Pd\}$ với $Pa = 00100011;\ Pb = 10000100;\ Pc = 00001000;\ Pd = 01010000$ như

trong **Ví dụ 2.4.** Yêu cầu xây dựng chỉ mục $Index2$ cho A_s.

Để thuận tiện trong quan sát, ví dụ này bỏ qua phần gây nhiễu chuỗi A_s, sau đó thực hiện tạo chỉ mục $Index2$ cho chuỗi A_s nguyên bản như sau:

- Giả sử cho số nguyên tố $q = 13$, số cơ sở $V = 20$, chọn giá trị ε_{a_i} sao cho $1 \leq \varepsilon_{a_i} \leq {}^{q}/_{2}$ $(\varepsilon_a = 3, \varepsilon_b = 1, \varepsilon_c = 5, \varepsilon_d = 4)$, số nguyên ngẫu nhiên $k_{A_s} = 5$.

- Tập $VA = \{Va, Vb, Vc, Vd\}$ là tập các giá trị bí mật đại diện cho các ký tự của A_s dựa trên q, khi đó VA được xây dựng theo **Định nghĩa 2.4** như sau: $VA = \{23, 21, 25, 24\}$.

- Xây dựng tập $IP_{A_s} = \{(I_a, IPa), (I_b, IPb), (I_c, IPc), (I_d, IPd)\}$. Cụ thể:

$I_a = 23 + 2 * 13 = 49 = 00110001_{(2)}$ với $k_i = 2$

$I_b = 21 + 1 * 13 = 34 = 00100010_{(2)}$ với $k_i = 1$

$I_c = 25 + 5 * 13 = 90 = 01011010_{(2)}$ với $k_i = 5$

$I_d = 24 + 9 * 13 = 141 = 10001101_{(2)}$ với $k_i = 9$

$IPa = Pa^{\gg k_{A_S} + Va} = Pa^{\gg 5+23} = 00110001^{\gg 28} = 00010011$

$IPb = Pb^{\gg k_{A_S} + Vb} = Pb^{\gg 5+21} = 00100010^{\gg 26} = 10001000$

$IPc = Pc^{\gg k_{A_S} + Vc} = Pc^{\gg 5+25} = 01011010^{\gg 30} = 01101001$

$IPd = Pd^{\gg k_{A_S} + Vd} = Pd^{\gg 5+24} = 10001101^{\gg 29} = 01101100$

- Chỉ mục $Index2$ được mô tả như **Bảng 2.1** dưới đây :

Bảng 2.1. Cấu trúc của chỉ mục $\boldsymbol{Index2}$

Chuỗi rõ	**Các ký tự phân biệt**	$\boldsymbol{Index2 = IP_{A_s}}$	
		$\boldsymbol{I_{a_i}}$	$\boldsymbol{IPa_i^N}$
$A_s =$ "$aabcdadb$"	a	00110001	00010011
	b	00100010	10001000
	c	01011010	01101001
	d	10001101	01101100

2.4. Truy vấn chuỗi con trên cặp chỉ mục (Index1, Index2)

2.4.1. Quy trình thực hiện truy vấn chuỗi con

Theo mô hình đề xuất như **Hình 2.1**, quá trình truy vấn chuỗi con A'_s trên CSDLQH mã hóa được thực hiện dựa trên 4 bước, gồm:

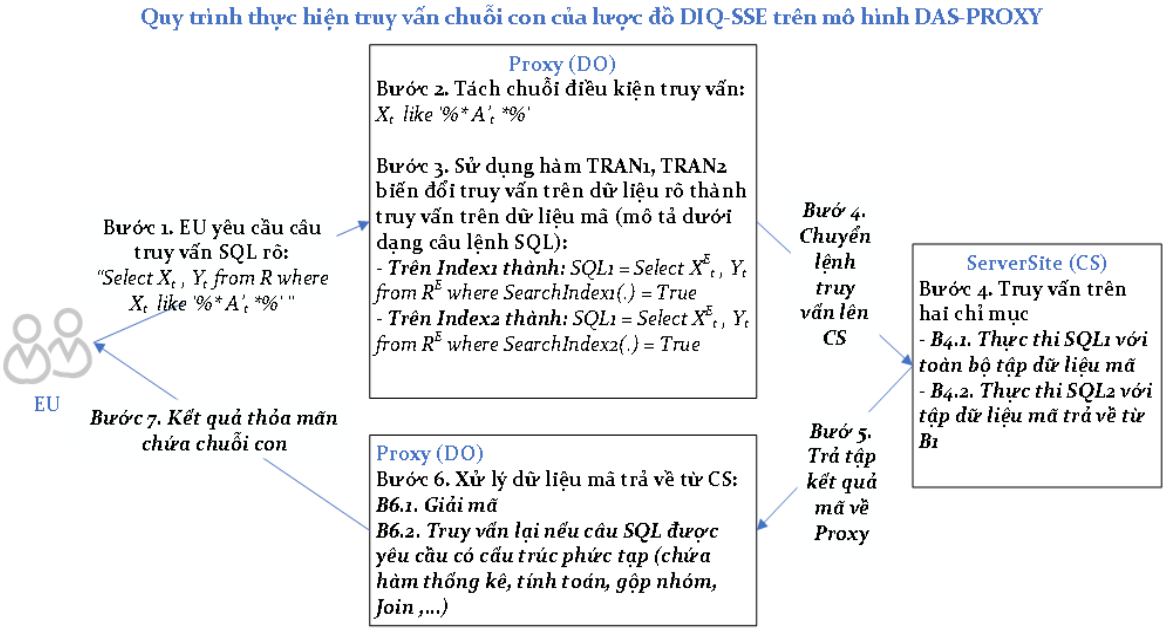
Bước 1 – Biến đổi lệnh truy vấn: Bước này mô tả **Giai đoạn 3.** $\boldsymbol{TranslateQuery}$ của lược đồ DIQ-SSE. Trọng tâm thực hiện hai thuật toán $I1 \leftarrow Tran1(A'_s, MetaData)$ và $I2 \leftarrow Tran1(A'_s, MetaData)$ tại Proxy.

Bước 2 – Truy vấn trên chỉ mục Index1: Bước này thuộc **Giai đoạn 4. *SearchIndex*** của lược đồ DIQ-SSE, mô tả quá trình thực thi thuật toán $SearchIndex1(Index1, I1)$ tại CS đối với toàn bộ tập bản ghi.

Bước 3 - Truy vấn trên chỉ mục Index2: Bước này thuộc **Giai đoạn 4. *SearchIndex*** của lược đồ DIQ-SSE, mô tả quá trình thực thi thuật toán $SearchIndex2(Index2, I2)$ trên CS đối với tập kết quả nhận được từ *Bước 2*.

Bước 4 – Giải mã dữ liệu: Bước này thuộc **Giai đoạn 5 Decrypt** của lược đồ DIQ-SSE, thực hiện giải mã kết quả nhận được từ *Bước 3*.

Dựa vào phạm vi của luận án, giả sử $A'_s = "abc"$, khi đó điều kiện truy vấn chuỗi con A'_s trên trường rõ X_t có dạng "$X_t\ LIKE\ '\%abc\%'$". Quá trình thực hiện truy vấn chuỗi con được mô tả như **Hình 2.4** và **Ví dụ 2.6**.



Hình 2.4. Quy trình thực hiện truy vấn chuỗi con trên lược đồ DIQ-SSE

Ví dụ 2.6. Cho câu truy vấn rõ $SQL = "\textbf{Select}\ X_t, Y_t\ \textbf{from}\ R\ \textbf{Where}\ X_t\ \textbf{LIKE}\ '\%A'_s\%'\ "$, khi đó quá trình biến đổi như sau:

(**1**) $SQL1 = "\textbf{Select}\ X_t^E, Y_t\ \textbf{from}\ R^E\ \textbf{Where}\ S1 = True"$

với $S1 \leftarrow SearchIndex1(Tran1(A'_s, Metadata), Index1)$.

(**2**) $SQL2 = "\textbf{Select}\ X_t^E, Y_t\ \textbf{from}\ Temp1\ \textbf{Where}\ S2 = True"$

với $Temp1$ là tập dữ liệu được trả về khi thực hiện lệnh truy vấn (**1**) và $S2 \leftarrow SearchIndex2(Tran2(A'_s, Metadata), Index2)$.

(**3**) $SQL3 = "\textbf{Select}\ X_t, Y_t\ \textbf{from}\ f^{Dec}(Temp2, sKey)$. Với $Temp2$ là tập kết quả trả về sau khi thực hiện lệnh truy vấn (**2**).

2.4.2. Biến đổi điều kiện truy vấn cho phép tìm kiếm trên chỉ mục $Index1$

Mặc dù phạm vi luận án tập trung truy vấn chuỗi con theo điều kiện "$LIKE\ '\%substring\%'$", nhưng thiết kế của $Index1$ vẫn cho phép hỗ trợ được cả các chuỗi điều kiện "$LIKE\ 'substring\%'$" và "$LIKE\ '\%substring'$".

Từ **Định nghĩa 2.1** và **2.2**, ta có $A_s^u = A_s^{uFirstCharater} \cup A_s^{uSingleCharater} \cup A_s^{uPairCharater} \cup A_s^{uLastCharater}$ và DA_s^u là *tập các cặp ký tự kết nối phân biệt theo khoảng cách u trên* A_s. Trong phần này, ta xét thêm một số phiên bản của A_s^u và DA_s^u:

$A_{s(left)}^u = A_s^{uFirstCharater} \cup A_s^{uSingleCharater} \cup A_s^{uPairCharater}$ và $DA_{s(left)}^u$ là *tập các cặp ký tự kết nối phân biệt theo khoảng cách u trên* A_s *từ bên trái.*

$A_{s(right)}^u = A_s^{uSingleCharater} \cup A_s^{uPairCharater} \cup A_s^{uLastCharater}$ và $DA_{s(right)}^u$ là *tập các cặp ký tự kết nối phân biệt theo khoảng cách u trên* A_s *từ bên phải.*

$A_{s(mid)}^u = A_s^{uSingleCharater} \cup A_s^{uPairCharater}$ và $DA_{s(mid)}^u$ là *tập các cặp ký tự kết nối phân biệt theo khoảng cách u trên* A_s *ở giữa.*

Rõ ràng $A_{s(left)}^u$, $A_{s(right)}^u$, $A_{s(mid)}^u$ là tập con của A_s^u và $DA_{s(left)}^u$, $DA_{s(right)}^u$, $DA_{s(mid)}^u$ cũng là tập con của DA_s^u. Công thức (2.1) cho thấy nếu A_s' là chuỗi con của A_s và $\forall u'(0 \le u' \le |A_s'|,\ u' \le u)$ thì $DA'_{s(left)}{}^{u'}, DA'_{s(right)}{}^{u'}, DA'_{s(mid)}{}^{u'}$ cũng là tập con của DA_s^u. Từ cơ sở này, thuật toán $Tran1(.)$ giúp biến đổi 3 biến thể của chuỗi điều kiện theo từ khoá $LIKE$ về tập $I1$ được đề xuất như sau:

Thuật toán 2.3: $Tran1$

Input

$A_s' = c_1'c_2' \dots c_t'$: chuỗi con tìm kiếm;

$Type$: giá trị phân loại bằng 1 hoặc 2 hoặc 3 tương ứng với điều kiện truy vấn sau từ khóa $LIKE$ là "$\boldsymbol{A_s'}$ %" hoặc "%$\boldsymbol{A_s'}$ " hoặc "%$\boldsymbol{A_s'}$%";

k: số lượng hàm băm;

$H\_KEY = \{hKey_1, hKey_2, \dots, hKey_k\}$: tập khóa sử dụng cho k hàm băm;

u: khoảng cách tối đa giữa các ký tự trong các cặp ký tự kết nối;

Output

Trả về tập $I1 = \{DW_1^{tran}, DW_2^{tran}, \dots, DW_{l'}^{tran}\}$ $(1 \le i \le l')$ với mỗi $DW_i^{tran} = \{p_1, p_2, \dots, p_{k'}\}$ $(1 \le j \le k',\ 1 \le\ k' \le k,\ p_j$ *là số nguyên*$)$;

Các bước thực hiện

(1) Chọn ngẫu nhiên số u', k' sao cho $0 \le\ u' \le |A_s'|,\ u' \le u,\ 1 \le k' \le k$;

(2) Chọn ngẫu nhiên k' khóa từ tập H_KEY, ta được tập khóa $H\_KEY' = \{hKey'_1, hKey'_2, \dots, hKey'_{k'}\}$, $H_{KEY}{}' \subset$ H$_KEY$;

(3) Kiểm tra giá trị tham số $type$ để xây dựng ${DA'_{s(.)}}^{u'} = \{DW_1, DW_2, \dots, DW_{l'}\}$ là *tập các cặp ký tự kết nối phân biệt theo khoảng cách u' trên A'_s* cho phù hợp (${DA'_{s(.)}}^{u'} \in \{{DA'_{s(left)}}^{u'}, {DA'_{s(right)}}^{u'}, {DA'_{s(mid)}}^{u'}\}$)

$If\ Type = 1\ then$

$${DA'_{s(.)}}^{u'} = {DA'_{s(left)}}^{u'}$$

$Else\ If\ Type = 2\ then$

$${DA'_{s(.)}}^{u'} = {DA'_{s(right)}}^{u'}$$

$Else$

$${DA'_{s(.)}}^{u'} = {DA'_{s(mid)}}^{u'}$$

(4) Với mỗi chuỗi $DW_i \in {DA'_{s(.)}}^{u'}$ với $1 \le i \le l'$, tính toán:

$DW_i^{tran} = \{p_1, p_2, \dots, p_{k'}\}$ như sau: $\{p_1 = h_1(hKey'_1, DW_i),\ p_2 = h_2(hKey'_2, DW_i), \dots,\ p_{k'} = h_{k'}(hKey'_{k'}, DW_i)\}$

(5) Trả về tập $I1 = \{DW_1^{tran}, DW_2^{tran}, \dots, DW_{l'}^{tran}\}$

Thuật toán $Tran1(.)$ cho thấy với cùng một chuỗi A'_s đầu vào thì kết quả đầu ra $I1$ không giống nhau giữa những lần thực thi khác nhau bởi sự thay đổi của các giá trị u', k' và tập H_KEY'. $Trans1(.)$ hoạt động dựa trên hàm băm, kẻ gian không thể suy luận được A'_s từ giá trị $I1$ nếu không biết $k, k', u, u', H\_KEY, H\_KEY'$.

2.4.3. Biến đổi điều kiện truy vấn cho phép tìm kiếm trên chỉ mục $Index2$

Theo **Định nghĩa 2.5**, chỉ mục $Index2$ của chuỗi A_s là tập IP_{A_s} với mỗi phần tử là một bộ (I_{a_i}, IPa_i) trong đó $I_{a_i} = Va_i + k_i * q$. Trên cơ sở này, chuỗi con A'_s cần biến đổi thành một tập giá trị với đặc tính cho phép nhận diện được các bộ (I_{a_i}, IPa_i) tương ứng với các ký tự của A'_s. Việc biến đổi được đề xuất trong thuật toán $Tran2\ (.)$

Thuật toán 2.4: $Tran2$

Input

$A'_s = c'_1 c'_2 \dots c'_t$: chuỗi con tìm kiếm;

p: số nguyên tố lớn;

$VA = \{Va_1,\ Va_2, \dots, Va_z\}$: tập giá trị bí mật đại diện cho các ký tự phân biệt của chuỗi rõ dựa trên q;

Output

Trả về tập $I2 = \{I_{c_1'}, I_{c_2'}, \dots, I_{c_t'}\}$ với $t = |A_s'|$;

Các bước thực hiện

(1) Đặt $t = |A_s'|$;

(2) Chuyển chuỗi A_s' thành tập các ký tự đơn $C = \{c_1', c_2', \dots, c_t'\}$

(3) Đặt $I2 = \left\{I_{c_1'}, I_{c_2'}, \dots, I_{c_t'}\right\}$ với $I_{c_i'} = 0$ $(1 \le i \le t)$, sau đó thực hiện:

For $i = 1$ ***to*** t

Begin

$k_i' \leftarrow PRF(\lambda)$ với λ là tham số bảo mật;

$Vc_i' \leftarrow f^v(c_i', VA)$ với f^v là hàm trả về phần tử Va_j của mảng VA sao cho $a_j = c_i'$ $(1 \le j \le z)$;

$I_{c_i'} = Vc_i' + k_i' * q$;

End

(4) Return $I2$;

Thuật toán $Trans2(.)$ chỉ ra với cùng một chuỗi A_s' đầu vào cho kết quả $I2$ không giống nhau giữa những lần thực hiện khác nhau, nguyên nhân đến từ sự thay đổi giá trị k_i'. Do $Vc_i' > q$ (**Định nghĩa 2.4**) nên từ $I_{c_i'}$ sẽ không có cơ sở để tìm ra Vc_i' ngay cả khi kẻ gian biết q.

2.4.4. Thực thi tìm kiếm trên chỉ mục $Index1$

Theo thuật toán ***BuildIndex1***, mỗi giá trị $Index1$ độ dài m bit là kết quả của việc nén *tập các cặp ký tự kết nối phân biệt theo khoảng cách u trên chuỗi* A_s bằng bộ lọc Bloom kết hợp với định danh ID của bản ghi chứa A_s.

Theo thuật toán ***Trans1***, chuỗi con A_s' được biến đổi thành tập giá trị $I1 = \{DW_1^{tran}, DW_2^{tran}, \dots, DW_{l'}^{tran}\}$ với $DW_i^{tran} = \{p_1, p_2, \dots, p_{k'}\}$ đại diện cho *một cặp ký tự kết nối phân biệt theo khoảng cách u trên chuỗi* $A_s', (1 \le j \le k', p_j$ nguyên). Rõ ràng, nếu A_s' là chuỗi con của A_s thì mỗi DW_i^{tran} cũng đại diện cho *một cặp ký tự kết nối phân biệt theo khoảng cách u trên chuỗi* A_s. Vì vậy quá trình truy vấn chính là đi xác định các DW_i^{tran} có thuộc $Index1$ hay không. Để làm điều này, tại CS ta tiếp tục áp dụng bộ lọc $Bloom$ giống như trong thuật toán ***BuildIndex1*** kết hợp với định danh ID của bản ghi để biến đổi $I1$ về một chuỗi bits BF' tương ứng có độ dài m. Cuối cùng tại mỗi bản ghi, chúng ta kiểm tra kết quả biểu thức $BF' = BF' \wedge BF$. Theo **PL2.2** nếu kết quả trả về $False$ thì chắc chắn bản

ghi có định danh ID không chứa A'_s, ngược lại bản ghi có thể chứa A'_s với tỷ lệ dương tính giả nhất định. Quá trình biến đổi $I1$ về BF' và thực hiện tìm kiếm trên $Index1$ sẽ được mô tả trong thuật toán ***SearchIndex1***, thuật toán này hỗ trợ cả ba phương thức tìm kiếm chuỗi con qua toán tử LIKE là: "$LIKE\ 'substring\%'$", "$LIKE\ '\%substring'$" và "$LIKE\ '\%substring\%'$".

Thuật toán 2.5: *SearchIndex1*

Input

$I1 = \{DW_1^{tran},\ DW_2^{tran},\ \dots,\ DW_{l'}^{tran}\}$: kết quả trả về của hàm ***Trans1***;

ID: giá trị định danh của bản ghi cần kiểm tra có chứa A'_s hay không;

BF: Giá trị $Index1$ tại bản ghi có định danh ID;

m: chiều dài chuỗi bit BF';

Output

$True$: bản ghi định có danh ID chứa chuỗi A'_s;

$False$: bản ghi định có danh ID không chứa chuỗi A'_s;

Các bước thực hiện

(1) Khởi tạo chuỗi BF' với m bits, đặt tất cả các bit của BF' bằng 0;

(2) Với mỗi $DW_i^{tran} \in I1$, $DW_i^{tran} = \{p_1,\ p_2,\ \dots,\ p_{k'}\}$ $(0 \le i \le l')$, thực hiện:

- Tính tập giá trị $\{q_1,\ q_2,\ \dots,\ q_{k'}\}$ sao cho: $q_1 = h_1(p_1, ID),\ q_2 = h_2(p_2, ID),\ \dots,\ q_{k'} = h_k(p_{k'}, ID)\}$;

- Đặt $BF'[q_j] = 1$ với $1 \le j \le k'$;

(3) Kiểm tra BF' có thuộc BF hay không:

If $BF' = BF' \wedge BF$ ***then*** $Result = True$

Else $Result = False$;

(4) Return $Result$;

2.4.5. Thực thi tìm kiếm trên chỉ mục $Index2$

Theo thuật toán ***BuildIndex2***, giá trị $Index2$ trên CS được biểu diễn qua tập $IP_{A_S} = \left\{\left(I_{a_1}, IPa_1\right), \left(I_{a_2}, IPa_2\right), \dots, \left(I_{a_g}, IPa_g\right)\right\}$. Trong khi đó thuật toán ***Trans2*** biến đổi chuỗi con A'_s thành tập $I2 = \{I_{c'_1}, I_{c'_2}, \dots, I_{c'_t}\}$ với $I_{c'_i} = Vc'_i + k'_i * q$ $(1 \le i \le t, k'_i$ là số ngẫu nhiên), sau đó $I2$ được gửi tới CS. Khi đó việc tìm kiếm A'_s trên $Index2$ sẽ dựa vào kiểm tra mối quan hệ giữa tập $I2$ và IP_{A_S} theo các bước:

Bước 1. Với mỗi phần tử $I_{c'_i} \in I2$, sẽ xác định được duy nhất một phần tử

$\left(I_{a_j}, IPa_j\right) \in IP_{A_s}$ $(1 \le i \le t, 1 \le j \le g)$ sao cho $(I_{c_i'} - I_{a_j})\ mod\ q = 0$, khi đó ta kết luận $I_{c_i'}$ và I_{a_j} cùng đại diện cho một ký tự $(c_i' = a_j)$, chứng minh như sau:

- Theo công thức (2.5): $I_{c_i'} = Vc_i' + k_i' * q;\ I_{a_i} = Va_i + k_i * q$, khi đó: $\left(I_{c_i'} - I_{a_j}\right) mod\ q = \left((Vc_i' - Va_i)\ mod\ q + \left((k_i' - k_i) * q\right)\ mod\ q\right) mod\ q = \left((Vc_i' - Va_i)\ mod\ q\right)\ mod\ q = (Vc_i' - Va_i)\ mod\ q.$

- Nếu $c_i' = a_i$ thì hiển nhiên $(I_{c_i'} - I_{a_j})\ mod\ q = 0$, với $c_i' \ne a_i$ theo **Định nghĩa 2.4** thì $1 \le \overline{Vc_i' - Va_i} \le q/2$, nên $(I_{c_i'} - I_{a_j})\ mod\ q \ne 0$.

Như vậy sau ***Bước 1*** sẽ xác định được tập $SB = \{(I_{c_1'}, IPc_1'), (I_{c_2'}, IPc_2'), \dots, (I_{c_t'}, IPc_t')\}$ với $\left(I_{c_i'}, IPc_i'\right) \in IP_{A_s}$.

Bước 2. Biến đổi các chuỗi bits $IPc_1', IPc_2', \dots, IPc_t'$ trong tập SB về trạng thái cho phép xác định chuỗi $A_s' = c_1'c_2' \dots c_t'$ có phải là chuỗi con của chuỗi $A_s = c_1c_2 \dots c_v$ thông qua thực hiện các phép toán nhị phân đơn giản giữa các chuỗi bits.

Theo công thức (2.3), để xác định A_s' là chuỗi con của A_s, thì cần kiểm tra $Pc_1'^{\gg 1} \wedge Pc_2'^{\gg 2} \wedge \dots \wedge Pc_i'^{\gg i} \wedge \dots \wedge Pc_t'^{\gg t} \ne 0$. Do đó, từ tập $SB = \{IPc_1', IPc_2', \dots, IPc_t'\}$ ta sẽ cố gắng biến đổi IPc_i' trở về Pc_i' với $1 \le i \le t$. Tuy nhiên theo công thức (2.5) thì $IPc_i' = Pc_i'^{\gg k_{A_s} + Vc_i'}$, rõ ràng không thể khôi phục Pc_i' từ IPc_i' nếu không biết các giá trị Vc_i' và k_{A_s}. Vì vậy, hướng tiếp cận khác để giải quyết vấn đề là cố định chuỗi IPc_1', sau đó với mỗi chuỗi IPc_i' $(2 \le i \le t)$ thực hiện phép quay vòng bit trái hoặc phải một độ lệch $bias$ phù hợp để trả về chuỗi IPc_i' mới sao cho $IPc_i' = Pc_i'^{\gg k_{A_s} + Vc_1'}$. Khi đó tương tự (2.3), để xác định A_s' là chuỗi con của A_s thì chỉ cần kiểm tra $IPc_1'^{\gg 1} \wedge IPc_2'^{\gg 2} \wedge \dots \wedge IPc_i'^{\gg i} \wedge \dots \wedge IPc_t'^{\gg t} \ne 0$ hay không.

Để xác định được độ lệch $bias$ giữa hai phần tử IPc_1' và IPc_i' ta xác định giá trị $\theta = (I_{c_1'} - I_{c_i'})\ mod\ q = ((Vc_1' - Vc_i')\ mod\ q + \left((k_1' - k_i') * q\right)\ mod\ q)\ mod\ q = (Vc_1' - Vc_i')\ mod\ q$. Theo (2.4) ta có: $Vc_1' > q, Vc_i' > q$ và $1 \le \overline{Vc_1' - Vc_i'} \le q/2$. Vì vậy nếu $Vc_1' - Vc_i' \ge 0$ thì $0 \le \theta < q/2$, ngược lại $q/2 \le \theta < q$. Ta xác định độ lệch $bias$ như công thức (2.7):

$$\begin{cases} bias = -1 * \left(\theta - \frac{q}{2}\right) \text{ nếu } \frac{q}{2} \le \theta < q \text{ (tương ứng } Vc_1' - Vc_i' \le 0) \\ bias = \theta \text{ nếu } 0 \le \theta < \frac{q}{2} \text{ (tương ứng } Vc_1' - Vc_i' \ge 0) \end{cases} \quad (2.7)$$

Như vậy với mỗi phần tử IPc_i' của tập SB với $(2 \le i \le t)$ ta thực hiện phép

quay vòng bít trái hoặc quay vòng bit phải một lượng $bias$ như công thức (2.8):

$$\begin{cases} IPc_i' = IPc_i'^{\ll \overline{bias}} \text{ nếu } bias < 0 \\ IPc_i' = IPc_i'^{\gg bias} \text{ nếu } bias \geq 0 \end{cases} \quad (2.8)$$

Dựa trên các ý tưởng đã trình bày, thuật toán ***SearchIndex2*** hỗ trợ truy vấn chuỗi con qua chuỗi điều kiện "$LIKE\ '\%\ A_s'\ \%'$" được mô tả như sau:

Thuật toán 2.6: *SearchIndex2*

Input

$I2 = \{I_{c_1'}, I_{c_2'}, \dots, I_{c_t'}\}$: kết quả thực thi ***Trans2***(.) với chuỗi con A_s' làm đầu vào;

$IP_{A_s} = \left\{(I_{a_1}, IPa_1), (I_{a_2}, IPa_2), \dots, \left(I_{a_g}, IPa_g\right)\right\}$: giá trị $Index2$ đại diện cho A_s;

Output

$True$: nếu A_s' là chuỗi con của A_s;

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

Các bước thực hiện

(1) Xây dựng tập $SB = \{(I_{c_1'}, IPc_1'), (I_{c_2'}, IPc_2'), \dots, (I_{c_t'}, IPc_t')\}$ với mỗi phần tử $\left(I_{c_i'}, IPc_i'\right)$ sẽ tương ứng với một phần tử $\left(I_{a_j}, IPa_j\right) \in IP_{A_s}$ $(1 \leq i \leq t, 1 \leq j \leq v)$. Mã giả xây dựng SB như sau:

$SB = \{\emptyset\}$;

$For\ i = 1\ to\ t$

$Begin$

$For\ j = 1\ to\ g$

$Begin$

$if\ \left(I_{c_i'} - I_{a_j}\right) mod\ q = 0\ then$

$Begin$

$I_{c_i'} = I_{a_j}$;

$IPc_i' = IPa_j$;

$SB = SB.Add((I_{c_i'}, IPc_i'))$;

$Break$;

End

End

End

(2) Biến đổi tập SB bằng cách giữ cố định IPc_1' và với mỗi IPc_i' thực quay vòng bít

trái hoặc quay vòng bít phải một lượng $bias, (2 \leq i \leq t)$. Mã giả như sau:

$For\ i = 2\ to\ t$

$Begin$

$\theta = (I_{c_1'} - I_{c_i'})\ mod\ q$

$if\ \frac{q}{2} \leq \theta < q\ then$

$Begin$

$bias = -1 * \left(\theta - \frac{q}{2}\right)$

$IPc_i'^N = IPc_i'^{N \ll \overline{bias}}$

End

$Else\ if\ 0 \leq \theta < \frac{q}{2}$

$Begin$

$bias = \theta$

$IPc_i'^N = IPc_i'^{N \gg bias}$

End

End

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

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

$Result = True$

$Else$

$Result = False$

(4) Return $Result$

2.5. Phân tích bảo mật của lược đồ DIQ-SSE

Mục này góp phần đánh giá ***tính hiệu quả của lược đồ*** DIQ-SSE qua tiêu chí ***bảo mật***. Ta đã biết Proxy là khu vực tin cậy giúp DO quản lý và thực hiện an toàn các nhiệm vụ: lưu trữ $metadata$, xây dựng chỉ mục, vận hành cửa sập (biến đổi lệnh truy vấn), mã và giải mã dữ liệu. Vì vậy vấn đề bảo mật trong DIQ-SSE sẽ tập trung vào khía cạnh chống rò rỉ thông tin bản rõ trước các dạng thức tấn công phổ biến trên CS vào chỉ mục và mẫu truy vấn.

Trong quan hệ R^E trên CS, trường X_t^E lưu giá trị mã hóa của trường dữ liệu nhạy cảm X_t dựa trên các thuật toán mã hóa truyền thống như AES, DES, Blowfish [37],

[13], [98]. Vì vậy chương này sẽ không đề cập tới vấn đề bảo mật của X_t^E chừng nào các thuật toán mã hóa và khóa vẫn an toàn. Luận án cũng không nghiên cứu về bảo mật đường truyền hay xác thực - toàn vẹn dữ liệu, mà sẽ tập trung vào phân tích ***khả năng chống rò rỉ của các chỉ mục và mẫu truy vấn*** trên CS, cụ thể như sau:

- Vấn đề bảo mật chỉ mục: Tại CS, kẻ gian có cách nào để suy luận được thông tin về dữ liệu rõ lưu trong X_t khi quan sát $Index1, Index2$ hay không?

- Vấn đề bảo mật mẫu truy vấn: Tại CS, kẻ gian có cách nào suy luận được thông tin về chuỗi con truy vấn A'_s khi quan sát $I1, I2$ nhận được từ Proxy hay không?

Trong mục **2.1**, lược đồ DIQ-SSE kế thừa thiết kế từ các lược đồ SSE dựa trên chỉ mục mù đáp ứng bảo mật IND-CKA2 đã được phân tích và chứng minh trong các công trình [43], [33], [76], [135], [6], [91], [9], [101], [132], [10]. Vì vậy, dựa vào các định nghĩa bảo mật của lược đồ SSE trong **PHỤ LỤC 1** và các đặc tính tương đồng trong thiết kế của lược đồ đề xuất với những lược đồ trước đó thì DIQ-SSE được coi đáp ứng mô hình bảo mật IND-CKA2. Do đó, trong mục này luận án sẽ chỉ bổ sung, làm rõ thêm một số luận điểm, phân tích về khả năng chống rò rỉ thông tin của các chỉ mục và mẫu truy vấn trước một số kịch bản tấn công phổ biến.

2.5.1. Phân tích bảo mật cặp chỉ mục

Các thuật toán $BuildIndex1, BuildIndex2$ được thiết kế như hàm một chiều, vì vậy nếu kẻ gian chỉ biết một giá trị $Index1$ hoặc $Index2$ thì không thể suy diễn được nội dung bản rõ tương ứng. Tuy nhiên thực tế kẻ gian có thể tiếp cận một số thông tin về: đặc điểm tập dữ liệu rõ, một số tập con của tập dữ liệu rõ (do tính phổ biến của dữ liệu rõ) và một số tiết lộ khác tại CS (do CS là môi trường "bán trung thực") như: tập các giá trị $Index1, Index2$ lưu trong R^E và tập giá trị $I1, I2$ sưu tập được sau nhiều lần truy vấn. Từ đó kẻ gian thực hiện phương pháp *"Tấn công dựa trên thống kê"* để suy diễn thông tin bản rõ. Phương pháp này sử dụng các số liệu thống kê về tần suất xuất hiện của các đặc tính trong chuỗi rõ ban đầu và tần suất xuất hiện của các đặc tính trong chỉ mục ($Index1$, $Index2$) để tìm sự tương đồng. Xét trường hợp thuận lợi nhất cho kẻ gian, luận án giả định họ biết trước:

Thứ nhất: biết phương pháp xây dựng $Index1$, $Index2$ từ các chuỗi rõ (nhưng không biết các $metadata$ sử dụng trong quy trình này như H_KEY, u, u', VA);

Thứ hai: biết trước tập các chuỗi rõ P và biết tập mẫu PI_1 (chứa các giá trị $Index1$), PI_2 (chứa các giá trị $Index2$) tương ứng với một tập con P' của P (nhưng

kẻ gian không biết P'). Giả sử tập P có 10000 chuỗi rõ, tập PI_1, PI_2 đều có 100 phần tử tương ứng với tập $P' \subset P$ (kẻ gian không biết 100 chuỗi rõ của P').

2.5.1.1. Phân tích bảo mật của $Index1$

Xét chuỗi ký tự rõ A_s, chỉ mục $Index1$ được xây dựng dựa trên sử dụng bộ lọc Bloom để nén các đặc tính của chuỗi A_s (chính là tập DA_s^u) về m bits. Theo **Định nghĩa 2.1**, nếu $u = 0, u' = 0$ khi đó tập DA_s^u sẽ chỉ bao gồm 2 cặp ký tự đầu cuối, các ký tự đơn và các cặp ký tự liền kề.

Định nghĩa 2.6. Cho $A = \{a_1, a_2, \dots, a_z\}$ là tập z ký tự phân biệt tạo nên các chuỗi rõ ($z \geq 1$). Tập D là *tập các cặp ký tự phân biệt kết nối theo khoảng cách u ($0 \leq u \leq z-2$) trên tập ký tự A* được định nghĩa như sau:

$D = D^{FirtCharater} \cup D^{SingleCharater} \cup D^{PairCharater} \cup D^{LastCharater}$ trong đó:

$$\begin{cases} D^{FirtCharater} = \{a_i || *\} \text{ với } 1 \leq i \leq z \\ D^{SingleCharater} = \{a_i\} \text{ với } 1 \leq i \leq z \\ D^{PairCharater} = \{(a_i || k || a_j)\} \text{ với } 1 \leq i, j \leq z;\ 0 \leq k \leq u \\ D^{LastCharater} = \{* || a_i\} \text{ với } 1 \leq i \leq z \end{cases} \quad (2.9)$$

Khi đó cho chuỗi $A_s = "c_1 c_2 \dots c_v"$ được định nghĩa trên tập A ($c_i \in A, 1 \leq i \leq v$), từ công thức (2.1) và (2.9) với cùng tham số u ta có DA_s^u là tập con của tập D. Theo **định nghĩa 2.1** và **định nghĩa 2.6** mọi chuỗi A_s đều được hình thành dựa trên z ký tự của tập A với $z > 1$, điều này khẳng định bài toán tìm chuỗi con phù hợp trên mọi bảng mã (bao gồm bảng mã Unicode). Vì vậy mọi ký tự thuộc bất cứ ngôn ngữ nào đều được hỗ trợ bởi lược đồ DIQ-SSE. Để thuận tiện cho việc phân tích các kịch bản, không mất tính tổng quát ta giả sử các ký tự sử dụng trong tập A là 26 chữ cái tiếng Anh và $u = 0$, khi đó $|D| = 26 + 26 + 26^2 + 26 = 754$ phần tử và $|DA_s^u| \leq 754$. Theo thuật toán $\boldsymbol{BuildIndex1}$ nếu xét trường hợp số hàm băm $k = 1$ thì mỗi phần tử của tập DA_s^u sẽ tương ứng với một bit giá trị 1 trên $Index1$. Với **giả định $\boldsymbol{u = 0, u' = 0, k = 1}$**, lúc này có hai kịch bản thuận lợi để kẻ gian suy luận thông tin A_s từ $Index1$ như sau:

a) Kịch bản 1

Thống kê tần suất mỗi phần tử của D trên tập P và thống kê tần suất của các bit 1 tại mỗi vị trí trên các $Index1$, rồi sau đó so sánh chúng với nhau để phát hiện mỗi vị trí bit 1 của $Index1$ sẽ ứng với đặc tính nào trong A_s.

Gọi $num_1, num_2, \dots, num_{754}$ lần lượt là số lượng chuỗi tương ứng trong tập P có chứa phần tử $D[j]$ với $1 \le j \le 754$. Khi đó tập mô tả tần suất xuất hiện các phần tử của tập D trên tập P sẽ được tính như sau:

$$F_{D/P} = (\frac{num_1}{10000}, \frac{num_2}{10000}, \dots, \frac{num_{754}}{10000}) \quad (2.10)$$

Gọi $num'_1, num'_2, \dots, num'_m$ lần lượt là số lượng phần tử trong tập PI_1 có giá trị bit tại vị trí thứ i $(1 \le i \le m)$ bằng 1, tập tần suất xuất hiện bit 1 trên $Index1$ là:

$$F_{Index1/PI_1} = (\frac{num'_1}{100}, \frac{num'_2}{100}, \dots, \frac{num'_m}{100}) \quad (2.11)$$

So sánh giữa F_{Index1/PI_1} và $F_{D/P}$ để tìm ra một số cặp phần tử có độ lớn tương đương ($F_{Index1/PI_1}[i] \approx F_{D/P}[j]$), kẻ gian có thể suy luận được tại vị trí bit thứ i trên chuỗi $Index1$ có thể tương ứng với một phần tử thứ j trong tập D. Tuy nhiên trong thực tế các tham số $\boldsymbol{u}, \boldsymbol{u'}, \boldsymbol{k}$ đều lớn hơn $\mathbf{1}$ và hoàn toàn được giữ bí mật, dẫn đến số lượng phần tử trong tập D không đoán được và số lượng bit đại diện cho một phần tử của tập DA_s^u sẽ nhiều hơn một bit. Đặc biệt trong thuật toán $\boldsymbol{BuildIndex1}$ còn có sự kết hợp với định danh bản ghi ID và bổ sung thêm một lượng bit ngẫu nhiên trong quá trình tạo $Index1$, vì vậy kẻ gian sẽ không thể xác định được $F_{D/P}$ và F_{Index1/PI_1}.

b) Kịch bản 2

Phương pháp tiếp cận dựa trên quan sát: chiều dài của chuỗi rõ sẽ tỷ lệ thuận với số lượng phần tử của tập DA_s^u, dẫn đến chiều dài chuỗi rõ có xu hướng tỷ lệ thuận với số lượng bit 1 trên chuỗi $Index1$. Kẻ gian thực hiện phân loại và đếm số lượng các chuỗi rõ có độ dài bất thường trong tập P, đếm số lượng các $Index1$ có số lượng bit 1 bất thường, sau đó so sánh để suy luận được một số chuỗi rõ từ $Index1$. Tuy nhiên theo thuật toán $\boldsymbol{BuildIndex1}$, khả năng sử dụng kịch bản tấn công này rất thấp, bởi các chuỗi $Index1$ đều có chung độ dài m, mỗi chuỗi $Index1$ đều được bổ sung thêm một lượng bit ngẫu nhiên nên việc thống kê số lượng bit 1 trong mỗi $Index1$ sẽ không chính xác. Hơn nữa, các tham số u và định danh ID cũng làm cho số lượng bit 1 và phân phối vị trí của chúng trong $Index1$ khác biệt giữa các bản ghi.

2.5.1.2. Phân tích bảo mật của Index2

Một phương pháp tấn công dựa trên thống kê đơn giản mà kẻ gian có thể áp dụng là so sánh độ dài của mỗi $Index2$ trong tập PI_2 (chính là độ dài của một chuỗi IPa_i bất kỳ trong chỉ mục này) với lần lượt độ dài của các chuỗi rõ trong tập P để tìm ra mối liên quan giữa chỉ mục với chuỗi rõ. Tuy nhiên, việc các chuỗi rõ được làm

nhiễu trước khi đưa vào thuật toán $BuildIndex2$ làm cho độ dài của chỉ mục và chuỗi rõ không còn quan hệ tuyến tính. Vì vậy, ta sẽ phân tích hai kịch bản tấn công khác có tính thực tiễn hơn như sau:

a) Kịch bản 1

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 trong mỗi chỉ mục $Index2$ thuộc tập PI_2. Sau đó thực hiện các so sánh và suy diễn như sau:

Gọi $N_1,\ N_2, \dots,\ N_{10000}$ lần lượt là các tập hợp mô phỏng số lượng các ký tự phân biệt trong mỗi chuỗi 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 chuỗi 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 của mỗi chỉ mục $Index2$ trong tập PI_2, với $N'_j = \{num'_1,\ num'_2,\ \dots\ ,\ num'_g\}$ và num'_i là số lượng bit 1 trong IPa_i thuộc $Index2$ thứ j của tập PI_2 $(1 \le j \le 100, 1 \le i \le g)$.

Với mỗi tập N'_j kẻ gian sẽ so sánh với từng tập N_j để tìm ra các điểm tương đồng và suy luận thông tin chuỗi ký tự rõ tương ứng với $Index2$. Tuy nhiên, trong thuật toán $BuildIndex2$ sử dụng chuỗi A_s đã được gây nhiễu làm đầu vào, vì vậy số bit 1 trên mỗi chuỗi IPa_i không phản ánh được số lượng thực của ký tự a_i trong chuỗi A_s nguyên bản. Số lượng chuỗi IPa_i trong mỗi chỉ mục cũng không phản ánh được số ký tự phân biệt trong chuỗi rõ. Bên cạnh đó cũng không có sự tương quan về số lượng bit 1 giữa hai chuỗi IPa_i và IPa_j bất kỳ của cùng một chỉ mục $Index2$ (ví dụ: một chuỗi A_s có x ký tự a_i và y ký tự a_j, khi đó trong $Index2$ sẽ thống kê được x' bit 1 trong chuỗi IPa_i và y' bit 1 trong chuỗi IPa_j. Kẻ gian không tìm thấy mối liên quan nào giữa cặp (x, y) và (x', y')). Vì vậy, kịch bản tấn công này trên $Index2$ không thể giúp kẻ gian đạt được mục đích.

b) Kịch bản 2

Cơ sở của kịch bản tấn công này dựa trên việc kẻ gian tính khoảng cách giữa các vị trí xuất hiện của cùng một ký tự trong chuỗi rõ, sau đó so sánh tìm sự tương đồng với khoảng cách các bit 1 trên từng chuỗi IPa_j trong mỗi $Index2$. Qua đó có thể đoán nhận được chuỗi IPa_j đại diện cho ký tự nào.

Cho chuỗi ký tự $A_s = "c_1c_2 \dots c_v"$ và $C = \{a_1, a_2 \dots, a_g\}$ là tập các ký tự phân biệt của A_s (với $1 \le i \le v, 1 \le j \le g,\ g \le v$ sẽ luôn tồn tại $c_i = a_j$), khi đó với mỗi

ký tự a_j ta có một mảng POS_{a_j} mô tả danh sách vị trí tương ứng của a_j trong chuỗi A_s như sau: $POS_{a_j} = \{i\} \mid c_i = a_j$.

Định nghĩa 2.7. Tập $DC_{A_s} = \{D_{a_1}, D_{a_2}, \dots, D_{a_g}\}$ được gọi là *tập khoảng cách đặc trưng của chuỗi* A_s nếu mỗi phần tử D_{a_j} $(1 \le j \le g)$ cũng là một tập hợp các số nguyên và thoả mãn điều kiện sau:

$$D_{a_j} = \left\{POS_{a_j}[k+1] - POS_{a_j}[k]\right\} với\ 1 \le k \le |POS_{a_j}| - 1 \quad (2.12)$$

Trong $Index2$, với mỗi chuỗi bit IPa_j ta mô tả tất cả các vị trí bit có giá trị 1 vào một mảng $POS_{Bit(a_j)}$ như sau: $POS_{Bit(a_j)} = \{i\} \mid IPa_j[i] = 1$.

Định nghĩa 2.8. Tập $DC_{Index2} = \{D_{Bit(a_1)}, D_{Bit(a_2)}, \dots, D_{Bit(a_g)}\}$ được gọi là *tập khoảng cách đặc trưng của chuỗi* IPa_j nếu mỗi phần tử $D_{Bit(a_j)}$ $(1 \le j \le v)$ thoả mãn điều kiện sau:

$$D_{Bit(a_j)} = \left\{POS_{Bit(a_j)}[k+1] - POS_{Bit(a_j)}[k]\right\} \quad (2.13)$$
$$với\ 1 \le k \le |POS_{Bit(a_j)}| - 1$$

Dựa trên **Định nghĩa 2.7** và **2.8** kẻ gian xây dựng các tập DC_{A_s} cho tất cả chuỗi rõ của tập P, các tập DC_{Index2} cho tất cả chỉ mục $Index2$ của tập PI_2. Sau đó thực hiện đối sánh các cặp (DC_{A_s}, DC_{Index2}), quá trình này thực chất là việc so sánh mức độ tương đồng của từng cặp phần tử $(D_{a_j}, D_{Bit(a_j)})$. Nếu tập giao $D_{a_j} \cap D_{Bit(a_j)}$ có số phần tử càng lớn thì kẻ gian có thể suy diễn hai tập D_{a_j} và $D_{Bit(a_j)}$ cùng đại diện cho một ký tự a_j. Nếu phần lớn các cặp $(D_{a_j}, D_{Bit(a_j)})$ đều đại diện chung cho một ký tự nào đó thì có thể suy luận $Index2$ liên quan mật thiết với A_s. Tuy nhiên, thuật toán $BuildIndex2$ có hai yếu tố khiến kịch bản này khó thực hiện là việc gây nhiễu chuỗi A_s trước khi thực thi $BuildIndex2$ và việc thực hiện dịch vòng phải chuỗi Pa_j đi một lượng bí mật $k_{A_s} + Va_j$. Việc gây nhiễu làm vị trí tương đối giữa các ký tự khác nhau và giữa các ký tự giống nhau đều bị ảnh hưởng, còn việc dịch vòng phải tác động đến khoảng cách vị trí của các bit 1 giữa các chuỗi Pa_j. Do đó tập $D_{Bit(a_j)}$ sẽ không phản ánh đúng khoảng cách giữa các vị trí của ký tự a_j trong chuỗi ký tự rõ A_s.

2.5.2. Phân tích bảo mật mẫu truy vấn

Cơ sở phương pháp tấn công này như sau: Do CS được coi là môi trường "bán trung thực", nên kẻ gian có thể thống kê tần suất các giá trị $I1, I2$ (các mẫu truy vấn)

nhận từ Proxy cùng với tập kết quả trả về tương ứng sau khi truy vấn trên CS. Dựa vào nhu cầu tìm kiếm các chuỗi con trong thực tiễn, kẻ gian sẽ đoán được $I1, I2$ có thể liên quan tới chuỗi con nào. Qua đó suy diễn các bản ghi mã hoá được trả về sau truy vấn chứa chuỗi con nào.

Giả sử trong thực tế người dùng có số lần tìm kiếm chuỗi con "Hà Nội" vượt trội hơn tất cả các cụm từ khác, khi đó tại CS kẻ gian thống kê tần suất xuất hiện của các giá trị ($I1, I2$), nếu tìm ra được một cặp ($I1, I2$) có tần suất xuất hiện tương đồng với số lần yêu cầu truy vấn chuỗi con "Hà Nội" thì có thể dự đoán $I1, I2$ chính là đại diện cho chuỗi "Hà Nội". Qua đó kẻ gian suy luận được tập dữ liệu mã trả về sau khi truy vấn sẽ tương ứng với tập dữ liệu rõ có chứa chuỗi con "Hà Nội".

Thuật toán $Tran1$ sử dụng ngẫu nhiên k' khóa từ tập H_KEY ($1 \leq k' \leq k$) để làm tham số cho k' hàm băm sử dụng trong quá trình biến đổi mẫu truy vấn A'_s thành tập $I1$. Tương tự thuật toán $Tran2$ sử dụng số ngẫu nhiên k'_i để biến đổi mỗi ký tự c'_i của mẫu truy vấn $A'_s = c'_1 c'_2 \dots c'_t$ thành giá trị đại diện $I_{c'_i}$ ($1 \leq i \leq t$) và cuối cùng trả về tập $I2$. Kết quả là với cùng một mẫu truy vấn A'_s mỗi lần được yêu cầu sẽ gửi tới CS một cặp ($I1, I2$) khác nhau. Vì vậy, ta có thể kết luận tấn công mẫu truy vấn là không khả thi trong lược đồ DIQ-SSE.

2.6. Thực nghiệm và đánh giá hiệu năng của lược đồ DIQ-SSE

Qua các phân tích về tình hình nghiên cứu chung trong lĩnh vực SE (tại **phần mở đầu** và **mục 1.9**) đã cho thấy các khó khăn khi cố gắng thực hiện mục tiêu đa dạng truy vấn. Các nghiên cứu và giải pháp SE ứng dụng gần đây [140], [141], [55] cũng giới hạn đáng kể sự đang dạng truy vấn để không đánh đổi chi phí hiệu năng, bảo mật và sự phức tạp khi triển khai lược đồ. Nói cách khác, tập lệnh truy vấn trên mã còn hạn chế hơn rất nhiều so với tập lệnh truy vấn trên dữ liệu rõ, việc tối ưu hoặc đưa ra các cách tiếp cận mới nhằm nâng cao bảo mật, hiệu năng đối với thực thi một số lệnh truy vấn phổ biến trên dữ liệu mã vẫn là ưu tiên hiện nay trong lĩnh vực SE.

Mặc dù vậy, việc đặt vấn đề thử nghiệm khả năng đáp ứng của lược đồ DIQ-SSE với một câu lệnh truy vấn phức tạp (gồm kết nối nhiều quan hệ, nhiều điều kiện truy vấn trên nhiều trường thông tin và cả các hàm thống kê) như dưới đây là vẫn có thể thực hiện được.

$SQL\_Complex$ = "**Select** $R.X_t$, $SUM(R.Y_t)$, $R'.X'_t$, $R'.Y'_t$
from R *Inner Join* R' *ON* $R.ID = R'.ID$

$$\boldsymbol{Where}\ X_t\ \boldsymbol{LIKE}\ '\%\boldsymbol{A'_{s1}}\%'\ And\ X'_t\ \boldsymbol{LIKE}\ '\%\boldsymbol{A'_{s2}}\%'$$

Trong giới hạn của thiết kế lược đồ DIQ-SSE, giả thiết các trường ID trên hai quan hệ cho phép ánh xạ trên môi trường CSDLQH mã hóa, khi đó câu lệnh $SQL\_Complex$ sẽ phải được biến đổi thành:

(**1**) $SQL_{Complex1} =$ "$\boldsymbol{Select}\ R^E.X_t^E, R^E.Y_t^E, R'^E.X_t'^E, R'^E.Y_t'^E, R^E.Index2,$
$R'^E.Index2\ \boldsymbol{from}\ R^E\ Inner\ Join\ R'^E\ ON\ R.ID = R'.ID$
$\boldsymbol{Where}\ S1 = True\ And\ S2 = True$"
với $S1 \leftarrow SearchIndex1(Tran1(A'_{s1}, Metadata), R^E.Index1)$ và
$S2 \leftarrow SearchIndex1(Tran1(A'_{s2}, Metadata), R'^E.Index1)$

(**2**) $SQL_{Complex2} =$ "$\boldsymbol{Select}\ X_t^E, Y_t^E, X_t'^E, Y_t'^E\ \boldsymbol{from}\ Temp1$
$\boldsymbol{Where}\ S3 = True\ And\ S4 = True$"
với $Temp1$ là tập dữ liệu được trả về khi thực hiện lệnh truy vấn (**1**) và
$S3 \leftarrow SearchIndex2(Tran2(A'_{s1}, Metadata), R^E.Index2)$ và
$S4 \leftarrow SearchIndex2(Tran2(A'_{s2}, Metadata), R'^E.Index2)$.

(**3**) $SQL_{Complex3} =$ "$\boldsymbol{Select}\ X_t, SUM(Y_t), X'_t, Y'_t\ \boldsymbol{from}\ f^{Dec}(Temp2, sKey)$
với $Temp2$ là tập kết quả trả về sau khi thực hiện lệnh truy vấn (**2**).

Biến đổi trên cho thấy DIQ-SSE thực thi một câu lệnh phức tạp là có cơ sở, tuy nhiên phải bổ sung bước truy vấn lại câu lệnh $SQL_{Complex3}$, thậm chí việc thực hiện sẽ phức tạp hơn nếu các trường ID trên hai quan hệ đều được mã phi tất định. Các phát sinh này được coi là một hạn chế (được mô tả trong bước 6.2 tại **Hình 2.4**).

Các đóng góp trong chương này tập trung đề xuất lý thuyết lõi giúp truy vấn hiệu quả chuỗi con trên một trường dữ liệu trong một quan hệ mã hóa, làm cơ sở cho các mở rộng dạng thức truy vấn phức tạp khác trong hướng phát triển tiếp theo của luận án. Vì vậy, luận án xây dựng các kịch bản, dữ liệu, mẫu lệnh thực nghiệm để hướng tới làm rõ sự hiệu quả của việc truy vấn trên $Index1$ và $Index2$ thông qua sử dụng các dạng truy vấn đơn giản như **Bảng 2.2**, nhằm loại bỏ hoàn toàn các chi phí phát sinh không liên quan tới chỉ mục như khi truy vấn các câu lệnh phức tạp, qua đó giúp đánh giá các mô hình, lý thuyết đề xuất được chính xác và công bằng.

2.6.1. Chuẩn bị điều kiện thực nghiệm

Mục này góp phần ***đánh giá tính hiệu quả của lược đồ*** DIQ-SSE qua tiêu chí ***hiệu năng*** truy vấn dựa trên một số thực nghiệm và phân tích về: tỷ lệ lọc và tỷ lệ lỗi của tập dữ liệu mã sau mỗi lần truy vấn trên $Index1$ và $Index2$; thời gian thực thi

truy vấn từ thời điểm EU yêu cầu cho tới khi nhận được kết quả giải mã. Thiết kế lược đồ DIQ-SSE ngoài tăng cường khả năng chống rò rỉ thông tin còn hướng tới cải thiện tỷ lệ lọc và tốc độ truy vấn trên $Index1$ và $Index2$, vì vậy việc sử dụng các đo lường về tỷ lệ lọc, tỷ lệ lỗi và thời gian truy vấn cho đánh giá là hoàn toàn phù hợp. Các thông số này cũng thường xuyên được sử dụng trong các nghiên cứu SE như [120], [76], [126], [139].

Định nghĩa 2.9. Cho N_R là số lượng bản ghi của quan hệ R, giả sử N_{Result} là số lượng bản ghi trả về khi truy vấn trên dữ liệu rõ, N_{Index1} là số bản ghi mã trả về sau khi truy vấn trên $Index1$. Định nghĩa tỷ lệ lọc (FIR_1) và tỷ lệ lỗi (FAR_1) khi truy vấn trên $Index1$ qua công thức sau:

$$FIR_1 = \frac{N_R - N_{Index1}}{N_R - N_{Result}} \qquad FAR_1 = \frac{N_{Index1} - N_{Result}}{N_{Index1}} \tag{2.14}$$

Định nghĩa 2.10. Cho N_R là số lượng bản ghi của quan hệ R, giả sử N_{Result} là số lượng bản ghi trả về khi truy vấn trên dữ liệu rõ, N_{Index1} là số bản ghi mã trả về sau khi truy vấn trên $Index1$, N_{Index2} là số bản ghi mã trả về sau khi truy vấn trên $Index2$. Định nghĩa tỷ lệ lọc (FIR_2) và tỷ lệ lỗi (FAR_2) khi truy vấn trên $Index2$ qua công thức sau:

$$FIR_2 = \frac{N_{Index1} - N_{Index2}}{N_{Index1} - N_{Result}} \qquad FAR_2 = \frac{N_{Index2} - N_{Result}}{N_{Index2}} \tag{2.15}$$

Hai định nghĩa trên cho thấy tỷ lệ lọc phản ánh khả năng loại bỏ các bản ghi không thoả mãn điều kiện truy vấn, trong khi đó tỷ lệ lỗi phản ánh lượng bản ghi dương tính giả trong kết quả trả về sau khi truy vấn trên CSDLQH mã hóa.

Định nghĩa 2.11. Thời gian thực thi truy vấn (ET) trên lược đồ được tính bằng tổng thời gian thực hiện 4 bước truy vấn chuỗi con. Ký hiệu là $ET = \sum_1^4 ET_i$ với ET_i là thời gian thực thi của bước thứ i.

Dữ liệu thử nghiệm: TPC-H [142] là một CSDLQH chuyên dụng được sử dụng trong các đánh giá, thử nghiệm giải pháp về xử lý dữ liệu trên các hệ quản trị CSDLQH hiện nay (xem **PL2.1**). Luận án sử dụng bảng LINEITEM trong CSDL này làm đối tượng thực hiện các thử nghiệm và sử dụng công cụ DBGen để sinh dữ liệu rõ cho bảng LINEITEM với một triệu bản ghi. Xây dựng CSDLQH mã hóa TPC_H_CS trong đó có bảng LINEITEM_CS chứa các trường dữ liệu mã hóa và dữ liệu chỉ mục tương ứng với trường dữ liệu rõ L_COMMENT trong bảng LINEITEM. Thực hiện chèn dữ liệu mã hóa (được sinh ra bởi các thuật toán tiêu chuẩn như AES [37]), dữ liệu chỉ mục $Index1, Index2$ tương ứng của cột L_COMMENT trong bảng

LINEITEM vào các cột (L_COMMENT_E, L_COMMENT_SI1, L_COMMENT_SI2) của bảng LINEITEM_CS.

Trong mỗi kịch bản thử nghiệm, luận án sử dụng một số điều kiện truy vấn như **Bảng 2.2**, với S_1, S_2, S_3 là các chuỗi con có số ký tự tương ứng là 2, 8, 16.

Bảng 2.2. Các điều kiện truy vấn sử dụng trong thử nghiệm lược đồ DIQ-SSE

Query	Condition	Query	Condition
Q1	L_COMMENT like '%S_1%'	QL1	L_COMMENT like '%S_1'
Q2	L_COMMENT like '%S_2%'	QL2	L_COMMENT like '%S_2'
Q3	L_COMMENT like '%S_3%'	QL3	L_COMMENT like '%S_3'

Trong phạm vi của luận án, các chỉ mục $Index1, Index2$ được thiết kế để hỗ trợ truy vấn chuỗi điều kiện "$LIKE\ '\%substring\%'$" , đảm bảo kết quả trả về sau khi truy vấn tại CS không có bản ghi dư thừa. Tuy nhiên phần thực nghiệm vẫn bổ sung thêm các biến thể khác (QL1, QL2, QL3) của chuỗi điều kiện, khi đó kết quả trả về sau truy vấn trên $Index2$ có thể tồn tại bản ghi dương tính giả nhưng tựu chung lược đồ vẫn giải quyết được thông qua quá trình giải mã và truy vấn lại tại Proxy, điều này càng cho thấy tính khả thi của lược đồ khi hỗ trợ đa dạng truy vấn trong thực tiễn.

Sử dụng câu truy vấn "$\boldsymbol{Select}\ *\ \boldsymbol{From}\ LINEITEM\ \boldsymbol{Where}\ Q$", với mỗi điều kiện Q = {Q1, Q2, Q3, QL1, QL2, QL3} luận án sử dụng 30 mẫu chuỗi con S_1, S_2, S_3 tương ứng để thử nghiệm. Các giá trị FIR, FAR, ET tại mỗi kịch bản sẽ được tính bằng trung bình cộng các kết quả trả về sau khi truy vấn 30 mẫu này.

Thiết bị sử dụng mô phỏng các khu vực của mô hình DAS-PROXY như sau:

ClientSite: Intel Core i5 4310U 2.0Ghz CPU, 8GB DDRAM, cài Windows 10.

ProxyServer: Intel Xeon W-1350 3.3Ghz CPU và 16GB DDRAM, cài Windows Server 2019 và MS SQL Server 2016.

CloudServer: Intel Xeon E5-2620 2.0Ghz CPU và 64GB DDRAM, cài Windows Server 2019 và MS SQL Server 2016.

Các tham số ảnh hưởng tới hiệu quả truy vấn trên các chỉ mục là N_R, u, l, m, k, trong đó u tác động tới số lượng l phần tử của tập DA_s^u. Theo công thức tối ưu (PL2.2), giá trị của k sẽ được xác định để tối thiểu hóa số bản ghi dương tính giả trả về sau truy vấn trên $Index1$. Luận án lựa chọn các tham số cho thử nghiệm như **Bảng 2.3**.

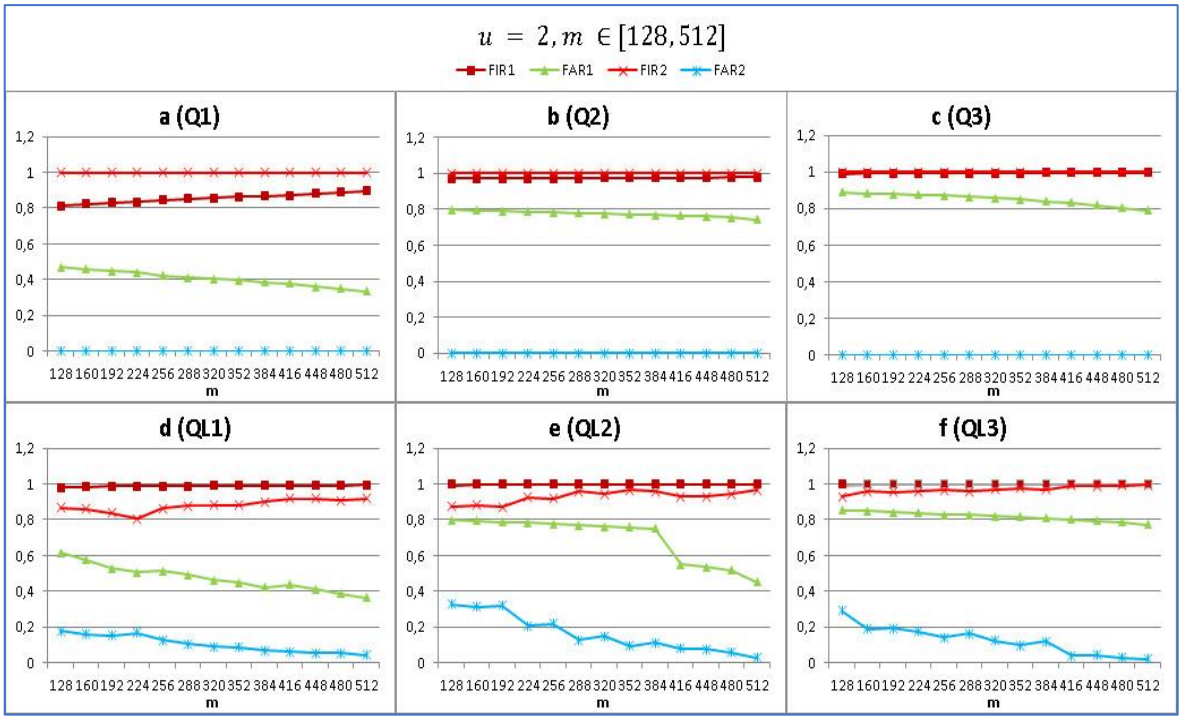
Bảng 2.3. Một số bộ giá trị $(\boldsymbol{u}, \boldsymbol{l}, \boldsymbol{m}, \boldsymbol{k})$ sử dụng trong thực nghiệm với $\boldsymbol{N_R = 1000000}$

$\boldsymbol{u}$	$\boldsymbol{l}$ trung bình	$(\boldsymbol{m}, \boldsymbol{k})$
1	105	(128,1), (160,1), (192, 1), (224,2), (256,2), (288,2), (320,2), (352,3), (384,3), (416,3), (448,3), (480,3), (512,3)
2	154	(128,1), (160,1), (192, 1), (224,1), (256,1), (288,1), (320,1), (352,2), (384,2), (416,2), (448,2), (480,2), (512,2)
3	201	(128,1), (160,1), (192, 1), (224,1), (256,1), (288,1), (320,1), (352,1), (384,1), (416,1), (448,2), (480,2), (512,2)
4	245	(128,1), (160,1), (192, 1), (224,1), (256,1), (288,1), (320,1), (352,1), (384,1), (416,1), (448,1), (480,1), (512,1)

2.6.2. Thực nghiệm và đánh giá hiệu năng tìm kiếm

2.6.2.1. Kịch bản thực nghiệm 1

Kịch bản 1: Chọn $N_R = 1000000$, cố định $u = 2$, với m tăng dần từ 128 đến 512 lần lượt thực hiện các truy vấn Q1, Q2, Q3, QL1, QL2, QL3 và tính toán FIR_1, FAR_1, FIR_2, FAR_2. Kết quả thử nghiệm gồm 6 biểu đồ như **Hình 2.5**.



Hình 2.5. Biểu đồ FIR1, FAR1, FIR2, FAR2 của các lệnh truy vấn trong kịch bản $\boldsymbol{N_R = 1000000, u = 2, m \in [128, 512]}$

(Trục hoành biểu thị giá trị m, trục tung biểu thị giá trị tỉ lệ lọc/tỉ lệ lỗi)

Từ các biểu đồ **a-c** (Q1, Q2, Q3) trong **Hình 2.5** ta có các nhận xét sau:

1) Giá trị FIR2 luôn đạt giá trị 1 và FAR2 luôn là hằng số 0 cho thấy $Index2$ hỗ trợ tìm kiếm chính xác chuỗi con theo điều kiện "$LIKE$ '$\%substring\%$'".

2) Đường biểu diễn FIR1 có xu hướng tiệm cận giá trị 1 khi m tăng dần từ 128

đến 512 bit. Theo thuật toán $BuildIndex1$, mỗi chuỗi rõ A_s được phân tích thành tập $DA_s^u = \{DW_1, DW_2, \dots, DW_l\}$, sau đó mỗi phần tử DW_i sẽ được đánh dấu bởi k bit 1 trên $Index1$ có độ dài m, việc tồn tại xác suất để hai DW_i và DW_j có chung vị trí bit 1 là nguyên nhân chính gây ra dư thừa dữ liệu khi truy vấn trên $Index1$. Do đó khi m tăng thì sẽ giảm xác suất trùng lặp bit 1 và giảm số bản ghi dương tính giả.

3) Tổng thể vị trí đường FIR1 của chuỗi con dài sẽ nằm cao hơn so với đường FIR1 của chuỗi con ngắn. Nguyên nhân số bản ghi chứa chuỗi con dài sẽ ít hơn số bản ghi chứa chuỗi con ngắn dẫn đến tăng tỷ lệ lọc khi truy vấn trên $Index1$.

4) Đường FAR1 hạ thấp dần khi m tăng từ 128 đến 512 bit. Cách lý giải tương tự như ý thứ 2 trong nhận xét này cho việc FIR1 tiệm cận giá trị 1 khi m tăng.

5) Đường FAR1 của chuỗi con dài nằm cao hơn so với đường FAR1 của chuỗi con ngắn. Điều này mâu thuẫn với ý thứ 3 trong nhận xét, bởi lý thuyết khi vị trí đường FIR1 càng cao thì đường FAR1 càng nằm thấp. Lý giải như sau: theo thuật toán $Tran1$, chuỗi con càng dài thì số phần tử của tập $I1$ càng lớn dẫn đến chuỗi BF' trong $SearchIndex1$ chứa nhiều bit 1 hơn, hệ quả biểu thức $BF' \wedge BF$ có xu hướng trả về $True$ hơn và tạo ra nhiều bản ghi dương tính giả, gây lên tăng FAR1.

Nhận xét chung các lệnh truy vấn Q1, Q2, Q3 trong kịch bản này đều đạt tỷ lệ lọc, tỷ lệ lỗi lý tưởng trên $Index2$ và tốt trên $Index1$. Khi truy vấn trên $Index1$, nếu m tăng thì FIR1 và FAR1 đều biến thiên theo hướng tích cực. Nếu chiều dài chuỗi con tăng thì FIR1 tiếp tục thay đổi tích cực, trong khi đó FAR1 có xu hướng thay đổi xấu hơn nhưng tổng thể vẫn hiệu quả khi xét trên tập bản ghi lớn ($N_R = 1000000$).

Tiếp tục nhận xét các biểu đồ **d-f** (QL1, QL2, QL3) trong **Hình 2.5** như sau:

1) Xu hướng các đường FIR1, FAR1 cơ bản như trong các biểu đồ **a-c**. Tuy nhiên đường FIR1 có tiệm cận giá trị 1 hơn, nguyên nhân do điều kiện "$LIKE$ '$\%Substring\%$'" làm giới hạn đáng kể các bản ghi thoả mãn, dẫn đến số bản ghi không hợp lệ bị loại bỏ nhiều hơn, giúp tỷ lệ lọc tốt hơn.

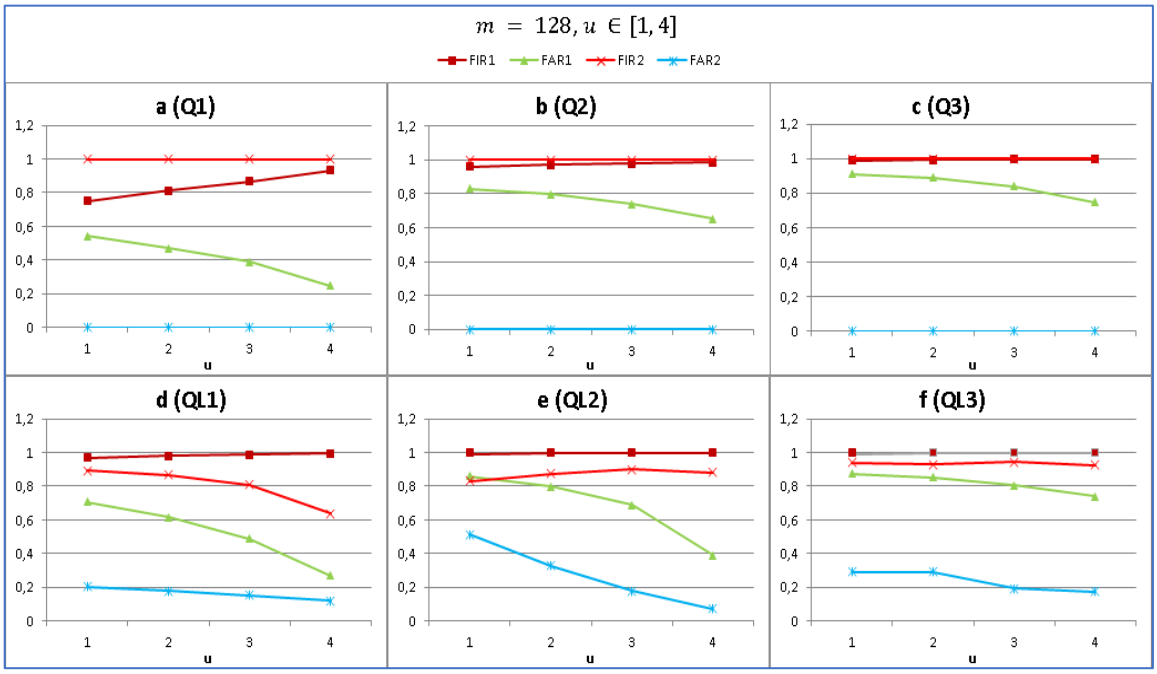
2) Giá trị FIR2 và FAR2 không còn là các hằng số lý tưởng, cho thấy kết quả truy vấn trên $Index2$ trả về bản ghi dương tính giả với các lệnh QL1, QL2, QL3. Nguyên nhân khi truy vấn trên $Index2$ với điều kiện "$LIKE$ '$\%Substring\%$'" sẽ trả về tập bản ghi chứa chuỗi con "$Substring$" nằm ở bất kỳ vị trí nào của chuỗi rõ. Tuy nhiên FIR2 và FAR2 vẫn biến đổi tích cực khi m hay độ dài chuỗi con tăng dần.

3) Mặc dù các lệnh QL1, QL2, QL3 không thuộc phạm vi luận án, nhưng thử

nghiệm vẫn cho thấy sự hiệu quả của lược đồ DIQ-SSE với các lệnh mở rộng này.

2.6.2.2. Kịch bản thực nghiệm 2

Kịch bản 2: Chọn $N_R = 1000000$, cố định $m = 128$, với u thay đổi ($u \in \{1, 2, 3, 4\}$) lần lượt thực hiện các truy vấn Q1, Q2, Q3, QL1, QL2, QL3 và tính toán FIR_1, FAR_1, FIR_2, FAR_2. Kết quả mô tả bởi các biểu đồ trong **Hình 2.6.**



Hình 2.6. Biểu đồ FIR1, FAR1, FIR2, FAR2 của các lệnh truy vấn trong kịch bản $N_R = 1000000, m = 128, u \in [1, 4]$

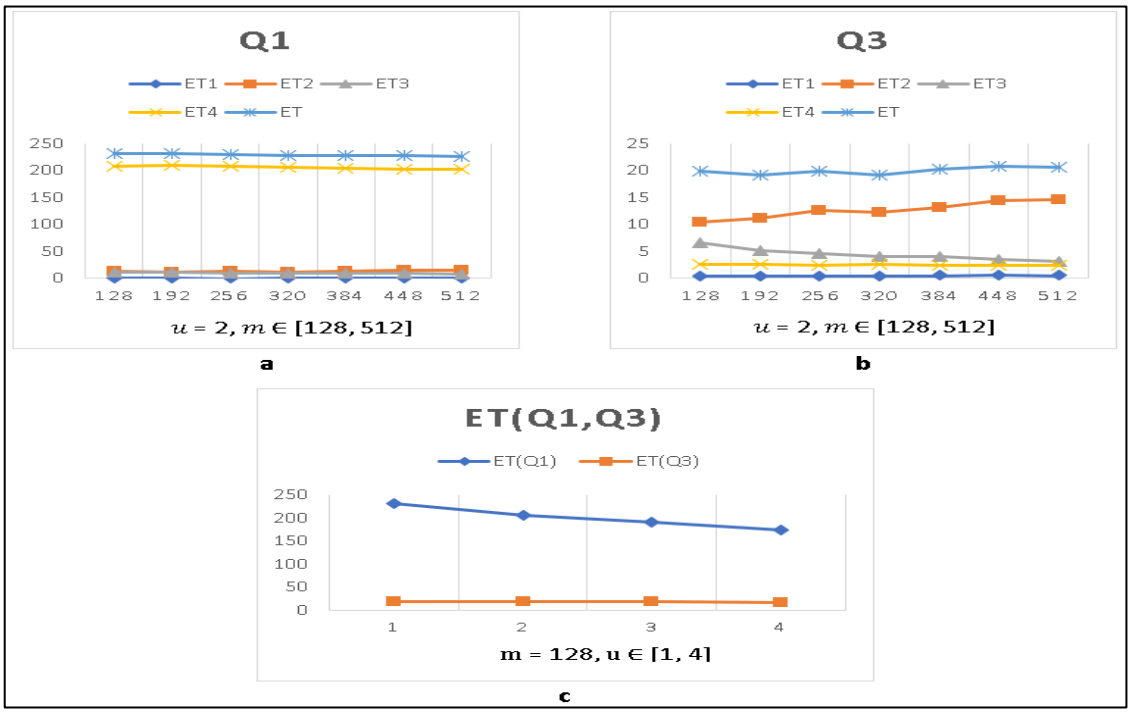
(Trục hoành biểu thị giá trị u, trục tung biểu thị giá trị tỉ lệ lọc/tỉ lệ lỗi)

Quan sát các biểu đồ **a-f** trong **Hình 2.6** cho thấy các đường FIR1, FAR1 đều thay đổi theo hướng tích cực (trong đó FIR1 tiệm cận 1) khi u tăng dần từ 1 tới 4. Nguyên nhân u tăng thì số lượng phần tử của tập DA_s^u cũng sẽ tăng và $Index1$ sẽ biểu diễn đầy đủ hơn các đặc tính của A_s, kết quả là FIR1, FAR1 được cải thiện. Đối với FIR2, FAR2 luôn ở trạng thái lý tưởng trong các biểu đồ **a-c**, khẳng định khả năng hỗ trợ truy vấn chính xác đối với điều kiện *"LIKE '%Substring%'"* của $Index2$. Trong khi đó biểu đồ **d-f** cho thấy các đường FIR2, FAR2 tiệm cận về giá trị tối ưu khi u hoặc độ dài của chuỗi con tăng, do xuất hiện bản ghi dương tính giả trong tập kết quả trả về khi truy vấn với điều kiện *"LIKE '%Substring'"*.

Đối với cả hai kịch bản, các biểu đồ đều chỉ ra FIR2, FAR2 tốt hơn FIR1, FAR1 với bất kỳ m, u, cho thấy tỷ lệ lọc lý tưởng khi truy vấn trên $Index2$. Vậy tại sao luận án đề xuất truy vấn tuần tự trên cặp $Index1$, $Index2$ thay vì chỉ thực thi duy nhất truy vấn trên $Index2$ cho toàn bộ CSDL? Ta sẽ xem xét về các thử nghiệm đo thời gian thực thi truy vấn trong **mục 2.6.3** để hiểu rõ nguyên nhân.

2.6.3. Thực nghiệm và đánh giá thời gian truy vấn

Trong kịch bản thực nghiệm và đánh giá thời gian truy vấn của lược đồ, luận án sử dụng $u \in [1,4]$, $m \in [128,512]$ và lựa chọn các câu lệnh Q1, Q3 để đo thời gian thực thi cho 4 bước (các bước mô tả trong **Mục 2.4.1**). Các giá trị thời gian ET_i (trong **Định nghĩa 2.11**) được biểu diễn qua biểu đồ như **Hình 2.7**.



Hình 2.7. Đánh giá thời gian truy vấn Q1, Q3 với $\boldsymbol{u \in [1,4]}$, $\boldsymbol{m \in [128,512]}$

(Trục hoành biểu thị giá trị m/u, trục tung biểu thị số giây thực hiện)

Biểu đồ **b** trong **Hình 2.7** chỉ ra thời gian thực thi 4 bước đối với câu lệnh Q3. Bước 1 cho thấy ET_1 chiếm không đáng kể do chỉ thực hiện hàm $Tran1(.)$, $Tran2(.)$ để biến đổi điều kiện truy vấn. Trong bước 2, mặc dù thuật toán $SearchIndex1$ có độ phức tạp rất thấp nhưng giá trị ET_2 lại lớn nhất trong các bước do phải duyệt trên 1000000 bản ghi. Tại bước 3 giá trị ET_3 tương đối nhỏ do chỉ thực hiện thuật toán $SearchIndex2$ (mặc dù độ phức tạp lớn hơn $SearchIndex1$) cho hơn 2000 bản ghi kết quả nhận được từ bước 2. Cuối cùng là thời gian ET_4 tại bước 4 cho quá trình giải mã tập dữ liệu (xấp xỉ 1000 bản ghi) nhận được từ bước 3. Với quy trình truy vấn tuần tự qua hai chỉ mục, thời gian truy vấn được cải thiện đáng kể nhờ vào việc lọc bỏ phần lớn bản ghi không hợp lệ qua thuật toán $SearchIndex1$ có hiệu năng cao. Qua đó cho thấy mối liên hệ chặt chẽ của chỉ mục $Index1$ và $Index2$ trong quá trình truy vấn. Việc truy vấn riêng lẻ (chỉ thực thi $Tran1$ và $SearchIndex1$ hoặc $Tran2$ và $SearchIndex2$) là hoàn toàn có thể nhưng phải trả giá bởi vấn đề dư

thừa dữ liệu khi chỉ truy vấn trên $Index1$ hoặc chi phí tìm kiếm quá cao khi chỉ truy vấn trên $Index2$. Các hàm $Tran1(.),\ Tran2(.)$ với chi phí thấp phù hợp với việc thực hiện tại Proxy cấu hình thấp để cung cấp các điều kiện truy vấn phù hợp trên dữ liệu mã. Vì vậy, việc thực thi đồng thời hai hàm này như mô tả trong **mục 2.1** và **Hình 2.4** là điều kiện tiên quyết và bắt buộc, đem lại hiệu quả truy vấn cao trên CSDLQH mã hoá chứa số bản ghi lớn.

Biểu đồ **a** trong **Hình 2.7** mô tả thời gian thực thi các bước đối với câu lệnh Q1, cho thấy độ lớn $ET_1,\ ET_2,\ ET_3$ là không đáng kể so với ET_4. Nguyên nhân do chuỗi con S_1 (mô tả trong **Bảng 2.3**) chỉ có 2 ký tự nên số lượng bản ghi có chứa S_1 là rất lớn, vì vậy mặc dù tỷ lệ lọc tại giai đoạn 2, 3 hiệu quả thì số lượng bản ghi mã phải trả về vẫn rất lớn và nó làm tăng chi phí giải mã tại bước 4.

Biểu đồ **c** trong **Hình 2.7** chỉ ra khi u tăng từ 1 đến 4 thì thời gian thực thi ET của mô hình đề xuất cũng được cải thiện. Đặc biệt với các truy vấn có kết quả trả về lớn thì việc tăng u sẽ góp phần giảm tỷ lệ lỗi FAR1 rất hiệu quả, giúp giảm tổng chi phí cho các bước truy vấn trên $Index2$ và giải mã.

2.7. So sánh hiệu quả lược đồ DIQ-SSE với một số nghiên cứu

Tính hiệu quả của lược đồ SE được mô tả trong mục **Phạm vi luận án** và trong **Nhận xét 1.2**, tập trung vào: *bảo mật, hiệu năng và tính khả thi trong triển khai*. **Mục 2.5**, luận án đã phân tích kỹ về tính *bảo mật* của DIQ-SSE thông qua khả năng chống rò rỉ thông tin chỉ mục và mẫu truy vấn. Tiếp theo, **Mục 2.6** tập trung thực nghiệm và phân tích *hiệu năng* lược đồ qua: các thông số FIR, FAR và thời gian truy vấn. Như vậy, *nội dung hai mục này vẫn mang **tính chủ quan** khi mới dừng lại ở việc **tự phân tích - nhận xét** về bảo mật và hiệu năng dựa trên các cơ sở thiết kế lý thuyết của lược đồ, cũng như **tự ghi nhận** kết quả thu được từ thực nghiệm trên dữ liệu mẫu đối với một số **kịch bản giới hạn***. Để có đánh giá thuyết phục và khách quan hơn về tính hiệu quả (dựa trên các tiêu chí trong **Bảng 1.3**) của lược đồ DIQ-SSE, luận án *tiếp tục so sánh với một số nghiên cứu liên quan tiêu biểu gần đây* như **Bảng 2.4**. Một số phân tích về ưu điểm và tồn tại của những nghiên cứu này đã được trình bày tích hợp trong **mục 1.9.1**, còn trong phần này luận án sẽ tiếp tục làm rõ hơn tính hiệu quả các giải pháp trên góc nhìn tổng hợp và đối sánh.

Đánh giá chung: Các công trình liệt kê đều tiêu biểu, uy tín và liên quan tới lược đồ SE hỗ trợ truy vấn chuỗi con, trong đó nhiều tài liệu cập nhật được tình hình

trong 5 năm trở lại đây. Cấu trúc dữ liệu hỗ trợ xây dựng chỉ mục trong lược đồ DIQ-SSE có tính mới so với các công trình trong **Bảng 2.4** khi đề xuất sử dụng tập DA_s^u và PA_s trong thiết kế cặp chỉ mục $Index1, Index2$.

Đánh giá về chi phí lưu trữ: Chi phí lưu trữ trong tất cả công trình liệt kê đều được đánh giá dựa trên tổng số bản ghi chứa chỉ mục trong CSDL, hàm ý đánh giá chi phí lưu trữ chỉ mục có tuyến tính với kích cỡ CSDL hay không. Các lược đồ đề xuất tại nghiên cứu [30], [67], [38], [102], [79], [131] đều có chi phí lưu trữ cỡ $O(c.N)$ với $c > 2$, trong đó các công trình [50], [83], [80], [130], [138] lại giới thiệu các lược đồ với chi phí lưu trữ tốt hơn là $O(N)$. Đối với lược đồ DIQ-SSE, do có tới 2 chỉ mục là $Index1$ và $Index2$ nên chi phí lưu trữ cũng tăng gấp đôi và đạt $O(2N)$. Vì vậy khi so sánh tương quan với các công trình đã nêu thì vấn đề chi phí lưu trữ của DIQ-SSE chỉ đạt mức độ trung bình và không phải là ưu thế của lược đồ.

Đánh giá về chi phí quy trình tìm kiếm: bao gồm các chi phí về: biến đổi lệnh truy vấn tại cửa sập; thực hiện các thuật toán tiền xử lý hoặc thuật toán tham chiếu tới các từ điển từ khóa tại Proxy/Client; truyền tải dữ liệu; tính toán – tìm kiếm trên CS và giải mã; truy vấn lại dữ liệu (nếu có) tại vùng an toàn. Chi tiết đối sánh các tiêu chí này giữa lược đồ DIQ-SSE với các công trình khác như sau:

Đánh giá các công trình liên quan:

- *Về chi phí tính toán và truy vấn trên CS*, xét trên độ phức tạp tìm kiếm trên mỗi tài liệu/bản ghi tại CS thì giải pháp [80] là thấp nhất $O(l_{substr})$, nhưng lại phát sinh chi phí không nhỏ cho truyền tải dữ liệu và tính toán phía EU (Client). Trong khi đó các giải pháp còn lại đều có chi phí từ $O(l_{substr} + occ)$ [79] cho đến $O(l_{substr}{}^2 + occ)$ [102], [130] và có thể yêu cầu thêm tài nguyên đáng kể cho Client hoặc việc truyền tải dữ liệu [30], [79], [130].

- *Về chi phí tính toán tại cửa sập (Trapdoor)*, một số ít giải pháp [50], [38], [138] được ghi nhận tiêu tốn một phần tài nguyên nhỏ (cỡ $O(l_{substr})$ đến $O(occ.l_{substr})$) cho quá trình biến đổi từ khoá, lệnh truy vấn về dạng tìm kiếm được trên dữ liệu mã.

- *Về chi phí tính toán phía Client*, phát sinh này được đánh giá là không phù hợp trong bối cảnh triển khai dữ liệu trên các hạ tầng thuê ngoài, nơi mà ưu tiên tính toán tập trung tại CS và giảm thiểu tối đa áp lực trên thiết bị người dùng (đặc biệt là các thiết bị di động có cấu hình thấp) cũng như chống nguy cơ rò rỉ thông tin tại Client. Nguyên nhân do một số giải pháp không sử dụng PROXY hoặc một số chức năng về

xử lý dữ liệu (bao gồm: giải mã, tham chiếu từ điển từ khóa, kiểm tra và trả lời các yêu cầu của CS tại mỗi vòng giao tiếp dữ liệu) lại được ủy quyền cho thiết bị người dùng cuối như trong các tài liệu [30], [79], [80].

- *Về chi phí truyền tải*, các nghiên cứu [30], [83], [79], [80], [130], [138] đều cho thấy phải tiêu tốn một lượng tài nguyên thường xuyên cho quá trình gửi nhận dữ liệu giữa Client/Proxy tới CS và ngược lại. Các chi phí này đánh giá dựa trên kích thước trung bình mỗi thông điệp được trao đổi tại mỗi vòng giao tiếp để phục vụ tính toán hoặc truy vấn (không phải là kích thước tập dữ liệu kết quả được trả về). Một số giải pháp [67], [102], [131] có nhiều hơn một vòng giao tiếp nhưng kích thước thông điệp trao đổi nhỏ cũng không quá ảnh hưởng đến chi phí truyền tải.

- *Về số vòng giao tiếp dữ liệu* (xem diễn giải trong **Bảng 1.3**), một số giải pháp trong [80], [38], [138] chỉ mất một vòng, trong khi đó các nghiên cứu khác đòi hỏi nhiều hơn. Nguyên nhân do quá trình truy vấn trên chỉ mục hoặc bản mã tại CS đòi hỏi nhiều bước tính toán mật nên cần gửi về Client hoặc Proxy thực hiện. Việc này tác động một phần đến chi phí truyền tải và áp lực cho các thiết bị ngoài CS.

- *Về bảo mật,* hầu hết các giải pháp những năm gần đây đều hỗ trợ chống rò rỉ thông tin đồng thời về chỉ mục tìm kiếm và mẫu truy vấn [79], [80], [130], [138].

Đánh giá về lược đồ DIQ-SSE khi so với các công trình liên quan:

- Các chi phí giao tiếp cũng như tính toán tại Proxy hay Client của DIQ-SSE là không đáng kể. Chi phí tìm kiếm chủ yếu tập trung tại CS qua việc thực thi hai thuật toán $SearchIndex1$ và $SearchIndex2$. Trong đó, thuật toán $SearchIndex1$ cần tới k lần tính toán trước khi thực hiện phép AndBitwise nên được đánh giá độ phức tạp cỡ $O(k)$. Thuật toán $SearchIndex2$ cần tới hai lần duyệt các ký tự chuỗi con nên độ phức tạp xấp xỉ cỡ $O(2.l_{substr})$. Thuật toán $SearchIndex1$ có chi phí rất tốt được áp dụng trên N bản ghi, trong khi đó thuật toán $SearchIndex2$ có độ phức tạp tính toán trung bình (không nổi trội hơn so với các công trình khác) nhưng lại có lợi thế do chỉ thực hiện trên tập dữ liệu nhỏ được trả về bởi $SearchIndex1$. Nếu gọi N_{QI1} là số bản ghi trả về sau truy vấn trên $Index1$ thì độ phức tạp tìm kiếm trên CS của DIQ-SSE cho toàn bộ CSDL là $O(N.k + N_{QI1}.2.l_{substr})$. Nếu cũng xét độ phức tạp tìm kiếm trên CS của các công trình khác trên phạm vi tập N văn bản / bản ghi thì càng thấy rõ ưu thế của lược đồ DIQ-SSE khi truy vấn liên tiếp trên hai chỉ mục.

- Xét về số vòng giao tiếp thì DIQ-SSE cũng cho thấy ưu thế khi chỉ mất 1 vòng.

Về khả năng chống rò rỉ chỉ mục và mẫu truy vấn cũng đạt yêu cầu. Qua đó cho thấy tính hiệu quả của lược đồ trên phương diện hiệu năng và bảo mật.

Bảng 2.4. So sánh hiệu quả lược đồ DIQ-SSE với một số nghiên cứu

($\boldsymbol{N}$ là số tài liệu hoặc bản ghi trong CSDL; $\boldsymbol{l_{str}}$ là chiều dài trung bình chuỗi dữ liệu rõ trong các tài liệu/bản ghi; $\boldsymbol{l_{substr}}$ là chiều dài chuỗi con; $\boldsymbol{l_d}$ là độ lớn của từ điển chứa tất cả các từ khoá được trích xuất trong CSDL; $\boldsymbol{occ}$ là số lần xuất hiện của chuỗi con/từ khoá trong tài liệu/bản ghi có độ dài l_{str}; $\boldsymbol{n_k}$ số lượng từ khóa được phép tìm kiếm; $\boldsymbol{\epsilon}$ là tỷ lệ dương tính giả; $\boldsymbol{avg}$ là số lượng trung bình các vị trí mà tại đó k-gram khớp với chuỗi con; $\boldsymbol{k}$ là số hàm băm; $\boldsymbol{kp}$ tham số tác động số lượng token được sinh ra, kp nhỏ thì số token lớn gây tăng chi phí truy vấn và ngược lại; $\boldsymbol{z}$ là số ký tự phân biệt trong N ký tự; $\boldsymbol{\lambda}$ là tham số bảo mật; $\boldsymbol{C_{dummy}}$ chi phí xử lý tăng thêm khi bổ sung gây nhiễu; $\boldsymbol{C_{KeywordSearch}}$ Chi phí thuật toán tìm kiếm từ khoá trên dữ liệu mã)

TT	Năm - Nghiên cứu	Cấu trúc dữ liệu hỗ trợ xây dựng chỉ mục	Chi phí lưu trữ	Chi phí cho quy trình tìm kiếm	Số vòng giao tiếp	Rò rỉ chỉ mục	Rò rỉ mẫu truy vấn
1	2015 - [30]	Suffix Tree, Dictionary, Array	$O(\lambda.N)$ với $\lambda \approx 20$	Server: $O(\lambda.l_{substr} + occ)$ Client: $O(\lambda.l_{substr} + occ)$ Communication: $O(\lambda.l_{substr} + occ)$	4	Yes	Yes
2	2015 - [38]	K-gram, Hash Table, Bloom Filter	$O(N.((l_{str} - kp) + 3)$	Server: $O((l_{substr}/kp).l_{str})$ Trapdoor: $O(occ.l_{substr})$	1	Yes	Yes
3	2016 - [102]	Position heap tree	$O(\lambda.N)$	Server: $O({l_{substr}}^2 + occ)$	3	No	No
4	2017 - [83]	Dictionary, Orthogonalized vectors	$O(N)$	Server: $O(l_d + occ)$ Communication: $O(l_{substr})$	3	No	No
5	2018 - [67]	FM index, BWT, Suffix array	$O(4N)$	Server: $O(l_{substr} + occ + C_{dummy})$	3	Yes	Yes
6	2018 - [50]	Map, List	$O(N)$	Server: $O(l_{substr}.avg)$ Trapdoor: $O(l_{substr})$	2	No	Yes
7	2019 - [79]	BWT, Suffix Array	$O(z.N)$	Server: $O(l_{substr} + occ)$ Client: $O((l_{substr} + occ).log^4(N))$ Communication: $O((l_{substr} + occ).log^2(N)))$	occ	No	No
8	2019 - [138]	Bilinear map	$O(N)$	Server: $O(l_{substr}.(l_{str} - l_{substr}))$ Trapdoor/Token: $O(l_{substr})$ Communication: $O(occ)$	1	No	No
9	2021 - [80]	BWT, Suffix Array	$O(N)$	Server: $O(l_{substr})$ Client: $O(l_{substr}.log^4(N) + occ.log^4(N/occ))$ Communication: $O(l_{substr}.log^2(N) + occ.log(N))$	1	No	No
10	2021 - [131]	Position heap	$O(N.(n_k + 1))$	Server: - Phase 1: $O(l_{substr} + occ)$ - Phase 2: $O(C_{KeywordSearch})$	2	No	Yes

11	2022 - [130]	Directed acyclic word graph (DAWG)	$O(N)$	Server: $O({l_{substr}}^2 + occ)$ Communication: $O({l_{substr}}^2 + occ)$	3	No	No
12	DIQ-SSE	Bloom Filter, Tập DA_s^u, Mảng PA_s	$O(2N)$	Server: - SearchIndex1: $O(k)$ - SearchIndex2: $O(2.l_{substr})$	1	No	No

2.8. Kết luận chương 2

Chương 2 đề xuất lược đồ DIQ-SSE dựa trên cặp chỉ mục mù đặc biệt $Index1, Index2$. Lựa chọn mô hình DAS-PROXY để triển khai và áp dụng ý tưởng tìm kiếm tuần tự trên cặp chỉ mục đã mang đến khả năng truy vấn chuỗi con hiệu quả cho lược đồ qua các tiêu chí về: bảo mật, hiệu năng, tính khả thi triển khai. Cụ thể:

- Thiết kế chỉ mục $Index1$ có ưu điểm chống rò rỉ thông tin và hỗ trợ truy vấn nhanh dựa trên cấu trúc dữ liệu "*tập các cặp ký tự kết nối phân biệt theo khoảng cách u trên A_s"* kết hợp cùng bộ lọc Bloom.

- Thiết kế chỉ mục $Index2$ với ưu điểm chống rò rỉ thông tin và tìm kiếm chính xác chuỗi con dựa trên cấu trúc "*tập các mảng nhị phân biểu diễn vị trí ký tự của chuỗi A_s"* kết hợp các phép dịch vòng bit bí mật.

- Thiết kế quy trình truy vấn tối ưu nhằm tận dụng thế mạnh của hai chỉ mục, bao gồm 4 bước tuần tự: *Biến đổi lệnh truy vấn, Truy vấn trên chỉ mục Index1, Truy vấn trên chỉ mục Index2, Giải mã dữ liệu*.

- Tính khả thi được kế thừa từ các lợi thế triển khai của mô hình DAS-PROXY, hỗ trợ truy vấn với một số lệnh mở rộng và cho phép thay đổi các tham số tác động liên quan. Trong khi đó, tính bảo mật (khả năng chống rò rỉ thông tin chỉ mục, mẫu truy vấn) và hiệu năng (tỷ lệ lọc, tỷ lệ lỗi, thời gian thực thi, độ phức tạp tính toán, số vòng giao tiếp) được làm rõ qua các kịch bản tấn công, kịch bản thực nghiệm và so sánh với các công trình tiêu biểu liên quan.

Ngoài các điểm nhấn của lược đồ DIQ-SSE nêu trên, thì một số nhược điểm cũng cần phải được ghi nhận như sau:

- Thiết kế của $Index1$ dẫn đến tồn tại bản ghi dương tính giả trong kết quả trả về sau truy vấn trên chỉ mục này. Trong khi đó $Index2$ mặc dù đã được làm nhiễu nhưng vẫn có thể để lộ ra một phần tính chất tuyến tính giữa độ dài chỉ mục với bản rõ khi xét trên các trường hợp chiều dài bản rõ quá dài hoặc quá ngắn.

- Tốc độ truy vấn trên $Index2$ bị ảnh hưởng bởi chiều dài chuỗi con.

- Chi phí lưu trữ chỉ mục của lược đồ DIQ-SSE chưa tốt, đạt mức $O(2N)$.

CHƯƠNG 3.
PHÁT TRIỂN PHƯƠNG PHÁP TRUY VẤN KHOẢNG HIỆU QUẢ TRÊN DỮ LIỆU SỐ TRONG CSDLQH MÃ HÓA

Chương 3 tập trung thiết kế lược đồ SSE dựa trên chỉ mục mù hỗ trợ truy vấn khoảng hiệu quả theo chuỗi điều kiện "$BETWEEN\ l\ and\ h$" trên dữ liệu số trong CSDLQH mã hóa. Các đóng góp nổi bật gồm: xây dựng lược đồ ESIT-SSE vận hành trên mô hình DAS-PROXY qua 4 giai đoạn; đề xuất phương pháp xây dựng chỉ mục mù NewBucketIndex, vector giấu tin (IHV) và cấu trúc IHV_B$^+$ Tree; đề xuất quy trình truy vấn khoảng theo 3 bước; cuối cùng là xây dựng các kịch bản thử nghiệm và đánh giá hiệu năng, phân tích bảo mật và so sánh lược đồ ESIT-SSE với các công trình tiêu biểu liên quan. Các ý tưởng và đề xuất trong chương 3 được NCS công bố tại các công trình liên quan là [CT4] và [CT5].

3.1. Đặt bài toán và dẫn luận ý tưởng thực hiện

Đặt bài toán:

Quan hệ $R(ID, X_1, X_2, \ldots X_t, \ldots X_n)$ có số lượng bản ghi lớn với X_t là trường dữ liệu rõ dạng số nhạy cảm, quan hệ mã $R^E(ID, X_1, X_2, \ldots X_t^E, \ldots X_n)$ lưu trên đám mây tương ứng với quan hệ R với X_t^E là trường dữ liệu mã tương ứng của X_t. Cho một cặp giá trị số bất kỳ $(\boldsymbol{l}, \boldsymbol{h})$, yêu cầu:

- Truy vấn trên trường dữ liệu X_t^E của quan hệ R^E để tìm ra ***chính xác*** những bản ghi thỏa mãn chuỗi điều kiện "$X_t\ BETWEEN\ l\ and\ h$".

- Quá trình truy vấn khoảng đạt hiệu quả tốt thông qua các tiêu chí về: ***bảo mật*** (chống rò rỉ thông tin theo mô hình IND-CKA2 (**Bảng 1.4**)); ***hiệu năng*** (thời gian thực thi tốt trên dữ liệu thực nghiệm lớn, quy trình tìm kiếm và độ phức tạp thuật toán tìm kiếm có ưu thế khi so sánh với các nghiên cứu gần đây); ***đa dạng truy vấn*** (hỗ trợ dạng thức truy vấn khoảng số phổ biến qua cú pháp BETWEEN trong CSDLQH, phù hợp với nhu cầu tìm kiếm trên dữ liệu số ở hầu hết các ứng dụng thực tiễn); ***khả thi triển khai*** (thuận lợi quản lý metadata và các giao dịch bảo mật, hỗ trợ tùy biến các tham số để điều chỉnh bảo mật và hiệu năng).

Dẫn luận ý tưởng thực hiện:

Về vấn đề chọn lược đồ và mô hình: Tương tự bài toán truy vấn chuỗi con trong CSDLQH mã hóa được trình bày ở **mục 2.1**, đối với bài toán này luận án cũng lựa chọn lược đồ SSE thiết kế dựa vào chỉ mục mù [43], [5], [33], [64], [136] và triển

khai trên mô hình DAS-PROXY [135], [6], [91], [9]. Khi đó lược đồ đề xuất sẽ tận dụng được các lợi thế đã phân tích trong **Nhận xét 1.1**, trong đó có ưu điểm đảm bảo cơ sở về tính bảo mật theo mô hình IND-CKA2 đã được chứng minh tại các lược đồ trong những nghiên cứu trước đó. Việc sử dụng DAS-PROXY cũng tạo sự thống nhất trong mô hình triển khai với lược đồ DIQ-SSE đề xuất trong **Chương 2**, giúp tăng tính khả thi khi đưa các đề xuất của cả 2 chương vào cài đặt thực tiễn.

Ý tưởng truy vấn chung: **Nhận xét 1.3** cho thấy cách tiếp cận dựa vào biến đổi 1-1 các giá trị số bản rõ về các giá trị chỉ mục bảo toàn thứ tự (giải pháp OPE [5], [17], [56]) đem lại hiệu năng truy vấn cao và thuận tiện cho biến đổi câu lệnh truy vấn rõ sang lệnh truy vấn trên chỉ mục. Các giải pháp OPE đòi hỏi xử lý dữ liệu ban đầu lớn và kẻ gian dễ dàng suy luận bản rõ từ chỉ mục dựa trên đặc tính bảo toàn thứ tự và mối quan hệ ánh xạ 1-1. Vì vậy ta sẽ lựa chọn giải pháp đơn giản hơn là chia nhóm/khoảng (hay còn gọi là Bucket [92], [47]) không nạp chồng và bảo toàn thứ tự cho tập dữ liệu rõ nhằm tiết kiệm chi phí xây dựng chỉ mục. Với mỗi BucketIndex đại diện cho nhiều giá trị bản rõ, khi đó dù kẻ gian tận dụng đặc tính bảo toàn thứ tự của các BucketIndex để suy diễn thì vẫn gặp khó khăn để xác định chính xác BucketIndex đang đại diện cho giá trị rõ nào trong khoảng (vấn đề này được các nghiên cứu [53], [113] chỉ ra liên quan tới các tham số như phương sai, entropy, pvd, ...). Mặc dù vậy, giải pháp Bucket lại có một nhược điểm quan trọng là trả về dữ liệu dương tính giả sau truy vấn. Nhằm nâng cao bảo mật và giảm thiểu kết quả dư thừa, luận án đề xuất tiếp tục cải tiến kỹ thuật chia khoảng Bucket (tạo ra ***chỉ mục mù NewBucketIndex*** [CT5]), xây dựng phương pháp che giấu đặc tính thứ tự của các NewBucketIndex (gọi là các ***vector giấu tin IHV*** [CT4]) và thiết kế cấu trúc dữ liệu (gọi là ***cây IHV_B⁺ Tree***) để tăng tốc truy vấn và loại bỏ dữ liệu dư thừa trả về.

Ý tưởng xây dựng NewBucketIndex: Để cải thiện khả năng bảo mật của Bucket (dựa vào [53], [113]), mỗi giá trị số sẽ thuộc nhiều NewBucketIndex và ngược lại. Khi đó với một giá trị rõ v_i có f_i lần xuất hiện trong CSDL thì có thể bị phân tán thành $f_i^1, f_i^2, .., f_i^n$ các phiên bản của v_i để thuộc về n NewBucketIndex kề nhau (với $f_i^1 + f_i^2 + \cdots + f_i^n = f_i$). Ý tưởng này giúp các Bucket có phân phối đều hơn, tăng bảo mật và giảm số v_i dương tính giả trả về sau truy vấn. Cơ chế chia khoảng dữ liệu số nạp chồng và bảo toàn thứ tự sẽ giúp thực hiện được ý tưởng trên.

Ý tưởng xây dựng vector giấu tin (IHV): Các NewBucketIndex bảo toàn thứ tự

nên cần có giải pháp che giấu đặc tính này (biến đổi thành vector IHV) nhưng vẫn phải hỗ trợ cơ chế bí mật khai thác các phép so sánh giúp thuận lợi truy vấn khoảng trên chỉ mục (dựa vào ý tưởng trong [58]). Mỗi vector IHV là kết quả của việc ẩn giá trị cần che giấu (là: giá trị rõ v_i trong trường X_t hoặc giá trị NewBucketIndex tương ứng của v_i hoặc giá trị l, h trong điều kiện truy vấn khoảng) vào các cặp vector trực giao. Không thể kiểm tra thứ tự giữa hai vector IHV đại diện cho hai giá trị v_i khác nhau hoặc hai giá trị NewBucketIndex khác nhau trên CS, nhưng vẫn hỗ trợ cơ chế so sánh chúng với các vector IHV đại diện cho l, h được yêu cầu từ EU.

Ý tưởng xây dựng cây IHV_B$^+$ Tree: dựa trên ý tưởng sử dụng cây B$^+$ Tree [34], [52], [121] để tăng tốc truy vấn dữ liệu số bản rõ trong hầu hết các hệ QTCSDL. Các nút của cây IHV_B$^+$ Tree chứa các vector IHV đại diện cho các giá trị NewBucketIndex, trong đó các nút lá tiếp tục liên kết đến danh sách các vector IHV đại diện cho các giá trị v_i tương ứng với NewBucketIndex. Quá trình truy vấn khoảng bản chất là duyệt cây IHV_B$^+$ Tree thông qua thực hiện các phép so sánh an toàn giá trị IHV trong các nút của cây với giá trị IHV đại diện l, h.

Như vậy, để triển khai các ý tưởng trên cần xây dựng lược đồ SSE (gọi là ESIT-SSE) dựa trên chỉ mục mù NewBucketIndex và vector IHV. Tuy nhiên các giá trị chỉ mục này không được lưu trữ vào các trường chỉ mục độc lập trong CSDLQH mã hóa R^E, mà sẽ được kết hợp biểu diễn và lưu trữ trong một cấu trúc dữ liệu bảo mật đi cùng CSDL gọi là IHV_B$^+$ Tree.

3.2. Đề xuất lược đồ và mô hình triển khai truy vấn khoảng trên dữ liệu số trong CSDLQH mã hóa

Dựa trên các giai đoạn cơ bản của lược đồ SSE (trình bày trong **Mục 1.5.1**) và ý tưởng được dẫn luận trong **Mục 2.1**. Luận án đề xuất lược đồ ESIT-SSE hỗ trợ truy vấn khoảng hiệu quả trên CSDLQH mã hóa gồm 5 giai đoạn (**Hình 3.1**) sau:

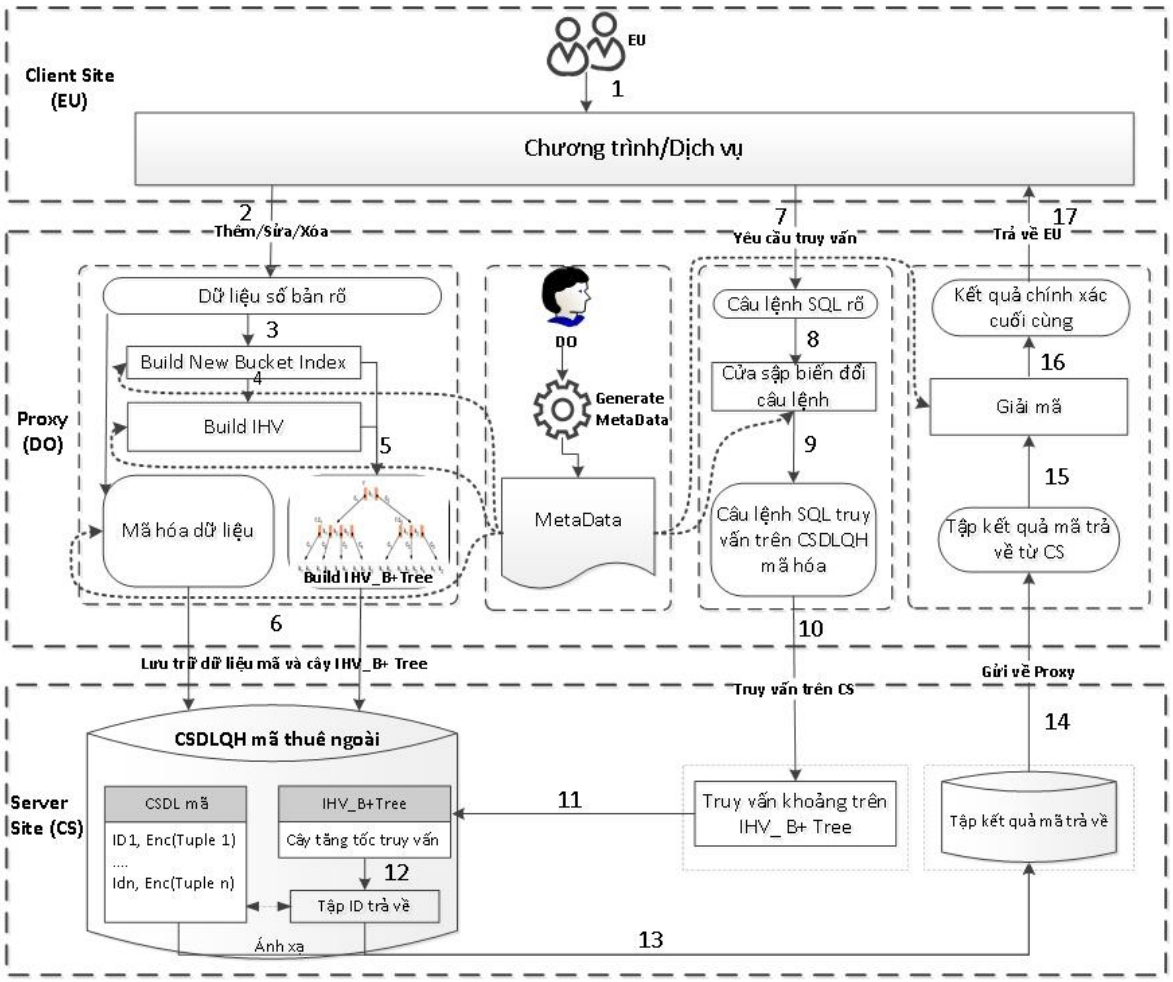
Giai đoạn 1. $\boldsymbol{GenSecretMetadata(\lambda)}$**:** trong giai đoạn này DO dùng thuật toán $GenSecretMetadata$ với tham số $\boldsymbol{\lambda}$ để sinh và lưu trữ các $MetaData$ (bao gồm các khóa và các tham số bí mật) tại vùng an toàn.

Giai đoạn 2. $\boldsymbol{Build\_IHV\_B^+Tree}$: nhiệm vụ chính giai đoạn này là xây dựng các NewBucketIndex từ dữ liệu rõ, sau đó chuyển đổi các NewBucketIndex về các vector IHV, cuối cùng xây dựng cây IHV_B$^+$Tree từ các vector giấu tin các để lưu trữ trên CS cùng với CSDLQH mã hóa.

Giai đoạn 3. ***Transform_RangeQueryCondition*:** trong giai đoạn này, câu lệnh truy vấn khoảng trên dữ liệu rõ của người dùng được biến đổi thành câu lệnh truy vấn trên dữ liệu mã, cụ thể điều kiện truy vấn sẽ được biến đổi thành các biểu thức so sánh trên các vector giấu tin qua thuật toán 3.3 $Transform\_ComparisonOperand\_To\_IHV$.

Giai đoạn 4. ***RangeQuery_On_IHV_B$^+$Tree***: đây là giai đoạn thực hiện thuật toán duyệt cây IHV_B$^+$Tree để tìm ra các giá trị thỏa mãn điều kiện (các biểu thức so sánh) nhận được từ Giai đoạn 3. Sau đó tham chiếu vào CSDLQH mã hóa để lấy ra các bản ghi mã tương ứng và trả về tập kết quả.

Giai đoạn 5. Decrypt: giải mã tập kết quả của **Giai đoạn 4** và trả về EU.



Hình 3.1. Triển khai lược đồ ESIT-SSE trên mô hình DAS-PROXY

Dựa trên phân tích tại **mục 3.1**, luận án lựa chọn mô hình DAS-PROXY với ba khu vực: Client Site, Proxy, Server Site để triển khai – vận hành lược đồ ESIT -SSE đã đề xuất (**Hình 3.1**), cụ thể:

- ***Client Site:*** là khu vực dành cho EU đưa ra các yêu cầu truy vấn khoảng.

- ***Proxy:*** là khu vực an toàn để DO ủy nhiệm thực hiện các thuật toán mang tính

riêng tư của lược đồ ESIT-SSE, như: $GenSecretMetadata$, $Build\_IHV\_B^+Tree$, $Transform\_RangeQueryCondition$, $Transform\_ComparisonOperand\_To\_IHV$. Một số MetaData bí mật được sinh ra, lưu trữ và quản lý tại Proxy gồm:

1) *Khóa bí mật* $sKey$: dùng để mã và giải mã dữ liệu, là tham số cho các hàm f^{Enc} và f^{Dec} (trong mô hình này sử dụng thuật toán AES với OFB Mode).
2) m là số lượng khoảng (Bucket) được tạo ra trên tập dữ liệu rõ kiểu số.
3) ε là hệ số mở rộng, sử dụng trong quá trình sinh NewBucketIndex.
4) $\overrightarrow{U_b}$, $\overrightarrow{U_c}$ là các vector trực giao, sử dụng để tạo vector IHV.
5) $(pos1, pos2)$ cặp vị trí sử dụng trong quá trình tạo vector IHV.
6) $(r_b,\ r_c)$ cặp số ngẫu nhiên nguyên dương hỗ trợ tạo nhiễu trong quá trình xây dựng vector IHV.
7) M là một ma trận khả nghịch dùng để tăng bảo mật vector IHV.

- ***Server Site:*** (CS) được quản lý bởi các nhà cung cấp dịch vụ (DSP). CS lưu trữ CSDLQH mã hóa cùng cây IHV_B^+Tree và thực hiện **Giai đoạn 4.** $\boldsymbol{RangeQuery\_On\_IHV\_B^+Tree}$ của lược đồ ESIT -SSE.

3.3. Xây dựng chỉ mục NewBucketIndex

Tại Proxy, mỗi giá trị v_i của trường dữ liệu X_t ngoài việc được biến thành giá trị mã v_i^E để lưu trên trường X_t^E tại CSDLQH mã hóa ($v_i^E = f^{Enc}(v_i,\ sKey)$) thì nó còn được biến đổi thành một NewBucketIndex cho phép giấu các thông tin về v_i nhưng vẫn bảo toàn thứ tự, là cơ sở quan trọng để thực hiện được các phép so sánh và lập các cấu trúc dữ liệu hỗ trợ tìm kiếm hiệu quả trên CSDLQH mã hóa tại CS.

3.3.1. Yêu cầu đối với chỉ mục NewBucketIndex

Luận án lựa chọn phương pháp tạo chỉ mục dựa trên Bucket [92], [47] là bước đầu trong quá trình xây dựng lược đồ ESIT-SSE bởi một số ưu điểm về hiệu năng của nó. Một các trực quan, dữ liệu rõ được phân vào các nhóm/khoảng (Bucket) và mỗi nhóm được đại diện bởi một số nguyên BucketID. Khi đó mỗi BucketID sẽ đại diện cho nhiều giá trị số, nếu kẻ gian biết BucketID thì cũng khó đoán được giá trị số thực sự được che giấu. Một số nghiên cứu chỉ ra để thiết kế được Bucket bảo mật tốt, ít dư thừa dữ liệu trả về sau truy vấn và khả thi triển khai thì cần được đánh giá qua nhiều tham số như Variance, Entropy [53], pvd [113].

Chương này đề xuất chỉ mục mù NewBucketIndex được xây dựng trên cơ sở chia khoảng Bucket với các yêu cầu như sau:

- Một giá trị rõ có thể được đại diện bởi nhiều NewBucketIndex và ngược lại.
- Hỗ trợ chống các dạng tấn công: rò rỉ thông tin động và rò rỉ thông tin tĩnh.
- Giảm dư thừa dữ liệu trả về sau truy vấn.
- Thuận lợi biến đổi lệnh truy vấn trên dữ liệu rõ sang lệnh truy vấn trên NewBucketIndex.
- Hỗ trợ truy vấn khoảng trực tiếp trên các NewBucketIndex với hiệu suất cao.

Để đạt yêu cầu trên, dựa trên ý tưởng đề xuất trong **Mục 3.1**, luận án đề xuất cải tiến phương pháp chia khoảng để tạo NewBucketIndex như sau:

- Tập dữ liệu rõ cần được ánh xạ sang tập giá trị mới lớn hơn (mở rộng) với các phần tử được sinh ngẫu nhiên và xác suất xuất hiện như nhau (phân bố đều).
- Sử dụng cách tiếp cận chia khoảng ngẫu nhiên liên tiếp trên tập giá trị mới để tạo các Bucket nạp chồng và bảo toàn thứ tự.

3.3.2. Xây dựng chỉ mục NewBucketIndex

Để xây dựng chỉ mục như yêu cầu trong **Mục 3.3.1**, trước hết cần thực hiện hai bước: là làm phẳng và chia khoảng với tập dữ liệu rõ. Quá trình làm phẳng dữ liệu dựa trên một số định nghĩa và được mô tả qua các ví dụ như dưới đây:

Định nghĩa 3.1. Cho $V = \{v_1,\ v_2, \dots, v_z\}$ là tập z các giá trị số lưu trong trường X_t của quan hệ R. Khi đó $D = \{d_1, d_2, \dots, d_n\}$ $(n \le z)$ được gọi là tập các phần tử phân biệt của tập V nếu các phần tử của D được xác định như sau:

$$d_i = \binom{f_i}{v_i},\ d_{i+1} = \binom{f_{i+1}}{v_{i+1}} \text{ với } 1 \le i \le n-1 \qquad (3.1)$$

trong đó $v_i, v_{i+1} \in V;\ v_i \le v_{i+1};\ f_i$ là tần suất xuất hiện của v_i trong tập V.

Ví dụ 3.1. Cho tập $V = \{3,\ 9,\ 7,\ 25,\ 14,\ 3,\ 7,\ 50,\ 9,\ 25,\ 3,\ 14,\ 50,\ 14,\ 7,\ 25,\ 3,\ 25,\ 25\}$, khi đó theo **Định nghĩa 3.1** tập D gồm 6 phần từ phân biệt:

$$D = \left\{\binom{f_i}{v_i}\right\} = \left\{\binom{4}{3}, \binom{3}{7}, \binom{2}{9}, \binom{3}{14}, \binom{5}{25}, \binom{2}{50}\right\}$$

Định nghĩa 3.2. Cho $D = \{d_1, d_2, \dots, d_n\}$ là tập các phần tử phân biệt của tập V và $d_i,\ d_{i+1} \in D$. Khi đó $P_i = \{p_{i1}, p_{i2}, \dots, p_{ik_i}\}$ $(1 \le i \le n)$ được gọi là tập giá trị ngẫu nhiên đại diện cho phần tử $d_i = \binom{f_i}{v_i}$ và tương tự P_{i+1} là tập giá trị ngẫu nhiên đại diện cho phần tử $d_{i+1} = \binom{f_{i+1}}{v_{i+1}}$. Tập $P_i,\ P_{i+1}$ phải có đặc điểm và mối quan hệ

với nhau như sau:

- $[P_i^{Min},\ P_i^{Max}]$ là miền giá trị của tập P_i với $P_i^{Min},\ P_i^{Max}$ là các số nguyên và $P_i^{Max} - P_i^{Min} \geq f_i$;
- $p_{ij} \leftarrow PRF\left(P_i^{Min}, P_i^{Max}\right)$ với $1 \leq j \leq k_i,\ k_i \geq f_i$ và $P_i^{Min} \leq p_{i1} \leq p_{i2} \leq \cdots \leq p_{ik_i} \leq P_i^{Max}$;
- $P_i^{Max} \leq P_{i+1}^{Min}$ với $1 \leq i \leq n$;
- Số phần tử k_i của tập P_i được xác định theo nguyên tắc: $\frac{k_1}{f_1} \approx \frac{k_2}{f_2} \approx \cdots \approx \frac{k_i}{f_i} \approx \cdots \approx \frac{k_n}{f_n} \approx \varepsilon$. Với ε gọi là hệ số mở rộng, ε nguyên và $\varepsilon \geq 1$. (3.2)

Ví dụ 3.2. Xét tập D trong **Ví dụ 3.1** với 6 phần tử, khi đó các tập giá trị ngẫu nhiên P_i $(1 \leq i \leq 6)$ trong trường hợp $\varepsilon = 1$ được xác định theo các bước sau:

Bước 1. Xác định miền giá trị cho mỗi tập P_i $(1 \leq i \leq 6)$:

$P_1 \in [1, 9],\ P_2 \in [25, 81],\ P_3 \in [90, 100],$

$P_4 \in [115, 165],\ P_5 \in [1024, 1254],\ P_6 \in [1255, 2003].$

Bước 2. Tạo danh sách phần tử ngẫu nhiên trong miền giá trị của từng P_i:

$P_1 = \{1, 2, 7, 9\},\ P_2 = \{45, 59, 74\},$

$P_3 = \{95, 99\},\ P_4 = \{116, 155, 161\},$

$P_5 = \{1123, 1198, 1215, 1217, 1252\},\ P_6 = \{1571, 1999\}.$

Định nghĩa 3.3. Cho $V = \{v_1,\ v_2, \ldots, v_z\}$ là tập z các giá trị số lưu trong trường X_t của quan hệ R, $D = \{d_1, d_2, \ldots, d_n\}$ là tập các phần tử phân biệt của tập V và P_i $(1 \leq i \leq n)$ là tập giá trị ngẫu nhiên đại diện cho phần tử $d_i \in D$. Khi đó P được gọi là tập giá trị phẳng đại diện cho tập V nếu $P = P_1 \cup P_2 \cup \ldots \cup P_i \cup \ldots \cup P_n$.

Từ **Ví dụ 3.2** và **Định nghĩa 3.3**, ta có: $P = \{1, 2, 7, 9, 45, 59, 74, 95,$

$99, 116, 155, 161, 1123, 1198, 1215, 1217, 1252, 1571, 1999\}$

Như vậy, ***giai đoạn mở rộng và làm phẳng tập dữ liệu rõ đã hoàn thành***. Tiếp theo là quá trình chia Bucket trên tập P, luận án đề xuất thêm **Định nghĩa 3.4**.

Định nghĩa 3.4. Cho $V = \{v_1,\ v_2, \ldots, v_z\}$ là tập z các giá trị số lưu trong trường X_t của quan hệ R và P được gọi là tập giá trị phẳng đại diện cho V. Tập $B = \{B_1, B_2, \ldots., B_m\}$ được gọi là tập các giá trị NewBucketIndex đại diện cho tập V nếu:

- Tập P được chia ngẫu nhiên thành m khoảng liên tiếp nhau $(m \leq |P|)$;
- Khoảng thứ j được đại diện bằng một giá trị nguyên ngẫu nhiên là B_j;
- Với $j, j' \in [1,\ m]$ và $j < j'$ thì $B_j \leq B_{j'}$.

Theo **Định nghĩa 3.4**, với mỗi giá trị $v_i \in V$ $(1 \le i \le z)$ sẽ được đại diện bởi một giá trị NewBucketIndex $v_i^B = B_j$ với $B_j \in B$ $(1 \le j \le m)$. Khi đó tập V^B sẽ chứa z phần tử v_i^B tương ứng với các phần tử $v_i \in V$.

Thuật toán ***BuildNewBucketIndex*** xây dựng tập B và V^B như sau:

Thuật toán 3.1. *BuildNewBucketIndex*

Input

$V = \{v_1, v_2, \dots, v_z\}$: là tập z các giá trị số lưu trong trường X_t của quan hệ R;

ε: là hệ số mở rộng $(\varepsilon \ge 1)$;

m: Số Bucket cần chia $(m \le |P|)$;

Output

Tập $V^B = \{v_1^B, v_2^B, \dots, v_z^B\}$: là tập các giá trị NewBucketIndex tương ứng với z phần tử trong V;

Các bước thực hiện

(1) Xây dựng tập $D = \{d_1, d_2, \dots d_i, \dots d_n\}$ là tập các phần tử phân biệt của tập V $(1 \le i \le n)$;

(2) Dựa trên hệ số ε, xây dựng $P_1, P_2, \dots, P_i, \dots P_n$ lần lượt là các tập giá trị ngẫu nhiên đại diện cho các phần tử $d_1, d_2, \dots d_i, \dots d_n$ $(1 \le i \le n)$;

(3) Xây dựng tập giá trị phẳng P đại diện cho tập V: $P = P_1 \cup P_2 \cup \dots \cup P_i \cup \dots \cup P_n$; Khi đó $P = \{p_1, p_2, \dots, p_{|P|}\}$;

(4) Chọn các giá trị biên mở rộng cho tập P là EP^{Min} và EP^{Max} sao cho $EP^{Min} \le p_1, EP^{Max} \ge p_{|P|}$;

(5) Xác định giá trị tại $m-1$ điểm giao nhau của m khoảng trên tập P bằng cách: Tính $m-1$ giá trị ngẫu nhiên phân biệt $\{ipv_1, ipv_2, \dots, ipv_{m-1}\}$ theo công thức: $ipv_t \leftarrow PRF(EP^{Min}, EP^{Max})$ với $1 \le t \le m-1$ và $ipv_t < ipv_{t+1}$;

(6) Xác định tập $B = \{B_1, B_2, \dots B_t, \dots B_m\}$ là tập các giá trị NewBucketIndex đại diện cho V như sau:

$B_1 \leftarrow PRF(EP^{Min}, ipv_1)$, B_1 đại diện cho tập EP_1 với miền giá trị $[EP^{Min}, ipv_1)$;

$B_2 \leftarrow PRF(ipv_1, ipv_2)$, B_2 đại diện cho tập EP_2 với miền giá trị $[ipv_1, ipv_2)$;

…

$B_t \leftarrow PRF(ipv_{t-1}, ipv_t)$, B_t đại diện cho tập EP_t với miền giá trị $[ipv_{t-1}, ipv_t)$;

…

$B_m \leftarrow PRF(ipv_{m-1}, EP^{Max})$, Giá trị B_m đại diện cho tập EP_m với miền giá trị $[ipv_{m-1}, EP^{Max})$;

(7) Dựa vào tập B, xác định tập $V^B = \{v_1^B, v_2^B, \dots, v_z^B\}$ tương ứng với V như sau:

(7.1) Với mỗi $v_j \in V$, xác định phần tử $d_i \in D$ sao cho $d_i = v_j$ $(1 \leq j \leq z;\ 1 \leq i \leq n)$;

(7.2) Xác định tập P_i tương ứng với d_i, lấy ngẫu nhiên giá trị $p \in P_i$;

(7.3) Kiểm tra nếu $p \in EP_t$ $(1 \leq t \leq m-1)$ thì B_t là giá trị Bucket đại diện cho v_j, gán $v_j^B = B_t$;

(7.4) $V^B = \{v_1^B, v_2^B, \dots, v_z^B\}$ với $v_j^B \in B$ $(1 \leq j \leq z)$;

(8) Return V^B;

Nhận xét 3.1:

- Các **Ví dụ 3.1** và **Ví dụ 3.2** sử dụng các giá trị số nguyên để thuận tiện mô phỏng thuật toán tạo NewBucketIndex. Thực tế tập V là tham số đầu vào thuật toán này cho phép mọi phần tử của nó là số thực, đồng thời đầu ra của thuật toán là tập V^B cũng cho phép so sánh các phần tử của nó với khoảng số thực $[l, h]$ được người dùng yêu cầu.

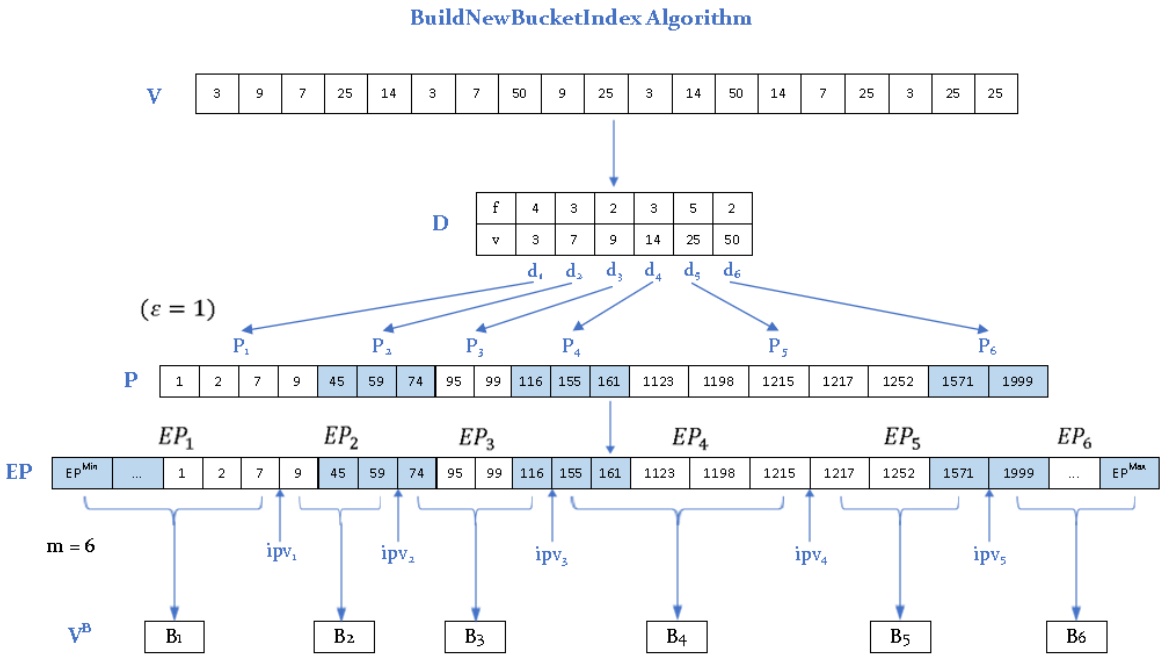
- Thuật toán $BuildNewBucketIndex$ nên được sử dụng trên quan hệ rõ R với số lượng bản ghi ban đầu đủ để mô tả hết các trường hợp giá trị khác nhau có thể có của trường X_t. Nói cách khác tập V chứa các giá trị rõ phân biệt đã bao phủ được hết miền giá trị của trường nàyX_t. Khi đó tập B chứa các NewBucketIndex có thể đại diện cho mọi giá trị phát sinh mới trên trường X_t trong quá trình vận hành (insert, update) CSDLQH mã hóa.

Sau khi xây dựng tập chỉ mục V^B đại diện cho các giá trị trong trường X_t, các thông tin bí mật liên quan tới quá trình sinh các NewBucketIndex (theo **Ví dụ 3.1** và **Ví dụ 3.2**) sẽ được lưu tại PROXY (xem **Bảng 3.1**) để phục vụ việc biến đổi các điều kiện truy vấn và sự thay đổi của CSDL trong quá trình vận hành.

Bảng 3.1. Các thông tin bí mật để sinh NewBucketIndex

$d_i.v$	P_i	P_i^{Min}	P_i^{Max}	B_i	ipv_{i-1}	ipv_i
3	P_1	1	9	B_1	EP^{Min}	ipv_1
3	P_1	1	9	B_2	ipv_1	ipv_2
7	P_2	25	81	B_2	ipv_1	ipv_2
7	P_2	25	81	B_3	ipv_2	ipv_3
9	P_3	95	99	B_3	ipv_2	ipv_3
14	P_4	115	165	B_3	ipv_2	ipv_3
14	P_4	115	165	B_4	ipv_3	ipv_4
25	P_5	1024	1254	B_4	ipv_3	ipv_4
25	P_5	1024	1254	B_5	ipv_4	ipv_5
50	P_6	1255	2003	B_5	ipv_4	ipv_5
50	P_6	1255	2003	B_6	ipv_5	EP^{Max}

Áp dụng với dữ liệu trong các **Ví dụ 3.1** và **Ví dụ 3.2**, thuật toán ***BuildNewBucketIndex*** được minh họa như **Hình 3.2**.



Hình 3.2. Mô tả thuật toán BuildNewBucketIndex

3.4. Biến đổi giá trị NewBucketIndex thành vector giấu tin (IHV)

Cho $v_i, v_j \in V$, theo thuật toán ***BuildNewBucketIndex*** nếu $v_i \leq v_j$ thì $v_i^B \leq v_j^B$, việc bảo toàn thứ tự này giúp quá trình biến đổi lệnh truy vấn trên dữ liệu rõ sang truy vấn trên NewBucketIndex được thuận lợi và thực thi truy vấn trên CSLDLQH mã đạt hiệu năng cao. Tuy nhiên nó gây ra rò rỉ dữ liệu khi kẻ tấn công biết được mối quan hệ thứ tự giữa các bản ghi trên CSDLQH mã hóa. Để giữ được tính chất bảo toàn thứ tự mà vẫn ngăn chặn được kẻ gian so sánh giữa các chỉ mục, NewBucketIndex được biến đổi về vector giấu tin (IHV) dựa trên ý tưởng [58].

3.4.1. Quy trình xây dựng vector giấu tin (IHV)

Xét điều kiện truy vấn khoảng của EU trên dữ liệu số là "$\boldsymbol{X_t}$ *Between* $\boldsymbol{l}$ *And* $\boldsymbol{h}$", tương đương tìm kiếm các phần tử $v_i \in V$ sao cho $l \leq v_i \leq h$. Khi đó tại Proxy sẽ thực hiện biến đổi l, h về dạng thức cho phép so sánh được với các giá trị NewBucketIndex tại CS như mô tả sau: *"tìm kiếm các phần tử* $v_i^B \in V^B$ *sao cho* $c_l \leq v_i^B \leq c_h$*"*, với $c_l = Min(l^B)$ là giá trị NewBucketIndex nhỏ nhất đại diện cho l và $c_h = Max(h^B)$ là giá trị NewBucketIndex lớn nhất đại diện cho h. Cách thức để xác định $Min(l^B), Max(h^B)$ được trình bày trong **Thuật toán 3.5**.

Bài toán bảo mật đặt ra là tại CS phải hỗ trợ cơ chế bảo mật để thực hiện được

biểu thức so sánh $Min(l^B) \le v_i^B \le Max(h^B)$ nhưng không có cơ sở nào để thực hiện so sánh hai phần tử v_i^B, v_j^B bất kỳ trong tập V^B. Như vậy, tại Proxy phải có giải pháp biến đổi tiếp v_i^B, $Min(l^B)$ và $Max(h^B)$ thành các dạng thức mới $\overrightarrow{ihv_b(v_i^B)}$, $\overrightarrow{ihv_c(Min(l^B))}$ và $\overrightarrow{ihv_c(Max(h^B))}$ (gọi là các vector giấu tin IHV) trước khi chuyển chúng lên CS. Tổng quát, ta xét việc biến đổi và so sánh các vector IHV cho biểu thức $Min(l^B) \le v_i^B$ (biểu thức $v_i^B \le Max(h^B)$ được vận dụng tương tự).

Đặt $c = c_l$ và $b = v_i^B$, khi đó biểu thức $c_h \le v_i^B$ được thay thế thành $c \le b$. Lúc này b đại diện cho các NewBucketIndex trên CS, còn c đại diện cho các toán hạng l, h. Mục tiêu là làm thế nào tạo ra các vector $\overrightarrow{ihv_b(b)}$ và $\overrightarrow{ihv_c(c)}$, trước tiên ta thực hiện biến đổi với biểu thức $c \le b$ như sau:

$$\begin{aligned} & c \le b \\ \Leftrightarrow\ & b - c \ge 0 \\ \Leftrightarrow\ & (b * 1) + (-1 * c) \ge 0 \\ \Leftrightarrow\ & \begin{pmatrix} b \\ -1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ c \end{pmatrix} \ge 0 \ \textit{(Tích vô hướng hai vector)} \end{aligned}$$

Các bộ giá trị $(b, -1)$ và $(1, c)$ được gọi là các vector nguyên thủy chứa các giá trị gốc b và c. Ký hiệu $\overrightarrow{O_b} = (b, -1)$ và $\overrightarrow{O_c} = (1, c)$, khi đó:

$$\overrightarrow{O_b} \cdot \overrightarrow{O_c} \ge 0 \Leftrightarrow \begin{pmatrix} b \\ -1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ c \end{pmatrix} \ge 0 \qquad (3.3)$$

Bước đầu che giấu thông tin thứ tự cho b và c trên CS dựa trên việc biến đổi $\overrightarrow{O_b}$ và $\overrightarrow{O_c}$ về các vector mới dài hơn nhưng vẫn đảm bảo tích vô hướng giữa chúng lớn hơn hoặc bằng 0. Ta thực hiện hoà nhập $\overrightarrow{O_b}$ và $\overrightarrow{O_c}$ lần lượt vào các vector trực giao $\overrightarrow{U_b} \perp \overrightarrow{U_c}$ tại các vị trí $pos1, pos2$ để tạo ra cặp vector nhúng $\overrightarrow{UO_b}$ và $\overrightarrow{UO_c}$.

Các bước tạo vector nhúng $\overrightarrow{UO_b}$ và $\overrightarrow{UO_c}$ được mô tả chi tiết như sau:

Bước 1. Chọn 2 vector cơ sở bí mật $\overrightarrow{U_b}$ và $\overrightarrow{U_c}$ trực giao với nhau trên không gian $k-2$ chiều ($k \ge 3$).

$$\overrightarrow{U_b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{k-2} \end{pmatrix},\ \overrightarrow{U_c} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_{k-2} \end{pmatrix} \text{ với } \overrightarrow{U_b} . \overrightarrow{U_c} = 0$$

Bước 2. Hoà nhập vector $\overrightarrow{O_b}$ vào vector $\overrightarrow{U_b}$ và hoà nhập vector $\overrightarrow{O_c}$ vào vector $\overrightarrow{U_c}$ tại các vị trí bí mật $(pos1, pos2) | \ 1 \le pos1, pos2 \le k$ để nhận được hai vector nhúng $\overrightarrow{UO_b}$, $\overrightarrow{UO_c}$ như (3.4):

$$\overrightarrow{O_b} = \begin{pmatrix} b \\ -1 \end{pmatrix}, \overrightarrow{O_c} = \begin{pmatrix} 1 \\ c \end{pmatrix}, \overrightarrow{UO_b} = \begin{pmatrix} b_1 \\ \boldsymbol{b} \\ b_2 \\ \vdots \\ \boldsymbol{-1} \\ b_{k-2} \end{pmatrix}, \overrightarrow{UO_c} = \begin{pmatrix} c_1 \\ \boldsymbol{1} \\ c_2 \\ \vdots \\ \boldsymbol{c} \\ c_{k-2} \end{pmatrix} \Rightarrow \overrightarrow{UO_b} . \overrightarrow{UO_c} = b - c \quad (3.4)$$

Đến đây mỗi vector $\overrightarrow{UO_b}$ đại diện cho một giá trị NewBucketIndex $b = v_i^B$. Giả định $(pos1, pos2)$ đã được giữ bí mật, thì kẻ gian vẫn có thể đoán được thứ tự giữa các vector $\overrightarrow{UO_b}$ với nhau (thứ tự ở đây được hiểu là vector $\overrightarrow{UO_b}$ nào đang đại diện cho giá trị b lớn hơn) nếu giả định họ có tập các vector $\overrightarrow{UO_b}$ và một vector $\overrightarrow{UO_c}$ bất kỳ. Cụ thể, quá trình tấn công biết trước bản mã này được thực hiện như sau:

- Tính tất cả các tích vô hướng $\overrightarrow{UO_b} . \overrightarrow{UO_c}$ để xác định được hiệu $b - c$.

- Sau đó so sánh các hiệu này với nhau để tìm ra thứ tự giữa các giá trị b.

Để chống lại phương pháp tấn công này ta thực hiện biến đổi tiếp như **Bước 3**.

Bước 3. Sử dụng cặp số ngẫu nhiên nguyên dương $(r_b,\ r_c)$ để làm nhiễu các giá trị trong các vector nhúng $\overrightarrow{UO_b},\ \overrightarrow{UO_c}$ như (3.5):

$$\overrightarrow{UO_b} = r_b * \overrightarrow{UO_b} = \begin{pmatrix} r_b * b_1 \\ \boldsymbol{r_b} * \boldsymbol{b} \\ r_b * b_2 \\ \vdots \\ \boldsymbol{r_b} * (\boldsymbol{-1}) \\ r_b * b_{k-2} \end{pmatrix},\ \overrightarrow{UO_c} = r_c * \overrightarrow{UO_c} = \begin{pmatrix} r_c * c_1 \\ \boldsymbol{r_c} * \boldsymbol{1} \\ r_c * c_2 \\ \vdots \\ \boldsymbol{r_c} * \boldsymbol{c} \\ r_c * c_{k-2} \end{pmatrix} \quad (3.5)$$

Ta có $\overrightarrow{UO_b}.\overrightarrow{UO_c} = (r_b * r_c) * (b - c) \geq 0$ nếu $b \geq c$. (3.6)

Như vậy kết quả $\overrightarrow{UO_b}.\overrightarrow{UO_c}$ chỉ là là một dấu hiệu nhận biết b có lớn hơn c hay không, không đủ cơ sở để so sánh thứ tự giữa các vector $\overrightarrow{UO_b}$. Quá trình gây nhiễu không ảnh hưởng tới cách thức và tốc độ tìm kiếm do việc tính tích vô hướng sau tạo nhiễu không phát sinh chi phí.

Ví dụ 3.3. Cho $k = 5$ và hai vector cơ sở $\overrightarrow{U_b}$, $\overrightarrow{U_c}$ như sau:

$$\overrightarrow{U_b} = \begin{pmatrix} -4 \\ 8 \\ 4 \end{pmatrix}, \qquad \overrightarrow{U_c} = \begin{pmatrix} 3 \\ -2 \\ 7 \end{pmatrix}$$

Chọn cặp giá trị $(pos1, pos2) = (2,4)$. Hoà nhập vector $\overrightarrow{O_b}$, $\overrightarrow{O_c}$ lần lượt vào vector $\overrightarrow{U_b}$, $\overrightarrow{U_c}$ để tạo ra các vector nhúng $\overrightarrow{UO_b},\ \overrightarrow{UO_c}$ có chiều dài là 5 như sau :

$$\overrightarrow{UO_b} = \begin{pmatrix} -4 \\ b \\ 8 \\ -1 \\ 4 \end{pmatrix}, \qquad \overrightarrow{UO_c} = \begin{pmatrix} 3 \\ 1 \\ -2 \\ c \\ 7 \end{pmatrix}$$

Dùng hai số nguyên dương ngẫu nhiên $r_b = 2,\ r_c = 3$ làm nhiễu $\overrightarrow{UO_b}, \overrightarrow{UO_c}$:

$$\overrightarrow{UO_b} = 2 * \overrightarrow{UO_b} = \begin{pmatrix} -8 \\ 2 * b \\ 16 \\ -2 \\ 8 \end{pmatrix}, \qquad \overrightarrow{UO_c} = 3 * \overrightarrow{UO_c} = \begin{pmatrix} 9 \\ 3 \\ -6 \\ 3 * c \\ 21 \end{pmatrix}$$

Cặp giá trị $(pos1, pos2)$ trong thiết kế $\overrightarrow{UO_b}, \overrightarrow{UO_c}$ ***bộc lộ một số hạn chế***:

- Thứ nhất là chúng được dùng chung cho việc giấu thông tin vị trí của tất cả giá trị $b = v_i^B$ $(1 \le i \le z)$ trên các vector nhúng $\overrightarrow{UO_b}$ tương ứng. Do đó khi bị lộ $pos1,\ pos2$ tại một vector nhúng sẽ gây rò rỉ thông tin giá trị b trên hàng loạt các vector nhúng còn lại.

- Thứ hai là kẻ gian có thể sử dụng phương pháp phân tích tập vector $\overrightarrow{UO_b}$, $\overrightarrow{UO_c}$ để dự đoán vị trí $(pos1, pos2)$. Bằng cách với mỗi bộ k phần tử của các vector $\overrightarrow{UO_b}, \overrightarrow{UO_c}$, thực hiện tìm ước chung lớn nhất của các phần tử trong mỗi bộ đó để suy diễn lại giá trị gốc của vector $\overrightarrow{UO_b}, \overrightarrow{UO_c}$ trước khi nhân tham số ngẫu nhiên (r_b, r_c). Sau đó kẻ gian sẽ suy diễn vị trí của giá trị -1 trong vector $\overrightarrow{UO_b}$ có khả năng cao là $pos2$ và vị trí giá trị 1 trong vector $\overrightarrow{UO_c}$ có khả năng cao là $pos1$.

Để giải quyết vấn đề lộ vị trí $(pos1, pos2)$, luận án tiếp tục tăng cường bảo mật cho $\overrightarrow{UO_b}, \overrightarrow{UO_c}$ dựa trên ý tưởng chuyển đổi các vetor này về các ma trận cỡ kx1 sau đó nhân chúng lần lượt với các ma trận bí mật M_b, M_c cỡ kxk. Trong đó $M_c = M^T, M_b = M^{-1}$ (M là một ma trận bí mật khả nghịch được chọn trước, M_c là ma trận chuyển vị của M và M_b là ma trận nghịch đảo của M, lúc này ${M_c}^T$ x M_b = 1).

Gọi UO_b, UO_c lần lượt là các ma trận cỡ kx1 được chuyển đổi tương ứng từ các vector $\overrightarrow{UO_b}, \overrightarrow{UO_c}$. Khi đó UO_b^M, UO_c^M lần lượt là kết quả của các phép nhân ma trận sau: $UO_b^M = M_b$x UO_b và $UO_c^M = M_c$ x UO_c.

Gọi $\overrightarrow{UO_b^M}, \overrightarrow{UO_c^M}$ lần lượt là các vector có chiều dài k được chuyển đổi lại từ các ma trận UO_b^M, UO_c^M cỡ kx1. Tiếp theo ta thực hiện phép nhân vô hướng hai vector $\overrightarrow{UO_b^M}, \overrightarrow{UO_c^M}$ như dưới đây:

$$\overrightarrow{UO_c^M}.\overrightarrow{UO_b^M} = (UO_c^M)^T \text{ x } UO_b^M$$
$$= (M_c \text{ x } UO_c)^T \text{ x } (M_b \text{ x } UO_b)$$
$$= \left(UO_c{}^T \text{ x } M_c{}^T\right) \text{ x } (M_b \text{ x } UO_b)$$
$$= UO_c{}^T \text{x } UO_b = \overrightarrow{UO_c}.\overrightarrow{UO_b} = (r_b * r_c) * (b - c) \quad (3.7)$$

Gọi $\overrightarrow{ihv_b(b)}, \overrightarrow{ihv_c(c)}$ lần lượt là các vector giấu tin (IHV) cho các giá trị b và c. Khi đó :

$$\overrightarrow{ihv_b(b)} = \overrightarrow{UO_b^M},\ \overrightarrow{ihv_c(c)} = \overrightarrow{UO_c^M} \quad (3.8)$$

Việc thực hiện so sánh giữa hai giá trị b và c tương đương với việc kiểm tra tích vô hướng $\overrightarrow{ihv_b(b)}.\overrightarrow{ihv_c(c)}$ như (3.9):

$$\begin{cases} \text{- Nếu } \overrightarrow{ihv_c(c)}.\overrightarrow{ihv_b(b)} > 0 \text{ thì } b > c \\ \text{- Nếu } \overrightarrow{ihv_c(c)}.\overrightarrow{ihv_b(b)} = 0 \text{ thì } b = c \\ \text{- Nếu } \overrightarrow{ihv_c(c)}.\overrightarrow{ihv_b(b)} < 0 \text{ thì } b < c \end{cases} \quad (3.9)$$

Nhận xét 3.2: Lưu ý hai vector giấu tin $\overrightarrow{ihv_b(b)}$ và $\overrightarrow{ihv_c(c)}$ có thiết kế và thời điểm xây dựng khác nhau. Ta sử dụng kết quả $\overrightarrow{ihv_b(b)}.\overrightarrow{ihv_c(c)}$ để xác định thứ tự của b và c, nhưng không thể sử dụng $\overrightarrow{ihv_b(b_i)}.\overrightarrow{ihv_b(b_j)}$ để kiểm tra quan hệ giữa hai giá trị b_i và b_j (với $b_i = v_i^B$, $b_j = v_j^B$). Các vector $\overrightarrow{ihv_b(b)}$ chủ yếu được sinh ra trong lần đầu biến đổi CSDLQH rõ về CSDLQH mã hóa, trong khi đó $\overrightarrow{ihv_c(c)}$ được sinh ra và sử dụng mỗi khi có các yêu cầu truy vấn khoảng $[l, h]$ của người dùng.

3.4.2. Thuật toán biến đổi NewBucketIndex thành vector giấu tin IHV

Thuật toán $\boldsymbol{Transform\_NewBucketIndex\_To\_IHV}$ trình bày dưới đây sẽ mô tả toàn bộ các bước biến đổi mỗi giá trị NewBucketIndex $b = v_i^B$ $(1 \le i \le z)$ về vector giấu tin tương ứng $\overrightarrow{ihv_b(b)} = \overrightarrow{UO_b^M}$ trong biểu thức (3.8).

Chú ý: Thuật toán $Transform\_NewBucketIndex\_To\_IHV$ cũng có thể được sử dụng để biến đổi các giá trị rõ v_i về vector giấu tin $\overrightarrow{ihv_b(v_i)}$ nếu coi $b = v_i$.

Thuật toán 3.2: $\boldsymbol{Transform\_NewBucketIndex\_To\_IHV}$
Input $b = v_i^B$: là giá trị NewBucketIndex đại diện cho giá trị rõ v_i; Vector cơ sở $\overrightarrow{U_b}$ bí mật có chiều dài $k - 2$; Cặp vị trí bí mật $(pos1, pos2) \mid 1 \le pos1, pos2 \le k$;

Ma trận bí mật M khả nghịch và có kích cỡ kxk;

Output

$\overrightarrow{\imath hv_b(b)}$ là vector giấu tin của giá trị b;

Các bước thực hiện:

Tạo vector $\overrightarrow{O_b} = (b, -1)$, sau đó thực hiện tiếp các bước sau:

(1) Hoà nhập vector $\overrightarrow{O_b}$ vào vector $\overrightarrow{U_b}$ tại các vị trí tương ứng $(pos1, pos2)$ để nhận được vector nhúng $\overrightarrow{UO_b}$;

(2) Chọn ngẫu nhiên số nguyên dương r_b, tính lại $\overrightarrow{UO_b} = r_b * \overrightarrow{UO_b}$;

(3) Tính $M_b = M^{-1}$, biến đổi vector $\overrightarrow{UO_b}$ thành ma trận UO_b cỡ kx1;

(4) Xác định ma trận UO_b^M bằng công thức: $UO_b^M = M_b$x UO_b;

(5) Biến đổi ma trận UO_b^M về vector $\overrightarrow{UO_b^M}$;

(6) $\overrightarrow{\imath hv_b(b)} = \overrightarrow{UO_b^M}$;

(7) Return $\overrightarrow{\imath hv_b(b)}$;

Cho tập $V^B = \{v_1^B, v_2^B, \dots, v_z^B\}$, áp dụng thuật toán $Transform\_NewBucketIndex\_To\_IHV$ cho tất cả các giá trị $b = v_i^B \in V^B$ $(1 \le i \le z)$ ta sẽ được tập vector $IHV(V^B) = \{\overrightarrow{\imath hv_b(v_1^B)}, \overrightarrow{\imath hv_b(v_2^B)}, \dots, \overrightarrow{\imath hv_b(v_z^B)}\}$. Tập $IHV(V^B)$ được lưu trữ và phục vụ truy vấn tại CS thay cho tập V^B.

3.4.3. Thuật toán biến đổi toán hạng so sánh thành vector giấu tin IHV

Thuật toán $\boldsymbol{Transform\_ComparisonOperand\_To\_IHV}$ trình bày dưới đây sẽ mô tả toàn bộ các bước biến đổi mỗi giá trị toán hạng so sánh c về vector giấu tin tương ứng $\overrightarrow{\imath hv_c(c)}$ trong biểu thức (3.8).

Chú ý: Tham số đầu vào c của **Thuật toán 3.3** có thể là: $c_l = Min(l^B)$, $c_h = Max(h^B)$ hoặc là l, h. Khi đó đầu ra sẽ tương ứng là các vector giấu tin $\overrightarrow{\imath hv_c(c_l)}$, $\overrightarrow{\imath hv_c(c_h)}$ hoặc $\overrightarrow{\imath hv_c(l)}$ và $\overrightarrow{\imath hv_c(h)}$.

Thuật toán 3.3: $\boldsymbol{Transform\_ComparisonOperand\_To\_IHV}$

Input

c: là các giá trị $c_l = Min(l^B)$ hoặc $c_h = Max(h^B)$;

Vector cơ sở $\overrightarrow{U_c}$ bí mật có chiều dài $k - 2$;

Cặp vị trí bí mật $(pos1, pos2) | \ 1 \le pos1, pos2 \le k$;

Ma trận bí mật M khả nghịch và có kích cỡ kxk;

Output

$\overrightarrow{\imath hv_c(c)} = \overrightarrow{UO_c^M}$ là vector giấu tin của giá trị c;

Các bước thực hiện:

Tạo vector $\overrightarrow{O_c} = (1, c)$, sau đó thực hiện các bước sau:

(1) Hoà nhập vector $\overrightarrow{O_c}$ vào vector $\overrightarrow{U_c}$ tại các vị trí tương ứng $(pos1, pos2)$ để nhận được vector nhúng $\overrightarrow{UO_c}$;

(2) Chọn ngẫu nhiên số nguyên dương r_c, tính lại $\overrightarrow{UO_c} = r_c * \overrightarrow{UO_c}$;

(3) Tính $M_c = M^T$, biến đổi vector $\overrightarrow{UO_c}$ thành ma trận UO_c cỡ kx1;

(4) Xác định ma trận UO_c^M bằng công thức: $UO_c^M = M_c$x UO_c;

(5) Biến đổi ma trận UO_c^M về vector $\overrightarrow{UO_c^M}$;

(6) $\overrightarrow{ihv_c(c)} = \overrightarrow{UO_c^M}$;

(7) **Return** $\overrightarrow{ihv_c(c)}$;

Nhận xét 3.3: Nhờ có các giá trị ngẫu nhiên r_b, r_c nên với cùng một giá trị b, c, mỗi lần thực thi các thuật toán $Transform\_NewBucketIndex\_To\_IHV$, $Transform\_ComparisonOperand\_To\_IHV$ sẽ cho ra các kết quả khác nhau. Điều này giúp tăng cường bảo mật cho vector giấu tin trên CS.

3.5. Xây dựng cấu trúc dữ liệu IHV_ B$^+$Tree hỗ trợ tổ chức lưu trữ và truy vấn khoảng hiệu quả trên các vector IHV

Một cách đơn giản trong tiếp cận lưu trữ vector giấu tin $\overrightarrow{ihv_b(v_i^B)}$ (chỉ mục mù hỗ trợ tìm kiếm trên CS) là bổ sung thêm thuộc tính IHV vào quan hệ $R^E(ID, X_1, X_2, \ldots X_t^E, IHV \ldots X_n)$. Khi đó biểu thức điều kiện truy vấn khoảng "$\left(\overrightarrow{ihv_c(c_l)}.\overrightarrow{ihv_b(b)} \geq 0\right)$ and $\left(\overrightarrow{ihv_c(c_h)}.\overrightarrow{ihv_b(b)} \leq 0\right)$" sẽ được biểu diễn trong lệnh SQL tại CS như sau: "*Select * from* R^E *where* $IHV.\overrightarrow{ihv_c(c_l)} \geq 0$ *and* $IHV.\overrightarrow{ihv_c(c_h)} \leq 0$". Tuy nhiên, nếu để thực hiện trực tiếp câu truy vấn này trên CSDLQH mã hóa gây ra một số hạn chế: *Thứ nhất* tốn rất nhiều chi phí cho quét duyệt toàn bộ R^E để kiểm tra từng bản ghi có thoả mãn biểu thức điều kiện hay không; *Thứ hai* là các giá trị trong trường IHV là các vector giấu tin đại diện cho các giá trị NewBucketIndex nên kết quả trả về sau truy vấn sẽ dư thừa (bao gồm những bản ghi thoả mãn điều kiện $c_l \leq v_i^B \leq c_h$ nhưng lại không thoả mãn $l \leq v_i \leq h$ với $1 \leq i \leq z$ do tính chất một NewBucketIndex có thể đại diện cho nhiều giá trị rõ khác nhau).

Ý tưởng để giải quyết hai hạn chế nêu trên là sử dụng cấu trúc dữ liệu B$^+$Tree truyền thống [34], [52], [121] để xây dựng cây chỉ mục mới nhằm tăng tốc tìm kiếm trên trường IHV (ta gọi cấu trúc dữ liệu này là cây **IHV_ B$^+$Tree**).

Để xây dựng IHV_ B$^+$Tree, trước tiên cần chuẩn bị dữ liệu như **Bảng 3.2 (**mở rộng từ **Bảng 3.1)**. Cụ thể, **Bảng 3.2** liệt kê tất cả các bản ghi với: $\boldsymbol{r\_id}$ là giá trị định

danh duy nhất của bản ghi trong quan hệ R^E (r_id lưu trữ trong trường ID); $\boldsymbol{v_i^B}$ là giá trị NewBucketIndex của v_i và $\boldsymbol{\overrightarrow{ihv_b(v_i)}}$, $\boldsymbol{\overrightarrow{ihv_b(v_i^B)}}$ lần lượt là các vector giấu tin cho v_i và v_i^B).

Bảng 3.2. Các thông tin sử dụng cho xây dựng cây IHV_ B$^+$Tree

$\boldsymbol{r\_id}$	$\boldsymbol{v_i}$	$\boldsymbol{v_i^B}$	$\boldsymbol{\overrightarrow{ihv_b(v_i)}}$	$\boldsymbol{\overrightarrow{ihv_b(v_i^B)}}$
2471	3	$B_1 = 7$	$\overrightarrow{ihv_b(3)}$	$\overrightarrow{ihv_b(7)}$
1206	3	$B_2 = 12$	$\overrightarrow{ihv_b(3)}$	$\overrightarrow{ihv_b(12)}$
3607	3	$B_2 = 12$	$\overrightarrow{ihv_b(3)}$	$\overrightarrow{ihv_b(12)}$
9347	3	$B_1 = 7$	$\overrightarrow{ihv_b(3)}$	$\overrightarrow{ihv_b(7)}$
7598	7	$B_2 = 12$	$\overrightarrow{ihv_b(7)}$	$\overrightarrow{ihv_b(12)}$
7176	7	$B_3 = 25$	$\overrightarrow{ihv_b(7)}$	$\overrightarrow{ihv_b(25)}$
9476	7	$B_2 = 12$	$\overrightarrow{ihv_b(7)}$	$\overrightarrow{ihv_b(12)}$
3552	9	$B_3 = 25$	$\overrightarrow{ihv_b(9)}$	$\overrightarrow{ihv_b(25)}$
2482	9	$B_3 = 25$	$\overrightarrow{ihv_b(9)}$	$\overrightarrow{ihv_b(25)}$
9340	14	$B_3 = 25$	$\overrightarrow{ihv_b(14)}$	$\overrightarrow{ihv_b(25)}$
9380	14	$B_4 = 48$	$\overrightarrow{ihv_b(14)}$	$\overrightarrow{ihv_b(48)}$
3367	14	$B_4 = 48$	$\overrightarrow{ihv_b(14)}$	$\overrightarrow{ihv_b(48)}$
8116	25	$B_4 = 48$	$\overrightarrow{ihv_b(25)}$	$\overrightarrow{ihv_b(48)}$
3485	25	$B_5 = 92$	$\overrightarrow{ihv_b(25)}$	$\overrightarrow{ihv_b(92)}$
8650	25	$B_5 = 92$	$\overrightarrow{ihv_b(25)}$	$\overrightarrow{ihv_b(92)}$
9159	25	$B_5 = 92$	$\overrightarrow{ihv_b(25)}$	$\overrightarrow{ihv_b(92)}$
7868	25	$B_4 = 48$	$\overrightarrow{ihv_b(25)}$	$\overrightarrow{ihv_b(48)}$
6500	50	$B_5 = 92$	$\overrightarrow{ihv_b(50)}$	$\overrightarrow{ihv_b(92)}$
1143	50	$B_6 = 119$	$\overrightarrow{ihv_b(50)}$	$\overrightarrow{ihv_b(119)}$
…	…	…	…	…
9669	320	$B_{16} = 499$	$\overrightarrow{ihv_b(320)}$	$\overrightarrow{ihv_b(499)}$

Thuật toán xây dựng cây IHV_ B$^+$Tree được mô tả như sau:

Thuật toán 3.4: *Build_IHV_B$^+$Tree*

Input

Các giá trị trong **Bảng 3.2**;

m: bậc của cây IHV_B^+Tree;

Output

IHV_B$^+$Tree

Các bước thực hiện:

(1) Xây dựng cây B$^+$Tree bậc m từ tập B là tập giá trị phân biệt của $V^B = \{v_1^B, v_2^B, \dots, v_z^B\}$, $B \subset V^B$.

(2) Mã hóa nút gốc và các nút trong của cây B$^+$Tree:

- Cấu trúc của nút gốc và nút trong có dạng: (ptr_node_0, k_1, ptr_node_1, k_2, ptr_node_2, k_3, … , ptr_node_{m-1}, k_m, ptr_node_m) với $k_i \in B$ gọi là khóa (được sắp theo thứ tự $k_i < k_{i+1}, 1 \le i \le m-1$) và ptr_node_i là con trỏ tham chiếu tới cây con mà giá trị khóa ($keyvalue$) ở tất cả các nút của cây con đều thỏa mãn điều kiện $k_i \le keyvalue < k_{i+1}$. Khi áp dụng cây B$^+$Tree trên CSDLQH, con trỏ ptr_node_i sẽ chứa địa chỉ của DiskBlock lưu trữ các nút.

- Để mã hoá nút gốc và nút trong của cây, ta thay các khóa k_i bằng vector giấu tin $\overrightarrow{ihv_b(k_i)}$ (chính là giá trị $\overrightarrow{ihv_b(v_i^B)}$ tương ứng). Khi đó cấu trúc nút gốc và nút trong có dạng: (ptr_node_0, $\overrightarrow{ihv_b(k_1)}$, ptr_node_1, $\overrightarrow{ihv_b(k_2)}$, ptr_node_2, $\overrightarrow{ihv_b(k_3)}$, … , ptr_node_{m-1}, $\overrightarrow{ihv_b(k_m)}$, ptr_node_m).

(3) Cấu trúc lại và mã hóa các nút lá của cây B$^+$Tree:

Trong cây B$^+$Tree, các nút lá chứa đầy đủ các khóa của nút gốc và các nút trong. Trong mỗi nút lá khóa sắp xếp tăng dần từ trái qua phải. Các nút lá được liên kết với nhau thành một danh sách và giữa các nút lá thì khóa của nút trái nhỏ hơn khóa của nút phải.

Giả sử có n nút lá, khi đó cấu trúc mặc định của nút lá thứ i ($1 \le i \le n$) trong cây B$^+$Tree có dạng: ($leaf\_previous_i$, $< k_{i(1)}, ptr\_row_{i(1)} >$, $< k_{i(2)}, ptr\_row_{i(2)} >$, $< k_{i(3)}, ptr\_row_{i(3)} >$ …, $< k_{i(m'-1)}, ptr\_row_{i(m'-1)} >$, $< k_{i(m')}, ptr\_row_{i(m')} >$, $leaf\_next_i$) với $m' \le m$, $leaf\_previous_i$ và $leaf\_next_i$ lần lượt là các con trỏ lưu địa chỉ của các DiskBlock chứa nút lá thứ $i-1$ và nút lá thứ $i+1$. Còn $ptr\_row_{i(j)}$ ($1 \le j \le m'$) là các DataPointer, mỗi $ptr\_row_{i(j)}$ trỏ tới một danh sách liên kết lưu trữ các bản ghi được đại diện bởi khóa NewBucketIndex $k_{i(j)}$.

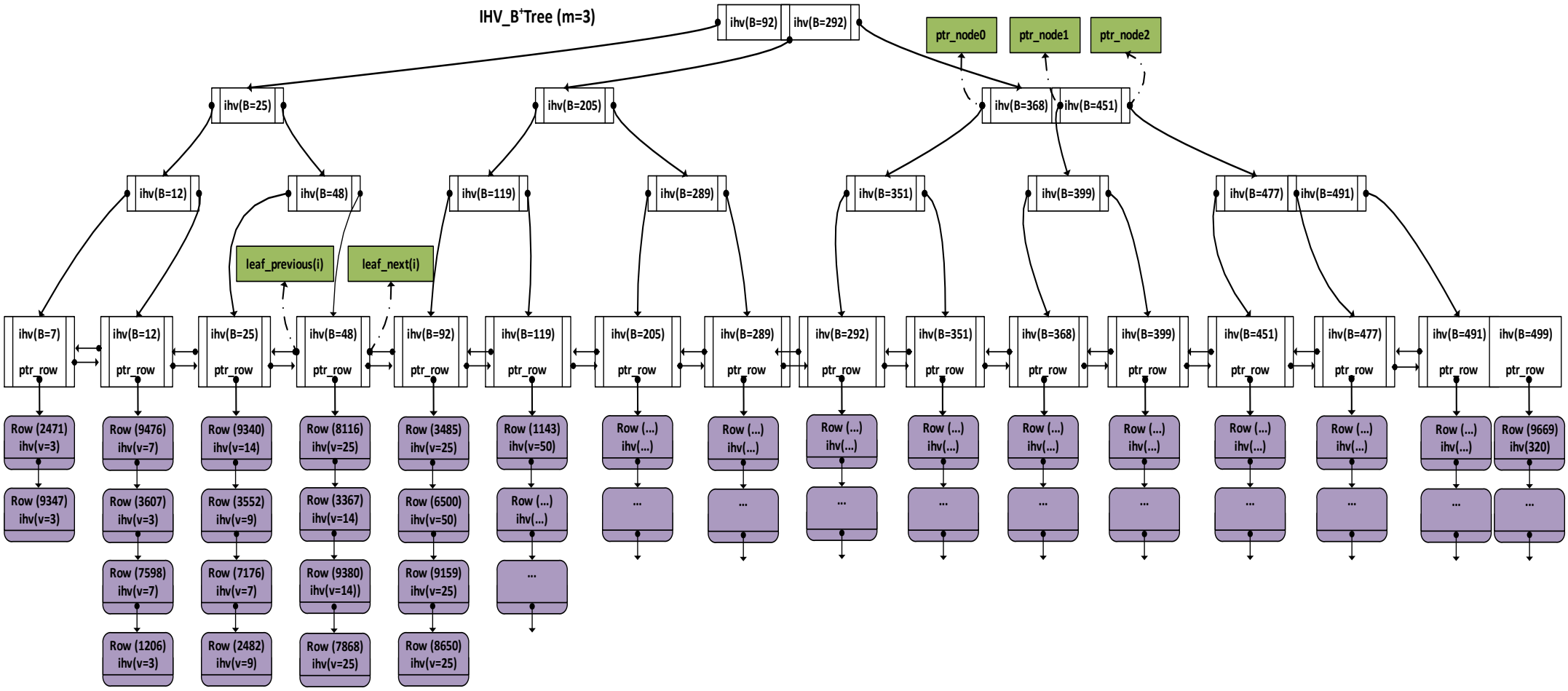
Để tái cấu trúc và mã hoá nút lá, thực hiện tuần tự như sau:

- *Thứ nhất:* biến đổi mỗi bộ $< k_{i(j)}, ptr\_row_{i(j)} >$ thành một bộ $< \overrightarrow{ihv_b(k_{i(j)})}, ptr\_row_{i(j)} >$, trong đó $k_{i(j)} \in B$.

- *Thứ hai:* $ptr\_row_{i(j)}$ trỏ tới đầu của danh sách liên kết $List_{i(j)} = \{row_1, row_2, \ldots row_{i'}, \ldots, row_{n'}\}$ với n' là số lượng bản ghi chứa NewBucketIndex $k_{i(j)}$. Mỗi phần tử trong danh sách liên kết là một bộ $row_{i'} = < row(r\_id), \overrightarrow{ihv_b(v_i)}, ptr\_row\_next >$ với $row(r\_id)$ là một bản ghi trong quan hệ $R^E(ID, X_1, X_2, \ldots X_t^E, \ldots X_n)$. Giá trị $\overrightarrow{ihv_b(v_i)}$ là vector giấu tin cho một giá trị rõ v_i được đại diện bởi NewBucketIndex $k_{i(j)}$. Còn ptr_row_next là con trỏ lưu địa chỉ của bộ $row_{i'+1}$ tiếp theo, chú ý thứ tự của các phần tử $row_{i'}$ trong danh sách liên kết được sắp xếp ngẫu nhiên.

(4) Return IHV_B^+Tree

Dựa trên dữ liệu trong **Bảng 3.2**, cây IHV_B^+Tree bậc 3 mô tả như **Hình 3.3**.



Hình 3.3. Mô tả cây IHV_B^+Tree

3.6. Truy vấn khoảng trong lược đồ ESIT-SSE triển khai trên mô hình DAS-PROXY

3.6.1. Quy trình truy vấn khoảng

Theo mô hình đề xuất trong **Hình 3.1**, từ yêu câu lệnh truy vấn khoảng rõ "*Select * from R where X_t between l and h"* được yêu cầu bởi EU, quá trình thực hiện truy vấn của ESIT-SSE trên R^E diễn ra qua 4 bước, bao gồm:

Bước 1 – Biến đổi toán hạng so sánh trong biểu thức điều kiện truy vấn khoảng: Bao gồm việc tiếp nhận câu truy vấn khoảng rõ từ EU, sau đó thực hiện tách lấy các toán hạng l, h. Xác định các giá trị NewBucketIndex cận dưới $c_l = Min(l^B)$ và cận trên $c_h = Max(h^B)$ tương ứng. Cuối cùng biến đổi $[l, h]$ và $[c_l, c_h]$ thành cặp khoảng truy vấn có dạng thức vector giấu tin là $[\overrightarrow{\imath hv_c(l)}, \overrightarrow{\imath hv_c(h)}]$ và $[\overrightarrow{\imath hv_c(c_l)}, \overrightarrow{\imath hv_c(c_h)}]$. Bước này thuộc **Giai đoạn 3** của lược đồ ESIT-SSE, thực thi tại Proxy qua *Thuật toán 3.3* $Transform\_ComparisonOperand\_To\_IHV$ và *Thuật toán 3.5* $Transform\_RangeQueryCondition$.

Bước 2 – Truy vấn khoảng trên cấu trúc dữ liệu IHV_B^+Tree: Bước này thuộc **Giai đoạn 4** của lược đồ ESIT-SSE. Sau khi tiếp nhận cặp khoảng truy vấn $[\overrightarrow{\imath hv_c(l)}, \overrightarrow{\imath hv_c(h)}]$, $[\overrightarrow{\imath hv_c(c_l)}, \overrightarrow{\imath hv_c(c_h)}]$ từ **Bước 1**, tại CS *Thuật toán 3.6* $RangeQuery\_On\_IHV\_B^+Tree$ sẽ mô tả quá trình thực thi qua ba bước:

Bước 2.1. Duyệt nút gốc và nút trong của cây IHV_B^+Tree để tìm các khóa (là các vector giấu tin đại điện cho NewBucketIndex) thuộc khoảng $[\overrightarrow{\imath hv_c(c_l)}, \overrightarrow{\imath hv_c(c_h)}]$. Lặp lại quá trình này cho đến khi nhận được một tập các khóa tại nút lá (các nút lá này liên kết với các nút con thỏa mãn điều kiện đã duyệt ở trên). Tập các khóa này tương ứng với tập các NewBucketIndex thuộc khoảng $[c_l, c_h]$.

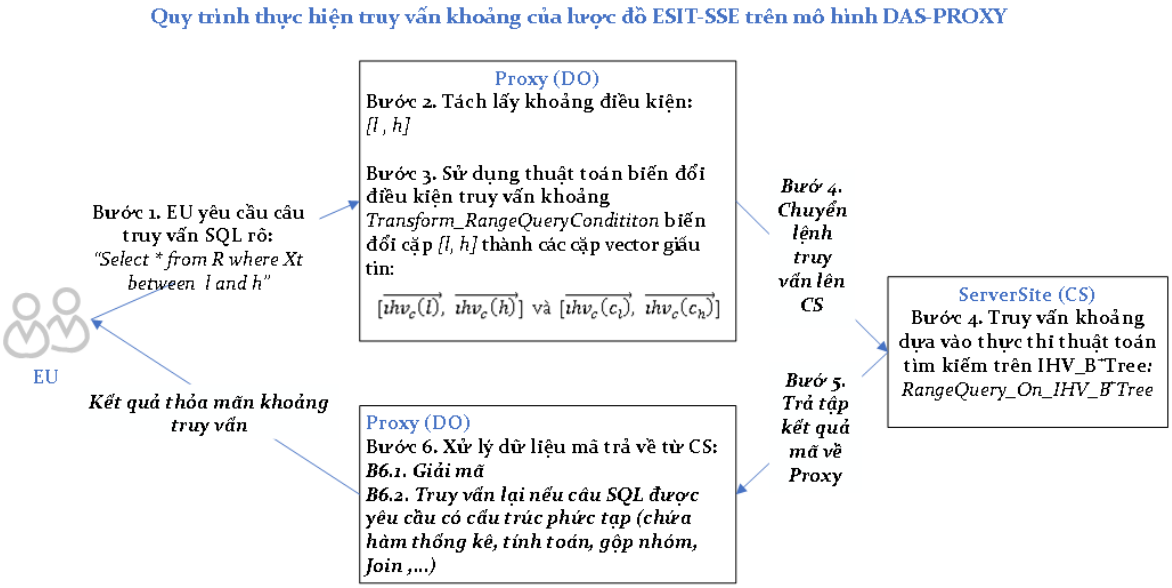
Bước 2.2. Duyệt từ trái qua phải các nút lá tìm được từ ***Bước 1*** và chỉ lấy ra những phần tử trong danh sách $List$ gắn với nút lá có giá trị thuộc khoảng $[\overrightarrow{\imath hv_c(l)}, \overrightarrow{\imath hv_c(h)}]$. Tập các phần tử này liên kết 1-1 đến các bản ghi mà giá trị rõ tương ứng chắc chắn thỏa mãn khoảng truy vấn rõ $[l, h]$.

Bước 2.3. Từ danh sách các phần tử nhận được trong ở ***Bước 2***, cho phép truy xuất đến các bản ghi mã $row(r\_id)$ tương ứng trong quan hệ R^E. Tập bản ghi mã sẽ được trả lại Proxy.

Bước 3 – Giải mã dữ liệu và truy vấn lại: Bước này thuộc **Giai đoạn 5** của

lược đồ ESIT-SSE. Giải mã giá trị cột X_t^E trong các bản ghi được trả về từ **Bước 2**. Trong trường hợp lược đồ phục vụ các yêu cầu truy vấn mở rộng phức tạp thì cần thực hiện truy vấn lại bằng câu lệnh rõ trên tập bản ghi đã giải mã. Giai đoạn này thực hiện tại Proxy và kết quả cuối được trả về EU.

Như vậy tại CS, cấu trúc dữ liệu IHV_B^+Tree hỗ trợ tìm kiếm các bản ghi mã trong quan hệ R^E thoả mãn các khoảng truy vấn trên vector giấu tin một cách nhanh chóng mà không phải sử dụng tới cơ chế quét từng bản ghi của bảng dữ liệu. Quá trình truy vấn khoảng $[l, h]$ được mô tả trong **Hình 3.4**.



Hình 3.4. Quy trình truy vấn khoảng trên lược đồ ESIT-SSE

3.6.2. Thuật toán biến đổi khoảng truy vấn trên dữ liệu rõ sang các khoảng truy vấn trên vector giấu tin (IHV)

Theo **Thuật toán 3.4: *Build_IHV_B+Tree***, nút gốc và các nút trong có khóa là vector giấu tin của NewBucketIndex ($\overrightarrow{ihv_b(v_l^B)}$), trong khi đó nút lá lại liên kết tới các danh sách ($LIST$) có các phần tử là vector giấu tin của dữ liệu rõ ($\overrightarrow{ihv_b(v_l)}$).

Theo **Mục 3.6.1** thì quá trình truy vấn khoảng bản chất là việc duyệt cây IHV_B^+Tree theo hai khoảng $[\overrightarrow{ihv_c(c_l)}, \overrightarrow{ihv_c(c_h)}]$ và $[\overrightarrow{ihv_c(l)}, \overrightarrow{ihv_c(h)}]$. Cụ thể:

- *Duyệt nút gốc và nút trong theo biểu thức:* $\overrightarrow{ihv_c(c_l)}.\ \overrightarrow{ihv_b(v_l^B)} \geq 0$ và $\overrightarrow{ihv_c(c_h)}.\ \overrightarrow{ihv_b(v_l^B)} \leq 0$.

- *Duyệt danh sách LIST trong các nút lá theo biểu thức:* $\overrightarrow{ihv_c(l)}.\ \overrightarrow{ihv_b(v_l)} \geq 0$ và $\overrightarrow{ihv_c(h)}.\ \overrightarrow{ihv_b(v_l)} \leq 0$.

Quy trình để tạo các khoảng truy vấn $[\overrightarrow{ihv_c(l)}, \overrightarrow{ihv_c(h)}]$ và $[\overrightarrow{ihv_c(c_l)},$

$\overrightarrow{\imath hv_c(c_h)}$] từ khoảng rõ $[l, h]$ để phục vụ quá trình nêu trên được trình bày trong *Thuật toán 3.5:* $Transform\_RangeQueryCondition$ như sau:

Thuật toán 3.5: $\boldsymbol{Transform\_RangeQueryCondition}$

Input

l, h: các toán hạng so sánh của điều kiện truy vấn khoảng trên dữ liệu rõ;

Bảng các thông tin bí mật để sinh NewBucketIndex (**Bảng 3.1**);

Vector cơ sở $\overrightarrow{U_c}$ bí mật có chiều dài $k - 2$;

Cặp vị trí bí mật $(pos1, pos2) | \ 1 \leq pos1, pos2 \leq k$;

Ma trận bí mật M khả nghịch và có kích cỡ kxk;

Output

Các khoảng truy vấn trên vector giấu tin:

$[\overrightarrow{\imath hv_c(l)}, \ \overrightarrow{\imath hv_c(h)}]$ và $[\overrightarrow{\imath hv_c(c_l)}, \ \overrightarrow{\imath hv_c(c_h)}]$;

Các bước thực hiện

Dựa vào "Bảng các thông tin bí mật để sinh NewBucketIndex", thực hiện :

(1) Xác định miền giá trị rõ thoả mãn điều kiện truy vấn khoảng $[l, h]$:

- Tìm $d_i.v$ lớn nhất mà nhỏ hơn hoặc bằng l, ký hiệu là v_l;
- Tìm $d_i.v$ nhỏ nhất mà lớn hơn hoặc bằng h, ký hiệu là v_h;

(2) Xác định l^B là danh sách các NewBucketIndex đại diện cho v_l, sau đó tìm $c_l = \ Min(l^B)$ là giá trị NewBucketIndex nhỏ nhất trong danh sách l^B;

(3) Xác định h^B là danh sách các NewBucketIndex đại diện cho v_h, sau đó tìm $c_h = \ Max(h^B)$ là giá trị NewBucketIndex lớn nhất trong danh sách h^B;

(4) Tính:

$\overrightarrow{UO_l^M} = \boldsymbol{Transform\_ComparisonOperand\_To\_IHV}(l, \overrightarrow{U_c}, pos1, pos2, M)$;

$\overrightarrow{UO_{\mathrm{h}}^M} = \boldsymbol{Transform\_ComparisonOperand\_To\_IHV}(h, \overrightarrow{U_c}, pos1, pos2, M)$;

$\overrightarrow{UO_{c_l}^M} = \boldsymbol{Transform\_ComparisonOperand\_To\_IHV}(c_l, \overrightarrow{U_c}, pos1, pos2, M)$;

$\overrightarrow{UO_{c_h}^M} = \boldsymbol{Transform\_ComparisonOperand\_To\_IHV}(c_h, \overrightarrow{U_c}, pos1, pos2, M)$;

(5) Đặt:

$\overrightarrow{\imath hv_c(l)} = \overrightarrow{UO_l^M}$ và $\overrightarrow{\imath hv_c(h)} = \ \overrightarrow{UO_h^M}$

$\overrightarrow{\imath hv_c(c_l)} = \overrightarrow{UO_{c_l}^M}$ và $\overrightarrow{\imath hv_c(c_h)} = \ \overrightarrow{UO_{c_h}^M}$

(6) Return $[\overrightarrow{\imath hv_c(l)}, \ \overrightarrow{\imath hv_c(h)}]$ và $[\overrightarrow{\imath hv_c(c_l)}, \ \overrightarrow{\imath hv_c(c_h)}]$;

Từ **Nhận xét 3.3**, ta rút ra nhận định: với cùng khoảng $[l, h]$ đầu vào thì cặp kết quả đầu ra $[\overrightarrow{\imath hv_c(l)}, \ \overrightarrow{\imath hv_c(h)}]$ và $[\overrightarrow{\imath hv_c(c_l)}, \ \overrightarrow{\imath hv_c(c_h)}]$ sẽ không giống nhau giữa những lần thực thi thuật toán $Transform\_RangeQueryCondition$.

3.6.3. Thuật toán truy vấn khoảng trên cấu trúc dữ liệu $\boldsymbol{IHV\_B^+Tree}$

Quy trình truy vấn với hai khoảng $[\overrightarrow{\imath hv_c(l)}, \ \overrightarrow{\imath hv_c(h)}]$ và $[\overrightarrow{\imath hv_c(c_l)}, \ \overrightarrow{\imath hv_c(c_h)}]$

trên cấu trúc dữ liệu IHV_B^+Tree được mô tả chi tiết trong **Thuật toán 3.6**.

Thuật toán 3.6: *RangeQuery_On_IHV_B+Tree*

Input

IHV_B^+Tree;

$R^E(ID, X_1, X_2, \dots X_t^E, \dots X_n)$;

$[\overrightarrow{\imath hv_c(l)}, \overrightarrow{\imath hv_c(h)}]$ và $[\overrightarrow{\imath hv_c(c_l)}, \overrightarrow{\imath hv_c(c_h)}]$;

Output

Tập bản ghi mã trong quan hệ $R^E(ID, X_1, X_2, \dots X_t^E, \dots X_n)$ thỏa mãn $l \le X_t \le h$;

Các bước thực hiện:

(1). Bắt đầu từ nút gốc của cây ***IHV_B+Tree***. So sánh $\overrightarrow{\imath hv_c(c_l)}$ với các khóa $\overrightarrow{\imath hv_b(k_\imath)}$ ở nút gốc (ptr_node_0, $\overrightarrow{\imath hv_b(k_1)}$, ptr_node_1, $\overrightarrow{\imath hv_b(k_2)}$, ptr_node_2, $\overrightarrow{\imath hv_b(k_3)}$, ... , ptr_node_{m-1}, $\overrightarrow{\imath hv_b(k_m)}$, ptr_node_m);

(2). Nếu $\overrightarrow{\imath hv_b(k_1)}.\overrightarrow{vec(c_l)} > 0$ thì chuyển đến cây con bên trái của $\overrightarrow{\imath hv_b(k_1)}$, được trỏ bởi ptr_node_0;

(3). Nếu $\overrightarrow{\imath hv_b(k_1)}.\overrightarrow{\imath hv_c(c_l)} = 0$ thì thực hiện so sánh tiếp với $\overrightarrow{\imath hv_b(k_2)}$:

- Nếu $\overrightarrow{\imath hv_b(k_2)}.\overrightarrow{\imath hv_c(c_l)} > 0$ thì chuyển đến cây con bên trái của $\overrightarrow{\imath hv_b(k_2)}$, được trỏ bởi ptr_node_1;
- Nếu $\overrightarrow{\imath hv_b(k_2)}.\overrightarrow{\imath hv_c(c_l)} < 0$ thì tiếp tục so sánh với $\overrightarrow{\imath hv_b(k_3)}$, $\overrightarrow{\imath hv_b(k_4)}$, ... $\overrightarrow{\imath hv_b(k_m)}$ như ở bước **(2)** và bước **(3)**;

(4). Lặp lại các bước trên trong mỗi cây con cho đến khi duyệt tới nút lá;

(5). Giả sử sau khi kết thúc bước **(4)**, quá trình duyệt dừng ở node lá thứ i có cấu trúc $(leaf_{previous_i}, < \overrightarrow{\imath hv_b(k_{\imath(1)})}, ptr_{row_{i(1)}} >, < \overrightarrow{\imath hv_b(k_{\imath(2)})}, ptr\_row_{i(2)} >, < \overrightarrow{\imath hv_b(k_{\imath(3)})}, ptr\_row_{i(3)} > \dots, < \overrightarrow{\imath hv_b(k_{\imath(m'-1)})}, ptr\_row_{i(m'-1)} >, < \overrightarrow{\imath hv_b(k_{\imath(m')})}, ptr\_row_{i(m')} >, leaf\_next_i)$. Khi đó thuật toán thực hiện tiếp như sau:

(5.1). Bắt đầu từ bộ $< \overrightarrow{\imath hv_b(k_{\imath(1)})}, ptr\_row_{i(1)} >$, nếu $\overrightarrow{\imath hv_b(k_{\imath(1)})}.\overrightarrow{\imath hv_c(c_l)} \ge 0 \; and \; \overrightarrow{\imath hv_b(k_{\imath(1)})}.\overrightarrow{\imath hv_c(c_h)} \le 0$ thì thực hiện tiếp bước **(5.2)**, ngược lại dừng bước **(5)**;

(5.2). Xét bộ $< \overrightarrow{\imath hv_b(k_{\imath(1)})}, ptr\_row_{i(1)} >$, truy xuất vào danh sách liên kết $List_{i(1)} = \{row_1, row_2, \dots row_{i'}, \dots, row_{n'}\}$ được trỏ bởi $ptr\_row_{i(1)}$;

(5.3). Duyệt tuần tự từng phần tử của danh sách liên kết $List_{i(1)}$. Với mỗi phần tử $row_{i'} = < row(ID), \overrightarrow{\imath hv_b(v_\imath)}, ptr\_row\_next >$ kiểm tra:

- Nếu $\overrightarrow{\imath hv_b(v_\imath)}.\overrightarrow{\imath hv_c(l)} \ge 0 \; and \; \overrightarrow{\imath hv_b(v_\imath)}.\overrightarrow{\imath hv_c(h)} \le 0$ thì trả về giá trị $row(ID)$;
- Ngược lại thì bỏ qua $row_{i'}$ và tiếp tục lặp lại quá trình kiểm tra ở trên với các phần tử kế tiếp trong danh sách cho đến khi $i' = n'$;

(5.4). Lặp lại bước **(5.1)** đến bước **(5.3)** đối với các bộ $< k_{i(2)}, ptr\_row_{i(2)} >$, $< k_{i(3)}, ptr\_row_{i(3)} > \dots < k_{i(m')}, ptr\_row_{i(m')} >$;

(5.6). Lặp lại bước **(5)** đối với các node lá thứ $i+1, \; i+2, \dots, \; n$;

(6). Tham chiếu các giá trị $row(ID)$ tìm được vào R^E để tìm ra các bản ghi mã

thỏa mãn $l \leq X_t \leq h$;
(7). Trả về tập các bản ghi mã thỏa mãn $l \leq X_t \leq h$;

3.7. Phân tích bảo mật của lược đồ ESIT-SSE

Mục này góp phần đánh giá ***tính hiệu quả của lược đồ*** ESIT-SSE qua tiêu chí ***bảo mật***. Tương tự cách luận giải xác định vấn đề bảo mật cần giải quyết trong lược đồ DIQ-SSE tại **Mục 2.5**, đối với lược đồ ESIT-SSE cũng không phân tích cơ chế giữ bí mật các $Metadata$ cùng các thuật toán ủy quyền thực thi tại PROXY, an toàn của cửa sập, bảo mật của dữ liệu lưu trong trường X_t^E hay bảo mật đường truyền và xác thực - toàn vẹn dữ liệu. Luận án tập trung vào phân tích: ***bảo mật NewBucketIndex***, ***bảo mật vector giấu tin IHV*** và ***bảo mật cây IHV_B$^+$Tree*** dựa trên khả năng chống rò rỉ thông tin động và tĩnh của chúng.

Đối với khả năng ***bảo mật mẫu truy vấn*** (các khoảng $[\overrightarrow{ihv_c(l)},\ \overrightarrow{ihv_c(h)}]$ và $[\overrightarrow{ihv_c(c_l)},\ \overrightarrow{ihv_c(c_h)}]$ nhận được từ Proxy) thì đã được chỉ ra trong **Nhận xét 3.3** và **Mục 3.6.2** nên sẽ không trình bày thêm trong phần này.

Trong mục **3.1. Đặt bài toán và dẫn luận ý tưởng thực hiện**, lược đồ ESIT-SSE kế thừa thiết kế từ các lược đồ SSE dựa trên chỉ mục mù đáp ứng bảo mật IND-CKA2 đã được phân tích và chứng minh trong các công trình [43], [5], [33], [64], [136], [10]. Vì vậy, dựa vào các định nghĩa bảo mật của lược đồ SSE trong **PHỤ LỤC 1** và các đặc tính tương đồng trong thiết kế của lược đồ đề xuất với những lược đồ trước đó thì ESIT-SSE được coi đáp ứng mô hình bảo mật IND-CKA2. Do đó, trong mục này luận án sẽ chỉ bổ sung, làm rõ thêm một số luận điểm, phân tích về khả năng chống rò rỉ thông tin của NewBucketIndex, IHV, IHV_B$^+$Tree trước một số kịch bản, giả thuyết tấn công phổ biến.

3.7.1. Vấn đề rò rỉ thông tin tĩnh và động

Trong các giải pháp truy vấn khoảng trên dữ liệu số mã hóa trước đó [58], [47], [53], [113], quá trình phân tích bảo mật chủ yếu tập trung vào khả năng chống tấn công rò rỉ thông tin tĩnh và rò rỉ thông tin động. Rò rỉ thông tin tĩnh được hiểu là ***kẻ gian (gọi tắt là A)*** khai thác được các thông tin dựa trên những kiến thức đã có về bản rõ và bản mã, một trong những phương pháp mà A hay sử dụng sử dụng là phân tích thống kê tần suất của các bản mã. Rò rỉ thông tin động được A khai thác thông qua việc quan sát cấu trúc mẫu truy vấn và tập kết quả trả về kết hợp cùng một số kiến thức biết trước về bản rõ, bản mã để suy diễn được mẫu truy vấn tương ứng với

tập kết quả nào, ngụ ý tập kết quả ẩn chứa thông tin gì liên quan tới bản rõ.

Ta xét **Ví dụ 3.5**, **Ví dụ 3.6** làm rõ về rò rỉ thông tin tĩnh và động:

Ví dụ 3.5. Giả sử phổ điểm môn toán của các sinh viên có xác suất phân phối như sau: 30% điểm trong khoảng [1, 4], 40% điểm trong khoảng [5, 6], 25% điểm trong khoảng [7, 8] và 5% điểm trong khoảng [9, 10]. Và sử dụng các Bucket để che giấu thông tin cho dữ liệu điểm như sau: khoảng [1,4] đại diện bởi BucketIndex = 1432, khoảng [5, 6] đại diện bởi BucketIndex = 1661, khoảng [7, 8] có BucketIndex = 1889 và BucketIndex của khoảng [9, 10] bằng 2561. Mô tả rò rỉ thông tin tĩnh: do biết trước xác suất phân phối trên bản rõ nên A chỉ cần thống kê tần suất xuất hiện BucketIndex là hoàn toàn có thể đoán được Bucket đại diện cho khoảng giá trị rõ nào.

Ví dụ 3.6. Giả sử có tập BucketIndex = {24, 75, 82, 99, 108, 154} được xây dựng liên tục và không nạp chồng trên tập giá trị rõ, số lượng phần tử trong các Bucket xấp xỉ bằng nhau. Quan sát 1000 truy vấn trước đó, A có được thống kê về Top 3 tập kết quả trả về nhiều nhất như **Bảng 3.3.**

Bảng 3.3. Thống kê kết quả sau quan sát 1000 truy vấn tại CS

Tập BucketIndex được trả về sau truy vấn	**Số lần trả về tập BucketIndex**	**Số mẫu truy vấn phân biệt tương ứng**
{24}	170	15
{75}	120	110
{82, 99, 108}	100	95

Bảng 3.3 chỉ ra có 170 lần trả về tập {24} tương ứng với 15 mẫu truy vấn, từ đó A có thể suy luận được hành vi người dùng hay thực hiện truy vấn 15 khoảng nào đó và các khoảng này có miền giá trị giao nhau lớn. Với tập {75} cũng cho thấy số lần trả về lớn và số mẫu truy vấn phân biệt cũng rất lớn, điều này ngụ ý phạm vi khoảng được đại diện bởi BucketIndex = 75 là rộng (mặc dù tổng số giá trị trong các Bucket là tương đương nhau do sử dụng equi-depth) nên tập kết quả trả về có khả năng chứa giá trị 75 cao. Ngoài ra tập {82, 99, 108} cũng trả về tương đối nhiều, A có thể suy luận phạm vi của các khoảng đại diện bởi BucketIndex = {82, 99, 108} là nhỏ hơn so với BucketIndex = 75. Qua đó đoán xác suất phân phối của mỗi giá trị trong BucketIndex 82, 99 và 108 là lớn hơn so với 75 với độ tin cậy cao.

3.7.2. Phân tích bảo mật NewBucketIndex

Theo Hore và cộng sự [53], variance (phương sai) và entropy (độ hỗn loạn) là hai đại lượng quan trọng để đo độ bảo mật của các Bucket. Phương sai (ký hiệu là

σ^2) đại diện cho độ phân tán của các giá trị trong mỗi Bucket, σ^2 càng lớn thì Bucket càng bảo mật, chống lại khả năng dự đoán một giá trị Bucket đại diện cho một giá trị rõ nào đó với xác suất cao. Phương sai rõ ràng liên quan tới khả năng chống tấn công rò rỉ thông tin động của mỗi Bucket. Trong khi đó entropy (ký hiệu là H) đại diện cho sự đa dạng của các giá trị phân biệt trong một Bucket, để đạt được điều này đồng thời cho tất cả các Bucket, các tác giả ngụ ý sử dụng kỹ thuật equi-depth để chia liên tục và không nạp chồng các giá trị rõ vào các Bucket. Entropy cao làm giảm khả năng A đoán được chính xác Bucket thỏa mãn một truy vấn khoảng bất kỳ cũng như chống được phương pháp suy luận tìm thông tin bản rõ dựa vào thống kê tần suất Bucket trên CS. Vì vậy H liên quan tới khả năng chống rò rỉ thông tin tĩnh. Tuy nhiên, không có giải pháp hoàn hảo nào để đảm bảo được cả H và σ^2 đồng thời đạt giá trị tốt nhất. Các công trình cũng chỉ ra số lượng Bucket càng ít và các phần tử trong Bucket phân phối càng đồng nhất thì càng bảo mật nhưng ảnh hưởng tới hiệu năng truy vấn.

Wang và cộng sự [113] chỉ ra hạn chế của phương sai khi chưa quan tâm đến sự phân phối đồng nhất của các giá trị trong mỗi Bucket, họ đã sử dụng thêm tham số pvd để đo tính chất phân phối đồng nhất trên phương diện độ lớn của các giá trị trong mỗi Bucket. Tổng quát, một Bucket càng bảo mật nếu các giá trị trong Bucket càng được phân phối đều về cả giá trị và tần suất. Giữa các Bucket số lượng các phần tử cũng phải xấp xỉ bằng nhau, xác suất phân phối giữa các phần tử giữa các Bucket cũng không nên chênh lệch quá lớn.

Với cách xây dựng Bucket bảo toàn thứ tự và không nạp chồng, sẽ rất khó để đảm bảo các giá trị về σ^2, H, pvd đồng thời tối ưu được. Giải pháp xây dựng Bucket đạt độ bảo mật cao nhất phải là nạp chồng giá trị giữa các Bucket và không bảo toàn thứ tự. Tuy nhiên, thuật toán xây dựng rất phức tạp, hiệu năng truy vấn thấp và không hỗ trợ truy vấn khoảng do không bảo toàn được thứ tự các Bucket. Vì vậy, các NewBucketIndex được thiết kế nạp chồng nhưng vẫn bảo toàn thứ tự để cố gắng nâng cao khả năng bảo mật nhưng vẫn giữ được lợi thế trong biến đổi điều kiện truy vấn khoảng và thực thi tìm kiếm trên các cấu trúc phát triển từ cây B^+Tree. Luận án sử dụng tập $D = \left\{\binom{f_i}{v_i}\right\} = \left\{\binom{4}{3}, \binom{3}{7}, \binom{2}{9}, \binom{3}{14}, \binom{5}{25}, \binom{2}{50}\right\}$ trong **Ví dụ 3.1** để thử nghiệm các giá trị σ^2, H, pvd trên các Bucket với hai phương pháp tạo khác nhau:

- ***Phương pháp 1:*** tạo 3 Buckets tuần tự và không nạp chồng: $B1 = \{\binom{4}{3}, \binom{3}{7}\}$; $B2 = \{\binom{2}{9}, \binom{3}{14}\}$; $B3 = \{\binom{5}{25}, \binom{2}{50}\}$

- ***Phương pháp 2:*** Sử dụng thuật toán $BuildNewBucketIndex$ tạo 6 Buckets tuần tự và nạp chồng: $NB1 = \{\binom{1}{-15}, \binom{3}{3}\}$; $NB2 = \{\binom{1}{3}, \binom{2}{7}\}$; $NB3 = \{\binom{1}{7}, \binom{2}{9}, \binom{1}{14}\}$; $NB4 = \{\binom{2}{14}, \binom{3}{25}\}$; $NB5 = \{\binom{2}{25}, \binom{1}{50}\}$; $NB6 = \{\binom{1}{50}, \binom{1}{108}\}$;

Tính toán σ^2, H, pvd theo công thức đề xuất trong [53], [113] ta có **Bảng 3.4**.

Bảng 3.4. So sánh $\boldsymbol{\sigma^2, H, pvd}$ giữa hai phương pháp tạo Bucket

Bucket	$\boldsymbol{\sigma^2}$	$\boldsymbol{H}$	$\boldsymbol{pvd}$
$B1$	4.5	0.986	32
$B2$	7.5	0.971	50
$B3$	148	0.869	1250
$NB1$	81	0.811	648
$NB2$	5.3	0.914	32
$NB3$	8.9	1.5	389
$NB4$	36.3	0.971	242
$NB5$	208.3	0.915	1250
$NB6$	1682	1	6728

Từ **Bảng 3.4**, có thể thấy hầu hết các giá trị σ^2, pvd đo được trên các Bucket tạo bởi phương pháp 2 đều có giá trị cao hơn so với phương pháp 1. Qua đó ngụ ý khả năng bảo mật của các NewBucketIndex tốt hơn so với Bucket liên tục và không nạp chồng. Tuy nhiên, giá trị của H thì không có sự khác biệt quá nhiều giữa hai phương pháp thiết kế Bucket, thậm chí một số NewBucketIndex có H còn nhỏ hơn so với các Bucket $B1, B2, B3$. Lý do là các NewBucketIndex không được chia theo phương pháp equi-depth mà hoàn toàn được thiết kế theo kỹ thuật nạp chồng ngẫu nhiên. Việc nạp chồng ngẫu nhiên cho phép một giá trị rõ nằm trong nhiều Bucket khác nhau mà không có quy luật, dẫn đến khả năng A khai thác thông tin dựa vào thống kê tần suất các NewBucketIndex trên CS sẽ không có cơ sở.

Để làm rõ hơn về ưu điểm của các NewBucketIndex, ta cùng phân tích khả năng chống rò rỉ thông tin động và tĩnh trong các giả thuyết về sự hiểu biết của A:

Giả thuyết 1: A biết trước một NewBucketIndex nào đó cùng tập giá trị rõ mà Bucket đó đại diện, A cũng biết trước phân phối của các giá trị rõ trong CSDLQH. Khi đó liệu A có đoán được NewBucketIndex đang đại diện chính xác cho giá trị nào trong tập giá trị rõ tương ứng. Khả năng chống tấn công loại này được Hore và cộng sự [53] chứng minh có liên hệ mật thiết với phương sai, theo đó với cách chia một giá trị rõ nằm trong nhiều Bucket và một Bucket chứa nhiều giá trị rõ thì phương sai trung bình theo **Bảng 3.4** có xu hướng tốt hơn so với cách chia không nạp chồng. Khi phương sai cao, xác suất để A đoán đúng giá trị rõ tương ứng với Bucket sẽ nhỏ.

Giả thuyết 2: A biết trước tập giá trị phân biệt trên bản rõ nhưng không biết rõ tần suất chính xác hoặc phân phối xác suất của các giá trị này. A cũng biết tổng số Bucket đại diện cho toàn bộ miền giá trị rõ. Khi đó với một khoảng truy vấn $[l, h]$ bất kỳ trên dữ liệu rõ thì liệu A có thể xác định được tập các Bucket mà giá trị rõ của nó hoàn toàn thuộc khoảng $[l, h]$ hay không. Nếu tập rõ có d giá trị phân biệt thì sẽ có tối đa $C_d^2 + d$ lệnh truy vấn khoảng khác nhau mà A có thể thực hiện. Do các NewBucketIndex được thiết kế dựa trên nguyên tắc nạp chồng một phần giá trị nên hầu hết các Bucket đều có số phần tử phân biệt lớn hơn 1, xét 6 bucket trong **Bảng 3.4**, khi đó chỉ có duy nhất một truy vấn khoảng ($-15 \leq v_i \leq 108$, truy vấn trên toàn miền giá trị) là trả về toàn bộ các Bucket thỏa mãn chính xác điều kiện truy vấn. Ta có thể kết luận việc chia Bucket nạp chồng theo thuật toán BuildNewBucketIndex giúp tăng khả năng chống suy luận kết quả đối với các truy vấn khoảng.

Giả thuyết 3: A biết được giới hạn miền giá trị của dữ liệu rõ nhưng không biết chính xác có bao nhiêu giá trị phân biệt khác nhau. A cũng biết xác suất phân phối của một số nhóm giá trị rõ trên toàn miền, nhưng không biết xác xuất phân phối của từng giá trị rõ trong các Bucket cũng như không biết chính xác mỗi Bucket đang đại diện cho những giá trị rõ nào. Ví dụ cho quan hệ R (StudentID, StudentName, Scores), A chỉ biết được miền giá trị điểm là [1, 100], tuân theo phân phối Gaussian. A không biết chính xác có bao nhiêu loại giá trị điểm số khác nhau trong khoảng [1, 100] và không biết tần suất chính xác của mỗi loại giá trị điểm trong miền này, cũng như không biết về nội dung, tần suất của các phần tử tương ứng trong mỗi Bucket.

- Trong giả thuyết này, A cố gắng thống kê trên CS để biết được có bao nhiêu NewBucketIndex phân biệt và tần suất xuất hiện của mỗi NewBucketIndex. Qua đó A suy luận Bucket nào có tần suất lớn thì sẽ đại diện cho các giá trị bản rõ có tần suất lớn, tuy nhiên cách tấn công này không có cơ sở vì theo thuật toán $BuildNewBucketIndex$ một giá trị rõ có thể bị chia thành nhiều phần và phân bổ vào nhiều Bucket khác nhau. Đồng thời việc phân bổ hoàn toàn ngẫu nhiên, do đó sẽ không có mối quan hệ tuyến tính giữa tần suất của NewBucketIndex trên CS với tần suất nguyên bản của dữ liệu rõ nữa. Trong ví dụ về điểm số ở trên, ta cũng chỉ biết được phổ điểm trong khoảng [50, 70] có xác suất cao nổi trội (nằm ở đỉnh chuông trong phân phối Gaussian). Theo đó A thống kê được 2 Bucket có tần suất cao bất thường trên CS, nhưng rõ ràng A không thể kết luận 2 Bucket đó đại diện cho các giá trị nằm trong khoảng [50, 70]. Nguyên nhân bởi thuật toán $BuildNewBucketIndex$

có thể đã phân mỗi giá trị trong khoảng [50, 70] vào nhiều NewBucketIndex khác nhau hoặc 2 Bucket được thống kê lại đại diện cho rất nhiều giá trị rõ khác.

Giả thuyết 4: A biết được tập Bucket và tần suất của chúng trên CS (đây được coi là điều hiển nhiên vì CS là môi trường bán trung thực). Bên cạnh đó A còn quan sát được tập kết quả trả về sau rất nhiều truy vấn khoảng. Vậy A có khả năng đưa ra nhận định sau: Nếu một Bucket B_i xuất hiện nhiều lần trong các tập kết quả trả về sau các truy vấn khoảng thì Bucket đó đại diện cho miền giá trị rõ rất lớn nên thỏa mãn được nhiều điều kiện truy vấn khoảng khác nhau. Trong trường hợp đặc biệt khi các Bucket trả về bởi truy vấn có tần suất xuất hiện đồng đều thì A kết luận thêm xác suất phân phối của mỗi phần tử rõ trong B_i sẽ thấp. Tuy nhiên mỗi NewBucketIndex có thể chỉ chứa một phần trong tổng số f lần xuất hiện của mỗi phần tử dữ liệu rõ. Vì vậy kết luận trên của A sẽ không có nhiều ý nghĩa.

Giả thuyết 5: A biết tập giá trị rõ và có được thống kê về m Bucket cùng tần suất và thứ tự của chúng (lưu ý A không biết chính xác Bucket đại diện cho tập giá trị cụ thể nào). Bằng cách sắp xếp tăng dần toàn bộ các giá trị trong tập dữ liệu rõ, sau đó chia toàn bộ tập này thành m khoảng liên tiếp dựa trên số lượng và tần suất mỗi Bucket đã thống kê. A tin rằng có thể đoán được miền giá trị mà mỗi Bucket đại diện. Tuy nhiên đối với NewBucketIndex, việc dự đoán của A sẽ có sai số nhiều hơn bởi: trong trường hợp xấu nhất nếu A biết được các tập EP_t (Bước 6 trong thuật toán $BuildNewBucketIndex$) thì không có nghĩa A có thể xác định được chính xác một giá trị rõ v_j thuộc Bucket B_t nào vì quá trình phân các giá trị rõ vào các Bucket hoàn toàn ngẫu nhiên (thể hiện tại Bước 7 trong thuật toán $BuildNewBucketIndex$).

3.7.3. Phân tích bảo mật vector giấu tin IHV

Qua các phân tích trong **Mục 3.7.2**, cho thấy hầu hết các giả thuyết tấn công của A đều gặp khó khăn. Tuy nhiên trong giả thuyết thứ 5, các NewBucketIndex không thể hoàn toàn chống được khả năng suy luận thông tin dựa trên đặc điểm bảo toàn thứ tự của các chỉ mục so với bản rõ. Để che giấu quan hệ thứ tự giữa các NewBucketIndex nhưng vẫn cung cấp được phương thức cho phép so sánh giữa chúng, luận án đã đề xuất biến đổi các NewBucketIndex về các vector IHV.

Mục 3.5 chỉ ra vector IHV được xây dựng trên hai giai đoạn:

- ***Thứ nhất*** là xây dựng các vector nhúng $(\overrightarrow{UO_b}, \overrightarrow{UO_c})$, các tham số bí mật cần dùng gồm: cặp vector cơ sở $(\overrightarrow{U_b}, \overrightarrow{U_c})$ và các cặp giá trị $(pos1, pos2)$, $(r_b,\ r_c)$.

- ***Thứ hai*** là xây dựng các vector IHV dựa trên các vector nhúng, quá trình này sử dụng thêm ma trận bí mật M cỡ kxk.

Để phân tích khả năng chống rò rỉ thông tin từ $(\overrightarrow{UO_b}, \overrightarrow{UO_c})$, ta xét giả thuyết tấn công biết trước bản mã. Trong giả thuyết này, A biết trước tập giá trị rõ và các vector $(\overrightarrow{UO_b}, \overrightarrow{UO_c})$, mục đích của A là phân tích nội dung vector này hoặc sự liên quan giữa các vector để cố gắng suy luận được các thông tin liên quan tới giá trị rõ b, c nhúng trong đó. Một số phương pháp phân tích A sử dụng trong kiểu tấn công này gồm:

- Thống kê tần suất xuất hiện của các vector $(\overrightarrow{UO_b}, \overrightarrow{UO_c})$ để đoán vector nào xuất hiện nhiều nhất hoặc ít nhất, qua đó suy luận được giá trị b, c tương ứng. Tuy nhiên do tác động của cặp số ngẫu nhiên $(r_b,\ r_c)$ nên cùng một giá trị rõ b hoặc c sẽ có nhiều trạng thái vector $\overrightarrow{UO_b}$, $\overrightarrow{UO_c}$ khác nhau. Cách tấn công này không khả thi.

- Dùng một vector cố định $\overrightarrow{UO_c}$, A thực hiện nhân $\overrightarrow{UO_c}$ với hàng loạt $\overrightarrow{UO_b}$ đã biết. Kết quả của các tích trên chính là hiệu $b - c$, dựa vào đó A hoàn toàn có thể xác định được thứ tự giữa các giá trị b. Tuy nhiên, cặp giá trị ngẫu nhiên $(r_b,\ r_c)$ tiếp tục có ý nghĩa quan trọng khi biến đổi kết quả $\overrightarrow{UO_b}.\overrightarrow{UO_c}$ thành ngẫu nhiên, khi đó kết quả này chỉ là một dấu hiệu (bằng 0, nhỏ hơn 0 hoặc lớn hơn 0) để xác định mối quan hệ giữa b với c, chứ không thể xác định mối quan hệ giữa các giá trị b với nhau.

- A phân tích nội dung trong các vector nhúng $(\overrightarrow{UO_b}, \overrightarrow{UO_c})$ và cố gắng dự đoán cặp vector trực giao $(\overrightarrow{U_b}, \overrightarrow{U_c})$ có chiều dài $k - 2$. Cách thức để thực hiện bằng phương pháp tìm ước chung lớn nhất trình bày trong **Mục 3.4**, bản chất của vấn đề là đi tìm cặp vị trí $(pos1, pos2)$. Để tìm được 2 vị trí trong k vị trí, ta cần tới $\frac{k(k-1)}{2}$ phép thử, độ phức tạp ước chừng $O(k^2)$. Rõ ràng, A có khả năng để thực hiện được tấn công này trong thời gian đa thức. Vì vậy, luận án đề xuất sử dụng thêm cặp ma trận bí mật (M_b, M_c) để biến đổi $\overrightarrow{UO_b}, \overrightarrow{UO_c}$ thành các vector giấu tin $\overrightarrow{UO_b^M}$, $\overrightarrow{UO_c^M}$ bảo mật hơn.

Để phân tích khả năng chống rò rỉ thông tin từ $\overrightarrow{UO_b^M}, \overrightarrow{UO_c^M}$, luận án xét giả thuyết tấn công biết trước bản rõ. Khi đó A biết trước tập vector $(\overrightarrow{UO_b}, \overrightarrow{UO_c})$ và tập vector $(\overrightarrow{UO_b^M}, \overrightarrow{UO_c^M})$ tương ứng. Mục tiêu cần tìm ra khoá là ma trận bí mật M khả nghịch cỡ kxk. Việc tìm M tương đương tìm M_b trong phương trình $UO_b^M = M_b \text{x}\ UO_b$. Với UO_b^M, UO_b là các ma trận cỡ $kx1$ được chuyển đổi từ vector $\overrightarrow{UO_b^M}$ và $\overrightarrow{UO_b}$.

Để dễ quan sát, ký hiệu $X = M_b, A = UO_b, B = UO_b^M$, khi đó có phương trình

$\boldsymbol{X}$ x $\boldsymbol{A} = \boldsymbol{B}$, biết A_{kx1} và B_{kx1} cần tìm X_{kxk}. Giả sử X, A, B được biểu diễn như sau:

$$X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1k} \\ x_{21} & x_{22} & \cdots & x_{2k} \\ \vdots & \vdots & \vdots & \vdots \\ x_{k1} & x_{k2} & \cdots & x_{kk} \end{pmatrix}, A = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_k \end{pmatrix}, B = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \end{pmatrix}$$

Khi đó $\boldsymbol{X}$ x $\boldsymbol{A} = \boldsymbol{B}$ được biểu diễn tương đương hệ phương trình (3.10) sau:

$$\begin{cases} a_1 x_{11} + a_2 x_{12} + \cdots + a_k x_{1k} = b_1 \\ a_1 x_{21} + a_2 x_{22} + \cdots + a_k x_{2k} = b_2 \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ a_1 x_{k1} + a_2 x_{k2} + \cdots + a_k x_{kk} = b_k \end{cases} \quad (3.10)$$

Với $A, B \neq 0, k \geq 3$ thì số biến x_{ij} (có k^2 biến) lớn hơn nhiều số phương trình, và hệ phương trình có vô số nghiệm. Nói cách khác có vô số ma trận M_b thoả mãn phương trình $UO_b^M = M_b$x UO_b. Với mỗi cặp (UO_b^M, UO_b) được quan sát, kẻ gian cũng nhận được vô số M_b khác. Rõ ràng việc tìm ra M chính xác (lưu trữ tại Proxy) từ vô số các M_b nói trên là một thách thức với kẻ gian.

Trong thực tiễn, lược đồ ESIT-SSE còn cho phép người dùng linh động tuỳ chỉnh k theo xu hướng lớn hơn để tăng mức độ phức tạp khi đối mặt dạng thức tấn công này của kẻ gian.

3.7.4. Phân tích bảo mật cây IHV_B$^+$Tree

Cấu trúc của cây IHV_B^+Tree được xây dựng theo thuật toán $Build\_IHV\_B^+Tree$ đáp ứng việc chống rò rỉ thông tin trên CS khi kẻ gian cố gắng khai thác mối quan hệ giữa các khóa trong mỗi nút và giữa các nút. Cụ thể:

- Đánh giá khả năng khai thác thông tin từ nút gốc và nút trong: Mỗi khóa (NewBucketIndex) trong các nút của cây có thể đại diện cho nhiều giá trị rõ v_i khác nhau và ngược lại. Thứ tự tăng dần của các khoá lân cận nhau trong mỗi nút sẽ không phản ánh hoàn toàn sự chính xác về thứ tự giữa các giá trị rõ v_i mà khóa đó đại diện. Tương tự, mối quan hệ về độ lớn của khoá giữa nút cha và nút con hoặc giữa các nút lá với nhau cũng không phản ánh chính xác việc thứ tự các giá trị rõ v_i mà các khoá đó đại diện. Hơn nữa, cấu trúc nút gốc và nút trong của cây IHV_B^+Tree không chứa các con trỏ lưu địa chỉ của các bản ghi, nên giả sử nếu kẻ gian xác định được khóa trong các nút nói trên đại diện chính xác cho giá trị rõ v_i nào thì cũng không thể tìm ra các bản ghi mã liên quan tới v_i.

- Đánh giá khả năng khai thác thông tin từ nút lá: Các khóa (NewBucketIndex) trong một nút lá cũng được sắp xếp tăng dần và các giá trị khóa của nút lá bên phải

lớn hơn giá trị khóa của nút lá bên trái. Mỗi khóa có một con trỏ ptr_row kèm theo để trỏ tới một danh sách liên kết lưu các phần tử chứa bản ghi mã hóa $row(id)$. Nếu kẻ gian dựa trên thứ tự khóa trên các nút lá để suy diễn ra thứ tự các bản ghi mã thì không có căn cứ bởi hai lý do: thứ nhất các bản ghi cùng chứa một giá trị v_i hoàn toàn có thể nằm trên các danh sách liên kết của các nút lá khác nhau; thứ hai thứ tự các bản ghi trong danh sách liên kết tại một nút lá được sắp xếp ngẫu nhiên.

3.8. Thực nghiệm và đánh giá hiệu năng của lược đồ ESIT-SSE

Tương tự cách tiếp cận thực nghiệm lược đồ DIQ-SSE tại **mục 2.6**, lược đồ ESIT-SSE có thể cho phép thực hiện các lệnh truy vấn phức tạp hơn với sự tham gia của nhiều quan hệ, nhiều trường và liên hợp nhiều điều kiện (kết hợp truy vấn nhiều khoảng trên nhiều trường hoặc kết hợp cả truy vấn chuỗi con). Bằng cách thực thi rời rạc các chuỗi điều kiện riêng lẻ theo quy trình mô tả tại **hình 2.4** và **hình 3.4**. Sau đó tại CS thực hiện liên hợp (toán tử And, Or, …) các tập kết quả mã được trả về ứng với mỗi chuỗi điều kiện, cuối cùng gửi tập kết quả liên hợp về Proxy để giải mã và truy vấn lại. Việc lựa chọn những lệnh truy vấn phức tạp như trên để thực nghiệm có thể tạo ra các thông số không ghi nhận hết sự hiệu quả của các thiết kế NewBucketIndex, IHV hay cây B^+Tree được đề xuất. Vì vậy, luận án tập trung vào các kịch bản, dữ liệu và mẫu lệnh thực nghiệm như **mục 3.8.1** để làm rõ tính hiệu quả của các lý thuyết được đề xuất cho bài toán truy vấn khoảng trên CSDLQH mã hóa.

3.8.1. Chuẩn bị điều kiện thực nghiệm

Mục này góp phần ***đánh giá tính hiệu quả của lược đồ*** ESIT-SSE qua tiêu chí ***hiệu năng*** truy vấn dựa trên một số thực nghiệm và phân tích về tổng thời gian để hoàn thành các truy vấn. Thiết kế của lược đồ ESIT-SSE đã được phân tích lý thuyết đảm bảo không có dữ liệu dư thừa trả về trên CS khi truy vấn khoảng, vì vậy việc sử dụng đo lường thời gian thực thi tại mỗi pha thực hiện là thang đo phù hợp để đánh giá tính hiệu quả của lược đồ, cách đo này cũng được sử dụng trong nhiều nghiên cứu liên quan tới truy vấn khoảng mà luận án có tham chiếu [47], [58], [5], [56], [99]

Thời gian truy vấn được đo dựa trên bốn pha:

- **Pha 1 (P1):** gồm tổng thời gian gửi câu lệnh truy vấn từ ClientSite lên Proxy, thời gian biến đổi điều kiện truy vấn trên Proxy và thời gian gửi điều kiện truy vấn sau khi biến đổi đến CS.
- **Pha 2 (P2):** thời gian truy vấn và trả về tập kết quả dữ liệu mã trên CS.

- **Pha 3 (P3):** thời gian gửi tập kết quả dữ liệu mã từ CS về Proxy.
- **Pha 4 (P4):** thời gian giải mã, truy vấn lại dữ liệu (đối với các câu lệnh phức tạp) tại Proxy và thời gian trả kết quả cuối cùng về ClientSite.
- **Sum4P:** tổng thời gian thực hiện của 4 pha (P1+P2+P3+P4).

Luận án dùng ba kịch bản thực nghiệm triển khai trên mô hình DAS-PROXY với cùng điều kiện truy vấn khoảng "$\boldsymbol{X_t}$ *Between* $\boldsymbol{l}$ *And* $\boldsymbol{h}$". Từ đó có kết luận khách quan về hiệu năng của lược đồ ESIT-SSE trên phương diện thực nghiệm. Cụ thể:

- Kịch bản thứ nhất: Biến đổi quan hệ rõ R thành $R^E(ID, X_1, X_2, \ldots X_t^E, X_t^B, \ldots X_n)$ trên CS. Trong đó X_t^E là cột lưu giá trị mã hóa và X_t^B là cột lưu giá trị NewBucketIndex tương ứng của X_t. Tương tự như các giải pháp OPES [5], [66], [18], ta xây dựng cây B$^+$ Tree trực tiếp trên các giá trị của trường X_t^B và tại CS sử dụng cây này tìm kiếm các bản ghi thoả mãn điều kiện truy vấn "$c_l \leq X_t^B \leq c_h$" với $c_l = Min(l^B)$ và $c_h = Max(h^B)$ (mô tả trong **Mục 3.4.1**).

- Kịch bản thứ hai: Quan hệ rõ R được biến đổi thành $R^E(ID, X_1, X_2, \ldots X_t^E, X_t^V, \ldots X_n)$ trên CS. Trong đó X_t^V là cột lưu giá trị IHV tương ứng của X_t. Chuỗi điều kiện truy vấn khoảng thực hiện trên CS có dạng "$X_t^V.\ \overrightarrow{ihv_c(l)} \geq 0\ and\ X_t^V.\overrightarrow{ihv_c(h)} \leq 0$" và sử dụng phương pháp duyệt qua tất cả các bản ghi để kiểm tra.

- Kịch bản thứ ba: Sử dụng lược đồ ESIT-SSE được đề xuất. Theo đó quan hệ rõ R được biến đổi thành $R^E(ID, X_1, X_2, \ldots X_t^E, \ldots X_n)$ và cây IHV_B^+Tree trên CS. Quá trình truy vấn trên CS chính là duyệt cây IHV_B^+Tree thoả mãn 2 khoảng $[\overrightarrow{ihv_c(l)},\ \overrightarrow{ihv_c(h)}]$ và $[\overrightarrow{ihv_c(c_l)},\ \overrightarrow{ihv_c(c_h)}]$ theo quy trình trong **Mục 3.6.1,** cụ thể:

+ Duyệt nút gốc và nút trong thoả mãn biểu thức: $\overrightarrow{ihv_c(c_l)}.\ \overrightarrow{ihv_b(v_i^B)} \geq 0$ và $\overrightarrow{ihv_c(c_h)}.\ \overrightarrow{ihv_b(v_i^B)} \leq 0$.

+ Duyệt danh sách LIST trong các nút lá thoả mãn biểu thức: $\overrightarrow{ihv_c(l)}.\ \overrightarrow{ihv_b(v_i)} \geq 0$ và $\overrightarrow{ihv_c(h)}.\ \overrightarrow{ihv_b(v_i)} \leq 0$.

Ngoài ra trong các kịch bản này, tiếp tục có sự điều chỉnh tham số m và k.

Dữ liệu thử nghiệm: Tương tự như chương 2, luận án tiếp tục sử dụng dữ liệu cột L_EXTENDEDPRICE của bảng LINEITEM trong CSDLQH TPC-H [142] cho các kịch bản thử nghiệm. Thực thi công cụ DBGen để sinh lần lượt 100 ngàn, 500 ngàn, 1 triệu, 2 triệu, 4 triệu và 8 triệu bản ghi cho LINEITEM. Xây dựng CSDLQH mã hóa TPC_H_CS trong đó có bảng LINEITEM_CS với trường dữ liệu mã hóa L_EXTENDEDPRICE_E tương ứng (lưu các giá trị mã được sinh ra bởi thuật toán

AES [37]).

Sử dụng câu truy vấn "***Select*** $*$ ***From*** *LINEITEM* ***Where*** *L_EXTENDEDPRICE* ***Between l*** *and* ***h***" với 15 khoảng $[l, h]$ khác nhau để thực nghiệm trên 3 kịch bản. Thời gian thực thi của các pha (P_i) được tính dựa trên trung bình cộng của các lần truy vấn ứng với tối thiểu 15 khoảng.

Thiết bị sử dụng mô phỏng các khu vực của mô hình DAS-PROXY như sau:

ClientSite: Intel Core i5 4310U 2.0Ghz CPU, 8GB DDRAM, cài Windows 10.

ProxyServer: Intel Xeon W-1350 3.3Ghz CPU và 16GB DDRAM, cài Windows Server 2019 và MS SQL Server 2016.

CloudServer: Intel Xeon E5-2620 2.0Ghz CPU và 64GB DDRAM, cài Windows Server 2019 và MS SQL Server 2016.

3.8.2. Thực nghiệm và đánh giá hiệu năng tìm kiếm

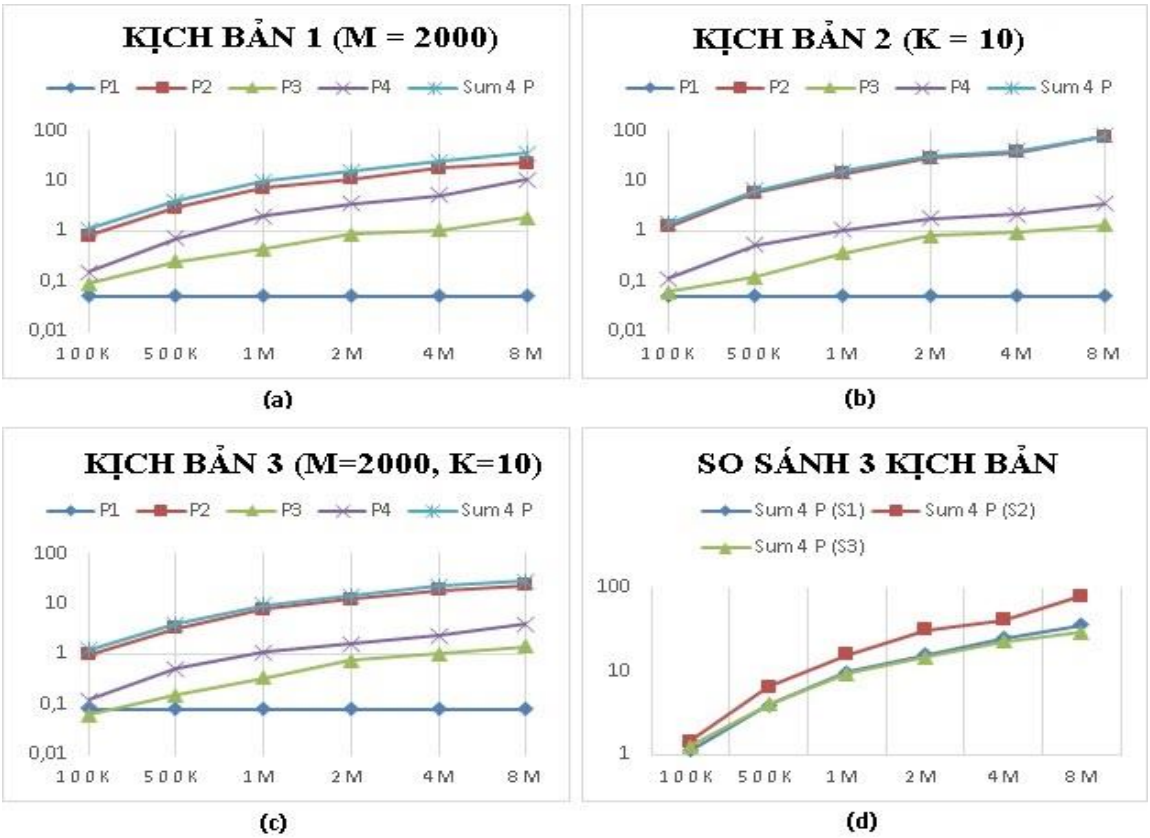
Luận án chọn tham số $m = 2000$ (số lượng NewBucketIndex) cho **Kịch bản thứ nhất** và **Kịch bản thứ ba**. Chọn tham số $k = 10$ (chiều dài IHV) cho **Kịch bản thứ 2** và **Kịch bản thứ ba**. Kết quả thời gian 4 pha (P1, P2, P3, P4, SUM4P) cho từng kịch bản khi số bản ghi tăng từ 100 ngàn đến 8 triệu được mô tả như **Hình 3.5**.

Quan sát các biểu đồ **a, b, c, d** trong **Hình 3.5** cho thấy:

- P1 gần như không có sự thay đổi nhiều bởi quá trình chuyển yêu cầu truy vấn từ ClientSite lên Proxy là như nhau với mọi kịch bản. Chi phí biến đổi điều kiện truy vấn trên dữ liệu rõ về dạng truy vấn trên NewBucketIndex và IHV là tương đương. Với biểu đồ **c**, tại Proxy phải thực hiện đồng thời cả hai biến đổi nêu trên nên tổng thời gian P1 có tăng nhẹ so với hai kịch bản còn lại. Qua đó có thể thấy P1 chiếm không đáng kể trong tổng chi phí thời gian thực hiện truy vấn.

- P2 cho thấy việc truy vấn trên dữ liệu mã tại CS chiếm nhiều thời gian nhất trong tất cả các kịch bản. P2 có xu hướng tăng nhanh khi số lượng bản ghi bùng nổ, đặc biệt trong Kịch bản 2 do thực hiện kỹ thuật quét toàn bộ bảng (Scan Table) nên thời gian truy vấn sẽ lớn nhất. Trong khi đó Kịch bản 1 và Kịch bản 3 có sử dụng cấu trúc cây B^+ $Tree$, IHV_B^+Tree nên tốc độ truy vấn vượt trội so với Kịch bản 2. Tuy nhiên trong Kịch bản 3, quá trình tìm kiếm trên IHV_B^+Tree phải tốn thêm thời gian duyệt các giá trị trong danh sách $LIST$ của mỗi nút lá để loại bỏ các bản ghi dương tính giả nên sẽ chậm hơn so với Kịch bản 1 khi chỉ tìm kiếm trên cây B^+ $Tree$ được xây dựng từ các NewBucketIndex. Bù lại Kịch bản 3 mang đến tính bảo mật cao cho

IHV_B^+Tree, đồng thời kết quả sau truy vấn không có bản ghi dương tính giả.



Hình 3.5. Đánh giá thời gian thực thi 4 pha trên 3 kịch bản thử nghiệm
(Trục hoành biểu thị số bản ghi, trục tung biểu thị số giây thực hiện)

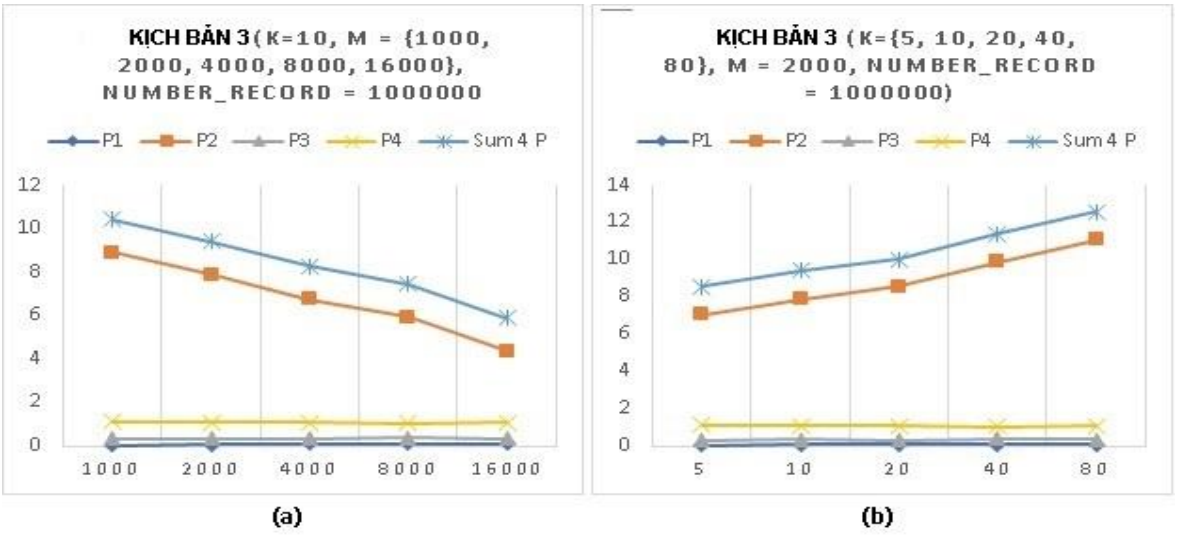
- P3 chỉ ra chi phí truyền tải dữ liệu chiếm một phần tương đối nhỏ so với chi phí truy vấn trên CS. Tuy nhiên vẫn có sự khác biệt giữa Kịch bản 1 với hai kịch bản còn lại bởi tập dữ liệu mã trả về sau khi truy vấn trong Kịch bản 1 bị dư thừa nhiều do sử dụng NewBucketIndex, trong đó hai kịch bản còn lại không trả về bản ghi dương tính giả. Vì vậy, đường P3 trong Kịch bản 1 có xu hướng cao hơn và biến thiên nhanh hơn khi có sự bùng nổ về tập dữ liệu trên CS.

- P4 là một hệ quả từ P3, khi số lượng bản ghi dương tính giả càng nhiều thì chi phí giải mã càng lớn theo. Đặc biệt với số lượng 8M bản ghi, kết quả trả về sẽ chứa một tập bản ghi dương tính giả rất lớn dẫn đến thời gian giải mã trong Kịch bản 1 tăng cao hơn nhiều so với hai kịch bản còn lại.

- Đường Sum4P trong biểu đồ **c** chỉ ra tổng thời gian hoàn thành một truy vấn dù ở bất kỳ kịch bản nào cũng đều phụ thuộc chủ yếu vào thời gian truy vấn trên CS và thời gian giải mã tại Proxy. Có thể thấy Kịch bản 2 là tệ nhất. Với hai kịch bản còn lại thì Kịch bản 3 tốt hơn Kịch bản 1 khi số lượng bản ghi trên CS tăng dần.

Qua **Hình 3.5**, Kịch bản 3 đã cho thấy được ưu thế về thời gian thực thi truy vấn. Để phân tích chi tiết hơn các yếu tố tác động tới hiệu năng của lược đồ ESIT-

SSE, luận án tiếp tục thử nghiệm Kịch bản 3 bằng việc cho biến động một trong hai tham số m và k khi truy vấn trên 1 triệu bản ghi. Kết quả thử nghiệm như **Hình 3.6**.



Hình 3.6. Đánh giá thời gian thực thi lược đồ ESIT-SSE khi thay đổi $\boldsymbol{k, m}$

(Trục hoành biểu thị giá trị M/K, trục tung biểu thị số giây thực hiện)

Biểu đồ **a, b** trong **Hình 3.6** chỉ ra đường P1, P3, P4 hầu như không thay đổi so với biểu đồ **c** trong **Hình 3.5** tại mốc 1 triệu bản ghi. Nguyên nhân được lập luận ở đây là do thuật toán $Transform\_RangeQueryCondition$ chỉ thực thi một lần nên việc tăng giá trị tham số m, k không làm tăng chi phí về thời gian cho P1. Thuật toán $RangeQuery\_On\_IHV\_B^+Tree$ luôn trả về tập kết quả chính xác, không tồn tại bản ghi dương tính giả nên các tham số m, k không ảnh hưởng tới thời gian truyền tải dữ liệu (P3) từ CS về Proxy và thời gian giải mã (P4) tập kết quả trên Proxy.

Từ biểu đồ **a** trong **Hình 3.6,** ta thấy đường P2 giảm dần khi m tăng dần. Khi số lượng NewBucketIndex tăng thì số lượng giá trị rõ được đại diện bởi một Bucket sẽ giảm, dẫn đến các nút lá của cây IHV_B^+Tree trỏ tới các danh sách $LIST$ chứa ít bản ghi hơn, điều này giúp việc duyệt danh sách liên kết được nhanh hơn. Có thể thấy bản chất của tăng m sẽ làm tăng hiệu năng truy vấn khoảng (do loại bỏ bớt bản ghi dương tính giả) nhưng lại làm giảm tính bảo mật của hệ thống.

Trong khi đó biểu đồ **b** trong **Hình 3.6** lại cho thấy khi k tăng dần thì đường P2 cũng tăng theo, nguyên nhân của vấn đề này là chi phí thực hiện phép tích vô hướng của các vector giấu tin tăng cao khi bị lặp lại nhiều lần trong quá trình duyệt cây IHV_B^+Tree. Tuy nhiên, nếu k tăng thì khả năng bảo mật của IHV cũng tăng theo.

Qua các phân tích về các biểu đồ trong **Hình 3.6**, luận án chỉ ra sự linh động trong việc điều chỉnh các tham số k, m lược đồ ESIT-SSE có thể đạt được các mục

tiêu về hiệu năng hoặc bảo mật tùy theo bối cảnh ứng dụng.

3.9. So sánh hiệu quả lược đồ ESIT-SSE với một số nghiên cứu

Các tiêu chí đánh giá tính hiệu quả của lược đồ ESIT-SSE được mô tả trong mục **Phạm vi luận án** và trong **Nhận xét 1.3**, tập trung vào: *bảo mật, hiệu năng và tính khả thi trong triển khai*. **Mục 3.7** đã phân tích kỹ về tính *bảo mật* của lược đồ thông qua khả năng chống rò rỉ thông tin chỉ mục NewBucketIndex, vector giấu tin IHV, cây IHV_B^+Tree và mẫu truy vấn. Trong khi đó **Mục 3.8** lại tập trung thực nghiệm lược đồ ESIT-SSE và phân tích đánh giá *hiệu năng* qua chi phí thời gian truy vấn. Như vậy, *nội dung hai mục này đang dừng lại ở việc **tự phân tích, nhận xét** về bảo mật, hiệu năng dựa trên đề xuất thiết kế của lược đồ, cũng như **tự ghi nhận** kết quả thu được từ thực nghiệm trên dữ liệu mẫu đối với một số **kịch bản giới hạn***. Để có đánh giá thuyết phục và khách quan hơn về tính hiệu quả (dựa trên các tiêu chí trong **Bảng 1.3**) của lược đồ ESIT-SSE, luận án *tiếp tục so sánh với một số nghiên cứu liên quan tiêu biểu gần đây* như **Bảng 3.5**. Một số phân tích về ưu điểm và tồn tại của những nghiên cứu này đã được trình bày tích hợp trong mục "**1.9.2. Tình hình nghiên cứu về truy vấn trên dữ liệu số mã hóa"**, còn trong phần này luận án sẽ tiếp tục làm rõ hơn các đánh giá về tính hiệu quả giữa các giải pháp trên góc nhìn tổng hợp và đối sánh:

Đánh giá chung:

- Các công trình đã liệt kê đều tiêu biểu và liên quan tới vấn đề truy vấn khoảng trên dữ liệu số mã hóa, được đăng trên các nhà xuất bản/tạp chí uy tín, trong đó nhiều tài liệu cập nhật được tình hình trong 5 năm trở lại đây.

- Cấu trúc dữ liệu hỗ trợ xây dựng chỉ mục trong lược đồ ESIT-SSE có tính mới so với các công trình trong **Bảng 3.5** khi đề xuất và sử dụng kết hợp NewBucketIndex, IHV và cây IHV_B^+Tree.

Đánh giá về chi phí lưu trữ:

- Tương tự các lược đồ hỗ trợ tìm kiếm chuỗi con trong **Bảng 2.4**, các nghiên cứu được liệt kê trong **Bảng 3.5** cũng đánh giá chi phí lưu trữ dựa trên tổng số bản ghi chứa chỉ mục trong CSDL. Các lược đồ đề xuất tại nghiên cứu [38], [35], [72], [94], [90], [82], [77] đều có chi phí lưu trữ cỡ $O(c.N)$ với $c > 2$. Một số nghiên cứu khác [136], [112] mặc dù chi phí lưu trữ tại Server đạt $O(N)$ nhưng lại phát sinh thêm lưu trữ tại Client. Trong khi đó các công trình [47], [60], [49], [105] lại giới

thiệu các lược đồ chỉ lưu trữ tập trung tại Server với chi phí tốt hơn là $O(N)$, thậm chí [64] còn đạt mức $O(log2^N)$.

- Chi phí lưu trữ của lược đồ ESIT-SSE chỉ tập trung chính tại Server (CS) cho cây IHV_B^+Tree với mức $O(N)$. Đây là chi phí được đánh giá là tốt khi so sánh tương quan với các tài liệu khác liệt kê trong **Bảng 3.5**.

Đánh giá về chi phí quy trình tìm kiếm:

Đánh giá các công trình liên quan:

- *Về chi phí tính toán và truy vấn trên CS*, ngoài các nghiên cứu [47] và [35] có chi phí tìm kiếm lần lượt tuyến tính với N là số bản ghi của CSDL và N_r là số lượng bản ghi thoả mãn truy vấn, thì hầu hết thuật toán tìm kiếm trong các giải pháp còn lại đều có độ phức tạp là hàm logarit của N hoặc các tham số nhỏ hơn N như $sR, eR, sD, N_{index}, N_r, E_r$, ... (có diễn giải chi tiết trong **Bảng 3.5**). Ngoại trừ các giải pháp [60], [49], [94] và [64] được coi là không phát sinh các chi phí ngoài khu vực CS thì các giải pháp khác đều tiêu tốn thêm tài nguyên tại Client, Proxy (Trapdoor) hay quá trình truyền tải (Communication).

- *Về chi phí tính toán tại cửa sập (Trapdoor),* một số giải pháp [38], [35], [72], [82], [105] được ghi nhận tiêu tốn một phần tài nguyên nhỏ (cỡ $O(1)$ đến $O(logN)$) cho quá trình biến đổi lệnh truy vấn về dạng tìm kiếm được trên dữ liệu số mã hoá. Một số nghiên cứu khác [82], [105] có chi phí thực hiện tại cửa sập lớn hơn $(O(k.N_b), O(n_d.\ell^2))$, tuy nhiên vẫn là không đáng kể khi so với các chi phí tính toán, tìm kiếm tại CS.

- *Về chi phí tính toán phía Client*, phát sinh này khiến các giải pháp giảm tính khả thi triển khai khi gây áp lực lên các thiết bị đầu cuối cấu hình thấp của EU. Giải pháp trong công trình [47] yêu cầu chi phí khá lớn $(O((N_{sr} - N_r).C_{decrypt} + N_{sr}))$ tại Client để giải mã và truy vấn lại do tồn tại dữ liệu dương tính giả. Một số giải pháp khác như [136], [112], [90] hay [77] phát sinh chi phí nhỏ hơn tại Client do không sử dụng Proxy.

- *Về chi phí truyền tải*, một số nghiên cứu như [47], [136], [90], [77] đều cho thấy phải tiêu tốn một lượng tài nguyên thường xuyên cho quá trình gửi nhận dữ liệu giữa Client/Proxy tới CS và ngược lại. Nguyên nhân các chi phí này hầu hết đều liên quan tới vấn đề dư thừa dữ liệu trả về sau truy vấn trên CS hoặc quá trình CS gửi dữ liệu các nút trong các cấu trúc dữ liệu dạng cây về Proxy/Client để giải mã, so sánh

và sau đó gửi ngược lại CS để xác định được nút tiếp theo cần duyệt.

- *Về số vòng giao tiếp dữ liệu* (xem diễn giải trong **Bảng 1.3**), hầu hết các giải pháp chỉ mất một vòng giao tiếp, trong khi đó có một số như [60], [112], [77] lại cần tới 2 vòng. Lưu ý: số vòng giao tiếp chỉ tác động một phần đến chi phí truyền tải.

- *Về bảo mật,* vấn đề chống rò rỉ chỉ mục tìm kiếm và mẫu truy vấn đối với dữ liệu dạng số khó thực hiện hơn so với dữ liệu ký tự, do việc cố gắng bảo toàn thứ tự chỉ mục hoặc xây dựng các cấu trúc hỗ trợ tăng tốc truy vấn dễ lộ thông tin bản rõ thông qua so sánh vị trí tương quan các nút dữ liệu. Vì vậy, một số công trình như [47], [38], [35], [72], [64], [77] vẫn chưa chống được các rò rỉ thông tin nêu trên.

Đánh giá về lược đồ ESIT-SSE khi so với các công trình liên quan:

- Các thuật toán $Transform\_ComparisonOperand\_To\_IHV$ và $Transform\_RangeQueryCondition$ đều khai thác dữ liệu từ **Bảng 3.1** nên lược đồ ESIT -SSE không tốn nhiều chi phí vào quá trình giao tiếp cũng như tính toán tại Proxy hay Client. Chi phí tìm kiếm chủ yếu tập trung tại CS qua việc thực thi thuật toán $RangeQuery\_On\_IHV\_B^+Tree$. Trong đó, quá trình duyệt cây đến các nút lá có chi phí tương tự duyệt cây B$^+$Tree với độ phức tạp trung bình $O(logN_b)$, quá trình duyệt các $LIST$ trên các nút lá vừa tìm kiếm được có chi phí cỡ $O(N_{br})$. Như vậy với tổng chi phí tìm kiếm tại Server cỡ $O(logN_b + N_{br})$ được đánh giá là rất tốt khi so sánh với các công trình khác trong bảng.

- Xét về số vòng giao tiếp dữ liệu giữa EU – PROXY - CS khi truy vấn thì ESIT-SSE cũng cho thấy ưu thế khi chỉ tốn 1 vòng. Về khả năng chống rò rỉ chỉ mục và mẫu truy vấn cũng đạt yêu cầu. Qua đó thêm một lần nữa chỉ ra tính hiệu quả của lược đồ qua các tiêu chí về hiệu năng và bảo mật.

Bảng 3.5. So sánh hiệu quả lược đồ ESIT-SSE với một số nghiên cứu

*(**N** là số tài liệu hoặc bản ghi trong CSDL; $\mathbf{N_{sr}}$ là số lượng bản ghi được trả về bởi Server sau truy vấn mã; $\mathbf{N_r}$ là số lượng bản ghi thoả mãn truy vấn ($N_r \leq N_{sr}$); $\mathbf{N_{index}}$ là tổng số lượng giá trị chỉ mục phân biệt; $\mathbf{N_b}$ là tổng số lượng Bucket phân biệt; $\mathbf{N_{br}}$ là số lượng Bucket thoả mãn truy vấn; $\mathbf{C_{decrypt}}$ là chi phí giải mã một bản ghi; $\mathbf{occ}$ là số lần xuất hiện của chuỗi con/từ khoá trong tài liệu/bản ghi có độ dài l_{str}; $\mathbf{k}$ là số hàm băm; $\boldsymbol{\epsilon}$ là tỷ lệ dương tính giả; $\boldsymbol{v}$ là một biến thể hiện sự khác biệt giữa các giá trị trong Dataset; $\mathbf{max\_value}$ giá trị số lớn nhất; $\mathbf{sD}$ là kích cỡ miền giá trị số trong CSDL; $\mathbf{sR}$ là kích cỡ khoảng truy vấn; $\mathbf{eDB}$ là số lượng phần tử/từ khoá trong CSDL; $\mathbf{eR}$ là số lượng phần tử/từ khoá trong khoảng truy vấn; $\mathbf{Lmax}$ là chiều dài tối đa của chỉ mục; $\boldsymbol{\lambda}$ là hệ số dương/tham số bảo mật; $\boldsymbol{\varepsilon}$ là tham số mở rộng; $\mathbf{E_r}$ là hàm mã hoá lại dựa vào PRE; $\mathbf{E_p^{-1}}$ là hàm giải mã dựa vào PRE; $\mathbf{r}$ là số lần lặp; $\mathbf{l_{csc}}$ là độ dài khối CSC-BF (biến thể của bộ lọc Bloom); $\mathbf{l_{enc}}$ là chiều dài bản mã; $\mathbf{n_d}$ là số chiều truy vấn; $\mathbf{l}$ là tham số lớn hơn 3 lần chiều dài nhị phân của dữ liệu; $\mathbf{n_{leaf}}$ là số nút lá được duyệt trên Rtree; $\mathbf{n_{node}}$ là số nút được duyệt trên Rtree; $\mathbf{C_h}$ là chi phí thực hiện hàm băm; $\mathbf{C_b}$ chi phí thực hiện ghép cặp song tuyến; $\mathbf{C_x}$ chi phí thực hiện phép nhân trên nhóm G; $\mathbf{C_{xt}}$ chi phí thực hiện phép nhân trên nhóm Gt)*

TT	Năm - Nghiên cứu	Cấu trúc dữ liệu hỗ trợ xây dựng chỉ mục	Chi phí lưu trữ	Chi phí cho quy trình tìm kiếm	Số vòng giao tiếp	Rò rỉ chỉ mục	Rò rỉ mẫu truy vấn
1	2002 - [47]	Bucket	Server: $O(N)$	Server: - Order Preserving $O(N)$; - Random $O(N.N_{br})$ Client: $O((N_{sr} - N_r).C_{decrypt} + N_{sr})$ Communication: $O(N_{sr} - N_r)$	1	Yes	Yes
2	2014 - [72]	Bloom Filter, PBTree	Server:$O(N.v.log\frac{1}{\epsilon})$	Server: $O(occ.N_r.logN)$ Trapdoor: $O(\log(occ.k))$	1	Yes	Yes
3	2015 - [38]	K-gram, Hash Table, Bloom Filter	Server:$O(N.((log_2($ max_value $))$	Server: $O(\log(N) + N_r)$ Trapdoor: $O(logN)$	1	Yes	Yes
4	2016 - [35]	List	**Quadratic:** Server:$O(N.sD^2)$ **Constant:** Server: $O(N)$ **Logarithmic:** Server:$O(N.logsD)$	**Quadratic:** Server: $O(N_r)$ Trapdoor: $O(1)$ **Constant:** Server: $O(sR + N_r)$ Trapdoor: $O(\log sR)$ **Logarithmic:** Server: $O(\log(sR) + N_r)$ Trapdoor: $O(\log sR)$	1	Yes	Yes
5	2017 - [60]	Array	Server: $O(N)$	Server: $O(\log(N) + N_r)$	>1	No	No
6	2018 - [136]	Binary Tree	Server: $O(N)$ Client: $O(2.eDB - 1)$	Server: $O(log\ eR)$ Client: $O(eDB + log\ eR)$ Communication: $O(N_r)$	1	No	No
7	2019 - [49]	List, R-tree	Server: $O(N)$	Server: $O(eDB.log^2(sD))$	1	No	No
8	2020 - [94]	Binary Tree	Server:$O(N_{index}.(\lambda + (Lmax - 1))$	Server: $O(log\ N_{index})$	1	No	No
9	2022 - [112]	Binary Tree	Server: $O(N)$ Client: $O(2eDB - 1)$	Server: $O(N_r)$ Client: $O(eR)$	2	No	No
10	2022 - [90]	Bucket, Dictionary	Server:$O(5.\lambda.N)$	Server: $O(\varepsilon.N_r.E_r)$, Client: $O(\varepsilon.N_r.E_p^{-1})$ Communication: $O(\varepsilon.N_r)$	1	No	No
11	2022 - [64]	2MTree	Server:$O(log2^N)$	Server: $O(logN)$	1	Yes	Yes
12	2023	Circular Shift, Coalesce	Server:$O(N.(r.l_{csc}.l_{enc}))$	Server: $O(k.N_b.r + ((N - N_r).\epsilon + N_r).r)$	1	No	No

	- [82]	Bloom Filter (CSC-BF)		Trapdoor/Token: $O(k.N_b)$			
13	2023 - [105]	Rtree, Bloom filter	Server: $O(N)$	Server: $O(n_d \,.\ell^2.n_{leaf}.logN)$ Trapdoor/Token: $O(n_d \,.\ell^2)$	1	No	No
14	2023 - [77]	G-tree, VG-tree	Server: $O(N.n_{node})$	$\boldsymbol{MARS^0}$**:** Server: $O(eR.C_x)$ Client: $O(n_{node}.C_h + eR \cdot C_x + (C_x + C_b + C_{xt}))$ Communication: O(n_{node})	2	Yes	Yes
15	ESIT-SSE	Bucket, IHV, IHV_B$^+$Tree	Server: $O(N)$	Server: $O(logN_b + N_{br})$	1	No	No

3.10. Kết luận chương 3

Chương này đề xuất lược đồ ESIT-SSE và quy trình triển khai trên mô hình DAS-PROXY cho phép truy vấn khoảng hiệu quả dữ liệu số trong CSDLQH mã hóa. Theo đó nội dung trọng tâm là xây dựng chỉ mục NewBucketIndex, vector giấu tin IHV và cấu trúc dữ liệu IHV_B$^+$ Tree nhằm tạo ra giải pháp truy vấn khoảng có nhiều ưu điểm trên các tiêu chí về: bảo mật, hiệu năng và khả thi triển khai. Cụ thể:

- Các NewBucketIndex được xây dựng đảm bảo nguyên tắc nạp chồng nhưng vẫn bảo toàn thứ tự. Điều này cho phép các NewBucketIndex tăng cường bảo mật, hỗ trợ tăng tốc truy vấn qua các cấu trúc như B$^+$Tree và giảm bớt bản ghi dương tính giả trả về. Tuy nhiên tính chất bảo toàn thứ tự vẫn gây rò rỉ thông tin cho NewBucketIndex. Để giải quyết vấn đề này, các NewBucketIndex tiếp tục được biến đổi thành vector giấu tin IHV với khả năng che giấu thứ tự nhưng vẫn hỗ trợ cơ chế so sánh bảo mật. Để tăng hiệu quả tìm kiếm dựa trên tính chất bảo toàn thứ tự của NewBucketIndex và bảo mật cao của IHV, cấu trúc IHV_B$^+$ Tree được xây dựng cho phép truy vấn khoảng thông qua thao tác duyệt cây tốc độ cao thay vì thực hiện quét toàn bộ các bản ghi trong bảng của CSDLQH mã hóa.

- Tính khả thi được kế thừa từ các lợi thế triển khai của mô hình DAS-PROXY, hỗ trợ truy vấn với một số lệnh mở rộng và cho phép thay đổi các tham số tác động liên quan. Trong khi đó, tính bảo mật (khả năng chống rò rỉ thông tin chỉ mục, mẫu truy vấn) và hiệu năng (kết quả trả về không dư thừa, thời gian thực thi, độ phức tạp tính toán, số vòng giao tiếp) được làm rõ qua các kịch bản tấn công, kịch bản thực nghiệm và bảng so sánh với các công trình tiêu biểu liên quan.

Bên cạnh các ưu điểm đã nêu, thì lược đồ ESIT-SSE cũng còn tồn tại một số hạn chế như sau:

- Số lượng m Bucket càng lớn thì độ bảo mật của NewBucketIndex càng giảm và ngược lại. Nếu điều chỉnh m nhỏ, một NewBucketIndex sẽ đại diện cho nhiều giá trị bản rõ hơn và kéo theo số phần tử của mỗi $List_{i(j)}$ gán với từng nút lá (trong Thuật toán 3.4: $Build\_IHV\_B^{+}Tree$) tăng lên làm chi phí duyệt cây tốn kém hơn.

- Các phần tử trong mỗi danh sách $List_{i(j)}$ được sắp xếp ngẫu nhiên nhằm tăng cường tính bảo mật, nhưng lại làm thao tác duyệt tìm các phần tử $\overrightarrow{ihv_b(v_i)}$ trong danh sách mà thoả mãn điều kiện truy vấn khoảng sẽ phức tạp hơn so với một danh sách đã được sắp xếp.

- Tham số k càng lớn càng tăng khả năng chống rò rỉ của vector IHV, tuy nhiên kéo theo chi phí lưu trữ vector IHV cũng tăng cao và chi phí thực hiện phép tích vô hướng giữa các vector giấu tin trong quá trình duyệt cây cũng bị ảnh hưởng.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Trong luận án này, NCS đã nghiên cứu, phân loại và mô tả đầy đủ bức tranh về các giải pháp liên quan trong lĩnh vực SE nói chung cũng như truy vấn trên dữ liệu kiểu số và ký tự mã hóa nói riêng. Từ đó chỉ ra các khoảng trống về tính hiệu quả của các lược đồ SE trước đó, là căn cứ để luận án tiếp tục nghiên cứu và đề xuất giải pháp cải thiện. Cụ thể, luận án đề xuất các lược đồ SSE mới dựa trên chỉ mục mù nhằm hỗ trợ truy vấn chuỗi con và truy vấn khoảng hiệu quả trên CSDLQH mã hóa. Tính hiệu quả được thể hiện qua các tiêu chí về: chống rò rỉ thông tin dữ liệu rõ, hiệu năng truy vấn, tính khả thi trong triển khai và hỗ trợ dạng thức truy vấn phổ biến.

A. Các kết quả đạt được của luận án

1) Áp dụng thành công mô hình DAS-PROXY để triển khai thống nhất cho hai lược đồ DIQ-SSE và ESIT-SSE đề xuất trong luận án. Máy chủ Proxy trong mô hình cho phép: quản lý và lưu trữ an toàn các khóa/metadata/thuật toán bí mật; giảm tải cho thiết bị cuối của EU; và hỗ trợ các vấn đề mở rộng liên quan tới vận hành.

Mô hình DAS-PROXY đảm bảo tiêu chí *"tính khả thi trong triển khai"* khi xét tới hiệu quả của các lược đồ đề xuất trong luận án.

2) Đề xuất lược đồ DIQ-SSE hỗ trợ truy vấn chuỗi con hiệu quả trong CSDLQH mã hóa. Các điểm đóng góp gồm: đề xuất các cấu trúc dữ liệu mới như *"Tập các cặp ký tự kết nối phân biệt theo khoảng cách trên chuỗi rõ"* hay *"Tập các mảng nhị phân biểu diễn vị trí ký tự của chuỗi rõ"* để xây dựng cặp chỉ mục $Index1, Index2$ hỗ trợ tìm kiếm; đề xuất quy trình truy vấn hiệu quả và các thuật toán liên quan trên cặp chỉ mục $Index1, Index2$.

Tính hiệu quả của lược đồ DIQ-SSE được đánh giá qua phân tích các kịch bản tấn công, kịch bản thực nghiệm trên số lượng bản ghi lớn của CSDL TPC-H cũng như qua so sánh tương quan với các công trình tiêu biểu liên quan. ***Về bảo mật:*** các chỉ mục và mẫu truy vấn có khả năng chống rò rỉ thông tin bản rõ; ***Về hiệu năng:*** ghi nhận ưu điểm về tỷ lệ lọc, tỷ lệ lỗi và thời gian thực thi truy vấn và quy trình truy vấn tuần tự tận dụng được lợi thế của các chỉ mục. Bên cạnh đó, quá trình truy vấn chỉ sử dụng 1 vòng giao tiếp dữ liệu từ EU đến CS với độ phức tạp tìm kiếm cỡ $O(k)$ trên $Index1$ và $O(2.l_{substr})$ trên $Index2$. ***Về hỗ trợ dạng thức truy vấn phổ biến:*** cho phép thực thi chuỗi "$LIKE\ '\%\ substring\ \%'$", là điều kiện phổ biến và bao quát nhất trong truy vấn trên dữ liệu ký tự. Ngoài ra lược đồ cho phép điều chỉnh các tham số

$u,\ m$ tác động đến hiệu năng, bảo mật và không gian lưu trữ, qua đó thể hiện tính khả thi của lược đồ DIQ-SSE trong triển khai thực tiễn.

3) Đề xuất lược đồ ESIT-SSE hỗ trợ truy vấn khoảng hiệu quả trên dữ liệu số trong CSDLQH mã hóa. Các điểm mới trong nội dung đề xuất gồm: xây dựng chỉ mục NewBucketIndex, vector giấu tin (IHV) và cây IHV_B^+ Tree; xây dựng quy trình và các thuật toán hỗ trợ truy vấn khoảng trên cây IHV_B^+ Tree.

Tính hiệu quả của lược đồ ESIT-SSE được đánh giá qua phân tích các kịch bản tấn công, kịch bản thực nghiệm trên số lượng bản ghi lớn của CSDL TPC-H cũng như qua so sánh tương quan với các công trình tiêu biểu liên quan. ***Về bảo mật:*** các chỉ mục NewBucketIndex, vector IHV, cây IHV_B^+ Tree và mẫu truy vấn có khả năng chống rò rỉ thông tin bản rõ; ***Về hiệu năng:*** hỗ trợ 1 vòng giao tiếp duy nhất giữa EU với CS; thuật toán duyệt cây IHV_B^+ Tree tốc độ cao (độ phức tạp cỡ $O(logN_b + N_{br})$) giúp giảm đáng kể chi phí thời gian truy vấn. ***Về hỗ trợ dạng thức truy vấn phổ biến:*** cho phép thực thi chuỗi "$\boldsymbol{X_t}\ Between\ \boldsymbol{l}\ And\ \boldsymbol{h}$", là điều kiện bao quát nhất trong truy vấn trên dữ liệu số. Ngoài ra lược đồ cho phép điều chỉnh các tham số $k,\ m$ tác động đến hiệu năng, bảo mật và không gian lưu trữ, qua đó thể hiện tính khả thi của lược đồ ESIT-SSE trong triển khai thực tiễn.

B. Hướng phát triển của luận án

Bài toán truy vấn trên dữ liệu mã được đánh giá phức tạp do các ràng buộc về tính bảo mật, hiệu năng, tính khả thi triển khai và đa dạng truy vấn. Luận án đã giới hạn phạm vi nghiên cứu tập trung chủ yếu vào truy vấn trên dạng thức dữ liệu ký tự và số với các điều kiện truy vấn điển hình và phổ quát. Các đóng góp này có ý nghĩa quan trọng, làm cơ sở để có thể mở rộng phạm vi nghiên cứu luận án, đem lại cơ hội ứng dụng cao hơn đối với các lược đồ đề xuất trong thực tiễn.

Ngoài các đóng góp đã được trình bày, NCS nhận thấy còn một số tồn tại trong luận án cùng các vấn đề mở rộng khác cũng nên được quan tâm giải quyết giúp hoàn thiện hơn các lược đồ đã đề xuất. Cụ thể:

- Nghiên cứu tiếp tục đa dạng hóa thêm các lệnh và điều kiện truy vấn.
- Nghiên cứu tối ưu chi phí không gian lưu trữ chỉ mục.
- Nghiên cứu chống rò rỉ thông tin mẫu truy cập (access pattern).
- Nghiên cứu các vấn đề mở rộng khác trong triển khai vận hành các lược đồ như: đổi khóa, biến động dữ liệu, cân bằng tải, …

DANH MỤC CÔNG TRÌNH KHOA HỌC ĐÃ CÔNG BỐ

[CT1] **C. Ngoc Hoang**, M. Hieu Nguyen, T. Thu Thi Nguyen, and H. Quang Vu, "A novel method for designing indexes to support efficient substring queries on encrypted databases," *Journal of King Saud University - Computer and Information Sciences*, vol. 35, no. 3, pp. 20–36, Mar. 2023, doi: 10.1016/j.jksuci.2023.02.008. **(*Journal: Scopus Index Q1, SCIE IF 5.2*)**

[CT2] **Hoàng Ngọc Cảnh**, Nguyễn Hiếu Minh, "Truy vấn hiệu quả dữ liệu ký tự mã hóa trong cơ sở dữ liệu thuê ngoài", *Kỷ yếu Hội nghị khoa học công nghệ quốc gia lần thứ XV – Nghiên cứu cơ bản và Ứng dụng công nghệ thông tin (FAIR 2022)*, ISBN: 978-604-357-119-6, Trang 8-15.

[CT3] **Hoang, N.C.**, Nguyen, T.T.T. (2024). "Build Feature Vector for String Supporting Pattern Matching on Encrypted Character Data". *In: Nguyen, T.D.L., Dawson, M., Ngoc, L.A., Lam, K.Y. (eds) Proceedings of the International Conference on Intelligent Systems and Networks. ICISN 2024.* Lecture Notes in Networks and Systems, vol 1077. Springer, Singapore. https://doi.org/10.1007/978-981-97-5504-2_41. **(*Proceedings: Scopus Index Q4*)**

[CT4] **Hoang, N.C.**, Nguyen, T.T.T., Tran, L.K.D., Vu, Q.H. (2024). "A Secure Approach to Financial Data Management: A Method for Constructing Encrypted Indexes to Support Efficient Querying without Revealing Order Information". *Hanoi University of Industry Journal of Science and Technology (P-ISSN 1859-3585, E-ISSN 2615-9619)*, vol.60, no.11E (Nov 2024), doi: http://doi.org/10.57001/huih5804.2024.344.

[CT5] **Hoang, N.C.**, Nguyen, T.T.T., Tran, L.K.D., Vu, Q.H. (2024). "Design Index Supporting Efficient Querying On Encrypted Numerical Data Based On Stacked Bucket Loading and Order Preservation". *In: Proceedings of the International Conference on Applied Mathematics and Computer Science (ICAMCS2024)*. Lecture Notes in Networks and Systems, vol 1, Springer, Singapore. **(*Proceedings: Scopus Index Q4*)**

TÀI LIỆU THAM KHẢO

Tiếng Việt

1. Nguyễn Anh Tuấn, (2011), Về một giải pháp bảo mật cơ sở dữ liệu — LUẬN ÁN TIẾN SĨ.
2. Phạm Thị Bạch Huệ, (2013), Nghiên cứu và phát triển một số giải pháp bảo mật và bảo vệ tính riêng tư trong cơ sở dữ liệu thuê ngoài (ODBS) - LUẬN ÁN TIẾN SỸ.
3. Hồ Kim Giàu, (2021), Nghiên cứu phát triển giải pháp xác thực an toàn và quản lý khoá cho cơ sở dữ liệu thuê ngoài - LUẬN ÁN TIẾN SĨ.

Tiếng Anh

4. How to Search on Encrypted Data: Deterministic Encryption (Part 2). <https://esl.cs.brown.edu/blog/how-to-search-on-encrypted-data-deterministic-encryption-part-2/>, accessed: 04/19/2024.
5. Agrawal R., Kiernan J., Srikant R. et al. (2004). Order preserving encryption for numeric data. *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, New York, NY, USA, Association for Computing Machinery, 563–574.
6. Almakdi S., Panda B., Alshehri M.S. et al. (2021). An Efficient Secure System for Fetching Data From the Outsourced Encrypted Databases. *IEEE Access*, **9**, 78474–78494.
7. Amorim I. and Costa I. (2023). Homomorphic Encryption: An Analysis of its Applications in Searchable Encryption. *Mathematics*, **11**(**13**), 2948.
8. Andola N., Prakash S., Yadav V.K. et al. (2022). A secure searchable encryption scheme for cloud using hash-based indexing. *Journal of Computer and System Sciences*, **126**, 119–137.
9. Azraoui M., Önen M., and Molva R. (2018). Framework for Searchable Encryption with SQL Databases:. *Proceedings of the 8th International Conference on Cloud Computing and Services Science*, Funchal, Madeira, Portugal, SCITEPRESS - Science and Technology Publications, 57–67.
10. Baron J., El Defrawy K., Minkovich K. et al. (2012). 5PM: Secure Pattern Matching. *Security and Cryptography for Networks*, Berlin, Heidelberg, Springer, 222–240.
11. Bast H., Buchhold B., and Haussmann E. (2016). Semantic Search on Text and Knowledge Bases. *Foundations and Trends® in Information Retrieval*, **10**, 119–271.
12. Bin L. and Jianguo J. (2016), *Search on encrypted data: State of arts*, .
13. Biryukov A. and De Cannière C. (2005). Data encryption standard (DES). *Encyclopedia of Cryptography and Security*. Springer US, Boston, MA, 129–135.
14. Bloom B.H. (1970). Space/time trade-offs in hash coding with allowable errors. *Commun ACM*, **13**(**7**), 422–426.
15. Blumer A., Blumer J., Haussler D. et al. (1985). The smallest automation recognizing the subwords of a text. *Theoretical Computer Science*, **40**, 31–55.
16. Boldyreva A. and Chenette N. (2014). Efficient Fuzzy Search on Encrypted Data. <https://eprint.iacr.org/2014/235>, accessed: 04/19/2024.

17. Boldyreva A., Chenette N., Lee Y. et al. (2009). Order-Preserving Symmetric Encryption. *Advances in Cryptology - EUROCRYPT 2009*, Berlin, Heidelberg, Springer, 224–241.
18. Boldyreva A., Chenette N., and O'Neill A. (2011). Order-Preserving Encryption Revisited: Improved Security Analysis and Alternative Solutions. *Advances in Cryptology – CRYPTO 2011*, Berlin, Heidelberg, Springer, 578–595.
19. Boneh D., Di Crescenzo G., Ostrovsky R. et al. (2004). Public Key Encryption with Keyword Search. *Advances in Cryptology - EUROCRYPT 2004*, Berlin, Heidelberg, Springer, 506–522.
20. Boneh D. and Waters B. (2007). Conjunctive, Subset, and Range Queries on Encrypted Data. *Theory of Cryptography*, Berlin, Heidelberg, Springer, 535–554.
21. Bösch C., Hartel P., Jonker W. et al. (2014). A Survey of Provably Secure Searchable Encryption. *ACM Comput Surv*, **47**(**2**), 18:1-18:51.
22. Brakerski Z. (2012). Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP. *Advances in Cryptology – CRYPTO 2012*, Berlin, Heidelberg, Springer, 868–886.
23. Burrows M., L D.J.W.D.I.G.I.T.A., Taylor R.W. et al. (1994). A Block-sorting Lossless Data Compression Algorithm. .
24. Cao N., Wang C., Li M. et al. (2011), *Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data*, .
25. Cash D., Jaeger J., Jarecki S. et al. (2014). Dynamic Searchable Encryption in Very-Large Databases: Data Structures and Implementation. *Proceedings 2014 Network and Distributed System Security Symposium*.
26. Cash D., Jarecki S., Jutla C. et al. (2013). Highly-Scalable Searchable Symmetric Encryption with Support for Boolean Queries. *Advances in Cryptology – CRYPTO 2013*, Berlin, Heidelberg, Springer, 353–373.
27. Chamili K., Nordin Md.J., Ismail W. et al. (2017). Searchable Encryption: A Review. *IJSIA*, **11**(**12**), 79–88.
28. Chang Y.-C. and Mitzenmacher M. (2005). Privacy Preserving Keyword Searches on Remote Encrypted Data. *Applied Cryptography and Network Security*, Berlin, Heidelberg, Springer, 442–455.
29. Chase M. and Kamara S. (2010). Structured Encryption and Controlled Disclosure. *Advances in Cryptology - ASIACRYPT 2010*, Berlin, Heidelberg, Springer, 577–594.
30. Chase M. and Shen E. (2015). Substring-Searchable Symmetric Encryption. *Proceedings on Privacy Enhancing Technologies*, **2015**.
31. Chen C., Zhu X., Shen P. et al. (2016). An Efficient Privacy-Preserving Ranked Keyword Search Method. *IEEE Transactions on Parallel and Distributed Systems*, **27**(**4**), 951–963.
32. Chen L., Xingming S., Xia Z. et al. (2014). An Efficient and Privacy-Preserving Semantic Multi-Keyword Ranked Search over Encrypted Cloud Data. *International Journal of Security and Its Applications*, **8**, 323–332.

33. Curtmola R., Garay J., Kamara S. et al. (2011). Searchable symmetric encryption: Improved definitions and efficient constructions. *Journal of Computer Security*, **19**, 895–934.
34. Damiani E., Vimercati S.D.C., Jajodia S. et al. (2003). Balancing confidentiality and efficiency in untrusted relational DBMSs. *Proceedings of the 10th ACM conference on Computer and communications security*, New York, NY, USA, Association for Computing Machinery, 93–102.
35. Demertzis I., Papadopoulos S., Papapetrou O. et al. (2016). Practical Private Range Search Revisited. *Proceedings of the 2016 International Conference on Management of Data*, New York, NY, USA, Association for Computing Machinery, 185–198.
36. Ding X., Liu P., and Jin H. (2017). Privacy-Preserving Multi-Keyword Top- Similarity Search Over Encrypted Data. *IEEE Transactions on Dependable and Secure Computing*, **PP**, 1–1.
37. Dworkin M.J. (2023), *Advanced Encryption Standard (AES)*, National Institute of Standards and Technology (U.S.), Gaithersburg, MD.
38. Faber S., Jarecki S., Krawczyk H. et al. (2015). Rich Queries on Encrypted Data: Beyond Exact Matches. <https://eprint.iacr.org/2015/927>, accessed: 04/19/2024.
39. Fan L., Cao P., Almeida J. et al. (2000). Summary cache: a scalable wide-area Web cache sharing protocol. *IEEE/ACM Transactions on Networking*, **8**(**3**), 281–293.
40. Fu Z., Xingming S., Xia Z. et al. (2013), *Multi-keyword ranked search supporting synonym query over encrypted data in cloud computing*, .
41. Gentry C. (2009), *A fully homomorphic encryption scheme*, phd, Stanford University, Stanford, CA, USA.
42. Gentry C., Halevi S., and Smart N.P. (2012). Homomorphic Evaluation of the AES Circuit. *Advances in Cryptology – CRYPTO 2012*, Berlin, Heidelberg, Springer, 850–867.
43. Goh E.-J. (2003). Secure Indexes. <https://eprint.iacr.org/2003/216>, accessed: 04/19/2024.
44. Goh E.-J. How to Search on Encrypted Data. .
45. Goldreich O. and Ostrovsky R. (1996). Software protection and simulation on oblivious RAMs. *J ACM*, **43**(**3**), 431–473.
46. Golle P., Staddon J., and Waters B. (2004). Secure Conjunctive Keyword Search over Encrypted Data. *Applied Cryptography and Network Security*, Berlin, Heidelberg, Springer, 31–45.
47. Hacigümüs H., Iyer B., Li C. et al. (2002), *Executing SQL over Encrypted Data in the Database-Service-Provider Model*, .
48. Hadavi M.A., Jalili R., Damiani E. et al. (2015). Security and searchability in secret sharing-based data outsourcing. *Int J Inf Secur*, **14**(**6**), 513–529.
49. Hahn F. (2019). Practical yet Provably Secure: Complex Database Query Execution over Encrypted Data. <https://publikationen.bibliothek.kit.edu/1000091007>, accessed: 09/07/2024.

50. Hahn F., Loza N., and Kerschbaum F. (2018). Practical and Secure Substring Search. *Proceedings of the 2018 International Conference on Management of Data*, New York, NY, USA, Association for Computing Machinery, 163–176.
51. Handa R., Krishna C.R., and Aggarwal N. (2019). Searchable encryption: A survey on privacy-preserving search schemes on encrypted outsourced data. *Concurrency and Computation: Practice and Experience*, **31**(**17**), e5201.
52. Ho C., Pak K., Pak S. et al. (2019). A Study on Improving the Performance of Encrypted Database Retrieval Using External Indexing System of B+ Tree Structure. *Procedia Computer Science*, **154**, 706–714.
53. Hore B., Mehrotra S., and Tsudik G. (2004). A Privacy-Preserving Index for Range Queries. *Proc Inte Conf on Very large data bases (VLDB'04)*.
54. Hwang Y.H. and Lee P.J. (2007). Public Key Encryption with Conjunctive Keyword Search and Its Extension to a Multi-user System. *Pairing-Based Cryptography – Pairing 2007*, Berlin, Heidelberg, Springer, 2–22.
55. Jaganathan Manonmani A. (2023). Confidential Computing with SQL Server using Always Encrypted Technique. .
56. Kadhem H., Amagasa T., and Kitagawa H. (2013). Optimization Techniques for Range Queries in the Multivalued-partial Order Preserving Encryption Scheme. *Knowledge Discovery, Knowledge Engineering and Knowledge Management*, Berlin, Heidelberg, Springer, 338–353.
57. Kamara S., Papamanthou C., and Roeder T. (2012). Dynamic Searchable Symmetric Encryption. <https://eprint.iacr.org/2012/530>, accessed: 10/28/2024.
58. Karras P., Nikitin A., Saad M. et al. (2016). Adaptive Indexing over Encrypted Numeric Data. *Proceedings of the 2016 International Conference on Management of Data*.
59. Katz J., Sahai A., and Waters B. (2008). Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. *Advances in Cryptology – EUROCRYPT 2008*, Berlin, Heidelberg, Springer, 146–162.
60. Kerschbaum F. and Tueno A. (2019). An Efficiently Searchable Encrypted Data Structure for Range Queries. *Computer Security – ESORICS 2019*, Cham, Springer International Publishing, 344–364.
61. Kim H.-I., Kim H.-J., and Chang J.-W. (2016), *A kNN query processing algorithm using a tree index structure on the encrypted database*, .
62. Krähenbühl C. and Perrig A. (2023). Searchable Symmetric Encryption. *Trends in Data Protection and Encryption Technologies*. Springer Nature Switzerland, Cham, 71–75.
63. Kumar D.V.N. and Thilagam P. (2018). Approaches and challenges of privacy preserving search over encrypted data. *Information Systems*, **81**.
64. Kwon H., Hur J., and Hahn C. (2022). Order-Hiding Range Query Over Encrypted Cloud Data. *IEEE Access*, **10**, 75604–75618.
65. Lai J., Zhou X., Deng R.H. et al. (2013). Expressive search on encrypted data. *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*, New York, NY, USA, Association for Computing Machinery, 243–252.

66. Lee S., Park T.-J., Lee D. et al. (2009). Chaotic Order Preserving Encryption for Efficient and Secure Queries on Databases. *IEICE Transactions*, **92-D**, 2207–2217.

67. Leontiadis I. and Li M. (2018). Storage Efficient Substring Searchable Symmetric Encryption. *Proceedings of the 6th International Workshop on Security in Cloud Computing*, New York, NY, USA, Association for Computing Machinery, 3–13.

68. Li F., Ma J., Miao Y. et al. (2023). A Survey on Searchable Symmetric Encryption. *ACM Comput Surv*, **56**(**5**), 119:1-119:42.

69. Li J., Wang Q., Wang C. et al. (2010). Fuzzy Keyword Search over Encrypted Data in Cloud Computing. *2010 Proceedings IEEE INFOCOM*, 1–5.

70. Li J., Wang Q., Wang C. et al. (2009). Enabling Efficient Fuzzy Keyword Search over Encrypted Data in Cloud Computing. <https://eprint.iacr.org/2009/593>, accessed: 04/19/2024.

71. Li K., Zhang W., Yang C. et al. (2015). Security Analysis on One-to-Many Order Preserving Encryption-Based Cloud Data Search. *IEEE Transactions on Information Forensics and Security*, **10**(**9**), 1918–1926.

72. Li R., Liu A.X., Wang A.L. et al. (2014). Fast range query processing with strong privacy protection for cloud computing. *Proc VLDB Endow*, **7**(**14**), 1953–1964.

73. Lipmaa H. (2005). An Oblivious Transfer Protocol with Log-Squared Communication. *Information Security*. Springer Berlin Heidelberg, Berlin, Heidelberg, 314–328.

74. Liu A.X. and Chen F. (2008). Collaborative enforcement of firewall policies in virtual private networks. *Proceedings of the twenty-seventh ACM symposium on Principles of distributed computing*, New York, NY, USA, Association for Computing Machinery, 95–104.

75. Liu C., Zhu L., Li L. et al. Fuzzy keyword search on encrypted cloud storage data with small index. *2011 IEEE International Conference on Cloud Computing and Intelligence Systems*. .

76. Liu L. and Gai J. (2009). Bloom Filter Based Index for Query over Encrypted Character Strings in Database. *2009 WRI World Congress on Computer Science and Information Engineering*, 303–307.

77. Liu Q., Peng Y., Xu Q. et al. (2023). {\sf MARS}$: Enabling Verifiable Range-Aggregate Queries in Multi-Source Environments. *IEEE Transactions on Dependable and Secure Computing*, **PP**, 1–16.

78. Lu W., Kawasaki S., and Sakuma J. (2017), *Using Fully Homomorphic Encryption for Statistical Analysis of Categorical, Ordinal and Numerical Data*, .

79. Mainardi N., Barenghi A., and Pelosi G. (2019). Privacy preserving substring search protocol with polylogarithmic communication cost. *Proceedings of the 35th Annual Computer Security Applications Conference*, New York, NY, USA, Association for Computing Machinery, 297–312.

80. Mainardi N., Barenghi A., and Pelosi G. (2022). Privacy-aware Character Pattern Matching over Outsourced Encrypted Data. *Digital Threats*, **3**(**1**), 1–38.

81. Mani M., Shah K., and Gunda M. (2013). Enabling Secure Database as a Service using Fully Homomorphic Encryption: Challenges and Opportunities. .

82. Miao Y., Yang Y., Li X. et al. (2023). Efficient Privacy-Preserving Spatial Range Query Over Outsourced Encrypted Data. *IEEE Transactions on Information Forensics and Security*, **18**, 3921–3933.

83. Moataz T., Ray I., Ray I. et al. (2018). Substring search over encrypted data. *Journal of Computer Security*, **26**(**1**), 1–30.

84. Moataz T. and Shikfa A. (2013). Boolean symmetric searchable encryption. *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*, New York, NY, USA, Association for Computing Machinery, 265–276.

85. Mullin J.K. (1983). A second look at bloom filters. *Commun ACM*, **26**(**8**), 570–571.

86. Örencik C. and Savaş E. (2012). Efficient and secure ranked multi-keyword search on encrypted cloud data. *Proceedings of the 2012 Joint EDBT/ICDT Workshops*, New York, NY, USA, Association for Computing Machinery, 186–195.

87. Örencik C. and Savas E. (2014). An efficient privacy-preserving multi-keyword search over encrypted cloud data with ranking. *Distributed and Parallel Databases*, **32**.

88. Pappas V., Krell F., Vo B. et al. (2014). Blind seer: A scalable private DBMS. *Proceedings - IEEE Symposium on Security and Privacy*, 359–374.

89. Park H.-A., Park J.H., and Lee D.H. (2011). PKIS: practical keyword index search on cloud datacenter. *EURASIP Journal on Wireless Communications and Networking*, **2011**(**1**), 64.

90. Peng Y., Wang L., Cui J. et al. (2022). LS-RQ: A Lightweight and Forward-Secure Range Query on Geographically Encrypted Data. *IEEE Transactions on Dependable and Secure Computing*, **19**(**1**), 388–401.

91. Pilyankevich E., Kornieiev D., and Storozhuk A. (2019). Proxy-Mediated Searchable Encryption in SQL Databases Using Blind Indexes. *IACR Cryptol ePrint Arch*.

92. Raybourn T. Bucketization Techniques for Encrypted Databases: Quantifying the Impact of Query Distributions. .

93. Raykova M., Vo B., Bellovin S. et al. (2009), *Secure anonymous database search*, .

94. Ren K., Guo Y., Li J. et al. (2020). HybrIDX: New Hybrid Index for Volume-hiding Range Queries in Data Outsourcing Services. *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, 23–33.

95. Salam M., Yau W.-C., Chin J.-J. et al. (2015). Implementation of searchable symmetric encryption for privacy-preserving keyword search on cloud storage. *Human-centric Computing and Information Sciences*, **5**, 19.

96. Saleh E., Alsa'deh A., Kayed A. et al. (2016). Processing Over Encrypted Data: Between Theory and Practice. *SIGMOD Rec*, **45**(**3**), 5–16.

97. Santos N., Raj H., Saroiu S. et al. (2014), *Using ARM TrustZone to Build a Trusted Language Runtime for Mobile Applications*, .

98. Schneier B. (1994). Description of a new variable-length key, 64-bit block cipher (Blowfish). *Fast Software Encryption*. Springer Berlin Heidelberg, Berlin, Heidelberg, 191–204.

99. Shi E., Bethencourt J., Chan T.-H.H. et al. (2007). Multi-Dimensional Range Query over Encrypted Data. *2007 IEEE Symposium on Security and Privacy (SP '07)*, 350–364.
100. Song D.X., Wagner D., and Perrig A. (2000). Practical techniques for searches on encrypted data. *Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000*, 44–55.
101. Song W., Wang B., Wang Q. et al. (2016). A privacy-preserved full-text retrieval algorithm over encrypted data for cloud storage applications. *Journal of Parallel and Distributed Computing*, **99**.
102. Strizhov M., Osman Z., and Ray I. (2016). Substring Position Search over Encrypted Cloud Data Supporting Efficient Multi-User Setup. *Future Internet*, **8**(**3**), 28.
103. Strizhov M. and Ray I. (2014). Multi-keyword Similarity Search over Encrypted Cloud Data. *ICT Systems Security and Privacy Protection*, Berlin, Heidelberg, Springer, 52–65.
104. Sun W., Wang B., Cao N. et al. (2013), *Privacy-preserving Multi-keyword Text Search in the Cloud Supporting Similarity-based Ranking*, .
105. Tu X., Bao H., Lu R. et al. (2024). PMRK: Privacy-Preserving Multidimensional Range Query With Keyword Search Over Spatial Data. *IEEE Internet of Things Journal*, **11**(**6**), 10464–10478.
106. Ucal M. (2007). Searching on Encrypted Data. *Current Sociology - CURR SOCIOL*.
107. Varri U., Pasupuleti S., and K V K. (2020). A scoping review of searchable encryption schemes in cloud computing: taxonomy, methods, and recent developments. *The Journal of Supercomputing*, **76**.
108. Wang B., Song W., Lou W. et al. (2015), *Inverted index based multi-keyword public-key searchable encryption with strong privacy guarantee*, .
109. Wang C., Cao N., Li J. et al. (2010). Secure Ranked Keyword Search over Encrypted Cloud Data. *2010 IEEE 30th International Conference on Distributed Computing Systems*, 253–262.
110. Wang C., Ren K., Yu S. et al. (2012). Achieving usable and privacy-assured similarity search over outsourced cloud data. *2012 Proceedings IEEE INFOCOM*, 451–459.
111. Wang G., Liu C., Dong Y. et al. IDCrypt: A Secure and Practical Searchable Encryption Scheme for Cloud Applications. .
112. Wang J. and Chow S.S.M. (2022). Forward and Backward-Secure Range-Searchable Symmetric Encryption. *Proceedings on Privacy Enhancing Technologies*, **2022**(**1**), 28–48.
113. Wang J. and Du X. (2008). LOB: Bucket Based Index for Range Queries. *2008 The Ninth International Conference on Web-Age Information Management*, 86–92.
114. Wang P., Wang H., and Pieprzyk J. (2009). An Efficient Scheme of Common Secure Indices for Conjunctive Keyword-Based Retrieval on Encrypted Data. *Information Security Applications: 9th International Workshop, WISA 2008, Jeju Island, Korea, September 23-25, 2008, Revised Selected Papers*. Springer-Verlag, Berlin, Heidelberg, 145–159.

115. Wang P., Wang H., and Pieprzyk J. (2007). Common Secure Index for Conjunctive Keyword-Based Retrieval over Encrypted Data. *Secure Data Management*, Berlin, Heidelberg, Springer, 108–123.

116. Wang P., Xiang T., Li X. et al. (2020). Public key encryption with conjunctive keyword search on lattice. *Journal of Information Security and Applications*, **51**, 102433.

117. Wang S., Ye J., and Zhang Y. (2018). A keyword searchable attribute-based encryption scheme with attribute update for cloud storage. *PLOS ONE*, **13**, e0197318.

118. Wang W. and Lakshmanan L. (2006), *Efficient Secure Query Evaluation over Encrypted XML Databases.*, .

119. Wang Y., Wang J., and Chen X. (2016). Secure searchable encryption: a survey. *Journal of Communications and Information Networks*, **1**, 52–65.

120. Wang Z.-F., Dai J., Wang W. et al. (2005). Fast Query Over Encrypted Character Data in Database. *Computational and Information Science*, Berlin, Heidelberg, Springer, 1027–1033.

121. Wang Z.-F., Tang A.-G., and Wang W. (2009). Fast query over encrypted data based on B+ tree. *2009 International Conference on Apperceiving Computing and Intelligence Analysis*, 132–135.

122. Wang Z.-F., Wang W., and Shi B.-L. (2005). Storage and Query over Encrypted Character and Numerical Data in Database. *The Fifth International Conference on Computer and Information Technology (CIT'05)*, 77–81.

123. Watanabe C. and Arai Y. (2009). Privacy-Preserving Queries for a DAS Model Using Encrypted Bloom Filter. *Database Systems for Advanced Applications*, Berlin, Heidelberg, Springer, 491–495.

124. Watanabe Y., NAKAI T., OHARA K. et al. (2022). How to Make a Secure Index for Searchable Symmetric Encryption, Revisited. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, **E105.A**.

125. Wong W.K., Kao B., Cheung D.W.L. et al. (2014). Secure query processing with data interoperability in a cloud database environment. *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, New York, NY, USA, Association for Computing Machinery, 1395–1406.

126. Wu Z., Xu G., Yu Z. et al. (2012). Executing SQL queries over encrypted character strings in the Database-As-Service model. *Know-Based Syst*, **35**, 332–348.

127. Xingming S., Zhu Y., Xia Z. et al. (2014). Privacy- Preserving Keyword-based Semantic Search over Encrypted Cloud Data. *International Journal of Security and Its Applications*, **8**, 9–20.

128. Xu Q., Shen H., Sang Y. et al. (2013). Privacy-Preserving Ranked Fuzzy Keyword Search over Encrypted Cloud Data. *2013 International Conference on Parallel and Distributed Computing, Applications and Technologies*, 239–245.

129. Yamamoto H. (2016). Secure Automata-Based Substring Search Scheme on Encrypted Data. *Advances in Information and Computer Security*, Cham, Springer International Publishing, 111–131.

130. Yamamoto H., Oda R., Wachi Y. et al. (2022). Substring Searchable Symmetric Encryption Based on an Improved DAWG. *IEICE Trans Fundamentals*, **E105.A**(**12**), 1578–1590.

131. Yin F., Lu R., Zheng Y. et al. (2021). Achieve efficient position-heap-based privacy-preserving substring-of-keyword query over cloud. *Computers & Security*, **110**, 102432.

132. Yin H., Qin Z., Zhang J. et al. (2021). Achieving Secure, Universal, and Fine-Grained Query Results Verification for Secure Search Scheme Over Encrypted Cloud Data. *IEEE Transactions on Cloud Computing*, **9**(**01**), 27–39.

133. Zhang X., Su Y., and Qin J. (2020). A Dynamic Searchable Symmetric Encryption Scheme for Multiuser with Forward and Backward Security. *Security and Communication Networks*, **2020**, e8893016.

134. Zhou Y., Li N., Tian Y. et al. (2020). Public Key Encryption with Keyword Search in Cloud: A Survey. *Entropy*, **22**(**4**), 421.

135. Zhu T. (2017). Executing SQL Queries over Encrypted Data: A Survey. **4**(**3**).

136. Zuo C., Sun S.-F., Liu J.K. et al. (2018). Dynamic Searchable Symmetric Encryption Schemes Supporting Range Queries with Forward (and Backward) Security. *Computer Security*, Cham, Springer International Publishing, 228–246.

137. Toward Securing Cloud-Based Data Analytics: A Discussion on Current Solutions and Open Issues | IEEE Journals & Magazine | IEEE Xplore. <https://ieeexplore.ieee.org/document/8678751>, accessed: 09/05/2024.

138. Privacy-Preserving Substring Search on Multi-Source Encrypted Gene Data | IEEE Journals & Magazine | IEEE Xplore. <https://ieeexplore.ieee.org/document/9034096>, accessed: 09/06/2024.

139. Zhu H., Cheng J., Jin R. et al. (2007). Executing Query over Encrypted Character Strings in Databases. *2007 Japan-China Joint Workshop on Frontier of Computer Science and Technology (FCST 2007)*, 90–97.

140. (2024). paragonie/ciphersweet. <https://github.com/paragonie/ciphersweet>, accessed: 04/19/2024.

141. (2024). cossacklabs/acra. <https://github.com/cossacklabs/acra>, accessed: 04/19/2024.

142. tpc-h_v3.0.0.pdf. <https://tpc.org/TPC_Documents_Current_Versions/pdf/tpc-h_v3.0.0.pdf>, accessed: 04/19/2024.

PHỤ LỤC 1. CÁC ĐỊNH NGHĨA BẢO MẬT CHO LƯỢC ĐỒ SSE

PL1.1. Bộ tạo giả ngẫu nhiên và hàm giả ngẫu nhiên

PL1.1.1. Bộ tạo giả ngẫu nhiên (PRG)

Theo Song và cộng sự [100], [111], [33], ta có các nhận xét và định nghĩa như sau:

- *Chức năng:* PRG lấy một chuỗi giá trị ngắn ngẫu nhiên làm đầu vào và kéo dài nó thành một chuỗi bit giả ngẫu nhiên dài hơn.

- *Bảo mật:* PRG được thiết kế để tạo ra đầu ra không thể phân biệt được với một chuỗi bit thực sự ngẫu nhiên. Điều này có nghĩa là kẻ tấn công không thể phân biệt hiệu quả giữa đầu ra của PRG và một chuỗi bit thực sự ngẫu nhiên.

- *Ứng dụng:* PRG được sử dụng để tạo ra các số ngẫu nhiên, khóa cho các thuật toán mật mã khác và để xây dựng các nguyên hàm mật mã khác như PRF.

Định nghĩa PL1.1. $G: \mathcal{K}_G \to S$ là một bộ tạo giả ngẫu nhiên đảm bảo (t, e) nếu mọi thuật toán A của kẻ gian với thời gian chạy tối đa là t có lợi thế $Adv\ A\ <\ e$. Lợi thế của một kẻ gian được định nghĩa là $Adv\ A = |\Pr\left[A\left(G\left(U_{\mathcal{K}_G}\right)\right) = 1\right] - \Pr[A(U_S) = 1]\ |$, với $U_{\mathcal{K}_G}$, U_S là các biến ngẫu nhiên phân bố đều trên $\mathcal{K}_G, S$.

PL1.1.2. Hàm giả ngẫu nhiên (PRF)

Theo Song và cộng sự [100], [111], [33], ta có các nhận xét và định nghĩa như sau:

- *Chức năng:* PRF lấy một đầu vào (khóa) và tạo ra một đầu ra (giá trị).

- *Bảo mật:* PRF được thiết kế để không thể phân biệt được với một hàm thực sự ngẫu nhiên. Điều này có nghĩa là với một PRF, kẻ tấn công không thể phân biệt hiệu quả giữa đầu ra của nó và đầu ra của một hàm thực sự ngẫu nhiên, ngay cả khi chúng có quyền truy cập vào nhiều cặp đầu vào-đầu ra.

- *Ứng dụng:* PRF được sử dụng rộng rãi trong nhiều giao thức mật mã khác nhau, bao gồm mã xác thực tin nhắn (MAC), mã khối và hàm băm an toàn.

Định nghĩa PL1.2. $F: K_F \times \mathrm{X} \to Y$ là một hàm giả ngẫu nhiên đảm bảo (t, q, e) nếu mọi thuật toán tiên tri A của kẻ gian thực hiện tối đa q truy vấn tiên tri và với thời gian chạy tối đa t có lợi thế $Adv\ A\ <\ e$. Lợi thế được định nghĩa là $Adv\ = |\Pr[A^{F_k} = 1] - \Pr[A^R = 1]\ |$, trong đó R đại diện cho một hàm ngẫu nhiên được chọn đồng nhất từ tập hợp tất cả các ánh xạ X đến Y với xác suất được lấy theo

sự lựa chọn của k và R.

PL1.2. Định nghĩa bảo mật lược đồ SSE

Dựa vào cấu trúc lược đồ SE trình bày trong **mục 1.2.2** và lược đồ SSE trình bày trong **mục 1.5.1**, tham khảo nghiên cứu [111], [33], ta có các nhận xét và định nghĩa như sau về đánh giá bảo mật đối với lược đồ SSE:

Định nghĩa PL1.3. Cho lược đồ SSE = (KeyGen, Enc, Trapdoor, Search, Dec), A là hiểu là các thuật toán/yêu cầu xác suất ngẫu nhiên của kẻ gian mà có trạng thái thực hiện với thời gian đa thức (PPT), S là một mô hình mô phỏng của lược đồ SSE với trạng thái PPT, L_1 và L_2 là các hàm rò rỉ trong kịch bản trò chơi bảo mật lý tưởng (sử dụng mô hình mô phỏng), $s \in \boldsymbol{N}$ biểu thị tham số bảo mật. Các trò chơi $Real_A(s)$ và $Ideal_{A,S}(s)$ được định nghĩa như sau:

- $Real_A(s)$: DO (người thách thức) thực thi hàm $KeyGen(1^s)$ để sinh khóa K. A yêu cầu tài liệu D và nhận được $(I, C) \leftarrow Enc_K(D)$ từ DO. Sau đó kẻ gian tiếp tục thực hiện một số lượng đa thức các truy vấn thích ứng Q và với mỗi lần truy vấn theo từ khóa w, sẽ nhận được từ DO giá trị trả về bởi cửa sập $TD \leftarrow Trapdoor_K(w)$. Cuối cùng A tổng hợp và trả về một bit b như là kết quả của trò chơi $Real_A(s)$.

- $Ideal_{A,S}(s)$: A yêu cầu tài liệu D, S tạo và gửi cặp giá trị (I, C) tới A, căn cứ vào đó A khai thác rò rỉ để xác định $L_1(D)$. Sau đó kẻ gian tiếp tục thực hiện một số lượng đa thức các truy vấn thích ứng Q và với mỗi lần truy vấn theo từ khóa w, S nhận được $L_2(D, w)$ và trả về giá trị cửa sập TD thích hợp. Cuối cùng A tổng hợp và trả về một bit b như là kết quả của trò chơi $Ideal_{A,S}(s)$.

- SSE đạt (L_1, L_2) an toàn về mặt ngữ nghĩa trước các cuộc tấn công thích ứng nếu đối với tất cả các thuật toán A (PPT) của kẻ gian và tham số s, tồn tại một mô hình mô phỏng S sao cho: $|\Pr[Real_A(s) = 1] - \Pr[Ideal_{A,S}(s) = 1] < e|$.

Từ các Định nghĩa 1.1, 1.2, 1.3 và độ an toàn bảo mật đã được chứng minh của của các thuật toán mã hóa như AES, DES, Blowfish [37], [13], [98], các nghiên cứu [111], [51], [107], [33], [124], [21], [27] đều kết luận rò rỉ L_1 trong các lược đồ SSE dựa trên chỉ mục không tiết lộ gì ngoài các thông tin hiển nhiên như kích thước của bản mã và chỉ mục. Trong khi đó rò rỉ L_2 có thể tiết lộ các thông tin về mẫu truy vấn và mẫu truy cập.

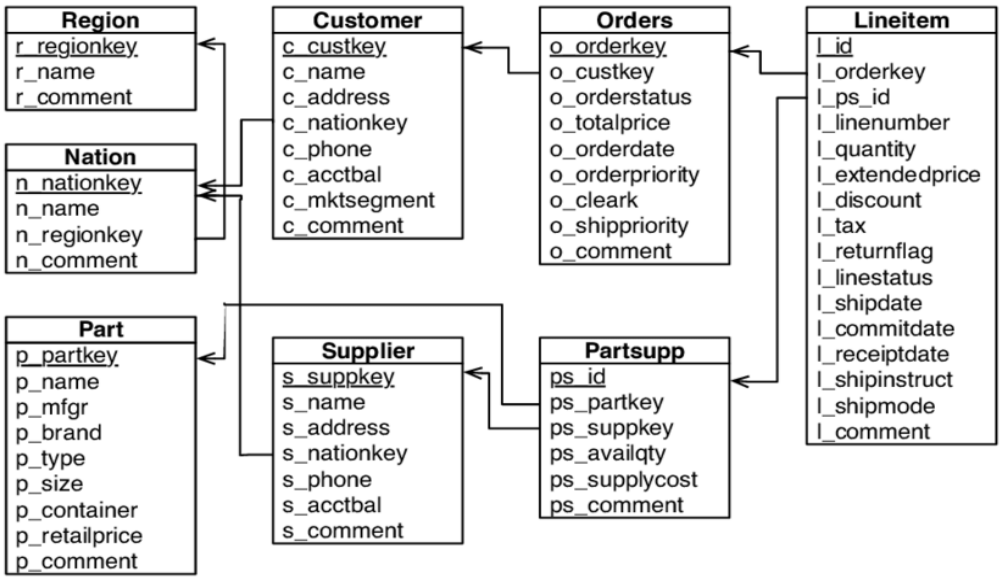
PHỤ LỤC 2. MỘT SỐ KIẾN THỨC LIÊN QUAN SỬ DỤNG TRONG LUẬN ÁN

PL2.1. Cơ sở dữ liệu quan hệ TPC-H

TPC [142] là một tổ chức phi lợi nhuận, cung cấp các phương pháp, công cụ, dữ liệu được kiểm định để đo đạc hiệu năng và chấm điểm cho khả năng xử lý dữ liệu của các hệ thống máy tính đối với nhiều lĩnh vực như: Xử lý giao dịch trực tuyến, Hỗ trợ ra quyết định, Ảo hóa, Dữ liệu lớn, Trí tuệ nhân tạo, IoT, …

TPC-H Benchmark [142] là một giải pháp bao gồm CSDL quan hệ TPC-H [142] và bộ các câu lệnh tổng hợp đặc biệt chạy trên TPC-H để đo hiệu năng thực thi của các hệ thống phần cứng máy tính. Ngoài việc phối hợp TPC-H và các bộ lệnh tiêu chuẩn đã nêu để đo đạc và chấm điểm đối với các hệ thống hỗ trợ ra quyết định và hệ thống dữ liệu lớn thì người dùng có thể sử dụng riêng CSDL quan hệ TPC-H để phục vụ làm nguồn dữ liệu tin cậy khi thử nghiệm, đánh giá các giải pháp liên quan tới khai thác, xử lý dữ liệu nói chung và CSDL quan hệ mã hóa nói riêng.

CSDL quan hệ TPC-H gồm 8 quan hệ (bảng) với các trường và mối liên kết, ràng buộc được mô tả như trong **Hình 1.16**.



Hình PL2.1. Lược đồ thực thể quan hệ của CSDL TPC-H [142]

Các bước để tạo CSDL quan hệ TPC-H:

- Bước 1: Tải mã nguồn mới nhất của bộ TPC-H Tool tại trang chủ *ww.tpc.org*, mục *Downloads → Download Programs and Specifications*.

- Bước 2: Giải nén và biên dịch mã nguồn vừa tải (sử dụng *Visual Studio Community Edition* với Windows hoặc sử dụng *Compiler clang/clang++* với

MacOS) để tạo ra tệp thực thi *dbgen.exe*.

- Bước 3: Tại thư mục chứa tệp *dbgen.exe*, mở cửa sổ dòng lệnh (ví dụ PowerShell trong Windows), gõ lệnh để sinh 8 tệp dữ liệu (*.tbl) tương ứng với 8 quan hệ ở **Hình PL2.1**. Cú pháp lệnh như sau:

```
dbgen [-{vf}][-T {pcsoPSOL}]
        [-s <scale>][-C <procs>][-S <step>]
```

Để sinh CSDL TPC-H với quy mô *nGB* dữ liệu, tham số *- s n* sẽ được sử dụng. Số bản ghi dữ liệu trong một số bảng sẽ biến động khi *n* thay đổi như **Bảng PL2.1**.

Bảng PL2.1. Số lượng bản ghi các bảng của CSDL TPC-H tương ứng với tham số *-s n* trong lệnh *dbgen*

	Số bản ghi trong bảng					
n-Scale (GB)	**lineitem**	**orders**	**partsupp**	**part**	**customer**	**supplier**
1 (mặc định)	6M	1.5M	0.8M	0.2M	0.15M	0.01M
10	60M	15M	8M	2M	1.5M	0.1M
20	120M	30M	16M	4M	3M	0.2M
50	300M	75M	40M	10M	7.5M	0.5M
100	600M	150M	80M	20M	15M	1M
200	1.2B	300M	160M	40M	30M	2M
1000	6B	1.5B	800M	200M	150M	10M

Ví dụ câu lệnh sinh ra CSDL TPC-H có dung lượng 1GB:

```
dbgen -vf -s 1
```

Lệnh *dbgen* cũng hỗ trợ trường hợp chỉ muốn sinh dữ liệu cho một số bảng được chỉ định cụ thể, khi đó cần dùng thêm tham số *-T {pcsoPSOL}*. Ví dụ chỉ sinh dữ liệu cho bảng *lineitem*, sẽ bổ sung tham số *-T L*:

```
dbgen -vf -s 1 -T L
```

Sau khi thực hiện lệnh *dbgen* thành công, các tệp dữ liệu tương ứng *lineitem.tbl, orders.tbl, partsupp.tbl, part.tbl, customer.tbl, supplier.tbl, nation.tbl, region.tbl* sẽ được sinh ra trong thư mục chứa tệp thực thi *dbgen.exe*. Nội dung dữ liệu trong các tệp hoàn toàn ngẫu nhiên, phù hợp để làm nguồn thử nghiệm cho bài toán nâng cao hiệu quả truy vấn trên dữ liệu mã mà luận án thực hiện.

Bước 4. Sử dụng hệ quản trị CSDL (ví dụ MS SQL Server) để tạo CSDL TPC-H và 8 bảng theo cấu trúc trong **Hình 1.16**.

Bước 5. Tải dữ liệu từ các tệp *lineitem.tbl, orders.tbl, partsupp.tbl, part.tbl, customer.tbl, supplier.tbl, nation.tbl, region.tbl* vào các bảng tương ứng qua lệnh *BULK INSERT* như sau:

```
dbgen -vf -s 1 -T L
BULK INSERT part FROM '~\dbgen\Debug\part.tbl'
BULK INSERT customer FROM '~\dbgen\Debug\customer.tbl'
BULK INSERT orders FROM '~\dbgen\Debug\orders.tbl'
BULK INSERT partsupp FROM '~\dbgen\Debug\partsupp.tbl'
BULK INSERT supplier FROM '~\dbgen\Debug\supplier.tbl'
BULK INSERT lineitem FROM '~\dbgen\Debug\lineitem.tbl'
BULK INSERT nation FROM '~\dbgen\Debug\nation.tbl'
BULK INSERT region FROM '~\dbgen\Debug\region.tbl'
```

PL2.2. Bộ lọc Bloom

Bộ lọc Bloom [14] là một cấu trúc dữ liệu đơn giản cho phép kiểm tra sự tồn tại của một phần tử trong một tập hợp một cách nhanh chóng với xác suất dương tính giả nhỏ. Bộ lọc Bloom mang đến hiệu quả về không gian lưu trữ, sử dụng mảng m bit để đại diện cho sự xuất hiện của các phần tử s trong tập $S = \{s_1, s_2, \ldots, s_n\}$. Trước tiên các bit trong mảng sẽ được đặt về 0, sử dụng k hàm băm ngẫu nhiên độc lập $h_1, h_2, \ldots, h_k$ $(\alpha \leftarrow h_i(Key_i, s) \mid s \in S,\ i = \{1, .., k\},\ 0 \leq \alpha \leq m-1)$ để xác định tập gồm k vị trí. Khi đó với mỗi phần từ $s \in S$, các bit tại vị trí $h_i(Key_i, s)$ trong mảng sẽ được đặt là 1. Để kiểm tra phần tử s có thuộc S hay không chỉ cần kiểm tra tất cả các vị trí $h_i(Key_i, s)$ trong mảng có bằng 1 hay không, nếu không thỏa mãn thì kết luận chắc chắn s không phải là thành viên của S, ngược lại s thuộc S với một xác suất dương tính giả có thể tối ưu được.

Theo Mullin [85], xác suất dương tính giả với một phần tử không thuộc S nhưng vẫn thỏa mãn giá trị bit tại các vị trí $h_i(s)$ bằng 1 được xác định qua hàm f^{FP} như sau:

$$f^{FP} = (1 - (1 - 1/m)^{kn})^k \approx \left(1 - e^{-kn/m}\right)^k \qquad \text{(PL2.1)}$$

Quan sát công thức tính f^{FP} ở trên cho thấy nếu m lớn, n nhỏ thì giá trị của f^{FP} sẽ nhỏ. Nhưng thực tế ta sẽ không thể can thiệp nhiều được n, vì vậy cần xác định số lượng k hàm băm phù hợp để f^{FP} đạt giá trị nhỏ nhất. Theo Fan và cộng sự [39], giá trị f^{FP} sẽ đạt giá trị nhỏ nhất khi:

$$k = (\ln 2) * (m/n) \qquad \text{(PL2.2)}$$

Trong trường hợp này xác suất dương tính giả sẽ là:

$$f^{FP} = (1/2)^k = (0.6185)^{m/n} \qquad \text{(PL2.3)}$$