BỘ GIÁO DỤC VÀ ĐÀO TẠO BỘ QUỐC PHÒNG

HỌC VIỆN KỸ THUẬT QUÂN SỰ

HỒ KIM GIÀU

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 KỸ THUẬT

HÀ NỘI - 2021

BỘ GIÁO DỤC VÀ ĐÀO TẠO BỘ QUỐC PHÒNG

HỌC VIỆN KỸ THUẬT QUÂN SỰ

HỒ KIM GIÀU

NGHIÊN CỨU PHÁT TRIỂN GIẢI PHÁP XÁC THỰC AN TOÀN VÀ QUẢN LÝ KHOÁ CHO CƠ SỞ DỮ LIỆU THUÊ NGOÀI

Chuyên ngành: CƠ SỞ TOÁN HỌC CHO TIN HỌC
Mã số: 9 46 01 10

LUẬN ÁN TIẾN SĨ KỸ THUẬT

NGƯỜI HƯỚNG DẪN KHOA HỌC:
PGS.TS NGUYỄN HIẾU MINH

HÀ NỘI - 2021

LỜI CAM ĐOAN

Tôi xin cam đoan các kết quả trình bày trong luận án là công trình nghiên cứu của tôi dưới sự hướng dẫn của cán bộ hướng dẫn. Các số liệu, kết quả trình bày trong luận án là hoàn toàn trung thực và chưa được ai công bố trong bất kỳ công trình nào trước đây.

Hà Nội, ngày 28 tháng 9 năm 2021

Nghiên cứu sinh

Hồ Kim Giàu

LỜI CẢM ƠN

Trong quá trình học tập, nghiên cứu và thực hiện luận án, nghiên cứu sinh đã nhận được sự hướng dẫn, giúp đỡ tận tình, các ý kiến đóng góp quý báu của các Thầy, Cô, các nhà khoa học; Sự động viên, chia sẻ của bạn bè, đồng nghiệp và gia đình.

Nghiên cứu sinh xin bày tỏ lòng biết ơn sâu sắc đến Thầy giáo hướng dẫn **PGS.TS Nguyễn Hiếu Minh**. Thầy đã nhiệt tình, tận tâm định hướng, hướng dẫn, giúp đỡ nghiên cứu sinh trong suốt quá trình nghiên cứu và hoàn thành luận án này.

Nghiên cứu sinh tỏ lòng biết ơn đến Thầy **TS Lều Đức Tân**, Thầy đã chỉ bảo cho nghiên cứu sinh những bài học, những kiến thức quý giá trong quá trình nghiên cứu. Nghiên cứu sinh trân trọng cám ơn quý Thầy, Cô giáo khoa Công nghệ Thông tin, Học viện Kỹ thuật Quân sự đã tận tình giảng dạy, giúp đỡ trong thời gian nghiên cứu sinh học tập, nghiên cứu tại đây.

Nghiên cứu sinh gửi lời cảm ơn đến trường Sĩ quan Thông tin, Binh chủng Thông tin Liên lạc, phòng Sau đại học, Bộ môn Công nghệ mạng, Học viện Kỹ thuật Quân sự đã giúp đỡ, tạo điều kiện cho nghiên cứu sinh được đi học tập, nghiên cứu, và hoàn thành luận án này.

Cuối cùng, nghiên cứu sinh gửi lời cảm ơn chân thành tới gia đình, bạn bè và đồng nghiệp, những người đã luôn ủng hộ, tạo niềm tin, động viên, chia sẻ những khó khăn với nghiên cứu sinh trong suốt thời gian vừa qua.

Hà Nội, tháng 9 năm 2021

Hồ Kim Giàu

MỤC LỤC

DANH MỤC TỪ VIẾT TẮT

Viết tắt	Nghĩa tiếng Anh	Nghĩa tiếng Việt
ADS	Authenticated Data Structure	Cấu trúc dữ liệu xác thực
Adv	Adversary	Kẻ tấn công
AES	Advanced Encryption Standard	Tiêu chuẩn mã hóa tiên tiến
API	Application programming interface	Giao diện lập trình
AQE	Adjustable Query-based Encryption	Mã hóa dựa trên truy vấn có thể điều chỉnh
CRT	Chinese Remainder Theorem	Định lý số dư Trung Hoa
CSDL	Database	Cơ sở dữ liệu quan hệ
DBMS	Database Management Systems	Hệ quản trị CSDL
DES	Data Encryption Standard	Tiêu chuẩn mã hóa dữ liệu
DET	Deterministic	Mã hóa tất định
DLP	Discrete Logarithm Problem	Bài toán logarit rời rạc
DO	Data Owner	Chủ sở hữu dữ liệu
DSA	Digital Signature Algorithm	Thuật toán chữ ký số
DSP	Database Service Provider	Nhà cung cấp dịch vụ CSDL
DSS	Digital Signature Standard	Chuẩn chữ ký số
FPE	Format Preserving Encryption	Mã hóa bảo toàn định dạng
FQE	Fuzzy Query Encryption	Mã hóa truy vấn mờ
FUSE	Filesystem in Userspace	Hệ thống tập tin trong không gian người dùng
GUI	Graphical User Interface	Giao diện đồ hoạ người dùng
HDFS	Hadoop Distributed File System	Hệ thống tập tin của Hadoop
HOM	Homomorphic Encryption	Mã hóa đồng cấu
IaaS	Infrastructure as a Service	Hạ tầng như một dịch vụ

IDS	Intrusion Detection System	Hệ thống phát hiện xâm nhập
IFP	Integer Factorization Problem	Bài toán phân tích số
IV	Initialization vector	Vector khởi tạo
JDBC	Java Database Connectivity	Chuẩn kết nối dữ liệu của java
KMT	Key Management Tree	Cây quản lý khóa
MHT	Merkle Hash Tree	Cây hàm băm Merkle
ODBS	Outsourced Database Service	Dịch vụ CSDL thuê ngoài
OPE	Order-Preserving Encryption	Mã hóa bảo toàn thứ tự
PaaS	Platform as a Service	Nền tảng như một dịch vụ
RPC	Remote Procedure Call	Gọi thủ tục từ xa
RST	Resource Set Tree	Cây tập tài nguyên
SaaS	Software as a Service	Phần mềm như một dịch vụ
SQL	Structured Query Language	Ngôn ngữ truy vấn có cấu trúc
SQLAE	SQL-aware Encryption	Mã hóa nhận thức SQL
SSE	Searchable Strong Encryption	Mã hóa mạnh có thể tìm kiếm
TLS	Transport Layer Security	Bảo mật tầng giao vận
UDF	User Defined Function	Hàm do người dùng định nghĩa
VFS	Virtual file systems	Hệ thống tệp tin ảo
VO	Verification Object	Đối tượng xác thực

DANH MỤC KÝ HIỆU

Kí hiệu	Ý nghĩa
$\{0,1\}^\infty$	Chuỗi bit có độ dài bất kỳ
$\|\|$	Phép toán nối chuỗi
$H()$	Hàm băm mật mã (Hash function)
$E_k(m)$	Hàm mã hoá (ví dụ: DES, AES...) dữ liệu m với khoá k
$D_k(c)$	Hàm giải mã dữ liệu c với khoá k (Hàm ngược của E_k)
$readKMT()$	Hàm đọc cây khoá KMT
$gcd(a,b)$	Ước số chung lớn nhất của a và b
$S(m)$	Hàm tạo chữ ký cho dữ liệu m
$V(\sigma_i)$	Hàm xác thực lô cho nhiều chữ ký σ_i
$len(a)$	Độ dài của a
$t_{join}(D_i)$	Chi phí trung bình (TB) ghép các bảng CSDL $D_1, D_2, ...$
$t_{insert}(T)$	Chi phí trung bình lưu trữ bảng T lên CSDL
t_{enc}	Chi phí TB thực hiện mã hoá
t_{dec}	Chi phí TB thực hiện giải mã
t_H	Chi phí TB cho một phép tính hàm H: $\{0,1\}^\infty \to \{0,1\}^h$.
t_{gen}	Chi phí TB thực hiện tạo khóa của hàm KeyGen()
t_{read}	Chi phí TB thực hiện đọc cây KMT
$t_M(N)$	Chi phí TB cho phép nhân hai số N-bít
$t_{Red}(N)$	Chi phí TB cho phép rút gọn một số 2N-bít theo modulo N-bít
$t_m(N)$	Chi phí TB cho phép nhân rút gọn theo modulo n với len(n) = N
$t_{exp}(N,L)$	Chi phí TB cho phép tính a^e mod n với len(e) = N và len(n) = L
$t_{gcd}(N)$	Chi phí TB cho phép tính gcd(a,b) với len(a), len(b) $\leq$ N
$t_{Jacobi}(N)$	Chi phí TB cho phép tính $(\frac{a}{n})$ với len(n) = N

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG

DANH MỤC THUẬT TOÁN

MỞ ĐẦU

1. Tính cấp thiết của đề tài luận án

Để quản lý hệ thống thông tin, các tổ chức, cá nhân thường lưu trữ dữ liệu dưới dạng cơ sở dữ liệu quan hệ (CSDL). Điều này giúp người chủ sở hữu dữ liệu (Data Owner - DO) dễ dàng truy xuất dữ liệu và chia sẻ cho nhiều người dùng. Trước kia, có hai hình thức lưu trữ CSDL là lưu trữ trong nội bộ tổ chức (In-house database) và lưu trữ trực tuyến (Online database).

Hình thức lưu trữ nội bộ: DO quản lý CSDL trên máy chủ của mình, không chia sẻ qua mạng Internet. Như vậy, DO phải có hệ thống máy chủ gồm: máy tính, hệ điều hành, hệ quản trị CSDL và nhân viên vận hành hệ thống. Khi nhu cầu lưu trữ và xử lý dữ liệu tăng đòi hỏi DO phải tốn chi phí cho nâng cấp phần cứng, cập nhập bản quyền phần mềm, phát triển đội ngũ nhân viên...

Hình thức lưu trữ trực tuyến: DO đặt máy chủ của mình ở môi trường mạng và toàn quyền quản lý máy chủ cũng như các dịch vụ của nó. Như vậy, ngoài chi phí như lưu trữ nội bộ, DO tốn thêm chi phí thuê đường truyền, đồng thời bảo vệ máy chủ khỏi các nguy cơ tấn công từ Internet.

Ngày nay, khi điện toán đám mây phát triển mạnh mẽ thì các tổ chức, cá nhân có thêm một phương án tiếp cận mới trong việc quản lý, khai thác CSDL, đó là dịch vụ CSDL thuê ngoài (Outsourced Database Service – ODBS). Với ODBS, các tổ chức và cá nhân sẽ được một nhà cung cấp dịch vụ (Database Service Provider – DSP) quản lý và duy trì hoạt động CSDL của mình. DO khai thác CSDL thông qua các phương thức do DSP cung cấp. Mô hình ODBS khác với hình thức lưu trữ trực tuyến ở chỗ DO chỉ sử dụng và trả tiền cho dịch vụ CSDL mà không quan tâm đến hệ thống máy chủ, đường truyền và nhân viên quản lý hệ thống.

Dữ liệu là tài sản quan trọng của DO. Nếu các thông tin của cá nhân như thẻ tín dụng, tài khoản ngân hàng... bị kẻ xấu đánh cắp và sử dụng trái phép thì sẽ gây ra thiệt hại nghiêm trọng. Hơn nữa, không ai có thể biết được hậu quả như thế nào nếu thông tin về an ninh quốc phòng, bí mật quốc gia bị tấn công. Mặc dù các hệ thống máy tính, hệ điều hành, phần mềm bảo mật, phần mềm ứng dụng... luôn luôn được cập nhập, vá lỗi, và các hệ thống bảo mật bằng phần cứng được triển khai nhưng chúng ta có thể thấy việc dữ liệu bị tấn công vẫn luôn diễn ra. Tính đến tháng 03/2019, Databreaches [74], Information Is Beautiful [83] đã thống kê được các tổ chức, doanh nghiệp bị tấn công đánh cắp dữ liệu từ CSDL nói chung và những tấn công này có thể xảy ra đối với ODBS nói riêng như bảng 1. Trong đó, nguyên nhân bảo mật kém là do các chính sách bảo mật chưa được thực hiện đúng để cho người dùng không có quyền vẫn có thể truy cập được dữ liệu. Bị tấn công là nguyên nhân xảy ra từ môi trường Internet. Kẻ tấn công khi truy xuất được vào máy chủ dịch vụ thì sẽ có được dữ liệu nếu đó là dữ liệu rõ. Một nguyên nhân nữa có thể gây thiệt hại về dữ liệu đó là mất thiết bị (máy tính, ổ cứng, token...). Khi mất thiết bị, kẻ tấn công không chỉ có được dữ liệu mà còn có thể có được các khóa mã, các thông tin phụ trợ cho quá trình xử lý trên dữ liệu mã, các hàm mật mã... Nguyên nhân khó kiểm soát nhất đó là tấn công từ bên trong của DSP. Nhân viên quản trị hệ thống của DSP có thể trực tiếp can thiệp vào hệ thống máy chủ. Do đó, việc dữ liệu của người dùng có thể bị nhân viên của DSP lấy cắp mà ngay cả DO cũng không biết.

Để bảo vệ "tài sản" quý giá của mình, DO phải có những giải pháp bảo đảm an toàn CSDL ngay từ khi triển khai hệ thống. Giải pháp cơ bản nhất là DO mã hoá dữ liệu trước khi lưu trữ lên ODBS để bảo vệ dữ liệu của mình. Khi dữ liệu bị mã hoá, kẻ tấn công có thể lấy cắp dữ liệu nhưng không sử dụng được do không biết nội dung của dữ liệu đó, nếu họ cố gắng trong việc giải mã thông tin thì cũng tốn nhiều thời gian, đôi khi là không thực hiện được. Nhưng khi đó, DO phải trả giá về chi phí về thời gian, tài nguyên hệ

Bảng 1: Tình trạng rò rỉ dữ liệu [83]

Thời gian	**Tổ chức/ Lĩnh vực**	**Số bản ghi bị rò rỉ**	**Mô tả**	**Nguyên nhân**
12/2018	Google+/ web	52.500.000	Một lỗ hổng mới có thể đã tiết lộ thông tin cá nhân của người dùng, ngay cả khi hồ sơ của họ được đặt ở chế độ riêng tư.	Bảo mật kém
03/2018	Facebook/ web	50.000.000	Cambridge Analytica (công ty tư vấn chính trị của Anh) đã thu thập 50 triệu hồ sơ người dùng facebook vào đầu năm 2014 để xây dựng một hệ thống tác động lên cử tri Mỹ và nhằm mục tiêu quảng bá chính trị.	Bị tấn công
09/2018	British Airways/ giao thông	380.000	Chi tiết cá nhân và tài chính của khách hàng đặt vé trong khoảng thời gian từ ngày 21 tháng 8 đến ngày 05 tháng 9 đã bị tấn công.	Bị tấn công
03/2017	Văn phòng đăng ký và bầu cử Hồng Kông/ Chính phủ	3.700.000	Thông tin cá nhân của 3,7 triệu cử tri thành phố có thể bị xâm phạm sau khi Văn phòng đăng ký và bầu cử báo cáo hai máy tính xách tay bị mất tích tại địa điểm dự phòng cho cuộc bầu cử giám đốc điều hành.	Mất thiết bị
01/2014	Phòng tín dụng Hàn Quốc/ tín dụng	20.000.000	Một nhân viên của công ty xếp hạng tín dụng cá nhân của Cục Tín dụng Hàn Quốc (KCB) đã bị bắt và bị buộc tội ăn cắp dữ liệu từ khách hàng của ba công ty thẻ tín dụng khi làm việc cho họ với vai trò là nhà tư vấn tạm thời.	Tấn công bên trong

thống cho truy xuất, tính toán trên dữ liệu mã. Các nhà khoa học đã đề xuất việc mã hoá dữ liệu lưu trữ và truy xuất, tính toán trên dữ liệu mã, tuy nhiên vấn đề giải quyết về mặt thời gian đôi khi vẫn chưa xem xét, hoặc có những nghiên cứu tập trung vào giải quyết vấn đề tính bí mật dữ liệu nhưng không đồng thời giải quyết về mặt bảo vệ các khoá mã, hoặc chưa có giải pháp thay thế khoá mã khi cần thiết. Với bài toán xác thực dữ liệu trên CSDL mã khi truy vấn ngẫu nhiên từ nhiều bảng hoặc CSDL động thì chưa giải quyết tốt

hoặc làm tăng số lượng tính toán của các đối tượng phụ trợ. Các nghiên cứu thường tách riêng việc xác thực dữ liệu với việc giải mã dữ liệu, do đó làm tăng thời gian xử lý của hệ thống...

Từ những nhận định như trên, việc nghiên cứu và đề xuất một số giải pháp nhằm xác thực an toàn, quản lý và thay đổi khoá mã cho ODBS là mang tính cấp thiết, có ý nghĩa khoa học và phù hợp với xu thế cách mạng công nghệ 4.0. Kết quả nghiên cứu của luận án sẽ là cơ sở khoa học và thực tiễn cho việc nâng cao tính an toàn của CSDL nói chung và ODBS nói riêng.

2. Đối tượng, phạm vi nghiên cứu

- Đối tượng nghiên cứu:
  - Mô hình ODBS, các hệ quản trị CSDL như: MySQL, SQL Server...
  - Các thuật toán mã hóa, chữ ký số, xác thực lô.
  - Mô hình lưu trữ khoá-giá trị, bảo mật hệ thống tệp tin trong không gian người dùng.
  - Các mô hình tính toán song song trên CPU, MapReduce...
- Phạm vi nghiên cứu:

  Luận án giới hạn nghiên cứu ODBS phục vụ nhu cầu của tổ chức, cá nhân. Trong đó, một DO thuê dịch vụ của một DSP để lưu trữ CSDL, chia sẻ dữ liệu cho nhiều người dùng. DO không được phép can thiệp vào máy chủ của DSP để lắp đặt các thiết bị, cài đặt các dịch vụ, chương trình hoặc phần mềm nào khác. Trong quá trình nghiên cứu, luận án sử dụng thêm một máy chủ trung gian để lưu trữ các dữ liệu hỗ trợ, xử lý các thuật toán như mã hóa/giải mã, các hàm do người dùng định nghĩa. Máy chủ trung gian do DO quản lý và toàn quyền can thiệp. Việc bảo mật máy chủ trung gian do DO thiết lập bằng các giải pháp phần cứng (router, firewall...), phần mềm (Antivirus, IDS...). Dữ liệu từ DSP truyền về máy chủ trung gian là dữ liệu mã, dữ liệu từ máy chủ trung gian đến

người dùng là dữ liệu rõ và dùng giao thức TLS để bảo mật. Vì vậy, máy chủ trung gian được giả định là an toàn và nằm ngoài phạm vi nghiên cứu của luận án.

3. Mục tiêu của luận án

Mục tiêu của luận án là nghiên cứu, phát triển các giải pháp nhằm xác thực an toàn và quản lý, thay đổi khoá cho ODBS. Từ đó, đề xuất các mô hình, thuật toán như: thuật toán xử lý song song để giảm thời gian truy vấn trên dữ liệu mã, các thuật toán xác thực lô và mô hình ứng dụng xác thực lô vào xác thực CSDL mã hóa, mô hình quản lý khoá và quyền truy cập, thuật toán mã hóa hệ thống tập tin trong hệ điều hành và thuật toán thay đổi khóa cho CSDL mã hóa của ODBS, làm nền tảng khoa học và có khả năng triển khai vào ứng dụng thực tế.

4. Phương pháp nghiên cứu

- Về lý thuyết: Dùng phương pháp khảo sát, phân tích, đánh giá, tổng hợp các phương pháp đã đề xuất về đảm bảo an toàn ODBS, từ đó đưa ra những định hướng nghiên cứu, phát triển. Bên cạnh đó, luận án nghiên cứu, vận dụng các kiến thức liên quan đến xử lý dữ liệu lớn, lý thuyết về số học, lý thuyết hệ điều hành để đề xuất mô hình, thuật toán; Nghiên cứu các mô hình tính toán song song trên CPU, MapReduce...
- Về thực nghiệm:
  - Luận án sử dụng các mô hình toán học để chứng minh tính đúng đắn của các phương pháp đề xuất. Luận án sử dụng các ngôn ngữ lập trình (Java, Python...) để cài đặt các thuật toán và sử dụng bộ dữ liệu mẫu TPC-H làm dữ liệu thử nghiệm đánh giá kết quả đề xuất.
  - Luận án sử dụng hệ thống mô phỏng ODBS với một máy chủ lưu trữ CSDL (tương ứng với DSP), một máy chủ trung gian để xử lý

(DO quản lý) và các máy người dùng. Việc mô phỏng vẫn đảm bảo đúng theo mô hình ODBS đồng thời hạn chế được sự ảnh hưởng của đường truyền mạng đến khả năng xử lý của các giải pháp đề xuất so với việc dùng dịch vụ ODBS trong thực tế.

5. Nội dung nghiên cứu của luận án

- Đánh giá tổng quát các nghiên cứu trước đây và những vấn đề tồn tại hiện nay trong bảo đảm an toàn ODBS. Từ đó đưa ra những nhận định nghiên cứu, đề xuất những vấn đề cần giải quyết đối với bảo đảm an toàn ODBS có khả năng ứng dụng vào thực tế.
- Nghiên cứu giải pháp giảm thời gian truy vấn trên dữ liệu mã.
- Tìm hiểu về các thuật toán mật mã, nghiên cứu và đề xuất các thuật toán chữ ký số, xác thực lô dựa trên hai bài toán khó và ứng dụng vào xác thực dữ liệu mã cho ODBS.
- Nghiên cứu giải pháp mã hoá hệ thống tệp tin trong hệ điều hành để bảo vệ nội dung tập tin.
- Nghiên cứu mô hình quản lý và giải pháp thay đổi khoá mã cho CSDL mã của ODBS.
- Đề xuất các mô hình, thuật toán để giải quyết các bài toán. Cài đặt, thực nghiệm nhằm kiểm chứng tính khả thi của các mô hình, thuật toán đã đề xuất.

6. Ý nghĩa khoa học và thực tiễn

Việc nghiên cứu, đề xuất và phát triển các giải pháp bảo đảm an toàn cho ODBS có khả năng triển khai trong thực tế luôn là mục tiêu của các nhà khoa học trên thế giới. Với các nghiên cứu được trình bày, luận án đã đóng góp về ý nghĩa khoa học và thực tiễn như sau:

- Kỹ thuật xử lý song song trên dữ liệu mã là nền tảng khoa học cho việc nghiên cứu giảm thời gian truy vấn trên dữ liệu mã. Khi mã hóa để đảm bảo tính bí mật và khi xác thực dữ liệu mã hóa thì thời gian truy vấn sẽ tăng đáng kể so với truy vấn trên dữ liệu rõ. Do đó, khi giảm thời gian truy vấn trên dữ liệu mã sẽ làm cho phương pháp mã hóa CSDL và xác thực dữ liệu mã có tính ứng dụng thực tế cao hơn, đảm bảo được tính an toàn cho ODBS.

- Thuật toán xác thực ODBS có ý nghĩa kiểm tra tính đúng đắn của dữ liệu khi truy vấn trên CSDL mã. Các thuật toán xác thực lô được đề xuất dựa trên hai bài toán khó còn có ý nghĩa về mặt nghiên cứu các thuật toán chữ ký số mới, đáp ứng yêu cầu bảo mật và có thể ứng dụng vào các mô hình xác thực khác nhau.

- Thuật toán mã hoá tệp tin trong không gian người dùng làm cơ sở khoa học cho việc phát triển hệ điều hành an toàn. Với tệp được mã hoá, cho dù người dùng bị mất thiết bị thì kẻ tấn công (Adversary - Adv) cũng không thể đọc được do không có khoá giải mã nội dung tệp tin, như vậy bảo vệ an toàn cho dữ liệu tệp tin người dùng.

- Thuật toán đổi khóa làm cơ sở khoa học để nghiên cứu và phát triển các kỹ thuật đổi khóa cho dữ liệu mã trên ODBS, có khả năng ứng dụng vào thực tế. Mô hình và thuật toán đổi khóa cho CSDL mã có ý nghĩa thực tiễn trong trường hợp nghi ngờ khóa mã bị lộ lọt hoặc nên thay đổi định kỳ để bảo vệ tính bí mật của dữ liệu.

7. Bố cục của luận án

Luận án gồm mở đầu, 3 chương, kết luận và hướng phát triển, các công trình công bố và tài liệu tham khảo. Nội dung 3 chương cụ thể như sau:

1. Chương 1: NHỮNG VẤN ĐỀ CHUNG VỀ AN TOÀN ODBS

   Trong chương này, luận án trình bày các vấn đề về bảo đảm an toàn

cho ODBS, tình hình nghiên cứu trong và ngoài nước. Từ đó luận án đưa ra những nhận định về hướng nghiên cứu, các bài toán cần giải quyết. Chương này cũng trình bày những kiến thức cơ sở, làm nền tảng cho các nghiên cứu, đề xuất trong các chương 2, 3. Cuối cùng là kết luận chương.

2. Chương 2: GIẢM THỜI GIAN TRUY VẤN VÀ XÁC THỰC KHI TRUY VẤN CSDL MÃ TRÊN ODBS

   Trong chương này, luận án đề xuất thuật toán giảm thời gian thực hiện khi truy vấn CSDL mã trên ODBS. Luận án đề xuất hai thuật toán xác thực lô mới dựa trên hai bài toán khó và áp dụng một thuật toán xác thực lô vào xác thực dữ liệu mã khi truy vấn trên môi trường thuê ngoài. Phân tích, đánh giá các thuật toán đề xuất và kết luận chương.

3. Chương 3: QUẢN LÝ, THAY ĐỔI KHOÁ MÃ CỦA ODBS

   Trong chương này, luận án trình bày bài toán quản lý khoá mã của CSDL mã và cơ chế quản lý truy cập CSDL người dùng. Luận án đề xuất cây KMT (Key Management Tree) để quản lý khoá, phương pháp mã hoá tệp trong hệ điều hành và phương pháp đổi khoá mã ở mức cột của ODBS. Phân tích thử nghiệm, đánh giá phương pháp đề xuất và kết luận chương.

Chương 1
NHỮNG VẤN ĐỀ CHUNG VỀ AN TOÀN ODBS

Nội dung chương 1 giới thiệu tổng quan về các vấn đề an toàn cho ODBS, khảo sát các nghiên cứu trong và ngoài nước có liên quan. Từ đó, luận án đưa ra những nhận định về định hướng nghiên cứu, phát triển liên quan đến các bài toán xác thực, quản lý và thay đổi khoá mã cho ODBS. Chương 1 còn trình bày các cơ sở lý thuyết làm nền tảng cho các nghiên cứu được trình bày trong chương 2, 3. Cuối cùng là kết luận chương.

1.1. Tổng quan về an toàn ODBS

1.1.1. Giới thiệu về CSDL quan hệ

Dữ liệu là thành phần cốt lõi của các hệ thống thông tin. Một hệ thống thông tin cần có dữ liệu đầu vào, xử lý tính toán và đưa ra dữ liệu kết quả. Trong quá trình xử lý, hệ thống cần lưu trữ dữ liệu đầu vào, giá trị tạm, dữ liệu kết quả... để dễ dàng sử dụng và chia sẻ. Các hình thức tổ chức lưu trữ dữ liệu như: Cấu trúc dữ liệu, hướng đối tượng, CSDL... Trong đó, mô hình CSDL được xây dựng trên cơ sở toán học chặt chẽ nên nó đảm bảo được hầu hết các yêu cầu khi sử dụng thông tin từ người dùng.

Định nghĩa 1.1 *Cho các tập $A_1, A_2, ..., A_n$ $(n \geq 1)$, R là được gọi là quan hệ trên n tập này nếu có ít nhất một bộ (tuple) gồm n giá trị, trong đó giá trị đầu tiên được lấy từ A_1, giá trị thứ hai lấy từ A_2,... [19]*

Đặc điểm của quan hệ R là các giá trị trong cùng một tập (A_1 hoặc $A_2, ...$) phải cùng kiểu dữ liệu và các giá trị này không có mối quan hệ về thứ tự. Khi biểu diễn quan hệ R dưới dạng một bảng thì mỗi cột tương ứng với một thuộc tính, mỗi dòng là một bộ và tất cả các dòng có nội dung khác nhau.

CSDL quan hệ được E. F. Codd [18] giới thiệu lần đầu tiên vào năm 1970.

CSDL quan hệ là cách thức tổ chức dữ liệu dưới dạng quan hệ (bảng) và các phép toán, thao tác dữ liệu trên các quan hệ đó.

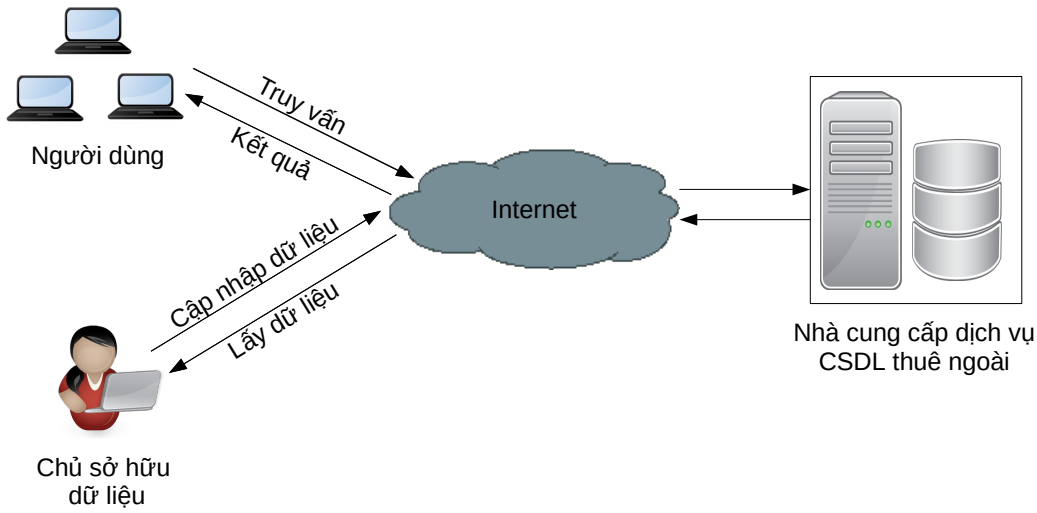
Để quản lý CSDL quan hệ trong máy tính thì ta dùng phần mềm quản lý gọi là hệ quản trị CSDL (Database Management Systems - DBMS). DBMS cho phép lập trình viên định nghĩa cấu trúc các bảng (tên trường, kiểu dữ liệu của trường...) để lưu trữ dữ liệu; tạo quan hệ giữa các bảng để đảm bảo sự toàn vẹn dữ liệu; thao tác thêm, sửa, xóa và trích xuất dữ liệu bằng các lệnh truy vấn có cấu trúc (Structured Query Language - SQL)

1.1.2. Giới thiệu ODBS

Điện toán đám mây (Cloud Computing) ra đời và phát triển đã giúp cho người dùng có được nhiều lợi ích khi sử dụng dịch vụ như: Không tốn chi phí đầu tư ban đầu về phần cứng, phần mềm, hạ tầng mạng,... thay đổi linh hoạt về tài nguyên hệ thống (CPU, RAM, ổ cứng...). Điện toán đám mây thường cung cấp cho người dùng các dịch vụ qua mạng Internet dưới dạng: Cơ sở hạ tầng như một dịch vụ (Infrastructure as a Service - IaaS), nền tảng như một dịch vụ (Platform as a Service - PaaS), phần mềm như một dịch vụ (Software as a Service - SaaS).

Khái niệm 1.1 *ODBS là một dịch vụ trên PaaS của điện toán đám mây [11]. Trong mô hình ODBS, DO thuê một nhà cung cấp dịch vụ quản lý và duy trì hoạt động CSDL của mình. DO chỉ khai thác và chia sẻ CSDL cho người dùng thông qua các phương thức do DSP cung cấp mà không có quyền truy cập vào các dịch vụ khác của máy chủ DSP.*

Với mô hình ODBS, DO chỉ sử dụng và trả tiền cho dịch vụ CSDL mà không quan tâm đến việc đầu tư và quản lý hệ thống máy chủ, đường truyền và nhân viên quản trị hệ thống. Do đó, DO sẽ giảm được chi phí trong việc đầu tư hạ tầng và nhân công để quản lý khi sử dụng CSDL. Việc lưu trữ, quản lý CSDL bởi DSP sẽ đảm bảo an toàn hơn với hệ thống phần cứng hiện đại, phần mềm cập nhập thường xuyên và đội ngũ nhân viên chuyên nghiệp.



Hình 1.1: Mô hình tổng quát của ODBS

Mô hình ODBS thường có ba đối tượng chính:

- Nhà cung cấp dịch vụ (Database Service Provider-DSP): DSP là tổ chức, doanh nghiệp cung cấp ODBS. Máy chủ DSP sẽ có nhiệm vụ lưu trữ, quản lý, bảo trì CSDL của DO. Các nhà cung cấp dịch vụ ODBS trên thế giới như: Amazon [5], Google [15], CMC [17]... Máy chủ của DSP có các DBMS như: MySQL, PostgreSQL, MariaDB, Oracle hoặc SQL Server...
- Chủ sở hữu dữ liệu (Data Owner-DO): Người tạo ra và toàn quyền quyết định dữ liệu. DO là người thuê dịch vụ của DSP để lưu trữ CSDL và chia sẻ dữ liệu cho người dùng.
- Người dùng (User): Người dùng đầu cuối hoặc ứng dụng truy cập đến dữ liệu của DO theo phân quyền của DO.

Nhận xét 1.1 *Mô hình ODBS khác với hình thức lưu trữ trực tuyến ở chỗ CSDL của DO được quản lý bởi bên thứ ba (DSP). Việc khai thác CSDL của hình thức ODBS có hai tương tác chủ yếu: Tương tác DO - DSP và tương tác người dùng - DSP. Tương tác giữa DO và DSP bao gồm các truy vấn liên quan đến quản lý CSDL như: Thêm, cập nhật, xóa dữ liệu, quản lý, phân*

quyền người dùng... Trong khi tương tác giữa người dùng và DSP bao gồm một số các truy vấn liên quan đến việc trích xuất dữ liệu.

1.1.3. Những nguy cơ mất an toàn ODBS

Khi sử dụng ODBS, các dữ liệu của DO được lưu trữ trên đám mây nên sẽ đối mặt với nhiều rủi ro từ môi trường Internet hoặc do chính máy chủ của DSP. Dựa trên những nguy cơ mà Namasudra và cộng sự [56] đưa ra, ta thấy một số vấn đề về an toàn ODBS như sau:

- CSDL chủ yếu bị tấn công bởi: Tấn công bên trong và tấn công bên ngoài. Tấn công bên trong là những nhân viên quản trị máy chủ thuộc về DSP can thiệp hệ thống. Tấn công bên ngoài là những người trên môi trường Internet tấn công vào hệ thống để đánh cắp dữ liệu. Tấn công bên trong khó kiểm soát do DSP toàn quyền quản lý CSDL.
- Khi một người dùng bị thu hồi quyền khỏi CSDL, người dùng đó không được phép truy cập dữ liệu. Nếu chương trình chưa giải quyết tốt vấn đề thu hồi quyền của người dùng thì sẽ dẫn tới tấn công theo hình thức "thoả hiệp". Hơn nữa, việc chưa quan tâm đổi khoá với CSDL mã khi nghi ngờ lộ lọt khoá sẽ dẫn đến rò rỉ dữ liệu.
- Trên đám mây, dữ liệu được lưu trữ và sao lưu dự phòng (backup) ở nhiều vị trí khác nhau, trên các thiết bị lưu trữ khác nhau trong các hệ thống máy chủ. Việc sao lưu dữ liệu do nhân viên của DSP quản lý và DSP có thể phục hồi dữ liệu (restore) từ các bản dự phòng cho dù DO đã xóa dữ liệu khỏi máy chủ đám mây. Như vậy, việc DO xoá dữ liệu không đồng nghĩa với việc DO bảo đảm dữ liệu của mình được an toàn.
- Cho đến nay, chưa có một tiêu chuẩn cụ thể giữa DSP, DO và người dùng để đảm bảo tuyệt đối an toàn dữ liệu cho DO, bảo mật thông tin cho người sử dụng.

Nhận xét 1.2 *Nguy cơ mất an toàn của ODBS phụ thuộc chính vào hệ thống DSP (phần cứng, phần mềm, nhân viên), các mối đe dọa trên đường truyền, sự quản lý của DO về khoá mã, thuật toán mật mã và những chính sách mà DO thiết lập đối với người tham gia vào hệ thống CSDL.*

1.1.4. Các bài toán bảo đảm an toàn ODBS

Từ những nguy cơ và rủi ro được trình bày, các nhà khoa học nhận thấy cần nghiên cứu các vấn đề nhằm bảo vệ an toàn dữ liệu trên ODBS. Có các bài toán bảo đảm an toàn cho ODBS như:

- Tính bí mật của dữ liệu (Data confidentiality) [32, 60, 76, 84]: Dữ liệu ban đầu (bản rõ) được biến đổi thành dữ liệu khác (bản mã). Nếu người có bản mã mà không có khoá mã sẽ không thể biết được nội dung do không thể chuyển lại thành bản rõ. DO thường mã hóa dữ liệu bởi một hàm mật mã với khóa bí mật trước khi lưu trữ trên đám mây. Các hàm mật mã được sử dụng như: Mật mã khóa đối xứng (DES, AES...), mã hóa bảo toàn thứ tự (OPE), mã hóa đồng cấu (HOM)... Với cách bảo vệ như vậy, chỉ DO mới có khóa mã để giải mã dữ liệu. Kẻ tấn công (kể cả DSP) nếu có được dữ liệu thì cũng không biết được nội dung do không có khóa mã. Tùy thuộc vào mức độ bảo mật cho CSDL mà các nhà nghiên cứu đề xuất các mức mã hóa CSDL khác nhau. Với mã hoá mức nào thì mức đó dùng chung một khoá. Các mức mã hóa CSDL theo mức độ an toàn từ thấp đến cao là: Mức bảng, mức cột, mức dòng và mức ô. Trong bài toán này, việc quản lý và thay đổi khoá mã cũng là một bài toán quan trọng. Thay đổi khoá là quá trình giải mã bằng khoá cũ và mã hoá bằng khoá mới toàn bộ dữ liệu của DO đã lưu trữ ở máy chủ của DSP.
- Tính riêng tư của dữ liệu (Data privacy) [25, 40, 91]: Các người dùng khác nhau sẽ có quyền khác nhau đối với việc truy xuất CSDL. Người dùng chỉ được phép truy cập vào những dữ liệu mà DO cấp quyền cho mình. DO quản lý tính riêng tư dữ liệu bằng cách sử dụng một cơ chế

quản lý truy cập sao cho mỗi đơn vị dữ liệu (dòng, cột) chỉ được truy cập bởi những người dùng mà DO cấp phép. Nếu CSDL được mã hóa thì DO phải có cơ chế quản lý khoá kết hợp với quản lý quyền truy cập người dùng, tránh các trường hợp tấn công đánh cắp khoá mã.

- Đảm bảo kết quả truy vấn (Query Assurance) [70, 45, 27]: Kết quả trả về từ server phải đảm bảo tính đúng (correctness), đầy đủ (completeness) và mới nhất (freshness). CSDL của DO có thể bị tấn công, làm sai lệch dữ liệu so với ban đầu, do đó, cần phải có phương pháp xác thực dữ liệu trả về khi truy vấn.

Một số bài toán bảo đảm an toàn cho ODBS khác như: Xác thực người dùng, bảo vệ tính riêng tư người dùng, ghi nhật ký hệ thống và bảo vệ siêu dữ liệu...

Nhận xét 1.3 *Tính bí mật dữ liệu giúp DO bảo vệ dữ liệu trước việc lộ lọt thông tin. Để thực hiện tốt tính bí mật thì các hàm mật mã đóng vai trò quan trọng trong việc mã hóa, giải mã, tính toán trên dữ liệu mã. Hàm mật mã phải đảm bảo độ an toàn, thực hiện nhanh và hỗ trợ các phép tính toán cơ bản. Các mô hình, thuật toán truy vấn trên CSDL mã phải đảm bảo về mặt thời gian thực hiện. Bên cạnh đó, việc quản lý tính riêng tư dữ liệu và quản lý khóa, quản lý người dùng cần thực hiện tốt, tránh trường hợp người dùng truy cập bất hợp pháp vào dữ liệu. Thay đổi khóa mã cho CSDL là biện pháp cần thiết để tăng tính an toàn cho ODBS. Việc xác thực dữ liệu trả về khi truy vấn sẽ giúp cho người khai thác dữ liệu tránh được những hậu quả đáng tiếc từ những thông tin không chính xác.*

1.2. Những nghiên cứu về an toàn cho ODBS

Từ những vấn đề nêu trên, các nhà khoa học trong và ngoài nước đã đề xuất các giải pháp để bảo đảm an toàn cho CSDL nói chung và ODBS nói riêng. Trong đó có các nghiên cứu như:

1.2.1. Nghiên cứu trong nước

Tác giả Nguyễn Anh Tuấn [2] nghiên cứu xây dựng mô hình mã hóa CSDL nói chung. Tác giả dùng các thuật toán mật mã để mã hóa dữ liệu và dùng các bảng ảo, trigger của DBMS để giải mã dữ liệu trước khi truy vấn. Phương pháp này chống lại các tấn công đánh cắp dữ liệu do Adv không có khóa mã để giải mã dữ liệu. Tuy nhiên, khi số lượng bảng ảo tăng lên thì sẽ làm giảm khả năng xử lý của máy chủ. Ngoài ra, nghiên cứu này chưa quan tâm giải quyết bài toán xác thực dữ liệu, quản lý và thay đổi khoá.

Tác giả Phạm Thị Bạch Huệ [1] nghiên cứu và phát triển một số giải pháp bảo mật và bảo vệ tính riêng tư trong ODBS. Trong đó, tác giả tập trung vào các vấn đề liên quan đến tính riêng tư người dùng, tính riêng tư dữ liệu, xác thực người dùng và ghi nhật ký hệ thống. Tác giả giải quyết tính riêng tư người dùng bằng phương pháp truy vấn tạo nhiễu, sau đó loại bỏ các bản ghi dư thừa ở phía người dùng, điều này hạn chế việc máy chủ dữ liệu thống kê tần suất truy vấn của người dùng. Việc bảo vệ tính riêng tư dữ liệu được tác giả sử dụng cơ chế quản lý truy cập mức cột, có thể sử dụng trong CSDL ít biến động. Để bảo vệ tính riêng tư dữ liệu,tác giả dùng thuật toán mã hóa dữ liệu dựa trên định lý số dư Trung Hoa (Chinese remainder theorem - CRT), sử dụng hàm một chiều để dẫn xuất khóa, dùng cây nhị phân để quản lý. Tuy nhiên, tác giả chưa giải quyết bài toán xác thực dữ liệu trả về. Bài toán thay khoá được tác giả đưa ra trong hướng nghiên cứu tiếp theo, nhưng đến nay, bài toán này vẫn chưa được giải quyết

1.2.2. Nghiên cứu ngoài nước

Khi mã hóa dữ liệu để đảm bảo tính bí mật thì đồng thời các nhà nghiên cứu đề xuất phương pháp truy vấn dữ liệu mã. Bởi vì khi mã hóa, dữ liệu đã bị biến đổi, không còn là kiểu dữ liệu ban đầu. Bên cạnh đó, nếu mã hóa theo các vector IV ngẫu nhiên trong các thuật toán mật mã thì sẽ không giữ được mối quan hệ giữa các bảng vì mặc dù cùng giá trị rõ nhưng hai giá trị

ở hai bảng khác nhau sẽ sinh hai giá trị mã khác nhau. Các công bố về truy vấn trên dữ liệu mã như: Dùng siêu dữ liệu (metadata) lưu trữ các thông tin phụ trợ để hỗ trợ truy vấn trên dữ liệu mã [9, 32]. Dùng máy chủ trung gian (proxy) để chuyển đổi câu truy vấn [12, 60]. Truy vấn trực tiếp trên dữ liệu mã (phương pháp này chỉ hỗ trợ một số dạng truy vấn) [53, 82].

Hacigumus và cộng sự [32] đề xuất giải pháp dùng chương trình biến đổi truy vấn (Query Translator) để chuyển đổi câu truy vấn dạng rõ của người dùng sang truy vấn dạng mã. Các dữ liệu được mã hóa trước khi lưu trữ ở máy chủ của DSP. Bên cạnh đó, dữ liệu có thêm các chỉ mục làm thông tin phụ trợ cho phép thực hiện truy vấn CSDL tại máy chủ của DSP mà không cần phải giải mã. DO dùng chỉ mục để chuyển đổi các truy vấn của người dùng thành các truy vấn thích hợp có thể thực thi trên máy chủ, và người dùng sẽ nhận được kết quả trả về từ máy chủ của DSP. Tuy nhiên, phương pháp này vẫn còn có những nhược điểm là tăng chi phí lưu trữ, chi phí để tính toán lại sau khi truy vấn CSDL và chưa quan tâm xác thực dữ liệu mã và đổi khoá mã.

Agrawal và cộng sự [3] đề xuất giải pháp bảo đảm tính bí mật dữ liệu và quyền riêng tư của các truy vấn bằng cách lưu trữ phân tán dữ liệu trên nhiều máy chủ thay vì mã hóa dữ liệu. Cách tiếp cận này tận dụng được số lượng lớn tài nguyên từ các máy chủ DSP tùy thuộc số lượng máy chủ mà DO thuê. CSDL được chia thành nhiều phần và lưu trữ mỗi phần tại DSP: $DSP_1, DSP_2, ..., DSP_n$ sao cho dữ liệu từ bất kỳ máy chủ $k(k \leq n)$ đều không thể tiết lộ được bí mật của dữ liệu. Khi truy vấn dữ liệu, các máy chủ trả về các giá trị thỏa điều kiện và được khôi phục lại giá trị ban đầu bằng một hàm do Agrawal định nghĩa trước. Tuy nhiên, phương pháp này chưa giải quyết vấn đề xác thực dữ liệu khi truy vấn.

A. Popa và cộng sự [60] trình bày một mô hình mã hoá và cách thức thực thi truy vấn trên CSDL mã bằng cách sử dụng một máy chủ làm trung gian gọi là CryptDB proxy. CryptDB proxy lưu trữ khoá bí mật và các lớp mã

hoá hiện tại của mỗi cột. Máy chủ DSP lưu trữ dữ liệu người dùng đã được mã hóa (trong đó tên bảng, tên cột cũng được mã hoá để ẩn đi ý nghĩa của bảng) và một số bảng phụ trợ được sử dụng bởi CryptDB. CryptDB cũng cung cấp cho các máy chủ với một số hàm do người dùng định nghĩa (User defined function - UDF) cho phép các máy chủ tính toán trên dữ liệu mã với một số phép toán nhất định. Điểm mạnh của CryptDB là cho phép các máy chủ tính toán trên dữ liệu mã hóa mà không cần giải mã. Proxy là hệ thống trung gian có thể tăng cường khả năng tính toán cho các ứng dụng khi truy vấn trên DBMS. Nó có thể thực hiện một số lượng lớn các truy vấn SQL trên các dữ liệu được mã hóa mà không đòi hỏi bất kỳ thay đổi bên trong của máy chủ DBMS và kết quả được giải mã bởi các user đáng tin cậy. Tuy nhiên, CryptDB chỉ thực thi trên MySQL mà chưa hỗ trợ các DBMS thương mại như SQL Server và Oracle. Hơn nữa, CryptDB vẫn chưa hỗ trợ hết tất cả các câu truy vấn dữ liệu.

Li và cộng sự [50] đề xuất mô hình L-EncDB truy vấn trên CSDL mã gồm 2 tầng: Tầng ứng dụng và tầng CSDL. Tầng ứng dụng mục đích mã hóa các dữ liệu và chuyển đổi các dạng câu truy vấn cho phù hợp các loại mã (FPE, FQE hoặc OPE). Tầng CSDL chỉ cung cấp dịch vụ CSDL mà không cho phép người dùng can thiệp hệ thống, người dùng chỉ tương tác với máy chủ bằng cách sử dụng các câu lệnh SQL. L-EncDB hỗ trợ được các loại truy vấn: truy vấn khoảng (dùng mã OPE), truy vấn mờ (dùng mã FPE). L-EncDB giải quyết truy vấn dữ liệu mã hơn CryptDB ở chỗ L-EncDB có thể tìm kiếm mờ trên toàn bộ dữ liệu chuỗi, trong khi CryptDB chỉ hỗ trợ một phần.

Đối với bài toán quản lý khóa để đảm bảo tính riêng tư dữ liệu, các nhà nghiên cứu thường sử dụng ma trận quản lý truy cập, các thuật toán dẫn xuất khóa, các cấu trúc quản lý khóa, đảm bảo chỉ có những người dùng có quyền mới có thể khai thác dữ liệu. Các mức riêng tư dữ liệu như: Mức bảng, mức cột, mức dòng.

A.Zych và cộng sự [91] đưa ra giải pháp đảm bảo tính riêng tư dữ liệu ở

mức dòng với cấu trúc quản lý khoá dựa trên mối quan hệ giữa các người dùng trong danh sách truy cập và xây dựng từ dưới lên (bottom up). Đồng thời dùng phương pháp dẫn xuất khoá dựa trên nghi thức trao đổi khoá Diffie – Hellman và phân phối khoá dẫn xuất đến từng người dùng. Tuy nhiên, thuật toán sinh khoá và thuật toán xây dựng cấu trúc quản lý khoá phức tạp, đòi hỏi chi phí tính toán cao.

Damiani và cộng sự [21] sử dụng ma trận truy cập để mô tả tập hợp quyền của người dùng. Trong đó, hàng biểu diễn người dùng và cột là quyền truy cập dữ liệu. Cây dẫn xuất khóa được xây dựng từ ma trận truy cập để kiểm soát truy cập. Thuật toán đề xuất tất cả các nút đại diện cho quyền truy cập theo ma trận truy cập. Mỗi nút sẽ được liên kết với một khóa bí mật sao cho khóa cho nút v sẽ được cấp cho người dùng u khi và chỉ khi người dùng thuộc quyền truy cập v và u không thuộc quyền truy cập của nút cha của u.

El Khory và cộng sự [25] đã đề xuất phương pháp quản lý truy xuất ở mức dòng với dẫn xuất khoá bằng hàm một chiều và cơ chế quản lý khoá bằng cây nhị phân. Mỗi mức của cây nhị phân tương ứng với các quyền của người dùng. Người dùng được cung cấp các khóa tương ứng với mức của anh ta trong cây và có thể lấy tất cả các khóa cho dữ liệu được phép truy cập. Giải pháp này có nhược điểm là gán các khóa một cách không bình đẳng. Người dùng ở mức đầu của cây chỉ sở hữu vài khóa, trong khi những người ở mức dưới nhận được một số lượng lớn khóa.

Các phương pháp đề xuất hầu hết vẫn chưa linh hoạt và dẫn đến không thể tránh khỏi việc tốn nhiều thời gian dựng lại cấu trúc khóa khi thay đổi quyền của người dùng thường xuyên. Ngoài ra, các phương pháp đề xuất chưa giải quyết vấn đề thay đổi khoá mã của CSDL khi cần thiết.

Đối với bài toán xác thực kết quả truy vấn trả về, các nhà khoa học thường sử dụng chữ ký số hoặc dùng các cấu trúc dữ liệu xác thực (Authenticated Data Structure - ADS) [44, 45, 58, 90]. Tuy nhiên, ADS thường không hiệu quả đối với dữ liệu động do thời gian thay đổi cấu trúc của ADS rất lớn. Vì

nếu một bản ghi được chèn hoặc xóa, tất cả các bản ghi kế tiếp bản ghi bị sửa đổi cũng thay đổi theo.

Einar Mykletun [54] và cộng sự đưa ra một giải pháp để đảm bảo tính đúng cho các câu truy vấn dạng chỉ đọc (read-only) và không có tính toán gộp (như SUM, AVERAGE,...). Mỗi bản ghi dữ liệu được lưu kèm theo chữ ký số của bản ghi đó. Kết quả trả về kèm theo với chữ ký số. Người dùng kiểm tra nội dung dữ liệu với chữ ký kèm theo để xác nhận được tính đúng của dữ liệu. Tuy nhiên, vì số lượng bản ghi trả về có thể lớn, vì vậy việc kiểm tra một số lượng lớn chữ ký số cho từng bản ghi dẫn đến thời gian tính toán lớn và tốn nhiều chi phí cho client. Để giải quyết vấn đề này, Mykletun đề nghị mô hình Condensed-RSA. Theo đó, thay vì kiểm tra riêng lẻ từng chữ ký của từng bản ghi, client chỉ cần kiểm tra tất cả các bản ghi cùng lúc dựa trên chữ ký tổng hợp (condensed signature) do server trả về là có thể xác định được tính đúng của dữ liệu. Mykletun cũng nêu ra một giải pháp khác nhằm đạt được tính đúng là sử dụng Merkle Hash Tree (MHT). MHT là cây mà các lá của nó là kết quả băm của dữ liệu của từng bản ghi tương ứng trong CSDL và đánh dấu node gốc bằng một chữ ký số. Nếu kèm theo hai bản ghi ở hai biên kết quả, ta có thể chứng minh được kết quả trả về đầy đủ. Cấu trúc MHT đòi hỏi phải lưu trữ kèm theo một cấu trúc dữ liệu chuyên dùng để phục vụ cho việc bảo đảm truy vấn. Mỗi cấu trúc này thường chỉ áp dụng cho một thuộc tính, như vậy, trong trường hợp CSDL có nhiều thuộc tính tìm kiếm (searchable attribute) đòi hỏi nhiều cấu trúc tương ứng, điều này có thể làm tăng chi phí phần cứng để lưu trữ tại server.

Yuan và cộng sự [88] đề xuất một lược đồ truy vấn gộp có thể xác thực cho CSDL thuê ngoài. Mỗi bản ghi được gán thẻ (tag) xác thực dùng để kiểm tra tính toàn vẹn của kết quả truy vấn gộp. Tuy nhiên, phương pháp này chỉ kiểm chứng trên dữ liệu rõ và phải kiểm tra thẻ của các bản ghi liên quan đến truy vấn. Nghĩa là hệ thống phải tính toán lại chữ ký của bản ghi mặc dù truy vấn chỉ lấy một vài thuộc tính trong bản ghi.

Maithili Narasimha [57] đã đề nghị một hướng tiếp cận mới dựa trên chuỗi chữ ký số. Khi đó, trong chữ ký của một bản ghi bao gồm nội dung của bản ghi liền trước nó. Trong kết quả trả về, server trả kèm thêm hai bản ghi ở biên để có thể đảm bảo được tính đúng và đầy đủ. Tuy nhiên, đề xuất của Narasimha được thực hiện trên CSDL rõ. Nếu áp dụng phương pháp này trên CSDL mã thì sau khi xác thực, Narasimha phải tốn thêm thời gian thực hiện giải mã các bản ghi trước khi trả kết quả rõ cho người dùng.

Không chỉ có các nhà nghiên cứu đưa ra các công bố khoa học mà các công ty cũng cho ra đời các sản phẩm thương mại để bảo đảm an toàn cho ODBS. Một số sản phẩm tiêu biểu là:

CipherCloud [16]: Giải pháp này cung cấp khả năng mã/giải mã dữ liệu vào/ra theo thời gian thực dựa trên việc sử dụng cổng mã hóa (gateway). Nó cung cấp giải pháp dựa trên mã hóa mạnh có thể tìm kiếm (SSE) theo tiêu chuẩn bởi FIPS 140-2. Dữ liệu chỉ được giải mã khi người dùng được quyền truy xuất dữ liệu từ đám mây. Các khóa mã được lưu trữ cục bộ và không chia sẻ với DSP. CipherCloud cung cấp giải pháp chống mất dữ liệu, phát hiện và giám sát phần mềm độc hại cho môi trường đám mây.

Thales eSecurity [71]: Cung cấp khả năng mã/giải mã trong suốt cho các hệ quản trị CSDL trên các dịch vụ đám mây. Giải pháp này cũng cho phép người dùng tự quản lý khóa.

Porticor [61] cung cấp các giải pháp để mã hóa dữ liệu và quản lý khóa. Giải pháp dữ liệu riêng ảo Porticor của công ty bảo vệ lớp dữ liệu của các ứng dụng trong đám mây như: Ổ cứng ảo, hệ thống tệp, CSDL và lưu trữ phân tán. Nó hoạt động phổ biến trong cơ sở hạ tầng như một dịch vụ (IaaS) và nền tảng như một dịch vụ (PaaS).

Các sản phẩm thương mại ngày càng hoàn thiện, giải quyết hầu hết các bài toán về an toàn cho ODBS. Tuy nhiên, các sản phẩm thương mại hoạt động cũng như một thành phần trung gian, cho phép mã hóa ODBS dưới dạng cổng mã hóa. Các sản phẩm này thường đắt tiền và vì là sản phẩm mã

nguồn đóng nên khó kiểm soát về mặt kỹ thuật, công nghệ và tính an toàn, người dùng chấp nhận sự tin tưởng để sử dụng. Bên cạnh đó, các sản phẩm này khó có khả năng tùy biến về mặt giải pháp mã hóa, không cho người dùng sử dụng các thuật toán mật mã phù hợp với yêu cầu riêng của cơ quan, tổ chức hoặc quốc gia của mình.

Nhận xét 1.4 *Các công bố đã giải quyết các bài toán: Truy vấn trên dữ liệu mã, xác thực dữ liệu mã, bảo vệ tính riêng tư dữ liệu... Tuy nhiên, với bài toán đảm bảo tính bí mật CSDL, các nghiên cứu đã đề xuất giải pháp để thực thi truy vấn trên dữ liệu mã mà chưa quan tâm đến thời gian xử lý truy vấn. Đây là vấn đề quan trọng trong mô hình CSDL quan hệ do xử lý trên dữ liệu mã tốn nhiều thời gian hơn so với truy vấn trên dữ liệu rõ. Đối với bài toán xác thực dữ liệu mã, các nghiên cứu sử dụng ADS sẽ khó tính toán lại giá trị xác thực khi truy vấn trên nhiều bảng. ADS hầu như không phù hợp với dữ liệu động vì phải xây dựng lại cấu trúc xác thực khi dữ liệu thay đổi. Mặc khác, quá trình xác thực dữ liệu mã độc lập với quá trình giải mã dữ liệu, do đó, các giải pháp đề xuất vẫn phải tốn thời gian giải mã dữ liệu sau khi xác thực truy vấn. Bên cạnh đó, bài toán thay khoá được giới thiệu trong công trình [1], tuy nhiên đến nay, bài toán này vẫn chưa được giải quyết.*

1.3. Các bài toán được giải quyết trong luận án

Theo khảo sát và phân tích như trên, các nghiên cứu đã được đề xuất chưa giải quyết hết các bài toán mà luận án quan tâm. Những bài toán được luận án nghiên cứu, giải quyết cụ thể như sau:

1. Giảm thời gian truy vấn trên dữ liệu mã: Đề xuất phương pháp xử lý song song trên dữ liệu mã để giảm thời gian truy vấn. Phương pháp này có thể được sử dụng ở các pha xử lý khác nhau như: Tính toán, giải mã, xác thực dữ liệu... giúp tăng hiệu năng của hệ thống.

2. Xác thực kết quả truy vấn: Đề xuất hai thuật toán xác thực lô dựa trên

hai bài toán khó và sử dụng thuật toán xác thực lô này vào xác thực dữ liệu mã khi truy vấn. Phương pháp đề xuất thích hợp với CSDL động do không phải xây dựng lại cấu trúc xác thực, đồng thời kết hợp xác thực và giải mã, trả kết quả rõ cho người dùng.

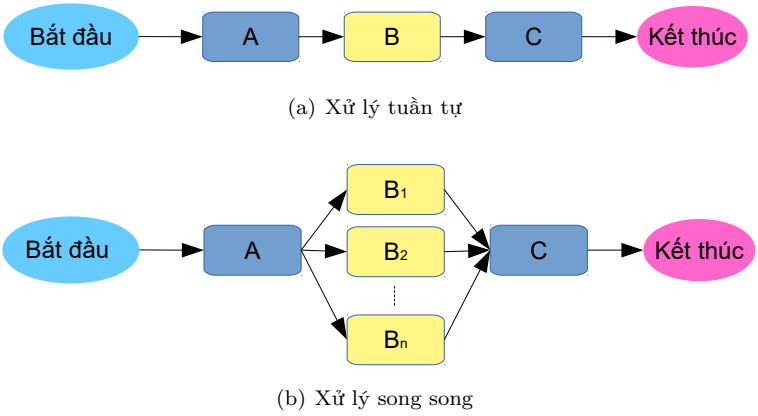
3. Quản lý quyền truy cập và thay đổi khoá mã: Đề xuất dùng cây KMT và ma trận quản lý truy cập để bảo vệ tính riêng tư dữ liệu. Đề xuất thuật toán mã hoá tệp tin người dùng để bảo vệ nội dung của cây KMT, tránh việc đánh cắp khoá mã. Đề xuất thuật toán thay đổi khoá mã của CSDL thuê ngoài dựa trên mô hình MapReduce.

1.4. Một số kiến thức liên quan

Trong mục này, luận án trình bày những kiến thức liên quan đến những đề xuất trong chương 2, 3 bao gồm: Xử lý song song để giảm thời gian truy vấn trên dữ liệu mã, chữ ký số và xác thực lô để xác thực dữ liệu mã khi truy vấn, hệ thống tệp tin trong không gian người dùng, kho lưu trữ khoá-giá trị sử dụng để mã hoá tệp tin quản lý khoá, mô hình lập trình MapReduce dùng để thay đổi khoá mã, tập dữ liệu mẫu TPC-H để thực nghiệm phân tích, đánh giá kết quả.

1.4.1. Xử lý song song

Xử lý song song là quá trình xử lý gồm nhiều tiến trình được kích hoạt đồng thời và cùng tham gia giải quyết một vấn đề thực hiện trên những hệ thống có nhiều bộ xử lý đồng thời. Xử lý song song cho phép chia nhỏ vấn đề lớn thành các vấn đề nhỏ để thực hiện cùng một lúc. Mục đích của xử lý song song là tận dụng triệt để tài nguyên của hệ thống trên các máy tính đa nhân, đa luồng. Các máy tính song song sẽ thực hiện những tính toán nhanh hơn do sử dụng nhiều bộ xử lý đồng thời. Việc ứng dụng xử lý song song vào các bài toán sẽ giải quyết được những vấn đề lớn hơn, phức tạp hơn trong thực tế.



(a) Xử lý tuần tự

(b) Xử lý song song

Hình 1.2: Quá trình xử lý tuần tự và song song

Các máy tính xử lý song song có thể được phân loại tùy theo mức độ hỗ trợ song song của phần cứng:

- Máy tính đơn với bộ xử lý đa nhân, đa luồng.
- Cụm máy tính, tính toán lưới (Grid computing) sử dụng nhiều máy tính để xử lý cùng một công việc.
- Những kiến trúc máy tính song song chuyên dụng xử lý cho những công việc đặc trưng.

1.4.2. Chữ ký số

Chữ ký số được dùng để xác thực về nguồn gốc và tính toàn vẹn của thông tin. Các thuật toán chữ ký số thường dựa trên hai hệ mật phổ biến là RSA và Elgamal. Hệ mật RSA dựa trên độ khó của bài toán phân tích thừa số nguyên tố. Elgamal dựa trên độ khó của bài toán logarit rời rạc. Có một số chữ ký số dựa trên hệ mật Elgamal được xem là chuẩn như: DSA (Digital Signature Algorithm) là giải thuật chữ ký số trong chuẩn chữ ký số (Digital Signature Standard – DSS) của chính phủ Mỹ và được đưa ra bởi FIPS-186 [28]. GOST là chuẩn chữ ký số của Liên bang Nga [31] và KCDSA là chuẩn chữ ký số của Hàn Quốc [51].

1.4.2.1. Lược đồ chữ ký số RSA

R.L.Rivest, A.Shamir và L.Adleman đề xuất một lược đồ chữ ký số lấy tên của ba người là RSA [63]. Các tham số trong lược đồ được tạo bằng cách chọn ngẫu nhiên các số nguyên tố lớn p, q, $p \neq q$ và tính $n = pq$. Thuật toán tạo khóa, ký và xác thực được mô tả như sau:

- Tạo khoá: Khoá bí mật d, khoá công khai (e, n).
  Tính $\phi(n) = (p-1)(q-1)$
  Chọn số ngẫu nhiên e sao cho $gcd(e, \phi(n)) = 1$.
  Tính $d = e^{-1} \bmod \phi(n)$.

- Ký: Tạo chữ ký σ cho dữ liệu $m \in \{0,1\}^*$
  Tính $\sigma = m^d \bmod n$.

- Xác thực: Kiểm tra chữ ký σ đúng/sai.

  1. Tính $m' = \sigma^e \bmod n$.
  2. Nếu $m' = m$, trả về 1 (đúng), ngược lại trả về 0 (sai).

1.4.2.2. Lược đồ chữ ký số Rabin

Michael.O.Rabin đề xuất một lược đồ chữ ký số dựa trên độ khó của bài toán phân tích số [62]. Thuật toán tạo khóa, tạo chữ ký và xác thực được mô tả như sau:

- Tạo khoá: Khoá bí mật (p, q), khoá công khai (n, b).
  Cho $n = pq$ là tích hai số nguyên tố lớn p, q, và chọn $0 \leq b \leq n$.

- Ký: Tạo chữ ký σ cho dữ liệu $m \in \{0,1\}^*$

  1. Chọn ngẫu nhiên phần đệm U và tính $c = H(mU) \bmod n$.
  2. Kiểm tra c thỏa mãn $x(x+b) = c \bmod n$. Nếu không tìm được x thì quay lại bước 1.
  3. Xuất $\sigma \leftarrow (U, x)$.

- Xác thực: Kiểm tra chữ ký σ đúng/sai.

  1. Tính $c = H(mU) \bmod n$.
  2. Nếu $x(x + b) = c \bmod n$, trả về 1 (đúng), ngược lại trả về 0 (sai).

1.4.2.3. Lược đồ chữ ký số Schnorr

Schnorr đề xuất một lược đồ chữ ký số [64] và được ứng dụng trong nhiều lĩnh vực xác thực. Cho các tham số p, q, g trên bài toán logarit rời rạc (DLP), p là một số nguyên tố lớn có kích thước trong khoảng từ 512 đến 1024 bit, q là một ước số nguyên tố của $p - 1$ có kích thước 160 bit, g là phần tử sinh cấp q thuộc $GF(p)$.

Thuật toán tạo khóa, ký và xác thực được mô tả như sau:

- Tạo khoá: Khoá bí mật x, khoá công khai y.
  Chọn số ngẫu nhiên $x \in \{1, ..., q - 1\}$ và tính $y = g^x \bmod p$.

- Ký: Tạo chữ ký σ cho dữ liệu $m \in \{0, 1\}^*$

  1. Chọn số ngẫu nhiên $t \in \mathbb{Z}_q$ và tính $r = g^t \bmod p$.
  2. Tính $h = H(m||r)$.
  3. Tính $s = t - hx \bmod q$.
  4. Xuất $\sigma \leftarrow (h, s)$.

- Xác thực: Kiểm tra chữ ký σ đúng/sai.

  1. Tính $r' = g^s y^h (\bmod p)$.
  2. Tính $h' = H(m||r')$.
  3. Nếu $h' = h$, trả về 1 (đúng), ngược lại trả về 0 (sai).

1.4.2.4. Lược đồ chữ ký số dựa trên hai bài toán khó

Các lược đồ chữ ký số có độ an toàn dựa trên tính khó của hai bài toán khó (DLP và IFP) được quan tâm từ năm 1994 do L. Harn đề xuất [36] và liên tục sau đó các công bố vào năm 2008 của ba tác giả E. S. Ismail, N. M. F.

Tahat và R. R. Amad [43], của E. S. Dermova năm 2009 [23], của S. Vishnoi và Shrivastava năm 2012 [78], của Binh V. Do, Minh H. Nguyên, Nikolay A. Moldovyal, năm 2013 [24],...

Với những phân tích chỉ ra thực chất về độ an toàn của các lược đồ chữ ký số dựa trên hai bài toán khó như phân tích của N. Y. Lee và T. Hwang về lược đồ của Harn công bố năm 1996 [49], phân tích của Shin-Yan Chiou và Yi-Xuan He về giao thức của Vishnoi và Shrivastava công bố năm 2013 [13]... Tóm lại chỉ còn hai lược đồ đưa ra trong [24] là đúng nghĩa với việc có độ an toàn dựa trên tính khó của hai bài toán khó đó là lược đồ Rabin-Schnorr và RSA-Schnorr.

Thuật toán tạo khóa, tạo chữ ký và xác thực trong [24] được mô tả như sau:

- Tạo khoá: Khoá bí mật (x, q', q), khoá công khai (p, g, y).
  Chọn số nguyên tố lớn độc lập q', q có dạng $4r + 3$ và tính $n = q'q$.
  Chọn ngẫn nhiên số x bí mật với $x \in Z_p^*$.
  Tính $y = g^x \bmod p$.

- Ký: Tạo chữ ký σ cho dữ liệu $m \in \{0, 1\}^*$

  1. Tính $R = g^k \bmod p$, với k là số bí mật ngẫu nhiên, $1 < k \leq n - 1$.
  2. Tính $E = H(m||R)$.
  3. Tính giá trị S sao cho $S^2 = k - xE \bmod n$.
  4. Xuất $\sigma \leftarrow (E, S)$.

- Xác thực: Kiểm tra chữ ký σ đúng/sai.

  1. Tính $R^* = g^{S^2} y^E \bmod p$.
  2. Tính $E^* = H(m||R^*)$.
  3. Nếu $E^* = E$, trả về 1 (đúng), ngược lại trả về 0 (sai).

1.4.3. Xác thực lô

Xác thực lô (batch verification) là hình thức xác thực nhiều chữ ký số cùng một lúc. Mục đích của xác thực lô là để giảm chi phí tính toán so với xác thực lần lượt từng chữ ký số và nó thích hợp cho các hệ thống được yêu cầu xác thực số lượng lớn chữ ký cùng lúc [10, 33]. Xác thực lô được ứng dụng vào các lĩnh vực xác thực như: Hệ thống giám sát tự động [85], kiểm soát truyền tải trong mạng không dây [81, 29], hệ thống vạn vật kết nối [47, 52]...

Định nghĩa 1.2 *Cho k dữ liệu m_i tương ứng với k chữ ký số σ_i được ký bởi khoá bí mật của người ký, $i = 1, ...k$. Với y là khoá công khai của người ký, hàm V được gọi là hàm xác thực lô nếu $V_y(\sigma_1, \sigma_2, ..., \sigma_k) \in \{0, 1\}$*

Nếu các chữ ký σ_i của các dữ liệu m_i là đúng của người ký thì kết quả của hàm $V_y = 1$. Ngược lại, nếu một trong những chữ ký σ_i không hợp lệ thì hàm $V_y = 0$.

Tuỳ thuộc vào các chữ ký số được ký bởi các lược đồ khác nhau sẽ có thuật toán xác thực lô khác nhau. Có hai loại xác thực lô: Xác thực lô tương tác và xác thực lô xác suất. Xác thực lô tương tác là dạng mà người ký tạo ra chữ ký σ thông qua sự tương tác với người xác thực, và sau đó người xác thực xác nhận tất cả các chữ ký σ này cùng một lúc. Xác thực lô xác suất là đối với mỗi dữ liệu m được ký, một số nguyên t ngẫu nhiên được chọn riêng và sau đó được tính lại giá trị r (ví dụ trong DSA, $r = (g^t \bmod p) \bmod q$). Thay vì ký dữ liệu m trực tiếp, các lược đồ chữ ký số phải ký dựa trên kết quả băm một chiều của m.

Naccache và cộng sự [55] đề xuất xác thực lô tương tác gồm hai phần: Phần tạo chữ ký và phần xác thực. Phần tạo chữ ký được người ký và người xác thực có sự tương tác với nhau. Phần xác thực được người xác thực kiểm tra nhiều chữ ký cùng một lúc. Cho n dữ liệu m_i, phần tạo chữ ký được thực hiện như sau:

- Với $i = 1, ..., n$, người ký chọn ngẫu nhiên $k_i \in \mathbb{Z}_q$, tính $\lambda_i = g^{k_i} \bmod p$

và gửi λ_i cho người xác thực;

- Người xác thực gửi lại dữ liệu ngẫu nhiên b_i có chiều dài e bit;
- Người ký tính $s_i = k_i^{-1}(H(m_i||b_i) + \lambda_i x) \bmod q$ và gửi s_i cho người xác thực.

Phần xác thực: Người xác thực kiểm tra

$$\prod_{i=1}^{n} \lambda_i = g^{\sum_{i=1}^{n} s_i^{-1} H(m_i||b_i)} y^{\sum_{i=1}^{n} s_i^{-1}\lambda_i} \bmod p$$

và thay thế $\{\lambda_i, s_i, b_i, m_i\}_{i=1,...,n}$ bởi $\{r_i = \lambda_i \bmod q, s_i, m_i||b_i\}_{i=1,...,n}$.

Bellare, Garay và Rabin [8] đưa ra phương pháp xác thực lô theo luỹ thừa modulo. Xác thực lô cho lũy thừa modulo được Bellare và cộng sự trình bày như sau: Cho $\mathbb{G}$ là một nhóm có cấp là q, g là phần tử sinh của $\mathbb{G}$. Hàm luỹ thừa modulo là $x \leftarrow g^x$, với $x \in \mathbb{Z}_q$. Định nghĩa biểu thức $EXP_{\mathbb{G},g}(x, y) = 1$ nếu $g^x = y, x \in \mathbb{Z}_q, y \in \mathbb{G}$. Nếu muốn xác thực nhiều giá trị $(x_1, y_1), ..., (x_n, y_n)$ là kiểm tra biểu thức $EXP_{\mathbb{G},g}(x_i, y_i) = 1$ với $i = 1, ..., n$ thì cách ngây thơ nhất là tính g^{x_i} và so sánh với y_i. Tuy nhiên, cách này sẽ tốn n lần tính luỹ thừa. Bellare đề xuất ba phương pháp tính toán nhanh xác thực lô cho biểu thức $EXP_{\mathbb{G},g}$ là kiểm tra tập hợp con ngẫu nhiên (random subset test), kiểm tra số mũ nhỏ (small exponents test) và kiểm tra khối (bucket test).

Harn giới thiệu lược đồ xác thực lô dựa trên RSA [37]. Cho p và q là hai số nguyên tố, $n = pq$. Giá trị e và d lần lượt là khóa công khai và khóa bí mật của người ký sao cho $ed = 1 \bmod \phi(n)$. Giả sử có k chữ ký $(\sigma_1, \sigma_2, ..., \sigma_k)$ tương ứng với k dữ liệu m_i, trong đó chữ ký $\sigma_i = H(m_i)^d \bmod n$ $(1 \leq i \leq k)$. Để xác thực k chữ ký cùng lúc, Harn kiểm tra, nếu thoả mãn biểu thức: $(\prod_{i=1}^{k} \sigma_i)^e = \prod_{i=1}^{k} H(m_i) \bmod n$ thì k chữ ký σ_i là hợp lệ.

Shao đề xuất thuật toán xác thực lô dựa trên lược đồ chữ ký số Schnorr [65]. Giả sử có k dữ liệu được ký bởi thuật toán Schnorr, nghĩa là có k chữ ký $\sigma_i(r_i, s_i), i = 1, 2, ..., k$, $i = 1, 2 \ldots, k$ được ký cùng một người ký với khoá

bí mật x. Để xác thực k chữ ký cùng lúc, Shao chọn ngẫu nhiên k số nguyên $u_i \in (1, 2^{32})$, $i = 1, 2, ..., k$ và kiểm tra nếu thoả mãn biểu thức:

$$\prod_{i=1}^{k} r_i^{u_i} = (g^{\sum_{i=1}^{k} u_i s_i} y^{\sum_{i=1}^{k} u_i H(m_i||r_i)}) \bmod p$$

thì k chữ ký đó là hợp lệ.

Hwang và cộng sự đề xuất lược đồ xác thực lô dựa trên lược đồ chữ ký của Shim's IBS [69] gọi là BV-Shim [41]. Các tham số dùng trong BV-Shim: Hàm ánh xạ $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, trong đó $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ là các nhóm có cùng cấp q. Cho P_1, P_2 được tạo ngẫu nhiên tương ứng của $\mathbb{G}_1, \mathbb{G}_2$. Với tham số bí mật λ, thuật toán chọn ngẫu nhiên $s \in \mathbb{Z}_q$ làm khoá chính và tính $P_{pub} = sP_2$ và $g = e(P_1, P_2)$. Các hàm băm: $H : \{0,1\}^k \to \mathbb{Z}_q, H_1 : \{0,1\}^* \to \mathbb{Z}_q, H : \{0,1\}^* \to \{0,1\}^l$. Như vậy, tham số BV-Shim là $pp = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P_1, P2, P_{pub}, g, H, H_1, H2)$. Với $ID \in \{0,1\}^k$ và khoá s, thuật toán tính $Z_{ID} = H(ID) \in \mathbb{Z}_q$ và khoá bí mật $S_{ID} = \frac{1}{S+Z_{ID}} P_1 \in \mathbb{G}_1$.

Tạo chữ ký: Chọn ngẫu nhiên $r \in \mathbb{Z}_q^*$ và tính $X = rP_1 \in \mathbb{G}_1$, $h = H_1(m, X)$, và $Y = (r + h)S_{ID} \in \mathbb{G}_1$. Chữ ký là $\sigma = (X, Y)$.

Kiểm tra chữ ký: Tính $z = H(ID) \in \mathbb{Z}_q, Q_{ID} = P_{pub} + zP_2 \in \mathbb{G}_2, h = H_1(m, X)$ và kiểm tra nếu $e(Y, Q_{ID}) = e(X, P_2)g^h$ thì chữ ký hợp lệ, ngược lại chữ ký không hợp lệ.

Xác thực lô: Kiểm tra k chữ ký $\sigma_i = (X_i, Y_i)$ tương ứng với ID_i, m_i, $1 \leq i \leq k$. BV-Shim tính $z_i = H(ID_i) \in \mathbb{Z}_q, h_i = H_1(m_i, X_i)$. Chọn ngẫu nhiên $\xi \in \{0,1\}^l$ và tính $\delta_i = H_2(\xi, h_i)$. Sau đó kiểm tra biểu thức $e(\sum_{j=1}^{k} \delta_j Y_j, P_{pub}) e(\sum_{j=1}^{k} (\delta_j Z_j Y_j - \delta_j X_j), P_2) = g^{\sum_{j=1}^{k} \delta_j h_j}$, nếu thoả mãn thì k chữ ký hợp lệ. Ngược lại, một trong những chữ ký σ_i không hợp lệ.

Kittur và cộng sự trình bày thuật toán xác thực lô dựa trên đường cong Elliptic [48]. Thuật toán của Kittur được mô tả như sau: Cho đường cong Elliptic $E(\mathbb{F}_p)$, chọn $P \in E(\mathbb{F}_p)$ có bậc n. Chọn ngẫu nhiên số nguyên d làm khoá bí mật, $2 \leq d \leq n - 2$. Tính $Q = dP$ là khoá công khai.

Tạo chữ ký: Chọn ngẫu nhiên số nguyên t, $2 \leq t \leq n - 2$. Tính $R = tP$,

$r = x(R) \bmod n$, với $x(R)$ là toạ độ x của R. Tính $s = k^{-1}(h(m) + dr) \bmod n$. Chữ ký $\sigma = (R, s)$.

Xác thực lô: Kiểm tra k chữ ký $\sigma_i = (R_i, s_i)$ tương ứng với dữ liệu m_i, $1 \leq i \leq k$. Chọn ngẫu nhiên k số nguyên tố $b_1, b_2, ..., b_k < n, \in \mathbb{F}_p$. Lần lượt tính $r_i = x(R_i)$, $w_i = s_i^{-1} \bmod n$, $u_i = h(m_i)w_i \bmod n$, $v_i = r_i w_i \bmod n$. Kiểm tra nếu biểu thức $\sum_{i=1}^{k} R_i b_i = (\sum_{i=1}^{k}(b_i u_i) \bmod n)P + (\sum_{i=1}^{k}(b_i v_i) \bmod n)Q$ đúng thì k chữ ký là hợp lệ. Ngược lại, một trong những chữ ký không hợp lệ.

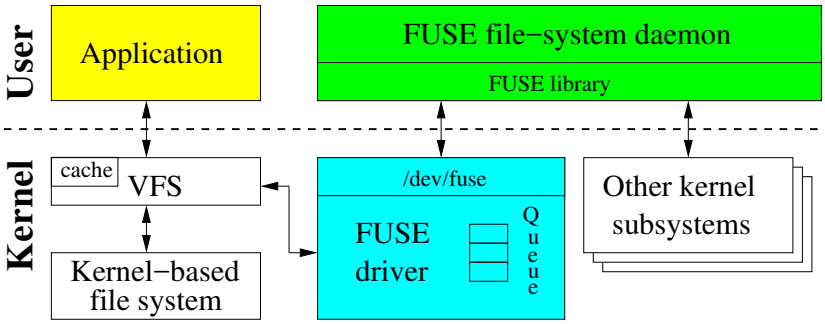
Nhận xét 1.5 *Các thuật toán xác thực lô được đề xuất thường dựa trên độ khó của bài toán phân tích số hoặc logarit rời rạc. Ưu điểm của các thuật toán này là thời gian xác thực chữ ký nhanh. Để tăng tính an toàn cho chữ ký, các nhà khoa học đưa ra ý tưởng kết hợp hai bài toán khó vào lược đồ chữ ký số. Tuy nhiên, nhược điểm của nó sẽ làm tăng độ phức tạp tính toán. Trong trường hợp cụ thể, hệ thống cần được xem xét giữa nâng cao độ an toàn và tốc độ xử lý để có sự lựa chọn thuật toán xác thực cho phù hợp. Có các lược đồ chữ ký số dựa trên hai bài toán khó [79, 42, 24, 14] nhưng theo khảo sát của luận án thì chưa có lược đồ xác thực lô dựa trên hai bài toán khó. Đề xuất lược đồ xác thực lô dựa trên hai bài toán khó sẽ có được sự an toàn từ việc khó giải của hai bài toán khó và thời gian xác thực đồng thời nhiều dữ liệu sẽ nhanh hơn xác thực thông thường bởi tính chất của xác thực lô.*

1.4.4. Hệ thống tệp tin trong không gian người dùng

Hệ thống tệp tin trong không gian người dùng (Filesystem in Userspace - FUSE) [73] là một thành phần trong nhân của hệ điều hành Linux. FUSE cung cấp thư viện giao diện lập trình (Application programming interface - API) cho phép người dùng tạo hoặc truy cập hệ thống tệp người dùng. FUSE có sẵn trên các môi trường như Android, Linux và macOS.

Khi một hệ thống tệp mới được thực thi, một chương trình xử lý sẽ tạo một liên kết đến thư viện FUSE (được gọi là libfuse). Chương trình này xác

định trạng thái của hệ thống tệp phản hồi khi nó được yêu cầu đọc/ghi/thống kê. Trình xử lý được đăng ký với nhân khi hệ thống tệp tin được gắn kết. Nếu người dùng thực thi các yêu cầu đọc/ghi/thống kê cho một hệ thống tệp được gắn kết, nhân sẽ chuyển tiếp các yêu cầu IO này đến trình xử lý và gửi phản hồi của trình xử lý trở lại cho người dùng.



Hình 1.3: Kiến trúc của FUSE [77]

Hình 1.3 cho thấy kiến trúc của FUSE gồm hai phần: Phần nhân và trình tiện ích người dùng. Phần nhân thuộc về nhân của Linux, nó đăng ký trình điều khiển hệ thống tệp tin FUSE trên hệ thống tệp tin ảo (Virtual file systems - VFS) của Linux. Trình điều khiển FUSE có thể được coi là một thành phần trung gian chuyển hướng yêu cầu đến trình tiện ích người dùng. Khi một người dùng tạo ra hệ thống tệp tin trên không gian người dùng bằng các ứng dụng (Application), thì tệp tin của người dùng được biên dịch thành một tệp tin nhị phân. Nếu người dùng thực hiện đọc/ghi tệp đó, VFS định tuyến hoạt động đến trình điều khiển nhân FUSE. Trình điều khiển cấp phát cấu trúc yêu cầu FUSE và đặt nó vào hàng đợi FUSE. Tại thời điểm này, tiến trình đã chỉ định hoạt động là thông thường và đặt trong trạng thái chờ. Sau đó, trình tiện ích người dùng FUSE lấy yêu cầu từ hàng đợi của nhân bằng cách đọc từ /dev/fuse và xử lý yêu cầu. Trình tiện ích đọc hoặc ghi từ các khối. Khi thực hiện xong việc xử lý yêu cầu, trình tiện ích FUSE ghi phản hồi trở lại /dev/fuse. Trình điều khiển nhân FUSE sau đó đánh dấu yêu cầu là đã hoàn thành và kích hoạt tiến trình người dùng ban đầu. Một số hoạt

động hệ thống tệp được gọi bởi một ứng dụng có thể hoàn thành mà không liên lạc với trình tiện ích FUSE của người dùng. Ví dụ, đọc từ một tệp có các trang được lưu trong bộ đệm của nhân, không cần phải chuyển tiếp đến trình điều khiển FUSE [77].

1.4.4.1. Giao diện lập trình FUSE

API FUSE cung cấp hai API: Một API đồng bộ hóa mức cao và một API không đồng bộ hóa ở mức thấp. Với hai API, các yêu cầu từ kernel được chuyển tiếp đến chương trình chính bằng cách sử dụng gọi ngược (callback). Khi sử dụng API mức cao, callback có thể hoạt động với tên tệp và đường dẫn thay vì các nút và việc xử lý yêu cầu kết thúc khi hàm callback trả về giá trị. Khi sử dụng API mức thấp, callback phải hoạt động với các nút và phản hồi phải được gửi bằng cách sử dụng một tập hợp hàm API riêng biệt.

API mức cao được định nghĩa chủ yếu trong *fuse.h*. API mức thấp chủ yếu được lưu trong *fuse_ lowlevel.h*.

Callback là một tập hợp các hàm để thực hiện các thao tác của tệp và cấu trúc *fuse_ operations* chứa các con trỏ tới các hàm callback:

struct fuse_ operations{

*int(*getattr)(const char *,struct stat *);*

*int(*mknod)(const char *, mode_ t, dev_ t);*

*int(*mkdir)(const char *, mode_ t);*

*int(*rmdir)(const char *);*

*int(*rename)(const char *, const char *);*

*int(*chmod)(const char *, mode_ t);*

*int(*open)(const char *, struct fuse_file_ info *);*

*int(*read)(const char *, char *, size_ t, off_ t, struct fuse_file_ info *);*

*int (*write)(const char *, const char *, size_ t, off_ t, struct fuse_file_ info *);*

.... };

Các thành phần của cấu trúc này là các con trỏ hàm. Mỗi hàm trong cấu trúc sẽ được FUSE gọi khi một sự kiện cụ thể xảy ra trên hệ thống tệp. Chẳng hạn, khi người dùng ghi vào một tệp tin, hàm được trỏ bởi thành phần "write" trong cấu trúc sẽ được gọi. Để thực hiện hệ thống tệp của mình, chúng ta cần sử dụng cấu trúc này và định nghĩa các hàm của nó sau đó hoàn thành cấu trúc với các con trỏ của các hàm đã được định nghĩa. Hầu hết các hàm ở đây là tùy chọn, ta không cần phải thực hiện tất cả.

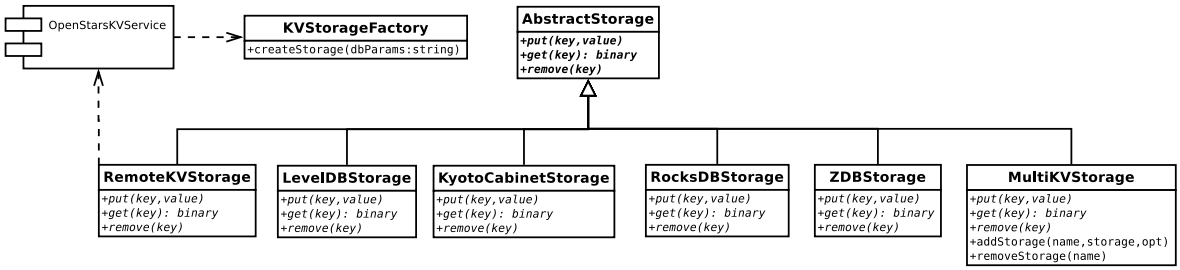
1.4.5. Lưu trữ khóa-giá trị

Lưu trữ khóa-giá trị là một loại CSDL NoSQL (Not Only SQL). Nó có hình thức đơn giản là một bảng với hai cột tương ứng với hai trường là khóa và giá trị. Kiểu của giá trị là chuỗi/nhị phân hoặc cấu trúc. Kiểu của khóa có thể là số nguyên hoặc chuỗi/nhị phân. Có nhiều triển khai và thiết kế lưu trữ khóa-giá trị bao gồm cả dựa trên bộ nhớ và trên đĩa. Lưu trữ khóa-giá trị dựa trên bộ nhớ thường được sử dụng để lưu trữ dữ liệu đệm, lưu trữ khóa-giá trị trên đĩa được sử dụng để lưu trữ dữ liệu vĩnh viễn trong hệ thống tệp. Một thư viện lưu trữ khóa-giá trị được viết bởi tác giả ThanhNT gọi là OpenStars [59]. OpenStars hỗ trợ nhiều CSDL khóa-giá trị như LevelDB [80], RockDB [86], Kyoto Cabinet, ZDB [72]...

1.4.5.1. Cấu trúc OpenStars

CSDL OpenStars sử dụng một số lớp trừu tượng để cung cấp truy cập lớp cụ thể như leveldb, rocksdb, ZDB [72]. Lớp *AbstractKVStorage* là lớp cơ sở, định nghĩa hàm get(), put(), multiget(), multiput(), remove() để ghi hoặc lấy dữ liệu. Các lớp kế thừa từ lớp này sẽ xử lý các hoạt động dữ liệu của lưu trữ khóa-giá trị. Lớp *AbstractCursor* là con trỏ cơ sở, con trỏ này trỏ đến từng bản ghi của CSDL. Lớp *KVStorageFactory* sử dụng để tạo *AbstractKVStorage* dựa trên chuỗi tùy chọn cụ thể.

Hình 1.4 mô tả các lớp của CSDL khóa-giá trị. Ta sử dụng *KVStorage-Factory* để tạo một thể hiện của *AbstractKVStorage*, truyền tham số đầu vào



Hình 1.4: Mô hình lớp của OpenStars [59]

cho hàm tạo của *KVStorageFactory* là một chuỗi cụ thể. Các nhà phát triển chỉ thực hiện trên *AbstractKVStorage* để xử lý với CSDL khóa-giá trị. Lưu trữ khóa-giá trị có thể được phục vụ với một dịch vụ phụ trợ từ xa. Chúng ta có thể đọc và ghi dữ liệu từ xa bằng cách sử dụng *RemoteKVStorage*.

1.4.5.2. Triển khai API OpenStars

Chúng ta có thể sử dụng kiểu của CSDL khóa-giá trị bằng cách truyền tham số qua chuỗi *configString* để khởi tạo CSDL. Cụ thể: *AbstractKVStorage* database = factory.createStorage(configString, name, rwmode);*

Sau khi khởi tạo CSDL, chúng ta có thể đọc và ghi dữ liệu bằng cách gọi các hàm put và get như sau:

```
string sVal ;
string sKey;
//Đọc dữ liệu từ CSDL
database->get (sVal, sKey);
//Ghi dữ liệu vào CSDL
database->put(sVal,sKey);
```

1.4.6. Mô hình lập trình MapReduce

MapReduce [22, 30, 68] là một mô hình lập trình cho phép xử lý khối dữ liệu lớn bằng cách chia công việc thành các nhiệm vụ độc lập, thực hiện các tác vụ song song thông qua các cụm máy tính (cluster) và tự động thu thập kết quả trên nhiều cụm để trả về một kết quả. Lợi thế lớn nhất của một

chương trình được thực hiện bằng MapReduce là nó chạy trên bất kỳ nút (node) nào trong trung tâm xử lý (một, hàng trăm hay hàng nghìn nút) mà không quan tâm đến mã lệnh của chương trình. Lập trình MapReduce là việc tạo ra hai hàm map() và reduce(), mỗi hàm được thực hiện song song trên các nút tính toán.

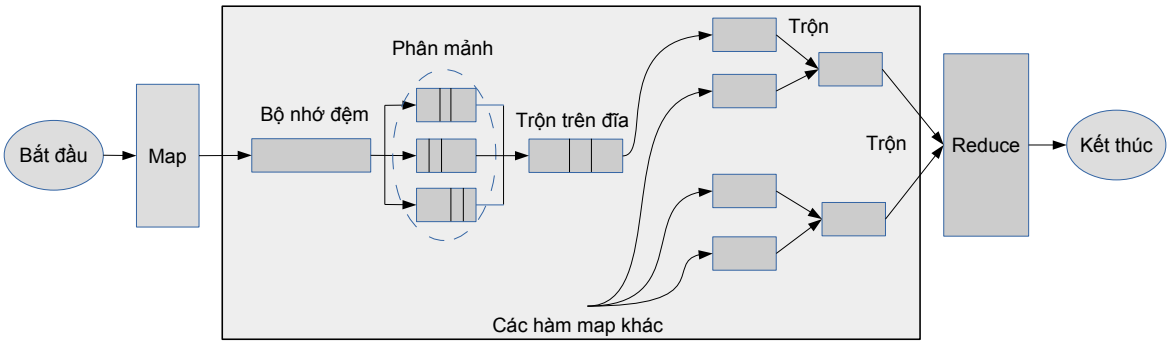
- Giai đoạn đầu của MapReduce là giai đoạn map: MapReduce sẽ tự động phân chia dữ liệu đầu vào cho các nút tính toán trong trung tâm dữ liệu. Mỗi nút tính toán đều chạy một hàm map() với mục đích là chia nhỏ dữ liệu. Trong giai đoạn map, hàm map() xử lý đầu vào là các cặp khóa-giá trị (key-value), tạo ra các cặp khóa-giá trị trung gian và được chuyển thành đầu vào cho giai đoạn Reduce.
- Giai đoạn thứ hai là giai đoạn Reduce: Giai đoạn này tổng hợp và xử lý các dữ liệu trung gian từ giai đoạn map. Reduce cũng được thực hiện trên các nút tính toán, mỗi nút đều nhận một khoá, tuỳ thuộc vào giá trị ở giai đoạn map mà chia sẻ các khoá này cho các nút xử lý, sau đó tổng hợp lại thành một giá trị cho mỗi khoá.

Quá trình thực hiện tiến trình của MapReduce được mô tả như sau:

- Giai đoạn Map: $<k_1, v_1> \rightarrow <k_2, v_2>$
- Giai đoạn Reduce: $<k_2, v_2> \rightarrow <k_3, v_3>$

Trong đó $<k_i, v_i>$, $1 \leq i \leq 3$ là các cặp khoá-giá trị do người dùng xác định trước khi thực hiện MapReduce. Một tập dữ liệu được truyền đến hàm map() dưới dạng các cặp $<k_1, v_1>$. Hàm map() xử lý các cặp $<k_1, v_1>$ và xuất kết quả trung gian ở dạng cặp $<k_2, v_2>$. Kết quả trung gian $<k_2, v_2>$ được tính toán lại trong hàm reduce(), đưa ra kết quả cuối cùng ở dạng $<k_3, v_3>$.

Mô hình lập trình MapReduce có thể dùng để xử lý dữ liệu lớn trên các nền tảng khác nhau, trong đó nó là một trong những thành phần chính của Apache Hadoop, ngoài ra nó có thể tương thích trên nền tảng Apache Spark.



Hình 1.5: Tiến trình xử lý của MapReduce [87]

- Apache Hadoop là một dự án phần mềm mã nguồn mở cho phép xử lý phân tán các tập dữ liệu lớn trên các cụm máy tính bằng các mô hình lập trình đơn giản. Hadoop cung cấp một dịch vụ có khả năng tính toán lớn trên các cụm máy tính với mỗi máy tính có khả năng chịu lỗi cao. Nó được thiết kế để có thể mở rộng quy mô từ vài máy đến hàng ngàn máy chủ, mỗi máy cung cấp khả năng tính toán và lưu trữ cục bộ. Thay vì dựa vào phần cứng để thực hiện các công việc xử lý thì các thư viện được thiết kế để có khả năng tính toán, phát hiện và xử lý các lỗi ở lớp ứng dụng. Hadoop thực hiện xử lý dữ liệu từ hệ thống tệp tin HDFS (Hadoop Distributed File System). HDFS gồm hai thành phần chính là "namenode" và "datanodes" với mục đích là quản lý các công việc đọc/ghi dữ liệu.

- Apache Spark cũng là một hệ thống xử lý dữ liệu lớn và tính toán nhanh. Nó cung cấp các API cấp cao trong các ngôn ngữ lập trình và công cụ được tối ưu hỗ trợ các mô hình thực thi chung. Nó cũng hỗ trợ một bộ công cụ cấp cao bao gồm Spark SQL cho SQL và xử lý dữ liệu có cấu trúc, MLlib để học máy, GraphX để xử lý đồ thị và Spark Streaming.

Sự khác biệt chính giữa Hadoop và Spark là ở cách tiếp cận xử lý và lưu trữ: Spark thực hiện xử lý dữ liệu trên bộ nhớ RAM, trong khi Hadoop phải đọc và ghi vào ổ cứng. Do đó, tốc độ xử lý của Spark có thể nhanh hơn tới 100 lần so với Hadoop. Tuy nhiên, Hadoop có thể thực hiện xử lý tính toán

với các tập dữ liệu lớn hơn nhiều so với Spark. Trong trường hợp dữ liệu kết quả trả về lớn hơn dung lượng RAM, Hadoop có thể vượt trội hơn Spark. Mặt khác, do cần nhiều bộ nhớ RAM để xử lý, nên Spark tốn nhiều chi phí đầu tư hơn so với Hadoop. Bên cạnh đó, vì Hadoop xử lý các bộ dữ liệu lớn được phân bố giữa các nút tạo thành một cụm các máy chủ nên người dùng không phải mua và bảo trì các thiết bị chuyên dụng đắt tiền. Hadoop lập chỉ mục và theo dõi trạng thái dữ liệu, giúp xử lý và phân tích hiệu quả hơn. Spark là một công cụ xử lý dữ liệu mà cho phép thực hiện các hoạt động khác nhau trên các bộ dữ liệu phân tán, nhưng không cung cấp cơ chế lưu trữ phân tán của chúng. Tuỳ thuộc vào mục đích sử dụng cụ thể mà người dùng có sự lựa chọn đúng đắn. Spark thích hợp với các chương trình tính toán xử lý dữ liệu theo thời gian thực, còn Hadoop phù hợp với các dữ liệu lớn về mặt dung lượng. Hadoop xử lý tuyến tính các bộ dữ liệu khổng lồ, trong khi Spark mang lại hiệu suất nhanh, xử lý lặp, xử lý đồ thị, học máy... Đối với trường hợp thực hiện đổi khoá cho toàn bộ CSDL thì chương trình sẽ thực thi trên các bảng của CSDL. Do dung lượng các bảng của CSDL lớn và không cần xử lý theo thời gian thực nên DO chỉ cần dùng nền tảng Hadoop để giảm chi phí đầu tư hoặc thuê các hệ thống máy chủ.

1.4.7. Bộ dữ liệu mẫu TPC-H

TPC-H bao gồm CSDL mẫu có thể thay đổi về số lượng dữ liệu và một bộ các câu truy vấn mẫu. Các truy vấn mẫu của TPC-H phù hợp với nhiều ngữ cảnh khác nhau. TPC-H hỗ trợ kiểm tra lượng dữ liệu lớn, thực hiện các truy vấn với mức độ phức tạp cao và phản ánh nhiều khía cạnh về khả năng của hệ thống để xử lý các truy vấn. Các khía cạnh này bao gồm kích thước CSDL được chọn dựa vào các truy vấn thực thi, sức mạnh xử lý truy vấn khi các truy vấn được gửi bởi một luồng duy nhất và thông lượng truy vấn khi các truy vấn được gửi bởi nhiều người dùng đồng thời [75]. TPC-H được sử dụng như là dữ liệu chuẩn để đánh giá trong các nghiên cứu đề xuất trước

đây về CSDL nói chung và ODBS nói riêng. CSDL của TPC-H có 8 bảng và các cột trong bảng tương ứng như sau:

1. **nation**(n_nationkey, n_name, n_regionkey, n_comment);
2. **region**(r_regionkey, r_name, r_comment);
3. **part**(p_partkey, p_name, p_mfgr, p_brand, p_type, p_size, p_container, p_retailprice, p_comment);
4. **supplier**(s_suppkey, s_name, s_address, s_nationkey, s_phone, s_acctbal, s_comment);
5. **partsupp**(ps_partkey, ps_suppkey, ps_availqty, ps_supplycost, ps_comment);
6. **customer**(c_custkey, c_name, c_address, c_nationkey, c_phone, c_acctbal, c_mktsegment, c_comment);
7. **orders**(o_orderkey, o_custkey, o_orderstatus, o_totalprice, o_orderdate, o_orderpriority, o_clerk,o_shippriority, o_comment);
8. **lineitem**(l_orderkey, l_partkey, l_suppkey, l_linenumber, l_quantity, l_extendedprice, l_discount, l_tax, l_returnflag, l_linestatus, l_shipdate, l_commitdate, l_receiptdate, l_shipinstruct, l_shipmode, l_comment);

Muốn tạo dữ liệu mẫu của TPC-H thì ta sử dụng lệnh DBGEN. Theo chế độ mặc định, DBGEN sẽ tạo 8 tệp tin chứa dữ liệu tương ứng với 8 bảng được định nghĩa trong lược đồ CSDL TPC-H. Các tệp tin sẽ được tạo trong thư mục hiện tại và có tên là <tên bảng>.tbl. Ví dụ: Tệp region.tbl sẽ chứa dữ liệu cho bảng region. Một số cú pháp mẫu sử dụng DBGEN để tạo dữ liệu:

- Tạo 8 bảng dữ liệu mẫu với tỉ lệ là 1: *dbgen -s 1.* Tỉ lệ có thể thay đổi là: 1, 10, 100, 300, 1000, 3000, 10000, 30000, 100000.
- Chỉ tạo bảng lineitem, cho CSDL tỷ lệ 10, và ghi đè lên tệp tin nếu có: *dbgen -s 10 -f -T L.*

Bảng 1.1: Số lượng bản ghi của các bảng trong TPC-H với dbgen -s 1

Tên bảng	nation	region	part	supplier	partsupp	customer	orders	lineitem
Số bản ghi	25	5	200000	10000	800000	150000	1500000	6001215

1.5. Kết luận

Chương 1 trình bày những kiến thức chung về ODBS, các bài toán đảm bảo an toàn cho ODBS và một số kiến thức liên quan đến các đề xuất. Chương 1 cũng khảo sát các nghiên cứu trong và ngoài nước về an toàn ODBS đồng thời đưa ra các định hướng, các bài toán được nghiên cứu và giải quyết.

Bối cảnh nghiên cứu của luận án là dựa trên mô hình ODBS với dữ liệu được mã hoá trước khi lưu trữ lên máy chủ DSP. Khi đó, vấn đề đặt ra cho luận án là nghiên cứu giảm thời gian truy vấn trên dữ liệu mã, đảm bảo kết quả truy vấn trả về là chính xác, quản lý, kiểm soát khoá của người dùng và thay đổi khoá mã của CSDL. Trong chương 2, luận án sẽ đề xuất các nghiên cứu liên quan đến vấn đề giảm thời gian truy vấn trên dữ liệu mã, xác thực dữ liệu mã khi truy vấn. Chương 3 sẽ đề xuất mô hình quản lý khoá người dùng, mã hóa hệ thống tệp tin trong hệ điều hành và phương pháp đổi khoá của CSDL mã. Cuối cùng là kết luận của luận án và đề nghị hướng phát triển nghiên cứu tiếp theo.

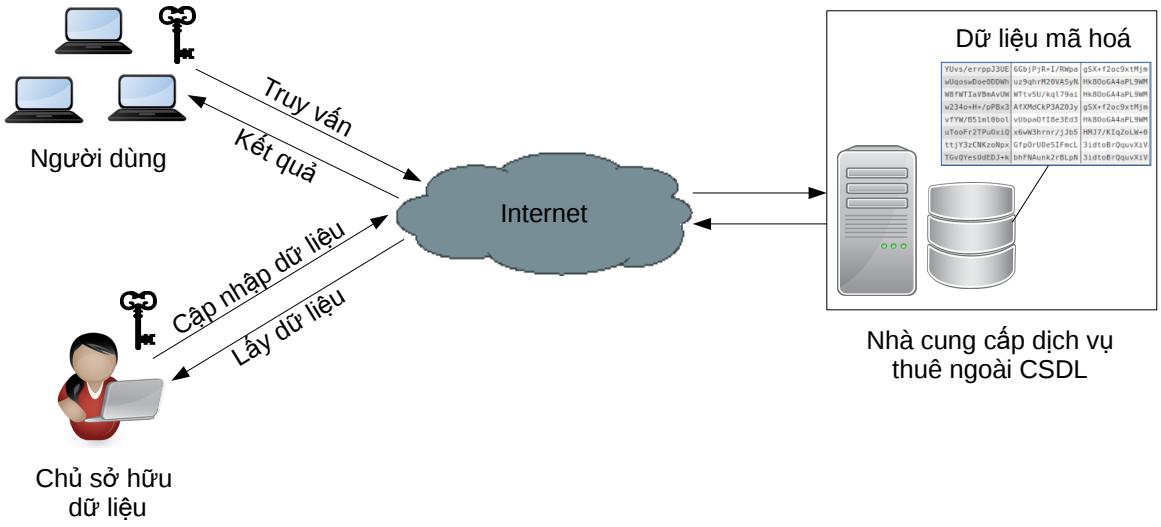
Chương 2
GIẢM THỜI GIAN TRUY VẤN VÀ XÁC THỰC KHI TRUY VẤN CSDL MÃ TRÊN ODBS

Nội dung chương 2 đề xuất mô hình, thuật toán giảm thời gian thực hiện khi truy vấn dữ liệu mã trên ODBS, đề xuất này được công bố trong [CT1]. Chương 2 cũng giới thiệu về vấn đề xác thực dữ liệu mã khi truy vấn trên ODBS, các lược đồ chữ ký số và xác thực lô. Luận án đề xuất hai thuật toán xác thực lô dựa trên hai bài toán khó là phân tích số (Integer Factorization Problem - IFP) và logarit rời rạc (Discrete Logarithm Problem - DLP) và áp dụng thuật toán xác thực lô được đề xuất là Rabin-Schnorr vào xác thực dữ liệu mã khi truy vấn trên môi trường thuê ngoài. Kết quả nghiên cứu xác thực dữ liệu mã khi truy vấn được công bố trong các công trình [CT2][CT3][CT6]. Cuối cùng là kết luận chương.

2.1. Giới thiệu chung

Mô hình ODBS luôn tiềm ẩn nhiều nguy cơ mất an toàn, gây nguy hại đến dữ liệu của DO. Để đảm bảo được tính bí mật dữ liệu của mình, DO mã hóa dữ liệu trước khi lưu trữ lên DSP (hình 2.1). Cùng với mã hóa, thì các nghiên cứu truy vấn trên dữ liệu mã được nghiên cứu và phát triển. Tuy nhiên, khi áp dụng phương pháp truy vấn trên dữ liệu mã thì thường làm cho máy chủ xử lý chậm hơn so với truy vấn trên dữ liệu rõ. Đặt biệt, nếu số lượng người dùng nhiều, số bản ghi trả về lớn thì các phương pháp truy vấn mã có thể khó thực hiện được.

Mặt khác, khi truy vấn CSDL từ ODBS, người dùng muốn kết quả trả về là chính xác. Tuy nhiên, trong thực tế có nhiều nguyên nhân xảy ra có thể làm thay đổi nội dung dữ liệu ban đầu của DO như:



Hình 2.1: Mô hình ODBS với CSDL mã

1. Phần mềm, phần cứng máy chủ DSP bị lỗi;
2. Thao tác của nhân viên quản trị hệ thống máy chủ DSP bị sai hoặc cố tình can thiệp bất hợp pháp vào dữ liệu;
3. Hệ thống của DSP bị tấn công;
4. Dữ liệu bị thay đổi trên đường truyền.

Để xác định được dữ liệu là đúng đắn thì khi trả về kết quả, DO phải kiểm tra (xác thực) dữ liệu phải đúng với dữ liệu mà mình tạo ra.

Các nghiên cứu xác thực ODBS thường xây dựng các cấu trúc dữ liệu xác thực (Authenticated Data Structure – ADS) [90, 58, 44]. Tuy nhiên, ADS sẽ tốn nhiều thời gian để tính toán lại nếu dữ liệu bị thay đổi. Chỉ cần một bản ghi được thêm, sửa hoặc xóa, ADS bắt buộc phải xây dựng lại cấu trúc xác thực. Một số nghiên cứu chỉ hỗ trợ vài dạng truy vấn đặc biệt mà chưa hỗ trợ truy vấn trên nhiều bảng; Hoặc các nghiên cứu chỉ đề xuất phương pháp xác thực độc lập với quá trình giải mã, khi đó, sau khi xác thực, DO tốn thêm thời gian để tiến hành giải mã, trả dữ liệu rõ cho người dùng.

Trong phạm vi nghiên cứu của chương này, luận án tập trung vào việc giảm thời gian truy vấn, xác thực dữ liệu trả về khi truy vấn trên dữ liệu mã

hoá. Khi người dùng truy vấn, DSP trả về dữ liệu mã, luận án dùng phương pháp xử lý song song để tính toán, giải mã dữ liệu, giảm thời gian cho quá trình truy vấn. Đối với xác thực dữ liệu mã, luận án sử dụng xác thực lô để kiểm tra kết quả trả về, nếu dữ liệu trả về là đúng thì tiến hành giải mã và trả kết quả rõ cho người dùng. Phương pháp đề xuất hiệu quả khi CSDL động do không phải xây dựng lại các cấu trúc xác thực. Đồng thời, phương pháp của luận án đề xuất tiến hành xác thực đồng thời với quá trình giải mã dữ liệu nên không tốn thời gian giải mã dữ liệu sau khi xác thực như các phương pháp xác thực trước đây.

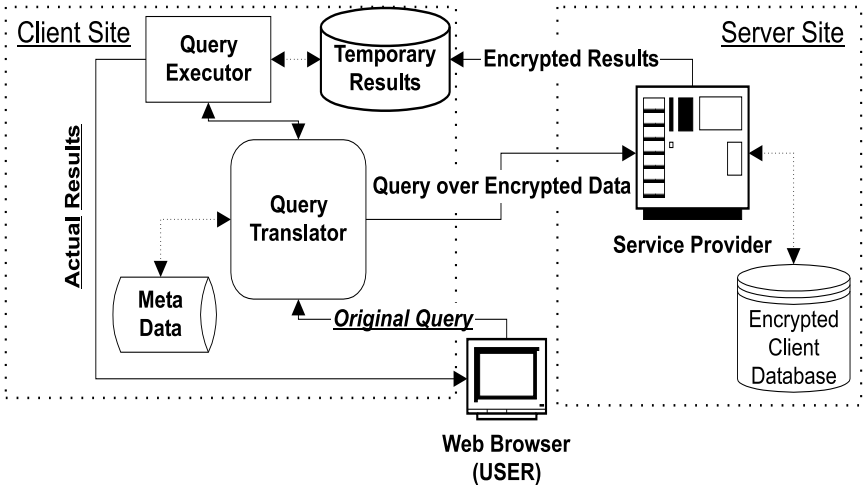
2.2. Giảm thời gian thực thi truy vấn trên dữ liệu mã

2.2.1. Một số phương pháp truy vấn trên dữ liệu mã

Tùy thuộc vào mô hình mã hóa CSDL mà có phương pháp truy vấn trên CSDL mã cho phù hợp. Có nhiều mô hình mã hóa CSDL, trong đó hai dạng thường các nhà nghiên cứu quan tâm là: Mã hóa bản ghi thành một giá trị mã và mã hóa từng ô dữ liệu.

Hacigumus và cộng sự [32] đề xuất giải pháp mã hóa bản ghi thành một giá trị mã và thực hiện truy vấn trên CSDL đã được mã hoá. Ý tưởng của giải pháp này là dùng chương trình biến đổi truy vấn để chuyển đổi các truy vấn của người dùng thành các truy vấn thích hợp để thực thi trên máy chủ của DSP.

Trong mô hình này, một câu truy vấn sẽ được chia thành hai phần: (1) Truy vấn ở phía máy chủ (Server Site) về các dữ liệu đã được mã hóa, truy vấn này được thực hiện tại máy chủ của DSP, (2) Truy vấn phía người dùng (Client Site), truy vấn này được thực hiện trên máy của người dùng và kết quả truy vấn có được sau khi được loại bỏ những bản ghi không thỏa mãn truy vấn. Như vậy, khi người dùng truy vấn dữ liệu, DSP sẽ trả về các bộ dữ liệu mã liên quan câu truy vấn. Người dùng giải mã các kết quả trả về và truy vấn trên dữ liệu rõ. Để cho phép máy chủ chọn một bộ dữ liệu trả về



Hình 2.2: Mô hình truy vấn trên dữ liệu mã được Hacigumus đề xuất [32]

thoả mãn truy vấn thì một tập hợp các chỉ mục được liên kết với bảng dữ liệu mã. Trong trường hợp này, máy chủ lưu trữ một bảng được mã hoá với một chỉ mục cho mỗi thuộc tính và các chỉ mục này cần phải xác định trước.

Để mô tả rõ hơn phương pháp của Hacigumus, luận án cho một CSDL rõ DB, với mỗi lược đồ $R(A_1, A_2, ..., A_n)$ trong DB được ánh xạ thành lược đồ $R^S(etuple, A_1^S, A_2^S, ..., A_n^S)$ trong CSDL mã hóa tương ứng. Ở đây, $etuple$ là một thuộc tính chứa bộ mã hoá mà giá trị thu được bằng cách sử dụng một hàm mã hoá E_k trên bộ dữ liệu rõ với k là khóa bí mật. Cụ thể, với bộ $t = (a_1, a_2, ..., a_n)$ trong R thì $etuple = E_k(t)$. Mỗi thuộc tính A_i^S là các chỉ mục tương ứng với các thuộc tính A_i của bộ rõ, và nó được dùng để thực hiện truy vấn trên máy chủ. Để có thể ánh xạ các chỉ mục A_i^S đến giá trị trong bản mã $etuple$ thì ta thường chia một khoảng thành các vùng liên tiếp, bằng nhau và không giao nhau gọi là bucket. Một bucket là một khoảng giá trị nằm trong giá trị thấp nhất $p.low$ và giá trị cao nhất $p.high$ của miền giá trị của thuộc tính. Với mỗi thuộc tính A_i, ta xác định các bucket là:

$$bucket(A_i) = \{p_1, p_2, ..., p_k\}$$

trong đó $p_i \subset [p.low, p.high], i \in [1, k]$

Cụ thể, giả sử giá trị của thuộc tính DIEMSV có miền giá trị từ [0,10], ta

Bảng 2.1: Bảng *DIEMSV* rõ và bảng $DIEMSV^S$ mã

DIEMSV

Tên SV	**Điểm**
Quang	7
Chiến	6
Mai	9
Bảo	8

$DIEMSV^S$

Etuple	**H**	**I**
C8dilQWWvc5dsqHU1w7WJ828	5	γ
mRSjcH01cIo2rB56ARMJg	3	α
FZpzMJd6VzkkY1LbPs8g	8	β
tNmYWHCJ6HSVD8qok6oaEA	3	γ

chia miền mày thành 5 phân hoạch như sau:

$bucket(DIEMSV) = \{[0, 2], (2, 4], (4, 6], (6, 8], (8, 10]\}$

Mỗi bucket được liên kết với một giá trị duy nhất và tập hợp các giá trị này là miền cho chỉ mục I tương ứng với A_i. Cho một bộ t^S trong R^S, giá trị của thuộc tính A_i^S trong t^S là một bucket. Điều quan trọng cần lưu ý là để bảo vệ bí mật dữ liệu tốt hơn, miền của chỉ mục A_i^S có thể không cùng thứ tự như là trong các thuộc tính bản rõ A_i.

Thuộc tính *Etuple* trong bảng 2.1 là giá trị mã của thuộc tính *Tên SV* và *Điểm*. Thuộc tính H và I là các chỉ số thu được bằng cách ánh xạ các giá trị của thuộc tính *Tên SV* và *Điểm* tương ứng của bảng *DIEMSV* vào các khoảng của bucket và được định danh thành các giá trị đại diện. Các giá trị I_1 và I_4 trong bảng $DIEMSV^S$ có cùng một bucket nên có giá trị bằng nhau và bằng γ. Để biết được các giá trị nằm trong khoảng nào thì Hacigumus dùng hàm $ident()$ để định danh và hàm $Map()$ để ánh xạ giá trị vào định danh đó.

Sau khi tạo ra được bảng $DIEMSV^S$, DO sẽ lưu trữ bảng mã này lên trên DSP. Quá trình xử lý một câu truy vấn Q sẽ thực hiện hai nhiệm vụ: (1) Máy khách tiếp nhận Q, chuyển đổi Q thành Q^S để có thể thực hiện truy vấn dữ liệu mã trên máy chủ. Máy khách gửi Q^S đến máy chủ. (2) Máy chủ thực thi Q^S và trả về các bản ghi thoả mãn điều kiện R^S cho máy khách (có thể trả về nhiều dữ liệu hơn yêu cầu của Q). Máy khách giải mã R^S, loại bỏ các dữ liệu không thoả mãn điều kiện Q và trả dữ liệu kết quả rõ theo yêu cầu người dùng.

Giả sử câu truy vấn Q có dạng:

"SELECT * FROM $DIEMSV$ WHERE Điểm = 7"

thì sẽ được chuyển thành Q^S là:

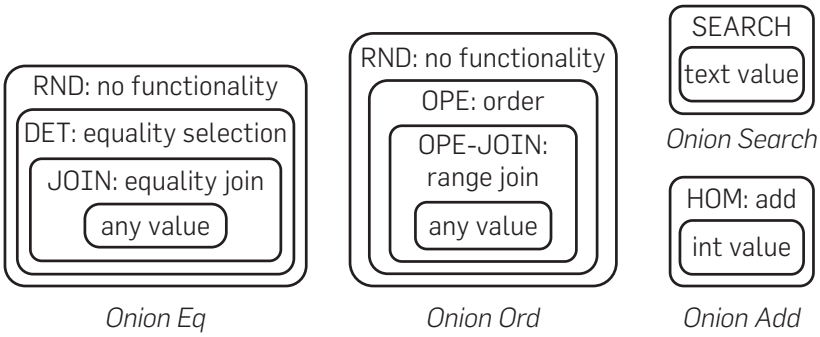
"SELECT etuple FROM $DIEMSV^S$ WHERE I = γ".

Sau khi thực hiện câu truy vấn Q^S, máy chủ DSP trả về cho người dùng với hai bản ghi là số 1 và 4 vì thỏa mãn điều kiện I = γ. Sau đó, phía người dùng giải mã các bản ghi này và thu được bản ghi rõ ban đầu và loại bỏ bản ghi số 4 vì không thỏa mãn điều kiện Điểm = 7". Kết quả cuối cùng người dùng thu được là bản ghi số 1, đây chính là kết quả của câu truy vấn Q trên CSDL mã.

Nhược điểm của phương pháp lập chỉ mục dựa trên bucket là phía máy chủ chỉ thực hiện các câu truy vấn với điều kiện bằng trong Q^S, các điều kiện này có thể được ánh xạ vào điều kiện tương đương trên chỉ số. Một điều kiện truy vấn trong Q có dạng $A_i = c$, trong đó c là một giá trị không đổi, thì điều kiện tương ứng trong Q^S trên chỉ mục I là $I = x$, trong đó x là bucket (một khoảng các giá trị). Trong trường hợp Q có điều kiện *Điểm = 7* thì điều kiện của Q^S được chuyển thành $I = \gamma$. Nhưng nếu Q có điều kiện khoảng (range quey) thì phương pháp này khó xử lý. Vì miền chỉ mục không nhất thiết bảo toàn thứ tự của miền bản rõ, một điều kiện phạm vi có dạng $A_i \leq c$, trong đó c là một giá trị không đổi, phải được ánh xạ thành một loạt các điều kiện bằng hoạt động trên chỉ số I có dạng $I = x_1$ hoặc $I = x_2$ hoặc $I = x_k$..., trong đó x_i là các giá trị liên quan đến bucket tương ứng với các giá trị bản rõ nhỏ hơn hoặc bằng c.

Như vậy nếu Q có điều kiện *Điểm* ≤ 8 thì điều kiện của Q^S được chuyển thành $I = \gamma$ hoặc $I = \alpha$ vì γ, α là các bucket đại diện cho *Điểm* ≤ 8. Trong trường hợp *p.high - p.low* là lớn thì ta có nhiều phân hoạch, do đó với điểu kiện khoảng thì sẽ là hợp của các phân hoạch sao cho thỏa mãn điền kiện truy vấn.

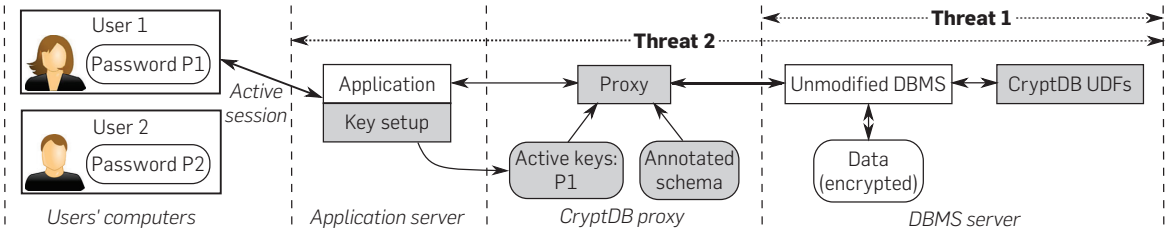
A. Popa và cộng sự [60] đề xuất một giải pháp truy vấn trên dữ liệu mã gọi là CryptDB, trong đó mô hình mã hoá theo phương pháp mã hóa từng ô dữ liệu và cách thức thực thi truy vấn trên CSDL mã bằng cách sử dụng một máy chủ làm trung gian gọi là CryptDB proxy. Với mã hóa theo từng ô dữ liệu thì CSDL vẫn đảm bảo được mối quan hệ giữa các bảng, nên có thể sử dụng các tính chất quan hệ của CSDL. CryptDB sử dụng nhiều thuật toán mã hóa khác nhau để phù hợp với từng loại câu truy vấn. Chính vì vậy, dữ liệu sau khi mã hóa sẽ tăng lên đáng kể do thêm nhiều cột mã hóa.



Hình 2.3: Mô hình mã hoá củ hành (Onion) do Popa đề xuất [60]

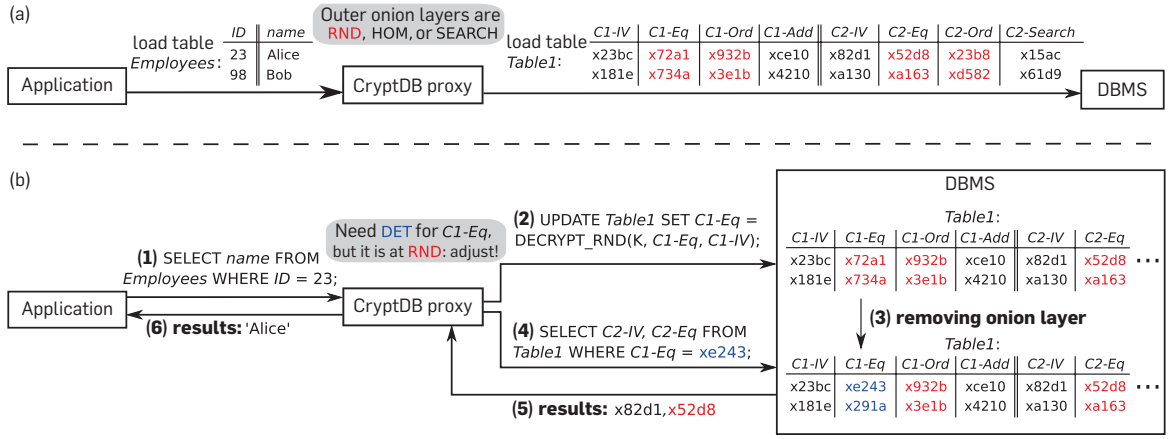
CryptDB sử dụng kết hợp hai kỹ thuật mã hóa là mã hóa nhận thức SQL (SQL-aware encryption - SQLAE) và mã hóa dựa trên truy vấn có thể điều chỉnh (adjustable query-based encryption -AQE). SQLAE sử dụng nhiều thuật toán mã hóa khác nhau như: Mã hóa ngẫu nhiên (Random - RND) để bảo mật tối đa của lớp mã hóa trong CryptDB, mã hóa tất định (Deterministic - DET) để thực hiện các phép bằng, GROUP BY, COUNT, DISTINCT..., mã hóa bảo toàn thứ tự (Order-preserving encryption - OPE) để thực hiện các phép toán ORDER BY, MIN, MAX, SORT..., mã hóa đồng cấu (Homomorphic encryption - HOM) để có thể tính toán trên dữ liệu mã mà không cần giải mã dữ liệu, mã hóa có thể tìm kiếm (Word search - SEARCH) để tìm kiếm trên chuỗi khi sử dụng phép toán LIKE. Chính vì vậy, SQLAE sử dụng cho hầu hết các truy vấn SQL được tạo thành từ một tập hợp các toán tử cơ bản như: So sánh bằng, so sánh thứ tự, tổng, và phép nối. Trong

khi đó, mục đích của AQE là điều chỉnh linh hoạt lớp mã hóa trên máy chủ DBMS. Ý tưởng của AQE là mã hóa từng mục dữ liệu trong một hoặc nhiều củ hành (Onion). Nghĩa là, mỗi giá trị mã hóa được nằm trong các lớp mã hóa mạnh hơn (hình 2.3). Mỗi lớp của củ hành cho phép thực hiện một số phép tính toán tương ứng với từng loại SQLAE.



Hình 2.4: Mô hình truy vấn trên dữ liệu mã do Popa đề xuất [60]

Cách thức thực hiện truy vấn trong CryptDB được mô tả như hình 2.5.



Hình 2.5: Ví dụ về (a) cách CryptDB biến đổi một lược đồ bảng và mã hóa CSDL và (b) truy vấn trên dữ liệu mã của CryptDB [60]

Một truy vấn trong CryptDB thực hiện qua bốn bước:

- Ứng dụng gởi truy vấn đến máy chủ trung gian và sẽ được proxy viết lại: nó sẽ ẩn danh tên bảng và cột, và sử dụng khóa MK mã hoá các thành phần cố định trong truy vấn với một lược đồ mã hóa thích hợp nhất cho các tính toán mong muốn. Proxy cũng thay thế một số phép toán với hàm do người dùng định nghĩa (User Defined Function - UDF).
- Proxy kiểm tra nếu server cần được cung cấp khoá để điều chỉnh các

lớp mã hóa trước khi thực hiện truy vấn, nếu như vậy, một truy vấn UPDATE tại server sẽ gọi một UDF để điều chỉnh lớp mã hóa của các cột cho thích hợp.

- Proxy sẽ gửi truy vấn được mã hóa đến server.
- Server trả về kết quả truy vấn mã hóa, proxy giải mã và trả kết quả về cho ứng dụng.

Li và cộng sự [50] đề xuất phương pháp hỗ trợ truy vấn mờ (fuzzy query) gọi là L-EncDB. Trong mô hình lưu trữ L-EncDB, Li đã mã hóa từng ký tự trong chuỗi. Giả sử chuỗi $t = t_1||t_2||...||t_n$, thì các từ $t_1, t_2, ..., t_n$ được thay thế bằng các giá trị mã của hàm FQE. Trong đó, hàm FQE được thực hiện bằng cách: Đầu tiên, duyệt các ký tự trong chuỗi và thêm vào các ký tự l giá trị 1, nghĩa là $str \leftarrow t_i||\{11...1\}$ và mã hóa chuỗi str thành chuỗi str' bằng mã hóa FPE, sau đó băm chuỗi str' thành một số nguyên và chuyển số nguyên này thành ký tự Unicode. Li gọi thuật toán mã từng ký tự là $gen(c)$. Khi đó giá trị mã của chuỗi t là $FQE_k(t) = gen(t_1)||gen(t_2)||...||gen(t_n)$. Để thực hiện truy vấn mờ, tầng ứng dụng của L-EncDB chuyển đổi giá trị rõ của từ khóa thành giá trị mã bằng hàm $FQE_k(x)$, sau đó đưa vào câu truy vấn. Giả sử câu truy vấn rõ là: "SELECT * FROM Table1 WHERE Field1 LIKE '%key1 %key2 %'" sẽ được chuyển thành: "SELECT * FROM Table1 WHERE Field1Extra LIKE '%$FQE_k(key1)$%$FQE_k(key2)$%'". Khi có kết quả trả về, tầng ứng dụng sẽ giải mã trả về kết quả rõ cho người dùng.

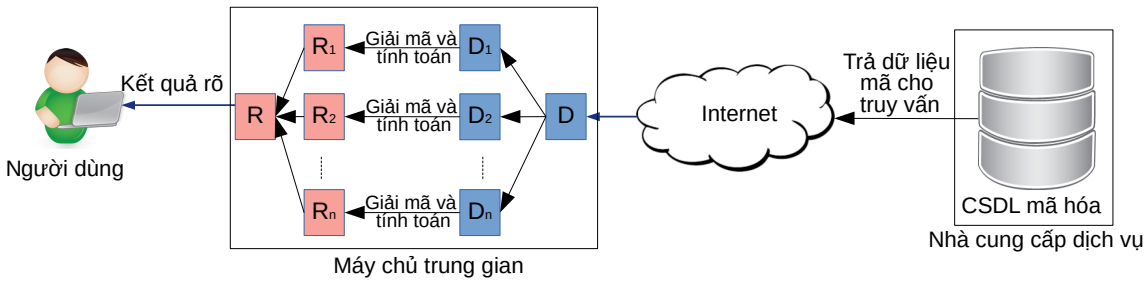
Nhận xét 2.1 *Mã hóa dữ liệu sẽ làm tăng sự phức tạp trong truy vấn dữ liệu, ảnh hưởng nhiều đến thời gian đáp ứng của CSDL. Với bảng kết quả trả về có n dòng và m cột thì phương pháp của Hacigumus tốn n lần giải mã, phương pháp của Popa là (n × m) lần giải mã, còn phương pháp của Li là (n × m × k) lần giải mã, với k là chiều dài trung bình của chuỗi. Nếu số lượng bản ghi trả về lớn thì thời gian tăng đáng kể so với phương pháp truy vấn trên dữ liệu rõ. Như vậy, muốn ứng dụng hiệu quả các phương pháp truy*

vấn trên dữ liệu mã vào thực tế để đảm bảo an toàn cho ODBS thì cần giảm thời gian truy vấn càng nhiều càng tốt.

2.2.2. Giảm thời gian khi truy vấn trên CSDL mã

Để giảm thời gian truy vấn trên CSDL, các nhà khoa học đã đề xuất những phương pháp khác nhau. Q.Zhang và cộng sự đề xuất công cụ Qscheduler [89] để truy vấn song song trên hệ thống CSDL. Tuy nhiên, giải pháp này thực hiện trên CSDL rõ và thực hiện song song các câu truy vấn cùng lúc truy xuất đến CSDL. Ying-Fu Huang [39] cũng đưa ra phương pháp truy vấn song song trên dữ liệu rõ và thực hiện các phép giao, nối, sắp xếp, nhóm,... của các bảng dữ liệu. Samraddhi Shastri [67] đưa ra giải pháp tăng tốc trên dữ liệu mã, tuy nhiên, phương pháp này thực hiện truy vấn song song tìm kiếm nhị phân trên dữ liệu mã. Nghĩa là, nếu muốn sử dụng phương pháp của Shastri thì các bản ghi dữ liệu phải được sắp xếp theo thứ tự.

Trong quá trình truy vấn trên dữ liệu mã, có nhiều tiến trình xử lý như: Giải mã kết quả rồi loại bỏ các bản ghi không phù hợp hoặc tính toán lại, giải mã kết quả sau truy vấn trả dữ liệu rõ cho người dùng. Các tiến trình này đều được xử lý trên tập dữ liệu quan hệ (bảng) của CSDL nên có thể dùng phương pháp xử lý song song để tính toán bằng cách chia quan hệ thành các tập con và xử lý đồng thời. Phương pháp này sẽ giảm đáng kể thời gian thực hiện truy vấn trên CSDL mã (hình 2.6).



Hình 2.6: Mô hình xử lý song song trên dữ liệu mã

Giả sử khi người dùng muốn hiển thị giá trị của bảng dữ liệu, hoặc xử lý

tính toán trên bản rõ của dữ liệu trả về thì người dùng phải giải mã từng bản ghi của dữ liệu. Lúc này, chúng ta chia nhỏ tập kết quả thành các tập con. Do các tập con có cùng cấu trúc nên ta có thể chia nhiều tiến trình để tính toán song song trên các tập con này. Việc thực hiện tính toán song song trên các tập con giống như giải quyết k dữ liệu trên x tiến trình của thiết bị có nhiều tiến trình tính toán. Thực hiện công việc f với đầu vào $S = \{Input_i\}_{(i=1...k)}$ có đầu ra $R = \{f(Input_i)\}_{(i=1...k)}$, được tiến hành theo 3 giai đoạn sau:

- Giai đoạn 1: Tách tập $S = \{Input_i\}_{(i=1...k)}$ thành x tập con:
  $S_j = \{Input_l\}_{(l=((j-1)[\frac{k}{x}]+1)...j[\frac{k}{x}])}$ $(j = 1...x)$
- Giai đoạn 2: Tại mỗi tiến trình j thực hiện tính:
  $R_j = \{f(S_j)\}$
  Các tiến trình j được thực hiện song song. Trong các tiến trình này, hàm f bao gồm các công việc như: Giải mã, tính toán trên các giá trị rõ, xác thực dữ liệu...
- Giai đoạn 3: Thực hiện gộp các tập dữ liệu trả về:
  $R = R_1 \cup R_2 ... \cup R_x$

Tương ứng với 3 giai đoạn trên, thuật toán 2.1 mô tả quá trình thực hiện truy xuất dữ liệu mã với tính toán song song có số luồng cụ thể là $x = 2$. Thuật toán bao gồm hàm f và hàm $main()$:

- Hàm $f(datatable, start, end, table)$: Thực hiện giải mã, tính toán (sum, count...) trên các giá trị rõ và trả về bảng dữ liệu rõ.
  - Tham số $datatable$: Có kiểu dữ liệu Datatable, tham số này là bảng dữ liệu mã hoá trả về khi truy vấn và nó tương ứng với tập dữ liệu đầu vào S.
  - Tham số $start, end$: Có kiểu dữ liệu số nguyên, là vị trí bản ghi bắt đầu và kết thúc trong bảng $datatable$.

Thuật toán 2.1: Thuật toán truy vấn CSDL mã song song

Input: Bảng dữ liệu mã D

Output: Bảng dữ liệu rõ R

```
1  Function f(datatable, start, end, table):
2      for i in range(start, end+1) do
3          row = datatable[i]
4          r = ∅
5          for j in range(0,datatable.columncount()) do
6              r.append(D_k(row[j]))
7              //caculate in D_k(row[j])
8          end
9          result.addrow(r)
10         table.put(result)
11     end
12 End Function
13 Function Main():
14     D ← Excute(SELECT c_1, ..., c_n FROM t)
15     //D = D_1 ∪ D_2
16     total = D.rowcount()
17     table = Queue()
18     p1 = Process(target=f, args=(D, 1, int(total/2), table)).start()
19     p2 = Process(target=f, args=(D, int(total/2)+1, total, table)).start()
20     p1.join(); p2.join()
21     //Get result
22     while not table.empty() do
23         result = table.get()
24         print result
25     end
26 End Function
```

  - Tham số *table*: Có kiểu dữ liệu Datatable, là kết quả bảng dữ liệu rõ sau khi giải mã, tính toán.

- Hàm $Main()$: Thực hiện chia nhỏ bảng dữ liệu D (là bảng dữ liệu mã khi thực hiện câu truy vấn trên dữ liệu mã) và thực hiện chia thành 2 luồng $p1, p2$ thực hiện song song giải mã, tính toán (nếu có). Kết quả trả về là bảng dữ liệu rõ $result$.

2.3. Lược đồ xác thực lô dựa trên hai bài toán khó

Trong mục này, luận án đề xuất hai lược đồ xác thực lô Rabin-Schnorr và RSA-Schnorr dựa trên độ khó của hai bài toán là IFP và DLP. Lược đồ xác thực lô dựa trên hai bài toán khó sẽ tăng tính an toàn của chữ ký. Giải đồng thời hai bài toán khó sẽ khó khăn hơn giải một bài toán khó. Việc kết hợp hai bài toán khó không có nghĩa là an toàn tuyệt đối trong mọi trường hợp. Kết hợp hai bài toán khó có ý nghĩa trong trường hợp tại một thời điểm, giả sử một trong hai bài toán bị phá vỡ thì thuật toán vẫn an toàn [42].

2.3.1. Cơ sở lý thuyết

Cơ sở lý thuyết trình bày trong mục này được trích dẫn từ tài liệu [20] [35] [46].

Một số kết quả:

Theo phương pháp Karatsuba thì:

$$t_M(N) = O(N^{ln3/ln2}) \text{ (phép toán bit)} \tag{2.1}$$

Theo phương pháp Mongomegy hoặc Barrett thì:

$$t_{Red}(N) = 2.t_M(N) \tag{2.2}$$

$$t_{gcd}(N) = t_{Jacobi}(N) = O(N^2) \text{ (phép toán bit)} \tag{2.3}$$

Một số quy đổi:

$$t_m(N) = 3.t_M(N) \tag{2.4}$$

$$t_m(L) = 3^{log_2(L/N)} \times t_m(N) \tag{2.5}$$

Đặc biệt

$$t_m(2N) \approx 3 \times t_m(N) \tag{2.6}$$

Theo phương pháp bình phương-nhân thì

$$t_{exp}(N, L) = 1.5 \times N \times t_m(L) \tag{2.7}$$

Theo phương pháp nhị phân thì

$$t_{gcd}(N) = t_{Jacobi}(N) \approx \sqrt{N}.t_M(N) \tag{2.8}$$

Công thức Garner

Cho $n = p.q$ là tích của hai số nguyên tố khác nhau. Khi đó với mỗi $a \in \mathbb{Z}_n$, ký hiệu $a_p = a \bmod p, a_q = a \bmod q$, thì:

$$a = (c(a_p - a_q) + a_q) \bmod n \tag{2.9}$$

$$\text{với } c = q.(q^{-1} \bmod p) \tag{2.10}$$

giá trị a được xác định từ (a_p, a_q) được ký hiệu là $CRT(a_p, a_q)$.

Bảng kích thước tương đương về độ an toàn của các tham số RSA và logarit rời rạc (DL) (bảng 2.2)

Tham số L = len(p) đối với DL và L = len(n) đối với RSA.

Tham số N = len(q) đối với DL.

Bảng 2.2: Kích thước tương đương về độ an toàn của các tham số RSA và DL

N	160	224	256	384	512
L	1024	2048	3072	8192	15360

2.3.2. Tham số miền

Hai lược đồ Rabin-Schnorr và RSA-Schnorr có chung bộ tham số miền là:

- Kích thước modulo L.
- Tập các dữ liệu $M = \{0,1\}^{\infty}$.
- Tập các chữ ký $S = \mathbb{N} \times \mathbb{N}$.
- Hàm băm H:$\{0,1\}^{\infty} \to \{0,1\}^{h}$. Giá trị h được gọi là độ dài hàm băm.

Mỗi thành viên tương ứng với bộ các tham số sau:

- Số nguyên tố p có dạng $p = 2.n + 1$ với len(p) = L.
- Hợp số n có dạng $n = q.q'$ là hai số nguyên tố lẻ khác nhau sao cho việc phân tích n ra thừa số là khó (với lược đồ Rabin-Schnorr thì $q, q' \equiv 3 \pmod 4$).
- Phần tử sinh g có cấp bằng n.
- Tham số mật $x \in \langle g \rangle$ (nhóm cyclic sinh bởi g trong GF(p)) và tham số công khai $y = g^x \bmod p$ (với lược đồ RSA-Schnorr thêm số mũ mật d, số mũ công khai e thỏa mãn $e.d \bmod \phi(n) = 1$ với ϕ là số Euler).

Khi đó khóa ký và khóa kiểm tra chữ ký:

- Trong lược đồ Rabin-Schnorr lần lượt là: (p, n, q, q', x) và (p, n, y).
- Trong lược đồ RSA-Schnorr lần lượt là: (p, n, q, q', d, x) và (p, n, e, y).

2.3.3. Lược đồ xác thực lô Rabin-Schnorr

Trong mục này, luận án đề xuất lược đồ xác thực lô dựa trên độ khó của hai bài toán là phân tích thừa số (Rabin) và logarit rời rạc (Schnorr). Để thiết kế lược đồ này thì luận án phải tính được giá trị căn bậc hai modulo từ số chính phương modulo.

Lược đồ ký, kiểm tra chữ ký và xác thực lô lần lượt là các thuật toán 2.2, 2.3, 2.4.

Thuật toán 2.2: Thuật toán tạo chữ ký $S(m)$ Rabin-Schnorr

Input: $m \in M$

Output: $(r, s) \in S$

1 $t \in_R (0, n)$

2 $r \leftarrow g^t \bmod p$

3 $a \leftarrow (t - H(m||r)x) \bmod n$

4 **if** $((\frac{a}{q}) = -1)$ or $((\frac{a}{q'}) = -1)$ **then** goto 1

5 $s_q \leftarrow a^{(q+1)/4} \bmod q$; $s'_q \leftarrow a^{(q'+1)/4} \bmod q'$

6 $s \leftarrow CRT(s_q, s_{q'})$

7 **return** (r, s)

Thuật toán 2.3: Thuật toán kiểm tra chữ ký Rabin-Schnorr

Input: $(m, (r, s)) \in M \times S$

Output: *"Accept"* nếu chữ ký là hợp lệ và *"Reject"* trong trường hợp ngược lại

1 $a = s^2 \bmod n$

2 $r' = g^a y^{H(m||r)} \bmod p$

3 **if** $r' = r$ **then return** *"Accept"*

4 **else return** *"Reject"*

Thuật toán 2.4: Thuật toán $V(\sigma_i)$ Rabin-Schnorr xác thực k chữ ký $\sigma_i(r_i, s_i)$ cho k dữ liệu $m_i, i = 1, 2, ..., k$, được ký bởi cùng một người ký

Input: Dữ liệu m_i, k chữ ký $\sigma_i(r_i, s_i)$, $1 \leq i \leq k$

Output: *"Accept"* nếu k chữ ký là hợp lệ và *"Reject"* trong trường hợp ngược lại

1 $a_i = s_i^2 \bmod n$

2 $u = \sum_{i=1}^{k} a_i \bmod p$

3 $v = \sum_{i=1}^{k} H(m_i||r_i) \bmod p$

4 **if** $(\prod_{i=1}^{k} r_i = g^u y^v \bmod p)$ (2.11) **then return** *"Accept"*

5 **else return** *"Reject"*

Mệnh đề 2.1 *Chữ ký $\sigma(r,s)$ tương ứng với dữ liệu m được ký bởi thuật toán 2.2 là chữ ký số hợp lệ.*

Chứng minh.
Ta có: $s = CRT(s_q, s_{q'}) = CRT(a^{\frac{q+1}{4}} \bmod q,\, a^{\frac{q'+1}{4}} \bmod q')$.
Do đó:

$$\begin{aligned} s^2 \bmod n &= CRT(a^{\frac{q+1}{2}} \bmod q,\ a^{\frac{q'+1}{2}} \bmod q') \\ &= CRT(a^{\frac{q-1}{2}+1} \bmod q,\ a^{\frac{q'-1}{2}+1} \bmod q') \\ &= CRT((\frac{a}{q})a \bmod q,\ (\frac{a}{q'})a \bmod q') \end{aligned}$$

Vì $((\frac{a}{q}) = 1)$ và $((\frac{a}{q'}) = 1)$ nên $s^2 \bmod n = a$
Tính: $r = g^t \bmod p = g^{a+H(m||r)x} \bmod p = g^a g^{H(m||r)x} \bmod p$
$= g^a y^{H(m||r)} \bmod p = r'$

Do đó chữ ký $\sigma(r,s)$ là chữ ký hợp lệ.

Mệnh đề 2.2 *Các chữ ký $(\sigma_1, \sigma_2, ..., \sigma_k)$ được ký bởi thuật toán 2.2 là k chữ ký số hợp lệ thì biểu thức* (2.11) *đúng.*

Chứng minh. Giả sử có k chữ ký (r_i, s_i) cho k dữ liệu $m_i, i = 1, 2, ..., k$, được ký bởi cùng một người ký sử dụng khoá bí mật (x, q, q'). Để xác định k chữ ký này là đúng thì từng chữ ký phải thoả mãn biểu thức: $r = g^a y^{H(m||r)} \bmod p$ với $a = s^2 \bmod n$.

Như vậy:

$$\begin{cases} r_1 = g^{a_1} y^{H(m_1||r_1)} \bmod p \\ r_2 = g^{a_2} y^{H(m_2||r_2)} \bmod p \\ \qquad\qquad \vdots \\ r_k = g^{a_k} y^{H(m_k||r_k)} \bmod p \end{cases}$$

Do đó: $\prod_{i=1}^{k} r_i = g^{a_1} y^{H(m_1||r_1)} \cdots g^{a_k} y^{H(m_k||r_k)} \bmod p$
$= g^{(a_1+\cdots+a_k)} y^{H(m_1||r_1)+\cdots+H(m_k||r_k)} \bmod p$
$= g^{\sum_{i=1}^{k} a_i} y^{\sum_{i=1}^{k} H(m_i||r_i)} \bmod p$

2.3.4. Lược đồ xác thực lô RSA-Schnorr

Lược đồ RSA-Schnorr dựa trên hai bài toán khó là bài toán IFP của RSA và DLP của Schnorr. Để phá vỡ lược đồ chữ ký này yêu cầu giải quyết đồng thời hai bài toán khó là tính toán logarit rời rạc trên $\mathbb{Z}_p$ và phân tích thừa số n. Lược đồ ký, kiểm tra chữ ký và xác thực lô lần lượt là thuật toán 2.5, 2.6, 2.7.

Thuật toán 2.5: Thuật toán tạo chữ ký RSA-Schnorr

Input: Dữ liệu $m \in M$

Output: Chữ ký $\sigma \in S$

1 $t \in_R (0, n)$

2 $r = g^t \bmod p$

3 $s = (t - H(m||r)x)^d \bmod n$

4 **return** (r, s)

Thuật toán 2.6: Thuật toán kiểm tra chữ ký RSA-Schnorr

Input: $(m, (r, s)) \in M \times S$

Output: *"Accept"* nếu chữ ký là hợp lệ và *"Reject"* trong trường hợp ngược lại

1 $a = s^e \bmod n$

2 $r' = g^a y^{H(m||r)} \bmod p$

3 **if** $(r = r')$ **then return** *"Accept"*

4 **else return** *"Reject"*

Mệnh đề 2.3 *Chữ ký $\sigma(r, s)$ tương ứng với dữ liệu m được ký bởi thuật toán 2.5 là chữ ký số hợp lệ.*

Chứng minh.

Tính: $a = s^e \bmod n = (t - H(m||r)x)^{ed} \bmod n = (t - H(m||r)x) \bmod n$ (do $ed = 1 \bmod \phi(n)$).

$\Rightarrow t = (a + H(m||r)x) \bmod n.$

Thuật toán 2.7: Thuật toán $V(\sigma_i)$ RSA-Schnorr xác thực k chữ ký $\sigma_i(r_i, s_i)$ cho k dữ liệu $m_i, i = 1, 2, ..., k$, được ký bởi cùng một người ký

Input: Dữ liệu m_i, k chữ ký $\sigma_i(r_i, s_i)$, $1 \leq i \leq k$

Output: *"Accept"* nếu k chữ ký là hợp lệ và *"Reject"* trong trường hợp ngược lại

1 $a_i = s_i^e \bmod n$

2 $u = \sum_{i=1}^{k} a_i \bmod p$

3 $v = \sum_{i=1}^{k} H(m_i||r_i) \bmod p$

4 **if** $(\prod_{i=1}^{k} r_i = g^u y^v \bmod p)$ (2.12) **then return** *"Accept"*

5 **else return** *"Reject"*

Ta có: $r = g^t \bmod p = g^{a+H(m||r)x} \bmod p = g^a g^{H(m||r)x} \bmod p$
$= g^a y^{H(m||r)} \bmod p = r'$

Do đó chữ ký $\sigma(r, s)$ là chữ ký hợp lệ.

Mệnh đề 2.4 *Các chữ ký $(\sigma_1, \sigma_2, ..., \sigma_k)$ được ký bởi thuật toán 2.5 là k chữ ký số hợp lệ thì biểu thức* (2.12) *đúng.*

Chứng minh. Theo thuật toán 2.6, để xác định k chữ ký $\sigma_i(r_i, s_i)$ là đúng thì từng chữ ký phải thoả mãn biểu thức: $r = g^a y^{H(m||r)} \bmod p$. Trong đó: $a = s^e \bmod n$.

Do đó:

$$\begin{cases} r_1 = g^{a_1} y^{H(m_1||r_1)} \bmod p \\ r_2 = g^{a_2} y^{H(m_2||r_2)} \bmod p \\ \qquad \vdots \\ r_k = g^{a_k} y^{H(m_k||r_k)} \bmod p \end{cases} \tag{2.13}$$

Từ (2.13), suy ra: $\prod_{i=1}^{k} r_i = g^{a_1} y^{H(m_1||r_1)} \cdots g^{a_k} y^{H(m_k||r_k)} \bmod p$
$= g^{a_1+\cdots+a_k} y^{H(m_1||r_1)+\cdots+H(m_k||r_k)} \bmod p$
$= g^{\sum_{i=1}^{k} a_i} y^{\sum_{i=1}^{k} H(m_i||r_i)} \bmod p$

2.3.5. Nâng cao tính an toàn và hiệu quả cho hai lược đồ Rabin-Schnorr và RSA-Schnorr

2.3.5.1. Nâng cao tính an toàn cho lược đồ Rabin-Schnorr và RSA-Schnorr

Sau khi đề xuất lược đồ 2.2, luận án nhận thấy lược đồ này còn những vấn đề như sau:

- *Thứ nhất.* Với n là tích của hai số nguyên tố khác nhau q và q' ta có: $gcd(a, n) \neq 1 \Leftrightarrow (a \bmod q = 0)$ hoặc $(a \bmod q' = 0)$. Vì vậy, nếu $gcd(a, n) \neq 1$ thì lược đồ sẽ giảm độ an toàn do chỉ còn dựa vào bài toán logarit rời rạc.

- *Thứ hai.* Nếu a là một thặng dư bậc hai thì sẽ có hai nghiệm là s và $-s$ thoả mãn phương trình $a = s^2 \bmod n$. Nói cách khác, nếu (r, s) được chấp nhận bởi thuật toán 2.3 thì $(r, -s)$ cũng được chấp nhận. Do đó, để tăng tính an toàn, luận án chỉ cần giữ lại một giá trị nghiệm s trong thuật toán tạo chữ ký.

Để nâng cao tính an toàn cho các thuật toán đề xuất, luận án cải tiến, bổ sung tại bước 3 và bước 6 của thuật toán 2.2 như sau:

3. $a = (t - H(m||r)x) \bmod n$; **if** $(gcd(a, n) \neq 1)$ **then** goto 1.

6. $s \leftarrow CRT(s_q, s_{q'})$; **if** $(s \geq n/2)$ **then** $s = n - s$.

Tương ứng với việc bổ sung ở bước 6 trong thuật toán 2.2 thì thuật toán 2.3 và 2.4 phải thêm điều kiện $s < n/2$.

Lược đồ RSA-Schnorr cũng có vấn đề như vấn đề thứ nhất của lược đồ Rabin-Schnorr nên cũng thêm điều kiện $gcd(s, n) \neq 1$ ở bước 3 trong thuật toán 2.5.

2.3.5.2. Nâng cao tính hiệu quả cho lược đồ Rabin-Schnorr và RSA-Schnorr

Xuất phát từ lược đồ ElGamal, các lược đồ phát triển của nó như DSA, GOST, Schnorr,... nhằm đạt được tính hiệu quả cao đều có chung một giải pháp đó là đưa thêm vào tham số miền N vừa đủ lớn sao cho bài toán tìm

các logarit trong nhóm $\langle g \rangle$ có kích thước N-bít là "khó".
Luận án có một số khuyến nghị như sau:

- *Thứ nhất.* Đưa vào tham số miền N cho các lược đồ cải tiến nếu nó được lấy tương ứng với L theo bảng 2.2, khi này bước 1 của các thuật toán ký nên chỉ là $t \in_R (2^{N-1}, 2^N)$. Tương tự tham số mật x cũng chỉ cần là $x \in_R (2^{N-1}, 2^N)$. Tham số $h = N$.

- *Thứ hai.* Số mũ công khai e nên có dạng $2^k + 1$, thậm chí lấy cố định là $2^{16} + 1$ như khuyến cáo đối với lược đồ RSA.

- *Thứ ba.* Đối với lược đồ Rabin-Schnorr, người ký tính sẵn và lưu như một tham số mật giá trị $c = q.(q^{-1} \bmod p)$. Khi này chi phí cho việc tính $CRT(x, y)$ chỉ còn là $t_m(L)$.

□ **Các tham số** Các tham số cho lược đồ Rabin-Schnorr và RSA-Schnorr giống như nêu trong mục 2.3.2 với một số bổ sung sau:

- Hàm tóm lược H:$\{0,1\}^\infty \to \{0,1\}^N$.
- Tham số mật $x \in_R (2^{N-1}, 2^N)$.
- Số mũ công khai $e = 2^{16} + 1$ dùng cho RSA-Schnorr.
- Tham số mật của người ký $c = q.(q^{-1} \bmod p)$ dùng cho Rabin-Schnorr.

Khi đó khóa ký và khóa kiểm tra chữ ký:

- Trong lược đồ Rabin-Schnorr lần lượt là: (p, n, q, q', x, c) và (p, n, y).
- Trong lược đồ RSA-Schnorr lần lượt là: (p, n, q, q', d, x) và (p, n, e, y).

□ **Lược đồ Rabin-Schnorr cải tiến:** Thuật toán 2.8, 2.9, 2.10.
□ **Lược đồ RSA-Schnorr cải tiến:** Thuật toán 2.11.

Thuật toán 2.8: Thuật toán tạo chữ ký Rabin-Schnorr cải tiến

Input: $m \in M$

Output: $(r, s) \in S$

1 $t \in_R (2^{N-1}, 2^N)$

2 $r \leftarrow g^t \bmod p$

3 $a \leftarrow (t - H(m||r)x) \bmod n$

4 $a_q \leftarrow a \bmod q$; $a'_q \leftarrow a \bmod q'$

5 **if** $((a_q = 0)$ or $(a_{q'} = 0))$ **then** goto 1. (thay cho $\gcd(a, n) \neq 1$)

6 **if** $((\frac{a}{q}) = -1)$ or $((\frac{a}{q'}) = -1)$ **then** goto 1

7 $s_q \leftarrow (a_q)^{(q+1)/4} \bmod q$; $s'_q \leftarrow (a_{q'})^{(q'+1)/4} \bmod q'$

8 $s \leftarrow (c.(s_q - s_{q'}) + s_{q'}) \bmod n$

9 **if** $(s \geq n/2)$ **then** $s \leftarrow n - s$

10 **return** (r, s)

Thuật toán 2.9: Thuật toán kiểm tra chữ ký Rabin-Schnorr cải tiến

Input: $(m, (r, s)) \in M \times S$

Output: *"Accept"* nếu chữ ký là hợp lệ và *"Reject"* trong trường hợp ngược lại

1 **if** $(s \geq n/2)$ **then return** *"Reject"*

2 $a \leftarrow s^2 \bmod n$

3 $r' \leftarrow g^a.y^{H(m||r)} \bmod p$

4 **if** $(r = r')$ **then return** *"Accept"*

5 **else return** *"Reject"*

2.4. Xác thực dữ liệu mã hóa thuê ngoài

Trong mục này, luận án đề xuất mô hình xác thực dữ liệu mã trả về từ DSP là đúng đắn. Khi trả về dữ liệu truy vấn, DSP kèm theo các chữ ký của DO tạo ra trước khi lưu trữ, máy chủ DO sẽ tiến hành xác thực chữ ký của các dữ liệu tương ứng, sau đó giải mã và trả kết quả rõ cho người dùng.

Thuật toán 2.10: Thuật toán $V(\sigma_i)$ Rabin-Schnorr cải tiến

Input: Dữ liệu m_i, k chữ ký $\sigma_i(r_i, s_i)$, $1 \leq i \leq k$

Output: *"Accept"* nếu k chữ ký là hợp lệ và *"Reject"* trong trường hợp ngược lại

1 **if** $(s_i \geq n/2)$ **then return** *"Reject"*

2 $a_i = s_i^2 \bmod n$

3 $u = \sum_{i=1}^{k} a_i \bmod p$

4 $v = \sum_{i=1}^{k} H(m_i||r_i) \bmod p$

5 **if** $(\prod_{i=1}^{k} r_i = g^u y^v \bmod p)$ **then return** *"Accept"*

6 **else return** *"Reject"*

Thuật toán 2.11: Thuật toán tạo chữ ký RSA-Schnorr cải tiến

Input: $m \in M$

Output: $(r, s) \in S$

1 $t \in_R (2^{N-1}, 2^N)$

2 $r \leftarrow g^t \bmod p$

3 $a \leftarrow (t - H(m||r)x) \bmod n$

4 $a_q \leftarrow a \bmod q$; $a_{q'} \leftarrow a \bmod q'$

5 **if** $((a_q = 0)$ or $(a_{q'} = 0))$ **then** goto 1. (thay cho $\gcd(a, n) \neq 1$)

6 $s_q \leftarrow (a_q)^{d \bmod q} \bmod q$; $s_{q'} \leftarrow (a_{q'})^{d \bmod q'} \bmod q'$

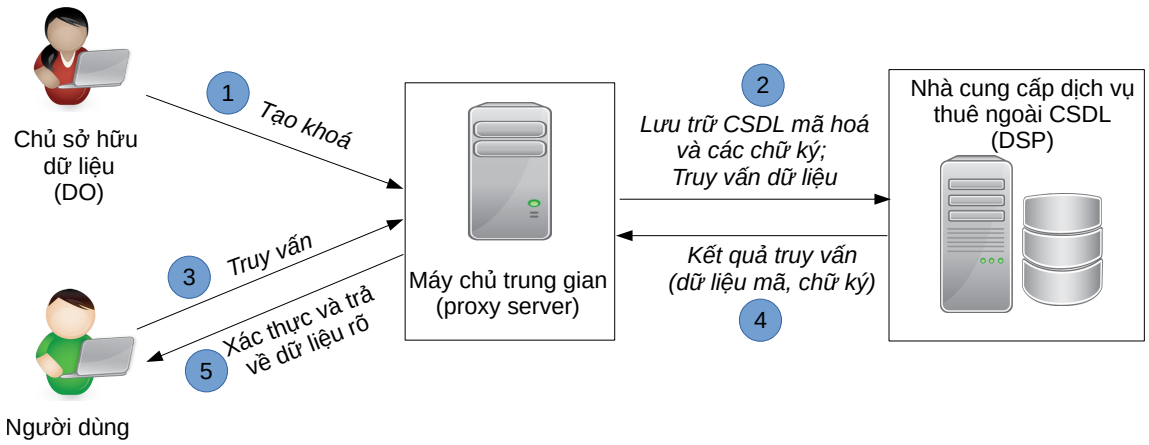
7 $s \leftarrow (c.(s_q - s_{q'}) + s_{q'}) \bmod n$

8 **return** (r, s)

2.4.1. Mô hình xác thực

Trong phạm vi nghiên cứu của mình, luận án sử dụng thêm một máy chủ trung gian để lưu trữ các dữ liệu hỗ trợ và xử lý tính toán. Quá trình khai thác CSDL trên ODBS có hai tương tác chủ yếu là: Tương tác người dùng - DSP (thông qua máy chủ trung gian của DO) và tương tác DO - DSP. Tương tác giữa người dùng và DSP bao gồm một số các truy vấn liên quan đến việc khai thác CSDL, còn tương tác giữa DO và DSP bao gồm các truy vấn liên

quan đến quản lý dữ liệu. Mô hình quá trình thực hiện xác thực dữ liệu mã được mô tả như hình 2.7. Trong đó, bước 1,2 là tương tác của DO - DSP;



Hình 2.7: Mô hình xác thực dữ liệu mã khi truy vấn ODBS

Bước 3,4,5 là tương tác của người dùng - DSP. Cụ thể quá trình tạo và lưu trữ CSDL lên DSP được DO thực hiện như sau:

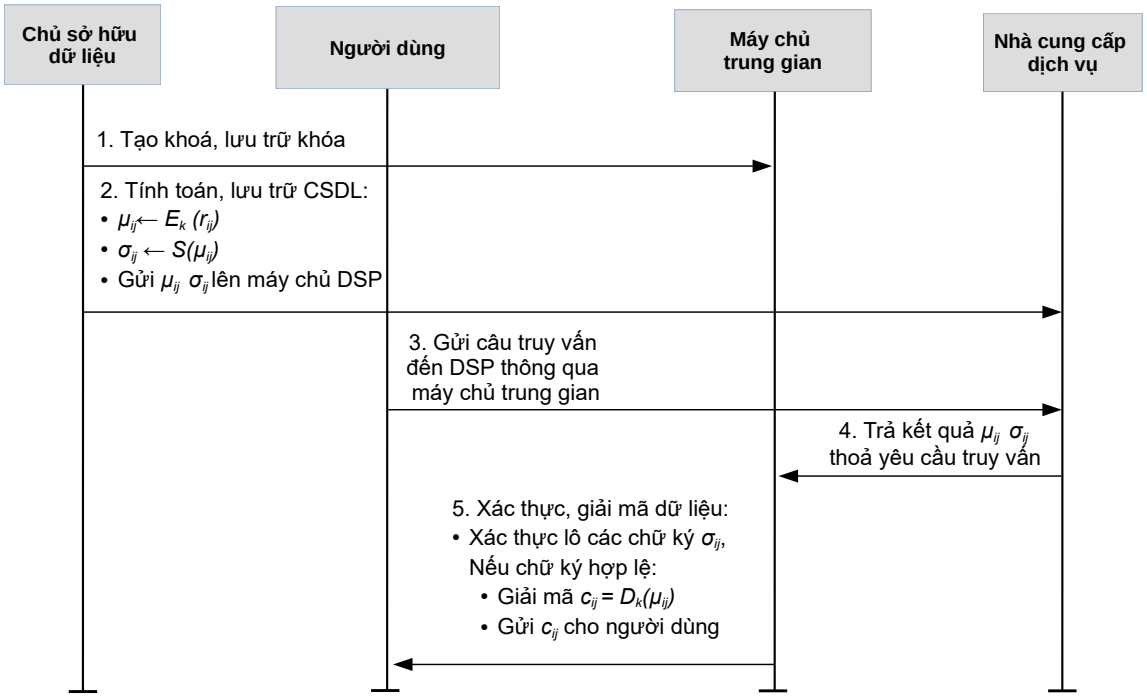
- Tạo các khoá: Khoá mã để mã hoá dữ liệu, khoá bí mật để tạo chữ ký và khoá công khai để xác thực chữ ký.
- Mã hoá dữ liệu và tạo chữ ký trên dữ liệu mã, sau đó lưu trữ dữ liệu đã mã hoá và chữ ký lên DSP.

Việc khai thác CSDL của người dùng được tiến hành như sau:

- Người dùng gửi câu truy vấn đến DSP thông qua máy chủ của DO để lấy dữ liệu theo nhu cầu. DO xác định các bảng dữ liệu liên quan đến việc trả lời các truy vấn của người dùng, DO gửi yêu cầu đến DSP để lấy các bảng này.
- Máy chủ DSP trả dữ liệu thoả điều kiện truy vấn kèm theo các chữ ký của dữ liệu về cho máy chủ trung gian.
- Máy chủ trung gian tiến hành xác thực dữ liệu, nếu dữ liệu đó hợp lệ sẽ tiến hành giải mã và trả về dữ liệu rõ cho người dùng.

2.4.2. Quá trình hoạt động

Quá trình xác thực CSDL mã thuê ngoài thực hiện qua ba giai đoạn: Giai đoạn tạo khoá (bước 1): Chủ sở hữu dữ liệu thực hiện chọn khoá để sử dụng trong toàn bộ quá trình tạo và xác thực dữ liệu. Việc tạo khoá được xử lý ngoài phạm vi của DSP và do DO quản lý; Giai đoạn lưu trữ ODBS (bước 2): thực hiện mã hoá dữ liệu, tạo chữ ký cho các dữ liệu mã và lưu trữ lên ODBS. Giai đoạn xác thực dữ liệu truy vấn (bước 3, 4, 5): Xác thực lô các chữ ký và giải mã các dữ liệu trả về. Nếu tất cả chữ ký là hợp lệ thì trả kết quả rõ cho người dùng. Nội dung các bước thực hiện xác thực dữ liệu mã với trường hợp ký trên bản mã khi truy vấn ODBS như hình 2.8.



Hình 2.8: Chi tiết các bước xác thực dữ liệu mã khi truy vấn ODBS

Thời gian thực hiện xác thực của ODBS phụ thuộc chính vào xác thực lô của lược đồ đề xuất. Theo đánh giá ban đầu, thuật toán RSA-Schorr cần tính mũ s^e khi xác thực nên sẽ có thời gian lớn hơn so với thuật toán Rabin-Schnorr do chỉ cần phép nhân $s.s$. Việc đánh giá độ phức tạp tính toán và thời gian thực hiện cụ thể ở phần 2.5.2.2. Vì vậy, luận án chọn thuật toán

Rabin-Schnorr vào việc xác thực CSDL mã trên ODBS.

Các tham số dùng cho hệ thống: Như mục 2.3.5.2, ngoài ra cần thêm khoá bí mật k để mã/giải mã dữ liệu. Khoá k được quản lý bởi DO. Khi đó khóa ký và khóa kiểm tra chữ ký lần lượt là: (p, n, q, q', x, c, k) và (p, n, y).

Giả sử với một CSDL D chứa nhiều bảng. Bảng T có m_T cột chứa n_T bản ghi $r = (r_{i1}, r_{i2}..., r_{im_T})$, với r_{ij} là dữ liệu tại dòng thứ i và cột thứ $j(1 \leq i \leq n_T, 1 \leq j \leq m_T)$. Để tránh trường hợp Adv hoán đổi giá trị giữa các bản ghi, mỗi bảng cần thêm một trường dữ liệu $AutoNum$ (kiểu số tự động tăng). Việc thêm trường $AutoNum$ không ảnh hưởng đến cấu trúc bảng của CSDL ban đầu. Khi truy vấn dữ liệu, câu lệnh SQL sẽ được máy chủ trung gian viết lại để bắt buộc DSP trả về trường $AutoNum$. Sau đó, máy chủ trung gian tiến hành xác thực và giải mã dữ liệu trả về. Khi người dùng gửi câu truy vấn đến DSP thông qua máy chủ proxy thì máy chủ DSP trả dữ liệu $T_r = \{AutoNum, \mu_{ij}, \sigma_{ij} | i = 1, 2, ...h_T; j = 1, 2, \ldots, k_T\}$ về máy chủ trung gian.

Các giai đoạn hoạt động xác thực ODBS được thực hiện lần lượt như thuật toán 2.12, 2.13.

2.4.3. Một số trường hợp truy vấn CSDL

2.4.3.1. Cập nhập dữ liệu

Các bản ghi của bảng T chứa dữ liệu mã và chữ ký. Những bản ghi này không phụ thuộc lẫn nhau nên khi thực hiện thao tác thêm, sửa, xoá thì dữ liệu được xử lý độc lập và cập nhập vào CSDL mà không cần phải tính toán, xây dựng lại bản ghi khác như các cấu trúc xác thực ADS.

Khi lưu trữ lên máy chủ DSP, bảng T có dạng $T = \{\mu_{ij}, \sigma_{ij} | i = 1...n_T, j = 1...m_T\}$. Giả sử muốn thêm bản ghi $r' = (r'_1, r'_2..., r'_{m_T})$ vào bảng T đầu tiên ta tính $\mu'_j = \{E_k(r'_j) | j = 1...m_T\}, \sigma'_j = \{S_x(\mu'_j) | j = 1...m_T\}$ sau đó thực hiện câu lệnh truy vấn "INSERT INTO T VALUES($\mu'_1, \sigma'_1, \mu'_2, \sigma'_2, ..., \mu'_{m_T}, \sigma'_{m_T}$);".

Khi thực hiện sửa dữ liệu thì người dùng tính toán các giá trị mã và chữ

Thuật toán 2.12: Thuật toán lưu CSDL mã

Input: Bảng T trong CSDL, khoá bí mật

Output: Lưu bảng dữ liệu mã kèm chữ ký lên máy chủ DSP

for $i = 1$ **to** n_T **do**
  $d = \varnothing$
  for $j = 1$ **to** m_T **do**
    $\mu_{ij} = E_k(r_{ij})$
    $t \in_R (2^{N-1}, 2^N)$
    $r \leftarrow g^t \bmod p$
    $a \leftarrow (t - H(AutoNum||\mu_{ij}||r)x) \bmod n$
    $a_q \leftarrow a \bmod q$; $a'_q \leftarrow a \bmod q'$
    if $((a_q = 0)\ or\ (a_{q'} = 0))$ **then** goto 5
    if $((\frac{a}{q}) = -1)\ or\ ((\frac{a}{q'}) = -1)$ **then** goto 5
    $s_q \leftarrow (a_q)^{(q+1)/4} \bmod q$; $s'_q \leftarrow (a_{q'})^{(q'+1)/4} \bmod q'$
    $s \leftarrow (c.(s_q - s_{q'}) + s_{q'}) \bmod n$
    if $(s \geq n/2)$ **then** $s \leftarrow n - s$
    d.append$(\mu_{ij}, (r, s))$
  end
  T'.addrow(d)
end
Lưu trữ bảng T' lên máy chủ DSP

ký của dữ liệu cần cập nhập, sau đó gửi câu truy vấn đến máy chủ DSP. Giả sử câu truy vấn rõ là: "UPDATE T SET $c1 = v1, c2 = v2$ WHERE $c3 = condition$;", câu truy vấn mã sẽ được viết lại: "UPDATE T SET $c1 = \mu_1, c1' = \sigma_1, c2 = \mu_2, c2' = \sigma_2$ WHERE $c3 = \mu_3$;" (hình 2.9).

Khi thực hiện xóa dữ liệu thì người dùng gửi câu truy vấn "DELETE FROM T WHERE <điều kiện>;". Máy chủ DSP sẽ xoá bản ghi thoả mãn <điều kiện> mà không thực hiện tính toán lại các cấu trúc liên quan đến bản ghi bị xoá như khi sử dụng ADS. Như vậy, việc thao tác cập nhập dữ

Thuật toán 2.13: Thuật toán xác thực chữ ký và giải mã dữ liệu

Input: Bảng T_r, khoá k, khóa công khai

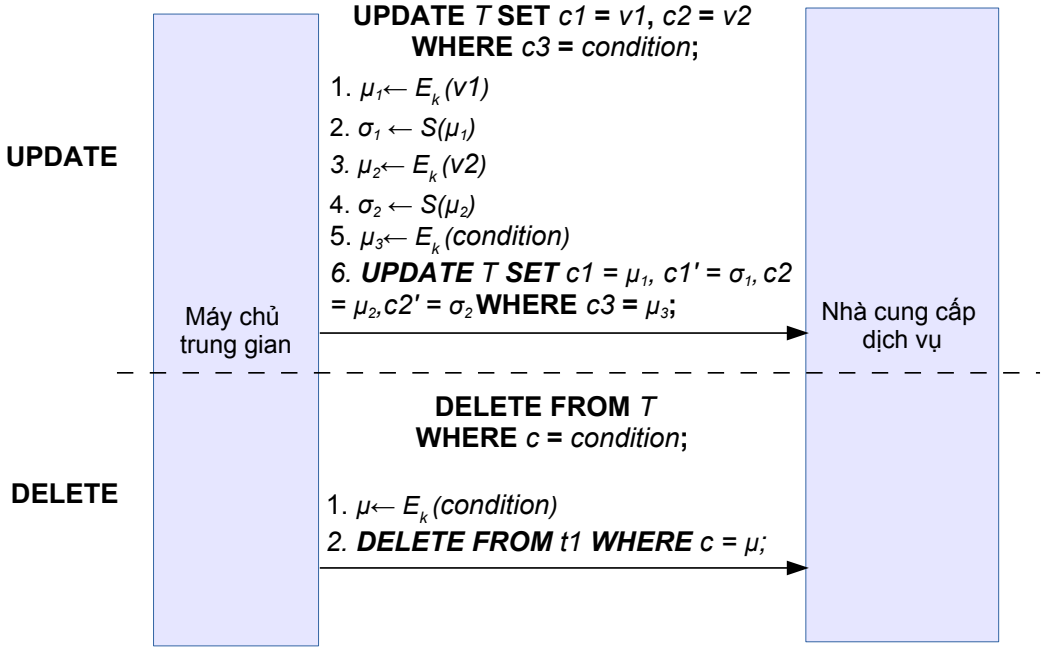
Output: Bảng dữ liệu rõ cho người dùng

$u = 0; v = 0; r = 1;$
for $i = 1$ **to** h_T **do**
  $d = \varnothing$
  for $j = 1$ **to** k_T **do**
    if $(s_{ij} \geq n/2)$ **then return** *"Reject"*
    $a = s_{ij}^2 \bmod n$
    $u+ = a \bmod p$
    $v+ = H(AutoNum||\mu_{ij}||r_{ij}) \bmod p$
    $r* = r_{ij} \bmod p$
    $a_{ij} = D_k(\mu_{ij})$
    d.append(a_{ij})
  end
end
if *(*$r == g^u y^v \bmod p$*)* **then** T.addrow(d)
else return *"Reject"*
Trả dữ liệu T về cho người dùng

liệu thực hiện dễ dàng và không ảnh hưởng đến các bản ghi trong bảng. Điều này thích hợp với CSDL động mà mô hình ADS khó giải quyết được.

2.4.3.2. Truy vấn dữ liệu từ nhiều bảng

Giả sử CSDL có hai bảng $T_1 = \{id1, \sigma_{id1}, \mu1_{ij}, \sigma1_{ij} | i = 1...n_T, j = 1...m_T\}, T_2 = \{id2, \sigma_{id2}, \mu2_{ij}, \sigma2_{ij} | i = 1...h_T, j = 1...k_T\}$. Người dùng gửi câu truy vấn "SELECT * FROM T_1 INNER JOIN T_2 ON $T_1.id1 = T_2.id2$;" đến DSP. Máy chủ DSP xử lý và trả kết quả $T_r = \{AutoNum1, id1, \sigma_{id1}, \mu1_{ij}, \sigma1_{ij}, AutoNum2, id2, \sigma_{id2}, \mu2_{ij}, \sigma2_{ij} | i = 1, 2, ...l_T; j = 1, 2, ..., p_T\}$ về máy chủ trung gian. Do $\sigma_{id1}, \sigma_{id2}, \sigma1_{ij}, \sigma2_{ij}$ được được tạo ra trên cùng khoá bí mật của DO nên ta có thể dùng thuật toán xác thực lô để xác thực cùng lúc



Hình 2.9: Quá trình cập nhập và xoá dữ liệu mã trên ODBS

mà không cần các thông tin phụ trợ kèm theo như các cấu trúc ADS.

2.5. Phân tích, đánh giá các phương pháp đề xuất

2.5.1. Phân tích, đánh giá phương pháp giảm thời gian truy vấn

Cho bảng CSDL $\mathscr{T}$ có n dòng và m cột. Khi truy vấn trên dữ liệu rõ, kết quả trả về được hiển thị mà không phải tiến hành giải mã. Vì số cột của bảng $\mathscr{T}$ là cố định nên chi phí xử lý phụ thuộc chủ yếu vào số lượng bản ghi. Do đó, chi phí truy vấn trên bản rõ là $O(n) = n \times m$. Khi truy vấn trên dữ liệu mã thì phải tiến hành giải mã nên tốn thêm chi phí giải mã. Vì vậy, truy vấn mã tuần tự có chi phí là $O(n) = n \times m \times t_{dec}$. Giả sử truy vấn mã 2 luồng, quá trình xử lý chia kết quả trả về là tập hợp D thành D_1 và D_2, và tiến hành giải mã trên hai tập này, sau đó kết hợp lại kết quả thì chi phí của song song 2 luồng là $max(T_{D_1}, T_{D_2}) + t_{join}(D_1, D_2)$, với T_{D_i} là chi phí giải mã tuần tự của tập D_i. Như vậy, chi phí xử lý 2 luồng thường giảm một nữa cộng thêm chi phí kết hợp kết quả so với việc giải mã tuần tự các bản ghi. Tuy nhiên, trên thực tế, việc chia nhỏ nhiều luồng không phải lúc nào cũng

mang lại kết quả tốt hơn. Do đó, khi triển khai vào thực tế cần phải đánh giá hệ thống hỗ trợ đa luồng, điều khiển luồng, điều khiển lỗi... để mang lại kết quả tốt nhất.

Để đánh giá kết quả của phương pháp đề xuất, luận án cài đặt thuật toán 2.1 bằng ngôn ngữ lập trình C# sử dụng Visual studio 2013 và thực hiện thử nghiệm trên máy tính Core™ i5-9000 CPU @ 2.9GHz, Ram 4GB, hệ điều hành windows 10, cơ sở dữ liệu MariaDB 10.4.17. Thuật toán mã hoá dữ liệu là AES được thiết kế thành hàm trong C#. Câu truy vấn *"SELECT* $f_1, f_2, \ldots, f_5$ *FROM t"* , trong đó $f_1, f_2, \ldots, f_5$ là các trường dữ liệu của bảng *t(ID,Hoten,Ngaysinh,Gioitinh,Diachi).*

Theo khảo sát của luận án, các đề xuất trước đây chưa sử dụng phương pháp song song trong truy vấn dữ liệu mã nên luận án thử nghiệm với hai kịch bản: Thực hiện truy vấn tuần tự trên dữ liệu mã và truy vấn song song 2 luồng trên dữ liệu mã để đánh giá tính hiệu quả của phương pháp đề xuất. Bên cạnh đó, luận án đưa ra phương pháp thực hiện đồng thời 3 truy vấn cùng một lúc để giả lập CPU bận đối với từng kịch bản. Kết quả thực hiện được mô tả ở bảng 2.3.

Bảng 2.3: Thời gian thực hiện truy vấn (ms)

CPU	Tuần tự		2 luồng	
Số bản ghi	Rỗi	Bận	Rỗi	Bận
1000	43	131	30	130
5000	234	647	125	503
10000	426	1381	274	896
50000	2070	6822	1191	4532
100000	4000	13192	2404	8594

Bảng 2.3 cho thấy thời gian thực hiện của các kịch bản truy vấn tuần tự trên dữ liệu mã gấp hơn 1.5 lần so với truy vấn 2 luồng. Điều này được giải thích do sau khi tính toán giải mã thì kịch bản 2 luồng phải cần có thời gian T_{join} để kết hợp các tập dữ liệu xử lý và phù hợp với đánh giá lý thuyết. Mặt khác, độ chênh lệch giữa truy vấn trên dữ liệu mã và dữ liệu rõ cho thấy sự

phức tạp tính toán trong việc đề xuất các phương pháp truy vấn trên dữ liệu mã. Trên thực tế, khi truy vấn dữ liệu, số lượng bản ghi trả về chỉ thoả mãn điều kiện truy vấn mà không phải lúc nào cũng là toàn bộ bảng dữ liệu nên số lượng bản ghi không nhiều. Mặt khác, khi số lượng bản ghi trả về lớn thì cần nhiều kỹ thuật xử lý phù hợp như: Cân bằng tải, phân trang (Pagination) để làm giảm tải quá trình xử lý.

Tác giả Ahmad và cộng sự [4] dựa trên đề xuất được trình bày ở mục 2.2.2 của luận án đã làm rõ hơn về phương pháp xử lý song song khi truy vấn trên dữ liệu mã. Ahmad tiến hành thử nghiệm với máy tính CPU Core i5-6200U, 8 GB DDR3 RAM, ổ cứng SANDISK M.2 Sata SSD 256 GB và sử dụng Visual Studio Ultimate 2012, SQL Server 2012 Express. Ahmad thực hiện truy vấn trên dữ liệu mã với 1000, 100.000 và 1.000.000 bản ghi. Ahmad thực hiện các câu truy vấn tuần tự và song song từ 2 đến 6 luồng trên hai kịch bản khác nhau: Kịch bản đầu tiên là khi CPU không hoạt động (nhàn rỗi), có nghĩa là nó không có bất kỳ truy vấn SQL nào khác để thực thi và kịch bản thứ hai hai là truy vấn liên tiếp trong 3 lần thực thi (CPU bận). Kết quả thời gian thực hiện truy vấn trên dữ liệu mã của Ahmad được mô tả trong bảng 2.4.

Bảng 2.4: Thời gian xử lý (ms) theo thử nghiệm của Ahmad [4]

Số bản ghi	1000		100.000		1.000.000	
CPU	Rỗi	Bận	Rỗi	Bận	Rỗi	Bận
Tuần tự	10	200	500	1284	5513	13.117
2 luồng	7	23	429	994	4598	11.550
3 luồng	3	21	574	880	5621	9038
4 luồng	3	18	473	760	5321	7816
5 luồng	3	15	464	582	5174	7755
6 luồng	4	16	426	507	5020	7112

Ở kịch bản thứ nhất, kết quả thử nghiệm của Ahmad cho thấy khi truy vấn 1000 bản ghi,thì không có thay đổi nhiều về thời gian xử lý với hệ thống sử dụng hơn 2 luồng. Thời gian xử lý trên 2 luồng là 7 ms và thời gian xử lý từ 3 đến 6 luồng ổn định ở mức 3 đến 4 ms. Như vậy, trong trường hợp này, thuật toán không có khác biệt rõ ràng khi sử dụng trên nhiều lõi. Tuy nhiên

nó được cải thiện khá nhiều so với xử lý tuần tự ở mức 10 ms. Khi Ahmad truy vấn 1.000.000 bản ghi, trong đó 2 luồng thực hiện 4598 ms là tốt nhất và xử lý tuần tự cũng tốt hơn 3 luồng. Kết quả cho thấy hiệu suất xử lý tuần tự gần như nhau trên 3 hoặc nhiều luồng.

Ở kịch bản thứ hai, khi Ahmad thực hiện truy vấn 1000 bản ghi thì tốt nhất là 5 luồng chỉ mất 15 ms. Trong khi đó xử lý tuần tự mất 200 ms. Như vậy, kết quả trong trường hợp này là tốt hơn khi sử dụng đa luồng. Khi Ahmad thực hiện truy vấn 100.000 bản ghi, việc xử lý 6 luồng là tốt nhất chỉ mất 507 ms, trong khi xử lý tuần tự mất 1284 ms. Khi Ahmad thực hiện truy vấn 1.000.000 bản ghi, kết quả thực hiện tốt nhất là 6 luồng chỉ mất 7112 ms, trong khi tuần tự mất 13.117 ms. Những kết quả này chứng minh được hiệu quả của xử lý song song trong trường hợp CPU luôn được sử dụng.

Bảng 2.5: So sánh phương pháp giảm thời gian truy vấn trên dữ liệu mã

Nhiệm vụ	Phương pháp của luận án	Phương pháp của Ahmad [4]
Truy vấn trên dữ liệu mã	Có	Có
Xử lý song song trong truy vấn	Có	Có
Sử dụng máy chủ trung gian	Có	Không
Thuật toán mã hoá	AES (Tuỳ biến)	AES (sẵn có của CSDL)

Trong mô hình đề xuất, luận án sử dụng một máy chủ trung gian để giải quyết mọi xử lý mà không can thiệp vào DBMS hay máy chủ của DSP. Trong khi đó, hạn chế của Ahmad đưa ra trong đề xuất của mình là sự phụ thuộc nền tảng của SQL Server vì tác giả đã sử dụng thuật toán mã hóa/giải mã được định nghĩa bởi Microsoft SQL Server và khóa mã được lưu trữ trong CSDL. Việc xử lý giải mã của Ahmad là do DBMS của DSP còn việc giải mã của luận án đề xuất là do máy chủ trung gian xử lý. Lợi ích của mô hình luận án đề xuất là tổng quát hóa cho các trường hợp xử lý bảo mật mà Ahmad khó làm được như: Dữ liệu truyền từ DSP về máy chủ trung gian là bản mã, thuật toán mật mã ở máy chủ trung gian có thể tùy biến không chỉ dùng

AES, như vậy sẽ dễ dàng phù hợp với các thuật toán mật mã của Chính phủ Việt Nam. Mặc dù vậy, kết quả của Ahmad một lần nữa khẳng định, nếu áp dụng mô hình xử lý song song vào các tiến trình tính toán khi truy vấn trên dữ liệu mã sẽ mang lại hiệu quả đáng kể khi số lượng truy vấn nhiều và số bản ghi trả về lớn.

2.5.2. Phân tích, đánh giá phương pháp xác thực dữ liệu mã

2.5.2.1. So sánh chi phí giữa những cặp lược đồ gốc và cải tiến

Do những cải tiến nêu ra trong mục 2.3.5.2 nhằm tăng tính hiệu quả chủ yếu trong các thuật toán tạo chữ ký cho nên luận án chỉ đánh giá trong từng cặp thuật toán này. Trong phân tích của luận án luôn giả thiết các cặp thuật toán cùng sử dụng việc lưu giá trị c để tính CRT và dùng chung số mũ e.

Mệnh đề 2.5 *Chi phí thuật toán ký của lược đồ Rabin-Schnorr cải tiến ít hơn của Rabin-Schnorr là:*

$$4.(1,5.(L-N)-1).t_m(L) \tag{2.14}$$

Chứng minh

Sự khác nhau giữa hai thuật toán là tham số t ở thuật toán 2.2 là số L-bít, còn ở thuật toán 2.8 là N-bít như vậy chi phí cho việc tính giá trị g^t mod p của hai thuật toán theo công thức (2.7) lần lượt sẽ là $1,5.L.t_m(L)$ và $1,5.N.t_m(L)$. Đối với thuật toán cải tiến có thêm việc kiểm tra (($a_q = 0$) or ($a_{q'} = 0$)) có chi phí $2.t_{Red}(\frac{L}{2}) < t_m(L)$.

Biết rằng xác suất để $(\frac{a}{q}) = -1$ (hoặc $(\frac{a}{q'}) = -1$) bằng 0,5 nên để qua được bước 4 của thuật toán 2.2 (tương tự bước 6 của thuật toán 2.8) trung bình cần thực hiện 4 lần kiểm tra $(\frac{a}{q}) = -1$ or $(\frac{a}{q'}) = -1$. Như vậy hiệu chi phí giữa thuật toán 2.2 và 2.8 sẽ là

$4.(1,5.L - (1,5.N + 1)).t_m(L) = 4.(1,5.(L-N)-1).t_m(L)$. Và đây là điều cần chứng minh.

Tương tự đối với cặp lược RSA-Schnorr ta có:

Mệnh đề 2.6 *Chi phí thuật toán ký của lược đồ RSA-Schnorr cải tiến ít hơn của RSA-Schnorr là*

$$(1,5.(L-N)-1).t_m(L) \tag{2.15}$$

Từ (2.14) và (2.15) ta tính được chi phí (theo $t_m(L)$) tiết kiệm được khi sử dụng các lược đồ cải tiến so với lược đồ ban đầu trình bày trong bảng 2.6.

Bảng 2.6: Chi phí tiết kiệm được của lược đồ cải tiến tương ứng với cặp (L, N) cho trong bảng 2.2

N	160	224	256	384	512
L	1024	2048	3072	8192	15360
Rabin-Schnorr	5180	10940	16892	46844	89084
RSA-Schnorr	1295	2735	4223	11711	22271

2.5.2.2. Chi phí thực hiện các thuật toán đề xuất

Thời gian thực hiện của các giai đoạn tạo khóa, tạo chữ ký và xác thực ODBS phụ thuộc vào số lượng các phép tính toán. Trong đó tập trung vào thời gian thực hiện các phép mũ, phép nhân modulo, thời gian tính hàm băm, thời gian mã hóa, giải mã và bỏ qua các phép tính cộng modulo vì thời gian thực hiện không đáng kể [42].

Luận án tiến hành tính toán chi phí cho các giai đoạn tạo chữ ký, xác thực chữ ký và xác thực lô của thuật toán Rabin-Schnorr lần lượt là các thuật toán 2.8, 2.9, 2.10; Thuật toán RSA-Schnorr lần luợt là các thuật toán 2.11, 2.6, 2.7.

Đối với thuật toán 2.8, tham số t là N-bít nên chi phí tính giá trị $g^t \bmod p$ là $t_{exp}(N,L) = 1,5.N.t_m(L)$ (theo công thức (2.7)). Chi phí cho việc tính a là $t_H + t_m(L)$. Việc kiểm tra $((a_q = 0)$ or $(a_{q'} = 0))$ có chi phí $2.t_{Red}(\frac{L}{2}) < t_m(L)$. Xác suất để $(\frac{a}{q}) = -1$ (hoặc $(\frac{a}{q'}) = -1$) bằng 0,5 nên để qua được bước 6 thì cần trung bình thực hiện 4 lần kiểm tra $((\frac{a}{q}) = -1$ or $(\frac{a}{q'}) = -1)$. Do đó, để qua bước 6, chi phí thực hiện là $4.((1,5.N+2).t_m(L) + t_H + t_{Jacobi}(\frac{L}{2}))$. Chi phí cho tính s_q (tương tự cho

$s_{q'}$) là $t_{exp}(\frac{L}{2}, \frac{L}{2}) = 1,5.\frac{L}{2}.t_m(\frac{L}{2})$. Chi phí tính s là $t_m(L)$. Như vậy, chi phí của thuật toán 2.8 là $4.((1,5.N+2).t_m(L)+t_H+t_{Jacobi}(\frac{L}{2}))+3.\frac{L}{2}.t_m(\frac{L}{2})+t_m(L) = (6.N+9)t_m(L)+4.t_H+4.t_{Jacobi}(\frac{L}{2})+3.\frac{L}{2}.t_m(\frac{L}{2})$

Thực hiện tính toán chi phí tương tự cho các thuật toán còn lại thì được chi phí các công việc của các thuật toán đề xuất như bảng 2.7. Trong đó, k là số lượng chữ ký cần xác thực lô.

Bảng 2.7: Thời gian thực hiện các công việc của thuật toán chữ ký số

Công việc	Thuật toán Rabin-Schnorr	Thuật toán RSA-Schnorr
Tạo chữ ký	$(6.N+9)t_m(L)+4.t_H$ $+4.t_{Jacobi}(\frac{L}{2})+3.\frac{L}{2}.t_m(\frac{L}{2})$	$(1,5.N+3).t_m(L)+t_H$ $+3.L.t_m(\frac{L}{2})$
Xác thực chữ ký	$(1,5.(L+N)+2).t_m(L)+t_H$	$(3.L+1,5.N+1).t_m(L)+t_H$
Xác thực k chữ ký	$k.((1,5.(L+N)+2).t_m(L)+t_H)$	$k.((3.L+1,5.N+1).t_m(L)+t_H)$
Xác thực lô	$(1,5.(L+N)+2.k+2).t_m(L)$ $+k(t_H+t_m(N))$	$(3.L+1,5.N+2k+1).t_m(L)$ $+k(t_H+t_m(N))$

Các giai đoạn thực hiện xác thực CSDL mã trên ODBS, ngoài việc tạo và xác thực chữ ký thì còn thêm quá trình mã hóa - giải mã dữ liệu. Giả sử CSDL có $\mathscr{T}_1$ bảng, mỗi bảng có n_T dòng và m_T cột. Khi truy vấn dữ liệu, máy chủ DSP trả về $\mathscr{T}_2$ bảng, mỗi bảng có h_T dòng, k_T cột.

Do thuật toán tạo chữ ký và xác thực dựa trên Rabin-Schnorr nên chi phí thực hiện tính toán của các giai đoạn xác thực CSDL mã trên ODBS được mô tả như bảng 2.8.

Bảng 2.8: Thời gian thực hiện các giai đoạn xác thực ODBS

Giai đoạn	Thời gian thực hiện
Lưu trữ dữ liệu	$n_T m_T(T_E+(6.N+9)t_m(L)+4.t_H$ $+4.t_{Jacobi}(\frac{L}{2})+3.\frac{L}{2}.t_m(\frac{L}{2})+t_{insert}(\mathscr{T}_1)$
Xác thực dữ liệu	$h_T(k_T(2.T_m(L)+t_m(L)+t_m(N)+t_H+T_D)$ $+(1,5.(L+N)+1)t_m(L)$

2.5.2.3. Thử nghiệm các phương pháp đề xuất

Để thử nghiệm các phương pháp được đề xuất, luận án tiến hành cài đặt các thuật toán bằng ngôn ngữ lập trình Python và thực hiện trên máy tính Core™ i5-4310U CPU @ 2.0GHz, Ram 8GB, hệ điều hành Ubntu 20.04.

Thời gian thực hiện tạo chữ ký với trên số lượng các dữ liệu tương ứng như bảng 2.9.

Bảng 2.9: Thời gian thực hiện tạo chữ ký (s)

Số lượng dữ liệu	Thuật toán 2.8	Thuật toán 2.11
10	0,25614	0,01634
100	1,535692	0,150983
1000	14,061793	1,228292
10000	136,160404	10,059431

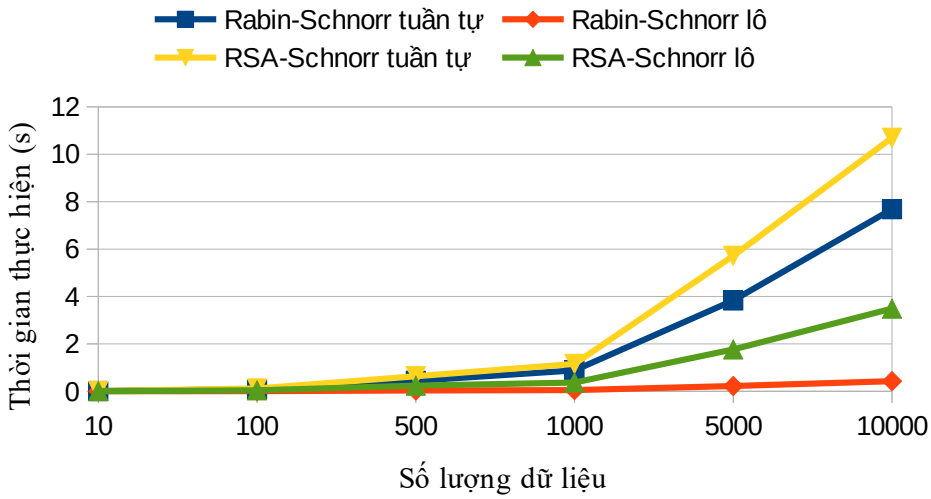
Do chọn xác suất giá trị a trong khi ký sao cho $\left(\frac{a}{q}\right) = 1$ và $\left(\frac{a}{q'}\right) = 1$ nên thuật toán Rabin-Schnorr có thời gian thực hiện lớn hơn thời gian ký của thuật toán RSA-Schnorr. Tuy nhiên, nếu xét về mặt ứng dụng trong mô hình ODBS, thời gian thực hiện ký xảy ra khi DO mã hóa và lưu trữ dữ liệu lên máy chủ DSP. Công việc này mỗi DO thực hiện lần lượt từng bản ghi, do đó không ảnh hưởng đến hiệu năng của hệ thống. Còn khi người dùng thực thi truy vấn thì số lượng bản ghi trả về lớn và nhiều người truy cập đồng thời nên vấn đề cần quan tâm là thời gian xác thực, giải mã dữ liệu trả về.

Để làm rõ hơn tính hiệu quả giữa phương pháp xác thực lô và xác thực tuần tự, luận án tiến hành thử nghiệm thời gian thực hiện trên hai phương pháp: Xác thực tuần tự k chữ ký và xác thực lô cho từng thuật toán.

Thời gian thực hiện xác thực của các thuật toán đề xuất được trình bày trong bảng 2.10 và hình 2.10. Trong đó thời gian thực hiện xác thực lô của thuật toán Rabin-Schnorr thấp hơn so với thuật toán RSA-Schnorr, điều này thể hiện đúng với chi phí đánh giá trong bảng 2.7. Do đó, phương pháp xác thực lô Rabin-Schnorr giúp làm giảm đáng kể thời gian khi thực hiện xác thực ODBS so với thuật toán RSA-Schnorr.

Bảng 2.10: Thời gian thực hiện xác thực (s)

Số lượng Dữ liệu	Rabin-Schnorr		RSA-Schnorr	
	Xác thực tuần tự	Xác thực lô	Xác thực tuần tự	Xác thực lô
10	0,01072	0,00137	0,01235	0,00459
100	0,08574	0,00575	0,11793	0,03903
500	0,44312	0,03519	0,63497	0,23705
1000	0,88124	0,05107	1,15748	0,36874
5000	3,83195	0,22150	5,71553	1,76460
10000	7,68620	0,42790	10,69711	3,48795



Hình 2.10: Thời gian xác thực chữ ký

2.5.2.4. Phân tích đánh giá các trường hợp tấn công

Trong phần này, luận án đưa ra một số khả năng mà Adv có thể thực hiện tấn công để xem xét độ an toàn của thuật toán đề xuất.

- Trường hợp 1. Giả mạo, thay đổi dữ liệu trong CSDL: Vì dữ liệu đã được mã hoá bởi khoá k nên Adv muốn tạo ra hoặc thay đổi dữ liệu thì Adv phải cần tấn công đánh cắp khoá k và thực hiện $\mu = E_k(r)$, sau đó cập nhập vào dữ liệu.

- Trường hợp 2. Giả mạo chữ ký cho các dữ liệu giả mạo: Giả sử Adv tấn công có được khoá k và tạo ra một giá trị mã μ hợp lệ. Khi đó, Adv tiến hành giả mạo chữ ký cho giá trị μ.

  Đối với thuật toán Rabin-Schnorr: Adv giả mạo chữ ký (r, s) cho μ

bằng cách tạo ra giá trị s dựa trên tính giá trị a tại bước thứ 7 của thuật toán 2.12. Muốn tính giá trị a, Adv cần có tham số x dựa trên khoá công khai y. Để làm được vậy, Adv phải giải bài toán DLP do $y = g^x \bmod p$. Trường hợp Adv giải được bài toán DLP và có giá trị x thì Adv vẫn cần phải giải bài toán IFP để phân tích n thành thừa số q và q' để tính s_q và $s_{q'}$ tại bước 11 mới có thể tính được s. Như vậy, muốn tạo ra chữ ký hợp lệ, Adv phải giải cả hai bài toán DLP và IFP. Việc giải một trong hai bài toán sẽ không thể tạo ra giá trị chữ ký hợp lệ.

Nếu sử dụng thuật toán RSA-Schnorr vào tạo chữ ký: Tương tự như thuật toán Rabin-Schnorr, Adv muốn giả mạo chữ ký phải giải được bài toán DLP để có giá trị x, đồng thời phân tích được n thành q và q' để tìm được tham số d dựa trên $\phi(n)$. Nghĩa là, để tính được s, Adv cũng phải giải đồng thời hai bài toán DLP và IFP.

- Trường hợp 3. Hoán đổi các giá trị giữa các bản ghi với nhau: Giả sử Adv hoán đổi một giá trị $\mu, (r, s)$ tại ô bất kỳ của bản ghi số 3 cho bản ghi số 5 và xác thực bản ghi số 5. Tuy nhiên, khi thực hiện thuật toán 2.13 để xác thực dữ liệu, DO tính giá trị $H(AutoNum||\mu_{ij}||r_{ij})$ tại bước 8, và kết quả sẽ không hợp lệ do chữ ký (r, s) có $AutoNum = 3$, còn xác thực $AutoNum = 5$. Do giá trị $AutoNum$ này là cột dữ liệu không trùng nhau nên Adv không đồng thời sửa $AutoNum$ của cột thứ 5 bằng giá trị 3.

2.5.2.5. Những vấn đề chưa giải quyết của phương pháp đề xuất

- Luận án thực hiện xử lý song song trên dữ liệu mã với câu lệnh SELECT và công việc giải mã, hiển thị dữ liệu rõ cho người dùng mà chưa đề xuất phương pháp chuyển đổi câu lệnh để truy vấn trên dữ liệu mã. Hiện nay, vẫn chưa có đề xuất nào giải quyết việc thực hiện cho tất cả các trường hợp truy vấn trên dữ liệu mã. Do đó, trong quá trình đề xuất xử lý truy vấn trên dữ liệu mã, việc kết hợp song song trong một số tiến trình xử

lý sẽ mang lại hiệu quả, có khả năng ứng dụng thực tế.

- Luận án đề xuất phương pháp xác thực để kiểm tra dữ liệu là đúng được tạo ra bởi DO. Phương pháp đề xuất phù hợp CSDL động, dữ liệu trả về nằm trên nhiều bảng, không phụ thuộc thứ tự bản ghi trong bảng.Tuy nhiên, phương pháp đề xuất chưa giải quyết được vấn đề bị tấn công xoá các ô dữ liệu, xoá bản ghi, thậm chí xoá cả bảng dữ liệu hoặc hoán đổi dữ liệu giữa các trường trong cùng bản ghi.

2.6. Kết luận

Chương 2 đề xuất giải pháp xử lý song song trên dữ liệu mã để rút ngắn thời gian thực thi truy vấn, đề xuất hai lược đồ xác thực lô dựa trên hai bài toán là IFP và DLP là: RSA-Schnorr và Rabin-Schnorr. Việc sử dụng hai bài toán khó sẽ giúp nâng cao tính an toàn cho chữ ký nhưng nhược điểm của nó sẽ tăng độ phức tạp tính toán. Tuy nhiên, kết quả thử nghiệm cho thấy tính hiệu quả của phương pháp đề xuất. Chương 2 cũng đề xuất thuật toán xác thực để kiểm tra tính đúng đắn của CSDL mã dựa trên xác thực lô, trả về kết quả rõ cho người dùng. Mô hình đề xuất hỗ trợ xác thực dữ liệu trả về của các câu truy vấn từ nhiều bảng và phù hợp với CSDL động. Hơn nữa, khi kiểm tra kết hợp với giải mã dữ liệu nên phương pháp đề xuất không tốn thời gian giải mã sau khi xác thực như các đề xuất trước đây. Nếu việc xác thực dữ liệu kết hợp với thuật toán 2.1 thì sẽ giảm thiểu về mặt thời gian xử lý, có thể triển khai ứng dụng vào thực tế.

Để cải thiện tốc độ truy vấn trên CSDL mã tốt hơn nữa thì cần các giải pháp bổ sung như: Sử dụng nhiều máy chủ trung gian và cân bằng tải, sử dụng kỹ thuật phân trang (Pagination) để chia nhỏ tập dữ liệu xử lý... Mặt khác, thuật toán mã hoá cũng là một yếu tố làm chậm quá trình tính toán trên dữ liệu mã. Nếu dùng thuật toán mã hoá đối xứng thì xử lý mã/giải mã nhanh nhưng không tính toán được trên dữ liệu mã. Muốn thực hiện truy vấn tính toán, máy chủ bắt buộc phải giải mã dữ liệu để được bản rõ, tính

toán trên bản rõ, rồi trả kết quả gửi về cho người dùng. Nếu dùng thuật toán mã hoá HOM thì hỗ trợ tính toán trên dữ liệu mã nhưng tốn thời gian hơn mã hóa đối xứng trong việc mã/giải mã... Vì vậy, việc kết hợp các thuật toán mã hoá, phân chia trường hợp truy vấn tương ứng với dữ liệu mã cho phù hợp là một ý tưởng cần tiếp tục nghiên cứu và phát triển.

Chương 3
QUẢN LÝ, THAY ĐỔI KHOÁ MÃ CỦA ODBS

Nội dung chương 3 giới thiệu về vấn đề quản lý, phân phối khóa và quản lý quyền truy cập. Từ đó, luận án đề xuất cách thức lưu trữ khóa, mô hình truy cập dữ liệu của người dùng và phương pháp thay đổi khóa mã mức cột của CSDL mã hóa dựa trên nền tảng MapReduce, các nghiên cứu này được công bố trong [CT4][CT5]. Kết quả thử nghiệm cho thấy phương pháp đổi khoá đề xuất có khả năng ứng dụng vào thực tế. Cuối cùng là kết luận chương.

3.1. Giới thiệu chung

Các nghiên cứu như [32, 60, 12, 82, 76] đề xuất các phương pháp truy vấn trên dữ liệu mã bằng cách lưu trữ các thông tin phụ trợ để hỗ trợ truy vấn hoặc truy vấn trực tiếp. Tuy nhiên, các nghiên cứu này chỉ dùng chung một khoá mã cho toàn bộ CSDL. Điều này có thể dẫn đến sự tấn công theo kiểu bắt tay: Nếu người dùng có khoá mã và nhân viên của DSP có toàn bộ CSDL thì họ có thể thoả hiệp để giải mã tất cả dữ liệu mà bỏ qua các bước xác thực khi truy cập.

Để hạn chế khả năng truy cập dữ liệu của người sử dụng, DO phải có các cơ chế quản lý truy cập người dùng với các quyền khác nhau. Có nhiều mức độ kiểm soát truy cập: Mức CSDL (toàn quyền truy cập CSDL), mức bảng, mức cột, mức dòng. Trong đó mức CSDL là đơn giản nhất với một khoá chung nhưng dễ bị rò rỉ thông tin nếu khoá bị lộ. Mức dòng là mức bảo mật cao nhất nhưng khó quản lý do có nhiều khoá. S.Hong và cộng sự [38] đã đề xuất cây RST (Resource Set Tree) để quản lý khoá truy cập người dùng. Cây RST là cây mà mỗi nút lá chứa một danh sách các lớp người dùng có cùng quyền truy cập vào tập dữ liệu, tuy nhiên phương pháp này không đưa ra mối quan hệ giữa người dùng với khoá mã của CSDL. Hang và cộng sự

[34] đã đề xuất mô hình ENKI cho phép quản lý quyền truy cập. Khi người dùng đã được xác thực bởi máy chủ thì họ sẽ có được khoá chính (masterkey) để giải mã khóa mã hóa được lưu trữ trong tập khóa. Hang dùng trình điều khiển JDBC (có sửa đổi) để viết lại câu truy vấn để thực thi câu truy vấn trên dữ liệu mã. Tuy nhiên, phương pháp này không đề xuất việc thay đổi khoá mã của CSDL.

Bài toán thay đổi khoá (rekey) đã được tác giả Phạm Thị Bạch Huệ đề ra trong hướng phát triển [1], tuy nhiên, theo khảo sát của NCS, hiện nay vẫn chưa có công bố nào liên quan đến bài toán thay đổi khóa. Bài toán thay khoá cho ODBS cần thiết khi: Lộ khóa từ người dùng, thu hồi quyền người dùng... Trong các vấn đề liên quan đến tính bí mật dữ liệu, thì bài toán quản lý và thay đổi khoá là những bài toán quan trọng góp phần bảo đảm sự an toàn cho ODBS.

Trong chương này, luận án đề xuất giải pháp quản lý truy cập của người dùng theo mức cột và phương pháp thay khoá cho CSDL với thời gian thực hiện có thể ứng dụng trong thực tế, đáp ứng được tính sẵn sàng của dữ liệu.

3.1.1. Quản lý khóa và quyền truy cập

3.1.2. Quản lý khóa

Quản lý khóa mã là một vấn đề quan trọng trong bài toán bảo mật. Khóa dùng để mã hóa - giải mã dữ liệu theo một thuật toán mã hóa cho trước. Cho một CSDL có t bảng, giả sử mỗi bảng có m cột và n bản ghi. Nếu ta thực hiện mã hóa theo mức dòng và quản lý $\mathscr{U}$ người dùng thì số lượng khoá tối đa cần phải có là $t \times n \times \mathscr{U}$.

Mã hoá dữ liệu theo mức cột (trường) thì tính bảo mật dữ liệu thấp hơn mức dòng do các dữ liệu trong cùng một cột sẽ dùng chung một khoá mã. Như vậy, số lượng khoá mã cần có là $t \times m$. Tuy nhiên, ta cần có một cơ chế quản lý truy cập để hạn chế quyền của người dùng vào các cột dữ liệu tương ứng. Giả sử với bảng $\mathscr{T}(f_1, f_2, \ldots, f_m)$, người dùng u muốn truy cập

vào thuộc tính f_i thì phải có khoá k_i, $i = 1, ..., m$.

3.1.3. Quản lý quyền truy cập dữ liệu

Quản lý quyền truy cập dữ liệu là bài toán bảo vệ tính riêng tư dữ liệu. Nghĩa là, người dùng chỉ được phép truy cập vào những dữ liệu mà DO cấp quyền. Trong mô hình ODBS, nếu DO phân quyền người dùng bằng các bảng CSDL và lưu trên đám mây thì DSP có thể xâm hại mà không thông qua cơ chế quản lý truy cập. Như vậy, muốn tăng tính an toàn cho CSDL, một đề xuất được đặt ra là DO phân quyền và quản lý truy cập trên máy chủ của mình, đồng thời DO phải kiểm tra quyền của người dùng trước khi cho phép người dùng truy vấn đến ODBS.

Để quản lý quyền truy cập dữ liệu của người dùng, DO dùng ma trận kiểm soát truy cập. Cho bảng $\mathscr{T}(f_1, f_2, \ldots, f_m)$, và người dùng $\mathscr{U}(u_1, u_2, ..., u_p)$. Ma trận kiểm soát truy cập $\mathscr{A}$ được biểu diễn như bảng 3.1 Trong đó, nếu

Bảng 3.1: Ma trận kiểm soát truy cập cho bảng $\mathscr{T}$

	f_1	f_2	...	f_m
u_1	0	1	...	1
u_2	1	1	...	0
⋮				
u_p	1	0	...	1

$\mathscr{A}[i, j] = 1$ thì người dùng thứ i được truy cập vào cột dữ liệu thứ j với $1 \leq i \leq p$, $1 \leq j \leq m$. Ngược lại, $\mathscr{A}[i, j] = 0$ nghĩa là người dùng thứ i không được truy cập vào cột dữ liệu thứ j. Tuỳ vào bài toán phân quyền cụ thể mà ma trận kiểm soát truy cập sẽ thay đổi các cột $f_1, f_2, \ldots, f_m$ thành mức bảng, dòng... Vector quyền truy cập người dùng $u_i = f_2, f_3, f_k, ...$ nghĩa là người dùng thứ i được quyền truy xuất vào các cột $f_2, f_3, f_k...$ Vector quyền truy cập cột $f_i = \{u_1, u_3, u_k, ...\}$ nghĩa là người dùng 1, 2, k... được quyền truy cập vào cột thứ i.

3.2. Đề xuất mô hình quản lý khoá và quyền truy cập

3.2.1. Mô hình quản lý khoá

Xét một bảng $\mathscr{T}(f_1, f_2, \ldots, f_m)$ với $f_1, f_2, \ldots, f_m$ là các thuộc tính; $\mathscr{T}$ chứa n bản ghi $r = (r_{i1}, r_{i2}..., r_{im})$, trong đó r_{ij} là dữ liệu tại dòng thứ i và cột thứ j với $1 \leq i \leq n$, $1 \leq j \leq m$.

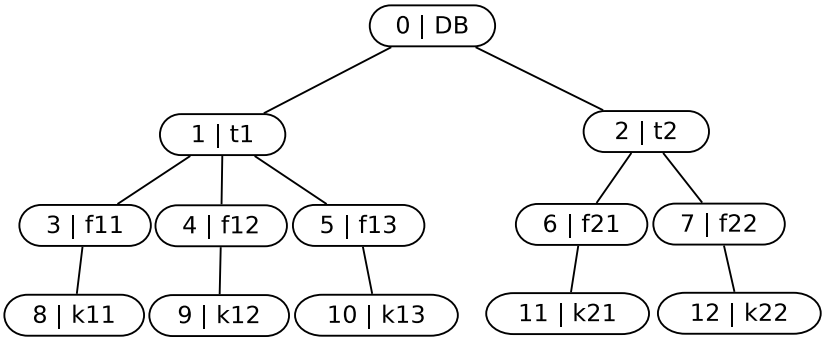
Định nghĩa 3.1 *Cho f là một thuộc tính bất kỳ trong một bảng của CSDL có n bản ghi. Mã hoá dữ liệu mức cột của thuộc tính f với khoá k được định nghĩa: $E_k(f) \coloneqq \{E_k(r_i) | r_i \in f;\ i = 1, ..., n\}$ với r_i là dữ liệu tại dòng thứ i của cột f.*

Mã hoá của bảng $\mathscr{T}(f_1, f_2, \ldots, f_m)$ là mã hoá tất cả các thuộc tính $f_1, f_2, \ldots, f_m$ và các dữ liệu trong các thuộc tính đó:

$$\begin{aligned} E(\mathscr{T}) &\coloneqq (E_{k_1}(f_1), E_{k_2}(f_2), ..., E_{k_m}(f_m)) \\ &= \{E_{k_1}(r_{i1}), E_{k_2}(r_{i2}), ..., E_{k_m}(r_{im}) | r_{ij} \in f_j;\ i = 1, ..., n;\ j = 1, ..., m\} \end{aligned} \tag{3.1}$$

Trong công thức (3.1), tập $\mathscr{K}(k_1, k_2, ..., k_m)$ là tập khoá mã của bảng $\mathscr{T}$. Ta gọi $\mathscr{K}$ là tập khoá mã theo mức cột của $\mathscr{T}$.

Đề xuất mỗi CSDL sẽ có một cây quản lý khoá gọi là cây KMT (Key Management Tree). Nút gốc (mức 1) của cây KMT là tên CSDL, mức 2 là tên các bảng, mức 3 là tên các trường và nút lá là các khoá của cột.



Hình 3.1: Cây quản lý khoá KMT của CSDL

Vì cây KMT quản lý tập trung các khoá của các cột trong CSDL nên dễ quản lý, thay đổi khoá khi cần thiết. Muốn truy xuất đến một khoá bất kỳ của một cột trong bảng, ta chỉ cần biết tên bảng và tên cột. Khi kết hợp cây KMT với ma trận kiểm soát truy cập (bảng 3.1), người dùng sẽ được phép truy cập vào cột với khoá của cột trong cây KMT. Cho CSDL DB gồm 2 bảng $t1(f11, f12, f13)$, $t2(f21, f22)$ và các tập khoá mã mức cột của bảng $t1$, $t2$ tương ứng là $\mathscr{K}_1(k11, k12, k13)$, $\mathscr{K}_2(k21, k22)$. Cây KMT được biểu diễn như hình 3.1, trong đó các giá trị 0, 1, 2... là vị trí của các nút trên cây.

Ta có thể biểu diễn KMT bằng một cấu trúc mảng quản lý khoá như bảng 3.2. Mỗi nút lưu trữ trong mảng quản lý khoá sẽ có cấu trúc *Node(index, value)*, trong đó: *index* chứa vị trí của nút, *value* chứa giá trị tên bảng, cột hoặc khóa mã. Ta dùng mảng K chứa các nút với I_X là vị trí của nút X trong mảng K.

Tính chất của mảng quản lý khoá:

- *Nếu R là nút gốc, thì* $I_R = -1$
- *Nút B là nút con của A thì* $B.index = I_A$

Bảng 3.2: Cấu trúc quản lý khoá của cây KMT

0	1	2	3	4	5	6	7	8	9	10	11	12	← Chỉ số mảng
DB	t1	t2	f11	f12	f13	f21	f22	k11	k12	k13	k21	k22	← Nhãn của nút trên cây
-1	0	0	1	1	1	2	2	3	4	5	6	7	← Chỉ số nút cha (=-1 nếu nút không có cha)

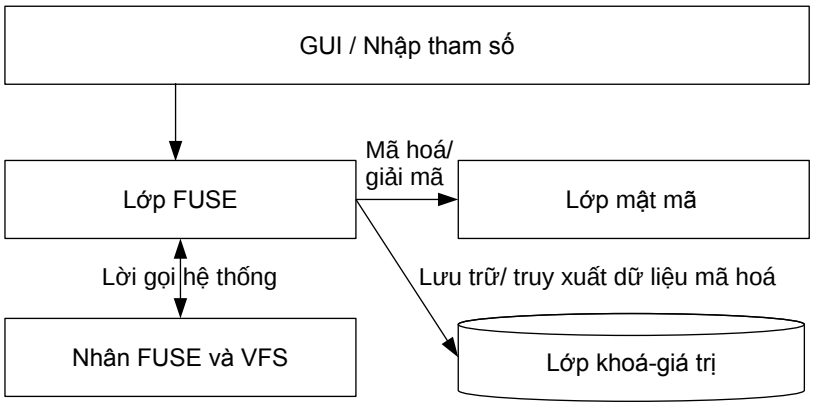
3.2.2. Xây dựng hệ thống tệp mã hóa trên Linux sử dụng kho lưu trữ khóa-giá trị

Việc mã/giải mã cây KMT được thực hiện bởi DO trước khi bắt đầu phiên làm việc. DO giải mã cây KMT bằng cách nhập mật khẩu riêng của mình. Cây KMT được lưu trữ trong tệp tin do DO (người dùng hệ điều hành) tạo ra. Để bảo vệ tệp người dùng, ta có thể mã hoá tệp bởi sự hỗ trợ của hệ điều hành. Trong phần này, luận án đề xuất một phương pháp xây dựng một hệ

thống mã hoá tệp người dùng gọi là KVEFS dựa trên các thư viện libfuse, openssl và openstars.

3.2.2.1. Mô hình của KVEFS

Mô hình của KVEFS gồm có 4 lớp: Lớp giao diện đồ hoạ người dùng (Graphical User Interface - GUI) là giao diện của hệ thống để người dùng dễ sử dụng và lấy các tham số của kho lưu trữ khoá-giá trị và các tuỳ chọn thao tác mã hóa; Lớp FUSE là lớp chính của hệ thống để giao tiếp với nhân của hệ điều hành và thao tác với các tệp và thư mục; Lớp mật mã để mã hóa/giải mã dữ liệu khi đọc hoặc ghi từ lưu trữ khóa-giá trị; Lớp khoá-giá trị để lưu trữ thông tin mã và dữ liệu về cấu trúc tệp và thư mục (Hình 3.2).



Hình 3.2: Mô hình của KVEFS

Các lớp của KVEFS cụ thể như sau:

Lớp GUI và nhập tham số: Giao diện dòng lệnh khó sử dụng cho những người ít hiểu biết về linux, vì vậy việc tạo giao diện đơn giản là điều cần thiết để người dùng dễ sử dụng chương trình. Mô-đun GUI chịu trách nhiệm xây dựng giao diện đơn giản, dễ sử dụng. Lớp GUI sẽ thực hiện gắn (mounting) tệp tin và gỡ (unmounting) tệp tin.

Lớp FUSE: Lớp này chịu trách nhiệm giao tiếp với nhân hệ điều hành để thực hiện các thao tác trên các tệp và thư mục như: Mở tệp tin, ghi vào tệp tin, tạo thư mục (mkdir),... KVEFS đã sử dụng thư viện *libfuse* để thực hiện công việc này. Lớp này cũng giao tiếp với lớp mật mã để mã hóa và giải mã

dữ liệu khi ghi và đọc.

Lớp mật mã: Để dữ liệu được bảo mật và không thể đọc được bởi một hệ thống khác, KVEFS cần một mô-đun chịu trách nhiệm mã hóa và giải mã dữ liệu, mã hóa dữ liệu trước khi dữ liệu được lưu vào CSDL. Để làm điều này, KVEFS sử dụng thuật toán mã hóa AES trong thư viện openssl. Tất cả các khóa và giá trị trong lớp khóa-giá trị được mã hóa trước khi lưu vào CSDL khóa-giá trị.

Lớp khoá-giá trị: Lớp này lưu trữ siêu dữ liệu của hệ thống tệp và dữ liệu của tệp. Cột khóa lưu trữ thông tin về tên tệp hoặc đường dẫn thư mục, cột giá trị chứa nội dung tệp hoặc các mục có trong thư mục. Trong quá trình triển khai, KVEFS sử dụng thư viện OpenStars với tuỳ chọn lưu trữ khóa-giá trị. Lớp này có thể điều chỉnh để sử dụng nhiều loại lưu trữ khóa-giá trị.

3.2.2.2. Quá trình hoạt động của KVEFS

Khi tạo hệ thống tệp bằng cách sử dụng lớp GUI và nhập tham số, hệ thống sẽ khởi tạo các hàm trong lớp FUSE. Các hàm trong Lớp FUSE có khả năng lấy các thuộc tính của tệp tin hoặc thư mục (getattr), đọc tất cả các mục trong một thư mục (readdir), đọc tệp (read), ghi tệp (write), tạo thư mục (mkdir), xóa thư mục (rmdir), xóa tệp (unlink)... Các hàm được khởi tạo sau khi hàm fuse_main() được chạy. Khi fuse_main() khởi chạy, một vòng lặp vô hạn được tạo ra để đáp ứng mọi hoạt động thời gian thực của người dùng. Điều đó có nghĩa là các hàm lấy thuộc tính tệp hoặc đọc tệp, ghi tệp được gọi liên tục bất cứ khi nào người dùng tương tác với hệ thống tệp.

Mục tiêu của KVEFS là thực hiện các hàm thao tác tệp thông qua *libfuse*. Tất cả các thao tác mã hóa và giải mã được thực hiện trong các hàm callback của *libfuse*. Các hàm trong *libfuse* có một điểm chung là chúng có tham số đường dẫn tệp hoặc thư mục, vì vậy KVEFS đã thiết kế lưu trữ tên đường dẫn của tệp hoặc thư mục vào trong cột khóa và lưu trữ nội dung của tệp hoặc danh sách tệp và thư mục con có trong thư mục trong cột giá trị. Cả

khóa và giá trị đều được mã hóa trước khi lưu vào CSDL khóa-giá trị. Như vậy, KVEFS có thể bảo vệ cấu trúc thư mục một cách hiệu quả. Để phân biệt một tệp từ một thư mục, KVEFS sử dụng tiền tố trước tên đường dẫn, tiền tố là FILE nếu đường dẫn trỏ đến tệp, tiền tố là DIR nếu đường dẫn trỏ đến thư mục. Thuật toán đọc, ghi file được mô tả như trong thuật toán 3.1, 3.2.

Trong hàm đọc, KVEFS tìm nội dung của tệp thông qua đường dẫn của chúng và truy xuất vào kho lưu trữ khoá-giá trị, bởi vì khoá lưu trữ đường dẫn, giá trị lưu nội dung của tệp, do đó, dữ liệu của tệp được đọc từ kho lưu trữ CTX_DB và giải mã.

Thuật toán 3.1: Thuật toán đọc file

```
1 Function KVEFS_read(path, buf, size, offset):
2     key = path_to_key(path, keylen, 0)
3     val = db_get(CTX_DB, key, keylen, vallen)
4     memcpy(buf, val+offset, size)
5 End Function
```

Trong hàm ghi, KVEFS mã hóa dữ liệu của tệp trước khi lưu trữ vào kho lưu trữ CTX_DB.

Thuật toán 3.2: Thuật toán ghi file

```
1 Function KVEFS_write(path, buf, size, offset):
2     key = path_to_key(path, keylen, 0)
3     val = db_get(CTX_DB, key, keylen, vallen)
4     memcpy(val, buf, size)
5     db_put(CTX_DB, key, keylen, val, vallen)
6 End Function
```

Khi thực hiện các hàm đọc, ghi tệp thì cần các thao tác mã/giải mã nội dung tệp từ kho lưu trữ khoá-giá trị. Các hàm mã/giải mã nội dung tệp được thực hiện như thuật toán 3.3, 3.4.

Thuật toán 3.3: Thuật toán mã hoá và lưu dữ liệu vào kho lưu trữ

```
1 Function db_put(db, key, klen, val, vlen):
2     //encrypt
3     cipher = calloc(vlen, sizeof(char))
4     encrypt_stream(val, cipher, vlen)
5     db->put(db->db, key, klen, cipher, vlen )
6 End Function
```

Thuật toán 3.4: Thuật toán giải mã dữ liệu từ kho lưu trữ

```
1 Function db_get(db, key, klen, vlen):
2     val = db->get(db->db, key, klen, vlen)
3     //decrypt
4     plaintxt = calloc(vlen, sizeof(char))
5     decrypt_stream(val, plaintxt, vlen)
6     return plaintxt
7 End Function
```

3.2.2.3. Quản lý khóa mã hoá tệp

Quá trình mã/giải mã tệp người dùng thì cần phải có khoá mã. Việc tạo và quản lý khoá mã được thực hiện theo các bước:

Đầu phiên, KVEFS tạo một khóa ngẫu nhiên cho AES:

aesDataKey = generateRandomKey()

Người dùng phải nhập mật khẩu và KVEFS sử dụng hàm dẫn xuất khoá để tạo ra một khóa gọi là *tempAESKey* từ mật khẩu người dùng:

tempAESKey = kdf(userPassword)

KVEFS mã hóa *aesDataKey* bằng *tempAESKey*:

eeKey = aesEncrypt(aesDataKey, tempAESKey)

KVEFS lưu trữ *eeKey* để lưu trữ khóa-giá trị bằng một khóa đặc biệt. Trước khi gắn hệ thống tệp, người dùng phải nhập mật khẩu để giải mã *aesDataKey*

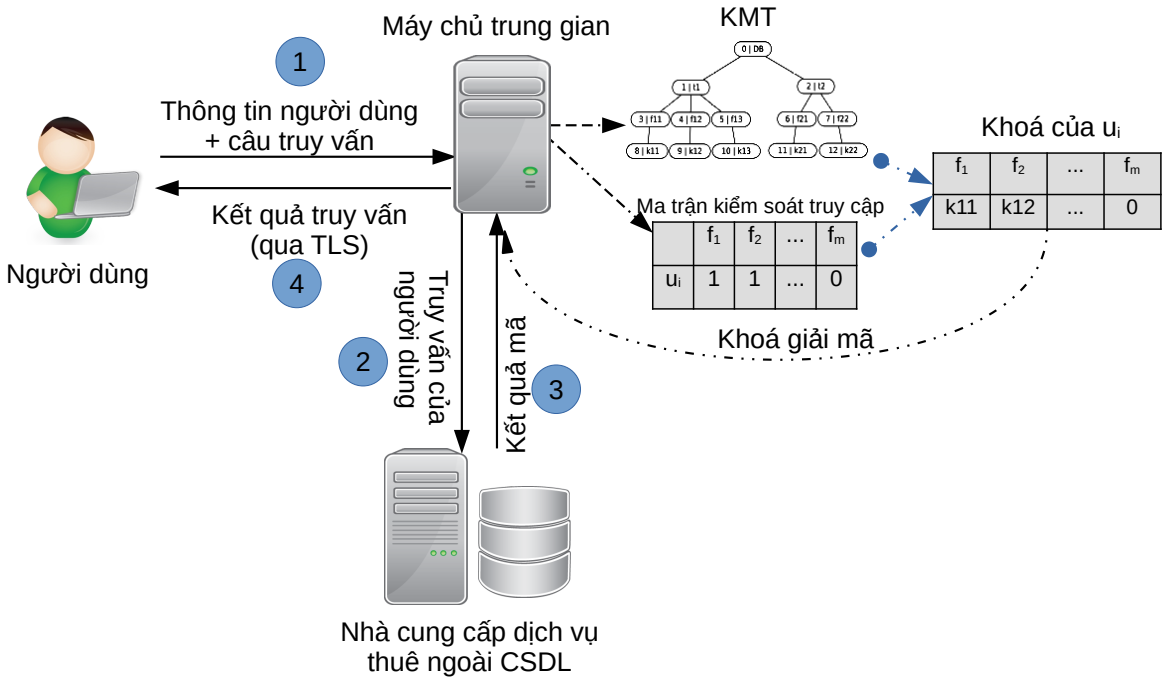
từ *eeKey* được lưu trữ. Để thay đổi mật khẩu, trước tiên người dùng giải mã và nhận *aesDataKey* bằng mật khẩu cũ và mã hóa lại bằng mật khẩu mới và cuối cùng lưu trữ khóa *eeKey* mới vào kho khóa-giá trị.

3.2.3. Mô hình quản lý truy cập dữ liệu mức cột của người dùng

Mỗi người dùng có một định danh riêng và có quyền truy cập dữ liệu mức cột khác nhau do DO quản lý. Khi người dùng u_i truy vấn dữ liệu và được máy chủ trung gian xác thực thì DO dựa vào ma trận quản lý truy cập $\mathscr{A}$ để có được các tên cột dữ liệu mà u_i có thể truy cập, đồng thời giải mã cây KMT từ đó biết được các khoá của các cột tương ứng quyền truy cập người dùng và tạo ra một bảng khoá tạm thời $\mathscr{K}_{u_i}$. Bảng $\mathscr{K}_{u_i}$ này giữ ở máy chủ trung gian và không can thiệp được từ người dùng. Sau đó, u_i truy vấn dữ liệu theo ma trận kiểm soát truy cập để lấy thông tin cần thiết và DO giải mã dữ liệu này bằng các khoá ở bảng $\mathscr{K}_{u_i}$. Việc trao đổi thông tin người dùng, kết quả truy vấn trả về giữa người dùng và máy chủ trung gian thông qua giao thức TLS (Transport Layer Security) để bảo đảm dữ liệu không bị tấn công theo dạng man in the middle.

Cơ chế quản lý truy cập dữ liệu của người dùng được thực hiện qua 4 bước như sau:

- Bước 1: Người dùng gửi thông tin người dùng thông qua giao thức TLS. Sau khi được xác thực, người dùng gửi câu truy vấn dữ liệu đến máy chủ trung gian. Máy chủ trung gian dựa vào ma trận kiểm soát truy cập và dùng khoá mã DO để giải mã cây KMT. Từ đó, máy chủ đưa ra được bảng khoá của người dùng.
- Bước 2: Máy chủ trung gian dựa vào ma trận kiểm soát truy cập để truy vấn các cột dữ liệu mà người dùng được phép truy cập trên DSP.
- Bước 3: DSP trả kết quả truy vấn là dữ liệu mã về máy chủ trung gian.
- Bước 4: Máy chủ trung gian dựa vào bảng khoá của người dùng và giải



Hình 3.3: Mô hình truy cập dữ liệu mức cột của người dùng

mã dữ liệu, trả kết quả dữ liệu rõ về cho người dùng thông qua giao thức TLS.

Với cơ chế truy cập dữ liệu của người dùng được đề xuất thì người dùng chỉ quản lý thông tin người dùng mà không nắm giữ bất kỳ thông tin khoá của cây khoá KMT nên không thể tấn công theo hình thức thoả hiệp. Bên cạnh đó, do cây khoá được quản lý ở máy chủ trung gian và số lượng khoá tương ứng với số cột của các bảng trong CSDL nên việc thay khoá tương đối dễ dàng, không ảnh hưởng và cũng không phụ thuộc đến người dùng tham gia trong hệ thống.

3.3. Đề xuất phương pháp thay đổi khóa mã cho ODBS

Bài toán thay đổi khóa là quá trình thực hiện thay đổi lại khoá mã của các dữ liệu mã trên ODBS. Bài toán này thường được đặt ra trong trường hợp DO lo ngại khóa của CSDL không còn an toàn.

3.3.1. Phương pháp đổi khoá ngây thơ

Đổi khoá CSDL mức cột là thay đổi khoá cũ của các cột trong bảng dữ liệu bằng các khoá mới tương ứng của cột đó. Để đổi khoá cho CSDL mã, ta phải tạo ra cây khoá KMT mới (bài toán tổng quát là thay đổi khoá của tất cả các cột trong bảng, các trường hợp riêng như thay đổi một số cột cụ thể hoặc mã hoá một số cột nhạy cảm trong CSDL là tập con của bài toán này) và tiến hành thay khoá cũ bằng khoá mới tương ứng từ cây khoá KMT cũ và KMT mới trên các dữ liệu trong bảng.

Thuật toán 3.5: Thuật toán NaiveKeyChanged

Input: Outsourced database name

Output: Change key of encrypted data

```
1  listOldKey ← readKMT("OldKMT.file")
2  listNewKey ← readKMT("NewKMT.file")
3  db ← connect to outsourced database
4  DataTable dt ← SELECT * FROM tableName
5  DELETE * FROM tableName
6  while dt.nextRow() do
7  |  list[] ← ∅
8  |  for each value ∈ dt.column do
9  |  |  plaintext = D_listOldKey(i)(value)
10 |  |  ciphertext = E_listNewKey(i)(plaintext)
11 |  |  list.append(ciphertext)
12 |  end
13 |  INSERT INTO tableName VALUES(list)
14 end
```

Phương pháp ngây thơ (naive method) giải quyết bài toán thay khóa được mô tả như sau:

- Bước 1: DO tải toàn bộ CSDL về máy chủ của mình,

- Bước 2: Giải mã CSDL bằng khoá đã có,
- Bước 3: Mã hoá toàn bộ dữ liệu bằng khóa mã mới,
- Bước 4: Lưu trữ dữ liệu được mã lên ODBS.

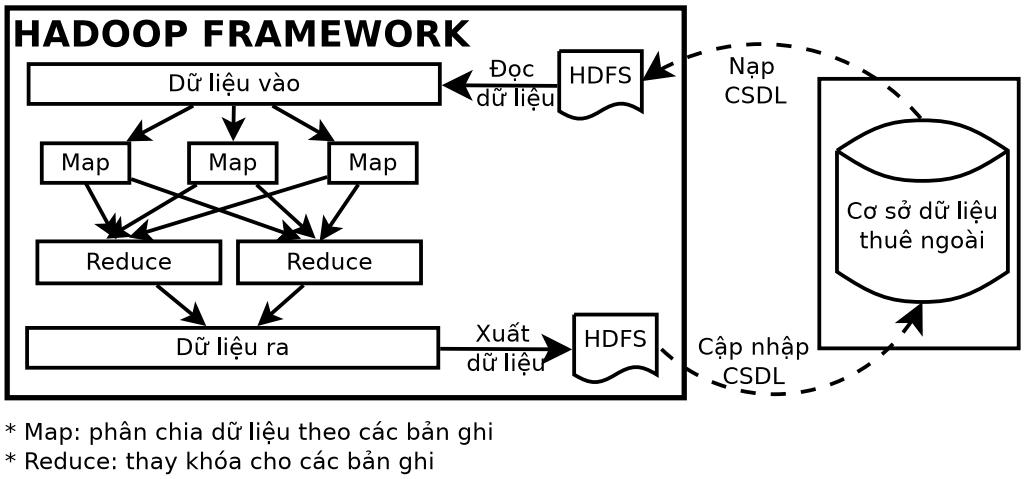
Phương pháp đổi khoá ngây thơ có thể được thực hiện như thuật toán 3.5. Ta xem thời gian thực hiện giải mã, mã hóa đối với các dữ liệu đầu vào khác nhau là thay đổi không đáng kể; Cho CSDL có bảng $\mathscr{T}$ với n dòng và m cột. Độ phức tạp thời gian tính toán của thuật toán 3.5 là: $O(n)) = 2.t_{read} + n.m.(t_{dec} + t_{enc})$. Vì vậy, khi CSDL có số lượng bản ghi n lớn thì phương pháp ngây thơ sẽ tốn nhiều thời gian và có thể không thực hiện được.

3.3.2. Đề xuất phương pháp đổi khoá dựa trên MapReduce

Luận án đề xuất thuật toán đổi khoá của dữ liệu mã ở mức cột thực hiện dựa trên MapReduce theo 3 bước sau đây:

- Bước 1: Chuyển dữ liệu mã của các bảng từ ODBS thành các tệp tin có định dạng HDFS trên hadoop framework.
- Bước 2: Sử dụng MapReduce để thực hiện đổi khoá với các tệp tin đầu vào chứa nội dung là các bảng dữ liệu. Quá trình này được thực hiện song song trên các cụm máy tính. Dữ liệu đầu ra là các tệp tin HDFS. (thuật toán 3.6)
- Bước 3: Cập nhập dữ liệu mã đã đổi khóa trong các tệp tin HDFS của hadoop lên ODBS.

Quá trình thay khóa dữ liệu mã được mô tả như trong hình 3.4. Trong phương pháp này, Hadoop framework được cấu hình trên máy chủ trung gian do DO toàn quyền quản lý. CSDL thuê ngoài được quản lý bởi DSP. Quá trình nạp và cập nhận CSDL thì dữ liệu được truyền qua mạng Internet. Dữ liệu từ các tệp tin HDFS được xử lý qua hai giai đoạn: Map và Reduce. Giai



Hình 3.4: Mô hình đổi khoá dựa trên MapReduce

đoạn Map, các dòng dữ liệu của bản ghi được chia thành nhiều phần và xử lý trên các cụm máy tính(cluster); Giai đoạn Reduce tiến hành giải mã và mã hóa (thay khóa) các giá trị kết quả của giai đoạn map. Kết thúc hai giai đoạn ta thu được tệp tin mã hóa với khóa mã mới.

Thời gian thực hiện đổi khoá theo mức cột là tổng thời gian của quá trình chuyển dữ liệu từ OBDS thành các tệp tin định dạng HDFS trên hadoop, thời gian xử lý thay khóa trên các bảng dữ liệu và thời gian cập nhập dữ liệu từ hadoop lên máy chủ dịch vụ. Trong đó, ta quan tâm đến thời gian xử lý thay khóa, vì lúc này số lượng dữ liệu cần xử lý là lớn nhất. Thời gian chuyển đổi dữ liệu từ DSP vào các trung tâm tính toán của MapReduce và ngược lại phụ thuộc quá trình phân chia xử lý, tính toán được quản lý bởi các cụm máy tính. Giai đoạn thực hiện đổi khoá mức cột của bảng CSDL được đề xuất trong thuật toán 3.6.

Hàm map() và reduce() làm việc trên các nút nên số lần thực hiện đồng thời của nó là x (x là số nút của MapReduce). Thời gian cho một lần lặp dữ liệu trong map() và reduce() trên 1 nút là n (n là số lượng bản ghi). Như vậy, thời gian cần thực hiện xong map() và reduce() trên x nút là n/x. Trong hàm reduce() thực hiện việc đổi khoá, nghĩa là giải mã và mã hoá lại dữ liệu nhưng vẫn giữ nguyên số lượng các bản ghi. Số lượng cột trong

Thuật toán 3.6: Thuật toán MR-EncColumnKeyChange

Input: HDFS file after imported database.

Output: HDFS file with key change.

```
1  listOldKey ← readKMT("OldKMT.file")
2  listNewKey ← readKMT("NewKMT.file")
3  Send (listOldKey, listNewKey) to REDUCE function
4  Function MAP(key, value):
5      emit(key, value)
6  End Function
7  Function REDUCE(key, value):
8      list[] ← split key by ', '
9      i=0
10     for each data ∈ list do
11         plaintext = D_listOldKey(i)(data)
12         ciphertext = E_listNewKey(i)(plaintext)
13         newkey.concat(ciphertext + "|")
14         i++
15     end
16     emit(newkey, value)
17 End Function
```

bảng CSDL là m. Như vậy, thời gian thực hiện đổi khoá xong mỗi bản ghi là $m.(t_{dec} + t_{enc})$. Do đó, độ phức tạp thời gian tính toán của thuật toán 3.6 là $O(n) = (n/x).m(t_{dec} + t_{enc})$.

3.3.3. Đảm bảo an toàn cho phương pháp thay đổi khoá

Phương pháp thay đổi khóa thực hiện chủ yếu trên Hadoop framework. Trong Hadoop, dữ liệu được lưu trữ phân tán trên nhiều DataNode, trong khi NameNode lưu trữ siêu dữ liệu và nhật ký hoạt động. Giao tiếp thực tế của các khối dữ liệu xảy ra giữa Client Node và DataNode. Vì vậy, một số vấn đề liên quan đến an toàn trong Hadoop framework [66] là:

1. Dữ liệu bị phân mảnh: Các cụm dữ liệu lớn cho phép có nhiều bản sao dữ liệu trao đổi giữa các nút khác nhau để đảm bảo dự phòng và khả năng phục hồi. Từ đó gây khó khăn trong việc bảo mật các bản sao dữ liệu. Từ đó gây khó khăn trong việc bảo mật các bản sao dữ liệu.
2. Máy tính phân tán: Hadoop framework có thể được cấu hình trên hệ thống máy tính phân tán để tận dụng tối đa tài nguyên. Vì vậy, để bảo mật dữ liệu trong Hadoop framework đòi hỏi cần phải bảo vệ tất cả các máy tính tham gia trong hệ thống tính toán.
3. DataNodes không có cơ chế kiểm soát truy cập. Do đó, kẻ tấn công có thể đọc các khối dữ liệu tùy ý từ DataNodes. Ngoài ra, kẻ tấn công có thể truy cập tệp HDFS qua RPC hoặc qua các giao thức HTTP mà không có bất kỳ sự kiểm soát nào.

Từ những nhận định trên, để đảm bảo an toàn cho phương pháp thay đổi khóa, ta cần thực hiện một số nội dung sau:

1. Đảm bảo tính toàn vẹn dữ liệu: Thực hiện tính toán số lượng bản ghi của các bảng thay đổi khóa trước và sau khi đổi khóa và so sánh hai dữ liệu này, tránh trường hợp lỗi hoặc sai sót dữ liệu khi tính toán trên DataNode và các cụm máy tính.
2. Do Hadoop framework triển khai trên máy chủ trung gian nên DO cần có cơ chế hạn chế truy cập tệp HDFS qua RPC hoặc qua các giao thức HTTP. Ngoài ra, cần xóa hết dữ liệu lưu trữ liên quan đến dữ liệu đổi khóa trên Hadoop framework sau mỗi phiên làm việc.
3. Mặc dù dữ liệu trong giai đoạn nạp và cập nhập dữ liệu là dữ liệu mã, tuy nhiên, để tránh các tấn công thay đổi dữ liệu trên đường truyền, ta nên dùng giao thức TLS để trao đổi dữ liệu.

3.4. Phân tích, thử nghiệm các phương pháp đề xuất

3.4.1. Phân tích KVEFS

Vì sử dụng thư viện lưu trữ khóa-giá trị OpenStars, KVEFS có thể tự động chọn các kho lưu trữ khóa-giá trị khác nhau và dữ liệu có thể được phân tán với RemoteKVStorage. Trong khi đó, EncFS [26] chỉ có thể lưu trữ dữ liệu trong hệ thống tệp cục bộ của hệ điều hành. Ta có thể so sánh sự khác biệt giữa KVEFS với EncFS như thể hiện trong bảng 3.3.

Bảng 3.3: So sánh giữa KVEFS và Encfs

Tính năng	KVEFS	EncFS [26]
Độ an toàn	AES 256 bit	AES, Blowfish
Lưu trữ có thể tùy chỉnh	Có, người dùng có thể chọn loại lưu trữ	Không, ghi trực tiếp vào tệp
Khả năng phân tán	Có, với RemoteKVStorage	Không, dữ liệu phải phù hợp với máy chủ cục bộ
Ẩn cấu trúc thư mục	Có, ẩn tất cả cấu trúc thư mục trong một lưu trữ khóa-giá trị	Không, Ta có thể thấy các mục trong một thư mục và thời gian truy cập

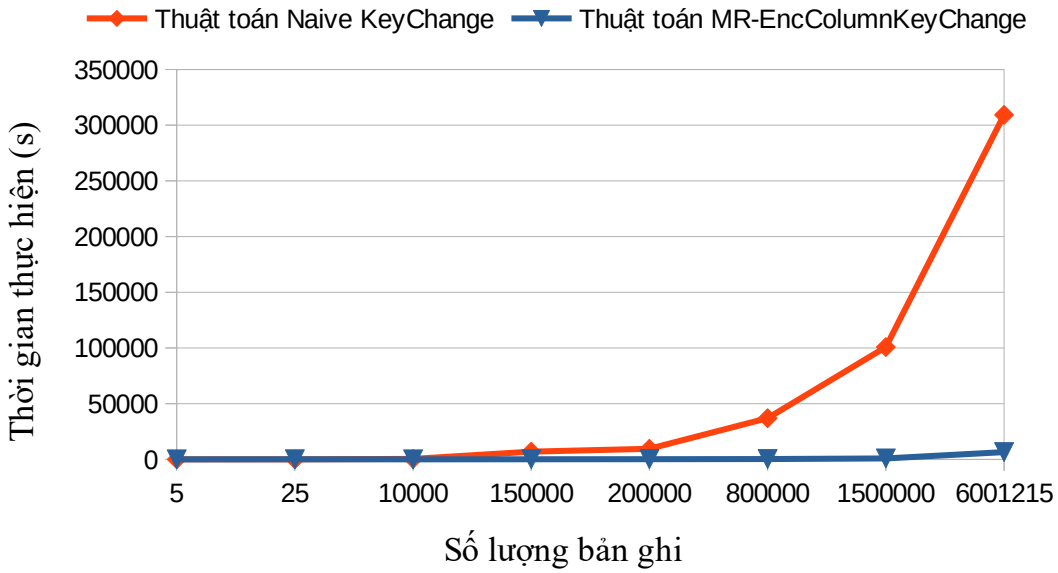
3.4.2. Thử nghiệm phương pháp thay đổi khoá

Trong quá trình nghiên cứu, luận án chưa phát hiện được công bố về phương pháp đổi khoá mã của CSDL. Vì vậy, luận án đánh giá kết quả giữa hai phương pháp: Đổi khoá ngây thơ và thuật toán 3.6. Luận án sử dụng Hadoop 2.7.0 [6] với chế độ mặc định của Hadoop chạy trên một máy tính và 1GB CSDL của TPC-H để thay đổi khoá tất cả các cột dữ liệu trong các bảng CSDL. Hai phương pháp được cài đặt bằng ngôn ngữ lập trình Java và thực hiện trên máy tính Core™ i7-6700 CPU @ 3.40GHz, Ram 8GB, ổ cứng HDD 1TB. Hệ điều hành Ubuntu 16.10. Sử dụng sqoop-1.4.4 [7] để chuyển đổi dữ liệu giữa CSDL và HDFS. Kết quả thời gian tính toán được mô tả trong bảng 3.4.

Bảng 3.4: Thời gian đổi khoá của thuật toán 3.5 và thuật toán 3.6

Tên bảng	Số bản ghi	Thuật toán Naive KeyChange(s)	Bước 1 CSDL → HDFS(s)	Bước 2 Đổi khoá(s)	Bước 3 HDFS → CSDL(s)
region	5	0,595	21,734	17,518	19,521
nation	25	1,499	16,571	18,67	38,101
supplier	10000	449,528	16,766	19,995	25,807
customer	150000	6.908,127	18,897	36,312	66,872
part	200000	9.526,048	28,198	43,239	92,563
partsupp	800000	36.963,637	26,125	76,68	219,674
orders	1500000	100.763,852	42,734	202,997	618,088
lineitem	6001215	309.241,736	218,252	1.376,162	4.963,679

Thời gian thực hiện đổi khoá mức cột trên MapReduce: $t_{MR} = t_I + t_C + t_E$; Trong đó t_I là thời gian nạp các bảng từ ODBS thành các tệp tin định dạng HDFS trên Hadoop, t_C là thời gian thực hiện thay đổi khoá trên MapReduce, và t_E là thời gian xuất dữ liệu từ Hadoop lên ODBS. Dựa vào bảng 3.4, toàn bộ thời gian đổi khoá của phương pháp đề xuất được so sánh với phương pháp NaiveKeyChange như hình 3.5.



Hình 3.5: Kết quả thời gian thực hiện đổi khoá

Kết quả thực nghiệm cho thấy, khi dữ liệu các bảng có ít bản ghi như region, nation thì thời gian thực hiện ngây thơ nhanh hơn so với phương pháp đề xuất tương ứng là 0.6s/58.773s, 1.5s/73.342s. Điều này xảy ra bởi vì

thuật toán 3.5 duyệt lần lượt các bản ghi và đổi khoá. Khi số lượng bản ghi ít (region: 5 dòng, nation: 25 dòng) thì quá trình duyệt và xử lý sẽ nhanh. Trong khi đó, thuật toán 3.6 sẽ thực hiện xử lý phân tán lên các nút (Map) rồi mới thực hiện đổi khoá (Reduce) nên sẽ chậm hơn thuật toán 3.5. Tuy nhiên, khi các bảng có số lượng bản ghi đáng kể như supplier với 10.000 dòng thì thời gian thực hiện của phương pháp ngây thơ là 7.4921 phút trong khi phương pháp đề xuất chỉ 62.568s. Thậm chí đối với bảng orders có 1.5 triệu bản ghi thì thời gian thực hiện của phương pháp ngây thơ $\approx$ 28 giờ, quá lớn so với 14 phút của phương pháp đề xuất. Khi số lượng bản ghi lớn như bảng lineitem với khoảng 6 triệu thì phương pháp ngây thơ thực hiện khoảng 3.57 ngày, quá lâu để tạo một phiên bảo trì CSDL và xem như không khả thi cho tính sẵn sàng của dữ liệu, trong khi phương pháp đề xuất thực hiện đổi khoá trong khoảng thời gian 1.8 giờ. Hình 3.5 cho thấy tính hiệu quả của thời gian thực hiện phương pháp đề xuất so với phương pháp ngây thơ khi số lượng bản ghi lớn. Khi dữ liệu càng lớn, hiệu quả phương pháp đề xuất càng cao.

3.4.3. Vấn đề tồn tại của phương pháp đề xuất

Trong quá trình đổi khoá, DO phải thực hiện chuyển đổi CSDL sang HDFS và ngược lại. Trong hai quá trình này, DO phải sử dụng công cụ chuyển đổi sqoop của Apache. Đây là phần mềm của bên thứ ba nên DO không kiểm soát được về công nghệ. Vì vậy, phương pháp đề xuất vẫn chưa giải quyết được trường hợp dữ liệu bị gián đoạn, hư hại khi chuyển tải dữ liệu do đường truyền hoặc phần mềm sqoop tạo ra.

3.5. Kết luận

Chương 3 giới thiệu về quản lý khoá của người dùng và bài toán thay đổi khóa cho ODBS. Luận án dùng một máy chủ trung gian để lưu trữ cây khóa và quyền truy cập của người dùng. Cây khóa cũng được mã hóa và chỉ có DO mới có quyền giải mã. Khi bắt đầu phiên làm việc, DO cung cấp mật khẩu

MK hoặc có thể dùng một USB token xác thực để giải mã cây khóa. Người dùng chỉ quản lý thông tin người dùng và không nắm giữ bất kỳ thông tin về khoá mã của CSDL. Khi người dùng truy cập dữ liệu, máy chủ trung gian sẽ dựa vào cây khoá và ma trận quản lý truy cập người dùng để có thể đưa ra khoá mã tương ứng với dữ liệu mà người dùng được quyền truy cập và DO sử dụng khoá đấy để giải mã dữ liệu. Do người dùng không quản lý khoá mã nên không thể tấn công theo hình thức thoả hiệp. Bên cạnh đó, số lượng khoá mã của CSDL bằng với số cột trong các bảng và được quản lý ở máy chủ trung gian nên việc thay khoá tương đối dễ dàng, không phụ thuộc đến người dùng tham gia trong hệ thống. Luận án cũng đề xuất thuật toán thay đổi khóa dựa trên mô hình MapReduce với thời gian thấp hơn đáng kể so với phương pháp tải toàn bộ CSDL về để thay đổi khóa. Vì vậy, phương pháp này đảm bảo cho CSDL luôn sẵn sàng hoạt động khi thay khóa. Kết quả chứng minh được khi dữ liệu càng lớn, tính hiệu quả của phương pháp càng cao. Ngoài nền tảng Hadoop, hướng nghiên cứu về mô hình, thuật toán đổi khoá trên nền tảng Spark có khả năng mang lại kết quả tốt hơn nếu như bỏ qua việc quan tâm đến chi phí đầu tư hay thuê dịch vụ Spark. Hoặc việc kết hợp cả hai nền tảng: nếu dung lượng bảng của CSDL thấp hơn bộ nhớ RAM thì dùng nền tảng Spark, ngược lại thì dùng nền tảng Hadoop. Tuy nhiên, nhược điểm của các nền tảng này là không kiểm soát được hai giai đoạn là chuyển đổi CSDL $\rightarrow$ HDFS và HDFS $\rightarrow$ CSDL. Trong tương lai, hướng nghiên cứu xây dựng một mô hình đổi khoá trực tiếp trên CSDL không phụ thuộc vào nền tảng của bên thứ ba sẽ khắc phục được nhược điểm này.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Luận án đã tiến hành khảo sát những kiến thức cơ bản về ODBS, các bài toán liên quan đến bảo đảm an toàn cho CSDL trên môi trường ODBS và các công trình nghiên cứu trong và ngoài nước có liên quan đến an toàn ODBS. Trên cơ sở nghiên cứu lý thuyết, luận án đã đề xuất các mô hình, thuật toán nhằm mục đích bảo vệ an toàn cho CSDL trên ODBS. Những kết quả thử nghiệm cho thấy tính hiệu quả của các phương pháp đề xuất, từ đó có thể áp dụng các mô hình, thuật toán được đề xuất để có thể triển khai vào các ứng dụng cụ thể.

A. Kết quả đạt được của luận án

1. Luận án đề xuất thuật toán giảm thời gian truy vấn trên dữ liệu mã dựa trên xử lý song song của CPU. Khi thực hiện truy vấn, kết quả trả về là một tập dữ liệu, ta chia nhỏ tập này để thực hiện xử lý song song. Phương pháp đề xuất sẽ giảm thời gian trong các tiến trình giải mã, tính toán dữ liệu và xác thực dữ liệu từ đó giảm thời gian thực hiện truy vấn.

2. Đề xuất hai thuật toán xác thực lô dựa trên hai bài toán khó. Thuật toán đề xuất giúp xác thực nhiều chữ ký cùng một lúc trong một phương trình xác thực. Đề xuất mô hình, thuật toán xác thực dữ liệu khi truy vấn trên CSDL mã thuê ngoài dựa trên xác thực lô. Phương pháp xác thực đề xuất hiệu quả trong trường hợp CSDL động. Khi dữ liệu thay đổi, DO chỉ tính toán chữ ký dựa trên giá trị thay đổi mà không tính lại toàn bộ giá trị xác thực trên dữ liệu như dùng phương pháp ADS.

3. Đề xuất cây KMT để quản lý khoá mã của CSDL. Cây KMT được mã hóa và chỉ có DO mới có thể mở được cây khóa trước phiên làm việc. Đề xuất phương pháp mã hoá hệ thống tệp tin (KVEFS), sử dụng để lưu trữ dữ liệu nhạy cảm, bảo vệ dữ liệu trong suốt đối với người dùng

bằng lưu trữ khóa-giá trị. Mô hình quản lý truy cập giúp DO kiểm soát quyền người dùng khi truy vấn dữ liệu. Phương pháp này sẽ giúp DO quản lý khoá và người dùng không giữ khoá, vì vậy tránh được tấn công theo hình thức "thoả hiệp". Bên cạnh đó, luận án đề xuất thuật toán đổi khoá mã của CSDL thuê ngoài ở mức cột dựa trên MapReduce. Đề xuất này giúp cho DO có thể thay đổi khoá mã của CSDL khi nghi ngờ khoá không còn an toàn.

B. Hướng phát triển tiếp theo

Mặc dù nghiên cứu sinh đã cố gắng trong việc nghiên cứu lý thuyết và đề xuất các mô hình, thuật toán nhằm nâng cao tính an toàn cho hệ thống ODBS. Tuy nhiên, theo nhận định chủ quan của nghiên cứu sinh thì vẫn có một số vấn đề cần tiếp tục được nghiên cứu và phát triển tiếp theo:

- Nghiên cứu, phát triển mô hình, thuật toán truy vấn trên CSDL mã thực hiện được các loại câu truy vấn trên ODBS kết hợp xử lý song song.
- Nghiên cứu mô hình đổi khóa đặc trưng cho CSDL quan hệ mà không phụ thuộc vào nền tảng công nghệ của bên thứ ba.
- Nghiên cứu các thuật toán xác thực dữ liệu đảm bảo tính đủ và tính mới thích hợp với CSDL động.

DANH MỤC CÁC CÔNG TRÌNH CÔNG BỐ

A. CÁC CÔNG TRÌNH SỬ DỤNG TRONG LUẬN ÁN

[CT1] **Kim Giau Ho**, Ly Vu, Nam Hai Nguyen, and Hieu Minh Nguyen, "Speed Up Querying Encrypted Data on Outsourced Database", In *Proceedings of the 2017 International Conference on Machine Learning and Soft Computing (ICMLSC '17)*, ISBN: 978-1-4503-4828-7, 2017, pp. 47–52. ACM. http://doi.acm.org/10.1145/3036290.3036299.

[CT2] **Giau Ho Kim**, Manh Tran Cong, and Minh Nguyen Hieu, "A Study on Batch Verification Scheme in Outsourced Encrypted Database", In *The Tenth International Symposium on Information and Communication Technology (SoICT 2019)*, ISBN: 978-1-4503-7245-9/19/12, 2019, pp. 502-507. ACM. http://doi.acm.org/10.1145/3368926.3369732.

[CT3] **Hồ Kim Giàu**, Nguyễn Hiếu Minh, "Xác thực cơ sở dữ liệu mã hoá thuê ngoài dựa trên xác thực lô", trong *Tạp chí Khoa học và Công nghệ - Đại học Thái Nguyên*, ISSN: 1859-2171, Tập 204, Số 11, 2019, pp. 107-116.

[CT4] **Kim Giau Ho**, Son Hai Le, Trung Manh Nguyen, Vu Thi Ly, Thanh Nguyen Kim, Nguyen Van Cuong, Thanh Nguyen Trung, and Ta Minh Thanh. "KVEFS: Encrypted File System based on Distributed Key-Value Stores and FUSE." In *International Journal of Network Security & Its Applications (IJNSA)*, ISSN: 0975-2307, Vol 11(2), 2019, pp. 55-66.

[CT5] **Hồ Kim Giàu**, Nguyễn Hiếu Minh, "Quản lý và thay đổi khoá cơ sở dữ liệu mã hoá trên môi trường thuê ngoài", trong *Chuyên san Công nghệ thông tin và Truyền thông - Học viện KTQS (LQDTU-JICT)*, ISSN: 1859-0209, Số 13, 2019, pp. 77-90.

[CT6] Lều Đức Tân, **Hồ Kim Giàu**, "Khắc phục lỗi và nâng cao tính hiệu quả cho các lược đồ chữ ký số dựa trên hai bài toán khó", trong *Tạp chí Khoa học Công nghệ Thông tin và Truyền thông - Học viện Công nghệ BCVT*, ISSN: 2525-2224, Số 01(CS.01), 2020, pp. 50-56.

B. CÁC CÔNG TRÌNH CÔNG BỐ KHÁC

[CT7] **Hồ Kim Giàu**, Vũ Thị Đào, Nguyễn Hiếu Minh, "Thay đổi khoá của cơ sở dữ liệu mã hoá trên môi trường thuê ngoài," trong *Hội thảo lần thứ II: Một số vấn đề chọn lọc về an toàn an ninh thông tin (SOIS) – TP. Hồ Chí Minh*, 2017, pp. 30-34.

[CT8] Nguyen Hieu Minh, Vu Thi Dao, **Kim Giau Ho**, Do Van Tuan, "Some new multisignature schemes based on discrete logarithm problem", trong *Một số vấn đề chọn lọc về an toàn an ninh thông tin 2018*, ISSN: 1859-3550, 2018, pp. 122-130.

[CT9] **Hồ Kim Giàu**, Trần Thanh Tùng, Nguyễn Hiếu Minh, "Xác thực cơ sở dữ liệu mã hoá thuê ngoài", trong *Hội thảo Công nghệ thông tin và Truyền thông*, ISBN: 978-604-67-1191-9, 2018, pp. 30-33.

TÀI LIỆU THAM KHẢO

Tiếng Việt

[1] 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 SĨ.*

[2] Nguyễn Anh Tuấn, (2011), *Về một giải pháp bảo mật cơ sở dữ liệu – LUẬN ÁN TIẾN SĨ.*

Tiếng Anh

[3] Agrawal Divyakant, et al. (2009), "Database management as a service: Challenges and opportunities", in: *2009 IEEE 25th International Conference on Data Engineering*, IEEE, 1709–1716.

[4] Ahmad Awais, et al.(2019), "Parallel query execution over encrypted data in database-as-a-service (DaaS)", *The Journal of Supercomputing*, 75 (4), 2269–2288.

[5] *Amazon Web Services (AWS) - Cloud Computing Services*, URL: `https://aws.amazon.com/` (visited on 10/2019).

[6] *Apache Hadoop*, URL: `http://hadoop.apache.org/` (visited on 10/2019).

[7] *Apache Sqoop*, URL: `https://sqoop.apache.org/` (visited on 10/2019).

[8] Bellare Mihir, Garay Juan A. Rabin Tal, (1998), "Fast batch verification for modular exponentiation and digital signatures", in: *International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, 236–250.

[9] Brinkman Richard, Doumen Jeroen, Jonker Willem, (2004), "Using secret sharing for searching in encrypted data", in: *Workshop on Secure Data Management*, vol. 13, 3, Springer, 18–27.

[10] Camenisch Jan, Hohenberger Susan, Pedersen Michael Østergaard, (2012), "Batch verification of short signatures", *Journal of cryptology*, 25 (4), 723–747.

[11] Chauhan Sanjeev Kumar, Sharma Aravendra Kumar, (2013), "Secure Access to Outsourced Databases", *IOSR Journal of Computer Engineering*, (5), 2278–661, URL: www.iosrjournals.org.

[12] Chen B. H. K. Et al.(2015), "CypherDB: A Novel Architecture for Outsourcing Secure Database Processing", *IEEE Transactions on Cloud Computing*, ISSN: 2168-7161, DOI: 10.1109/TCC.2015.2511730.

[13] Chiou S, He Y, (2013), "Remarks on new Digital Signature Algorithm based on Factorization and Discrete Logarithm problem", *International Journal of Computer Trends and Technology (IJCTT)*, 4, 3322–3324.

[14] Chiou Shin-Yan, (2016), "Novel digital signature schemes based on factoring and discrete logarithms", *International Journal of Security and Its Applications*, 10 (3), 295–310.

[15] *Cloud Computing Services*, Google Cloud, URL: https://cloud.google.com/ (visited on 10/2019).

[16] *Cloud Security Company | CASB Vendors*, CipherCloud, URL: https://www.ciphercloud.com/ (visited on 10/2019).

[17] *CMC TSSG*, CMC TSSG, URL: https://www.cmctssg.vn/en/homepage/ (visited on 10/2019).

[18] Codd Edgar F, (1970), "A relational model of data for large shared data banks", *Communications of the ACM*, 13 (6), 377–387.

[19] Codd Edgar F, (1990), *The relational model for database management: version 2*, Addison-Wesley Longman Publishing Co., Inc.

[20] Crandall Richard, Pomerance Carl B, (2006), *Prime numbers: a computational perspective*, vol. 182, Springer Science & Business Media.

[21] Damiani Ernesto, et al.(2007), "Selective data encryption in outsourced dynamic environments", *Electronic Notes in Theoretical Computer Science*, 168, pp. 127–142.

[22] Dean Jeffrey, Ghemawat Sanjay, (2010), "MapReduce: A Flexible Data Processing Tool", *Commun. ACM*, 53 (1), 72–77, ISSN: 0001-0782, DOI: 10.1145/1629175.1629198, URL: https://doi.org/10.1145/1629175.1629198.

[23] Dermova E. S. "Information Authentication Protocols on two hard problems. Ph. D. Dissertation", 2009.

[24] Do Binh V, Nguyen Minh H, Moldovyan Nikolay A, "Digital Signature Schemes from Two Hard Problems", in: *Multimedia and Ubiquitous Engineering*, Springer, 2013, 817–825.

[25] El-Khoury Vanessa, Bennani Nadia, Ouksel Aris M. (2009), "Distributed key management in dynamic outsourced databases: A trie-based approach", in: *2009 First International Confernce on Advances in Databases, Knowledge, and Data Applications*, IEEE, 56–61.

[26] *EncFS: an Encrypted Filesystem for FUSE*, URL: https://github.com/vgough/encfs (visited on 10/2019).

[27] Etemad Mohammad, Küpçü Alptekin, (2018), "Verifiable database outsourcing supporting join", *Journal of Network and Computer Applications*, 115, 1–19.

[28] FIPS PUB, (1994), "186", *Digital Signature Standard*, 1.

[29] Gayathri NB, et al.(2018), "Efficient pairing-free certificateless authentication scheme with batch verification for vehicular ad-hoc networks", *IEEE Access*, 6, 31808–31819.

[30] Gopalani Satish, Arora Rohan, (2015), "Comparing apache spark and map reduce with performance analysis using k-means", *International journal of computer applications*, 113 (1).

[31] GOST R. (1994), "R 34.10-94. Russian Federation Standard", *Information Technology. Cryptographic data Security. Produce and check procedures of Electronic Digital Signature based on Asymmetric Cryptographic Algorithm. Government Committee of the Russia for Standards.*

[32] Hacigumus Hakan, et al. (2002), "Executing SQL over encrypted data in the database-service-provider model", in: *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, 216–227.

[33] Hakuta Keisuke, et al. (2013), "Batch Verification Suitable for Efficiently Verifying a Limited Number of Signatures", in: *Information Security and Cryptology - ICISC 2012*, ed. by Kwon Taekyoung, Lee Mun-Kyu, Kwon Daesung, Berlin, Heidelberg: Springer Berlin Heidelberg, 425–440, ISBN: 978-3-642-37682-5.

[34] Hang Isabelle, Kerschbaum Florian, Damiani Ernesto, (2015), "ENKI: Access Control for Encrypted Query Processing", in: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of*

Data, SIGMOD '15, event-place: Melbourne, Victoria, Australia, ACM, 183–196.

[35] Hankerson Darrel, Menezes Alfred J, Vanstone Scott, (2006), *Guide to elliptic curve cryptography*, Springer Science & Business Media.

[36] Harn L, (1994), "Public-key cryptosystem design based on factoring and discrete logarithms", *IEE Proceedings-Computers and Digital Techniques*, 141 (3), 193–195.

[37] Harn Lein, (1998), "Batch verifying multiple RSA digital signatures", *Electronics Letters*, 34 (12), 1219–1220.

[38] Hong Seungtae, Kim Hyeong-Il, Chang Jae-Woo, (2017), "An efficient key management scheme for user access control in outsourced databases", *World Wide Web*, 20 (3), 467–490.

[39] Huang Yin-Fu, Chen Wei-Cheng, (2015), "Parallel query on the in-memory database in a CUDA platform", in: *2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, IEEE, 236–243.

[40] Hue T. B. P. Et al. (2012), "An Efficient Fine-grained Access Control mechanism for database outsourcing service", in: *Information Security and Intelligence Control (ISIC)*, IEEE, 65–69.

[41] Hwang Jung Yeon, et al.(2015), "New efficient batch verification for an identity-based signature scheme", *Security and Communication Networks*, 8 (15), 2524–2535.

[42] Ismail E. S. Tahat N. M. F. (2011), "A New Signature Scheme Based on Multiple Hard Number Theoretic Problems", *CN*, 2011, ISSN: 2090-4355,

DOI: 10.5402/2011/231649, URL: https://doi.org/10.5402/2011/231649.

[43] Ismail Eddie Shahril, Tahat NMF, Ahmad Rokiah R, (2008), "A new digital signature scheme based on factoring and discrete logarithms", *Journal of mathematics and statistics*, 4 (4), 222–225.

[44] Jain Rohit, Prabhakar Sunil, (2013), "Trustworthy data from untrusted databases", in: *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, IEEE, 529–540.

[45] Jiang S. Et al.(2019), "Publicly Verifiable Boolean Query Over Outsourced Encrypted Data", *IEEE Transactions on Cloud Computing*, 7 (3), 799–813.

[46] Katz Jonathan, et al.(1996), *Handbook of applied cryptography*, CRC press.

[47] Kittur Apurva S, Jain Ashu, Pais Alwyn Roshan, (2017), "Fast verification of digital signatures in IoT", in: *International Symposium on Security in Computing and Communication*, Springer, 16–27.

[48] Kittur Apurva S, Pais Alwyn R, (2019), "A new batch verification scheme for ECDSA signatures", *Sadhana*, 44 (7), p. 157.

[49] Lee N-Y, Hwang T, (1996), "Modified Harn signature scheme based on factorising and discrete logarithms", *IEE Proceedings-Computers and Digital Techniques*, 143 (3), 196–198.

[50] Li Jin, et al.(2015), "L-EncDB: A lightweight framework for privacy-preserving data queries in cloud computing", *Knowledge-Based Systems*, 79, 18–26.

[51] Lim Chae Hoon, Lee Pil Joong, (1998), "A study on the proposed Korean digital signature algorithm", in: *International Conference on the Theory and Application of Cryptology and Information Security*, Springer, 175–186.

[52] Liu Jingwei, et al.(2018), "A large-scale concurrent data anonymous batch verification scheme for mobile healthcare crowd sensing", *IEEE Internet of Things Journal*, 6 (2), 1321–1330.

[53] Mallaiah Kurra, Ramachandram S. (2014), "Applicability of Homomorphic Encryption and CryptDB in Social and Business Applications: Securing Data Stored on the Third Party Servers while Processing through Applications", *International Journal of Computer Applications*, 100, 5–19, ISSN: 0975-8887.

[54] Mykletun Einar, Narasimha Maithili, Tsudik Gene, (2006), "Authentication and integrity in outsourced databases", *ACM Transactions on Storage (TOS)*, 2 (2), 107–138.

[55] Naccache David, et al. (1994), "Can DSA be improved?—Complexity trade-offs with the digital signature standard—", in: *Workshop on the Theory and Application of of Cryptographic Techniques*, Springer, 77–85.

[56] Namasudra Suyel, Roy Pinki, (2016), "Secure and efficient data access control in cloud computing environment: A survey", *Multiagent and Grid Systems*, 12 (2), 69–90.

[57] Narasimha Maithili, Tsudik Gene, (2005), "DSAC: An Approach to Ensure Integrity of Outsourced Databases using Signature Aggregation and Chaining.", *IACR Cryptology ePrint Archive*, 2005, p. 297.

[58] Niaz Muhammad Saqib, Saake Gunter, (2015), "Merkle Hash Tree based Techniques for Data Integrity of Outsourced Data.", in: *GvD*, 66–71.

[59] *OpenStars*, URL: `https://github.com/openstars` (visited on 10/2019).

[60] Popa Raluca Ada, et al.(2012), "CryptDB: Processing queries on an encrypted database", *Communications of the ACM*, 55, 103–111, ISSN: 0001-0782.

[61] *Porticor Cloud Security - Overview | Crunchbase*, URL: `https://www.crunchbase.com/organization/porticor-cloud-security` (visited on 10/2019).

[62] Rabin Michael O, *Digitalized signatures and public-key functions as intractable as factorization*, tech. rep., Massachusetts Inst of Tech Cambridge Lab for Computer Science, 1979.

[63] Rivest Ronald L. Shamir Adi, Adleman Leonard, (1978), "A method for obtaining digital signatures and public-key cryptosystems", *Communications of the ACM*, 21 (2), 120–126.

[64] Schnorr Claus-Peter, (1989), "Efficient identification and signatures for smart cards", in: *Conference on the Theory and Application of Cryptology*, Springer, 239–252.

[65] Shao Zuhua, (2001), "Batch verifying multiple DSA-type digital signatures", *Computer Networks*, 37 (3-4), 383–389.

[66] Sharma Priya P, Navdeti Chandrakant P, (2014), "Securing big data hadoop: a review of security issues, threats and solution", *Int. J. Comput. Sci. Inf. Technol*, 5 (2), pp. 2126–2131.

[67] Shastri S. Kresman R. Lee J. K. (2015), "An Improved Algorithm for Querying Encrypted Data in the Cloud", in: *2015 Fifth Interna-*

tional Conference on Communication Systems and Network Technologies (CSNT), 653–656.

[68] Shi Juwei, et al.(2015), "Clash of the titans: Mapreduce vs. spark for large scale data analytics", *Proceedings of the VLDB Endowment*, 8 (13), 2110–2121.

[69] Shim Kyung-Ah, (2012), "A round-optimal three-party ID-based authenticated key agreement protocol", *Information Sciences*, 186 (1), 239–248.

[70] Song Wei, et al.(2017), "Tell me the truth: Practically public authentication for outsourced databases with multi-user modification", *Information Sciences*, 387, 221–237.

[71] *Thales eSecurity: Cloud and Data Security | Encryption | Key Management | Digital Payment Security Solutions*, URL: `https://www.thalesesecurity.com/` (visited on 10/2019).

[72] Thanh Trung Nguyen, Minh Hieu Nguyen, (2015), "Zing database: high-performance key-value store for large-scale storage service", *Vietnam Journal of Computer Science*, 2 (1), 13–23.

[73] *The Linux FUSE (Filesystem in Userspace) interface*, Oct. 2019, URL: `https://github.com/libfuse/libfuse` (visited on 10/2019).

[74] *The Office of Inadequate Security: DataBreaches.net*, URL: `https://www.databreaches.net/` (visited on 10/2019).

[75] *TPC-H*, URL: `http://www.tpc.org/tpch/` (visited on 10/2019).

[76] Tu Stephen, et al. (2013), "Processing analytical queries over encrypted data", in: *Proceedings of the VLDB Endowment*, vol. 6, VLDB Endowment, 289–300.

[77] Vangoor Bharath Kumar Reddy, Tarasov Vasily, (2017), "To FUSE or Not to FUSE: Performance of User-Space File Systems", in: *15th USENIX Conference on File and Storage Technologies (FAST) 17*, 59–72.

[78] Vishnoi Sushila, Shrivastava Vishal, (2012), "A new DigitalSignature Algorithm based on Factorization and Discrete Logarithm problem".

[79] Wang Ching-Te, Lin Chu-Hsing, Chang Chin-Chen, (2003), "Signature schemes based on two hard problems simultaneously", in: *17th International Conference on Advanced Information Networking and Applications, 2003. AINA 2003.* IEEE, 557–560.

[80] Wang Lei, et al. (2017), "Optimization of leveldb by separating key and value", in: *2017 18th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)*, IEEE, 421–428.

[81] Wang Yimin, et al.(2016), "ECPB: Efficient Conditional Privacy-Preserving Authentication Scheme Supporting Batch Verification for VANETs.", *IJ Network Security*, 18 (2), 374–382.

[82] Wang Zheng-Fei, Tang Ai-Guo, (2010), "Implementation of encrypted data for outsourced database", in: *Computational Intelligence and Natural Computing Proceedings (CINC), 2010 Second International Conference on*, vol. 2, IEEE, 150–153.

[83] *World's Biggest Data Breaches & Hacks*, Information is Beautiful, URL: `https://informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/` (visited on 10/2019).

[84] Xiang Tao, et al.(2016), "Processing secure, verifiable and efficient SQL over outsourced database", *Information Sciences*, 348, 163–178.

[85] Yang Anjia, et al.(2015), "A new ADS-B authentication framework based on efficient hierarchical identity-based signature with batch verification", *IEEE Transactions on Services Computing*, 10 (2), 165–175.

[86] Yang F. Et al. (2015), "Optimizing NoSQL DB on Flash: A Case Study of RocksDB", in: *2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom)*, 1062–1069.

[87] Yin Linzi, et al.(2019), "An efficient attribute reduction algorithm using MapReduce", *Journal of Information Science*, 1–17.

[88] Yuan Jiawei, Yu Shucheng, (2013), "Flexible and publicly verifiable aggregation query for outsourced databases in cloud", in: *Communications and network security (cns), 2013 ieee conference on*, IEEE, 520–524.

[89] Zhang Q. Li S. Xu J. (2014), "QScheduler: A Tool for Parallel Query Processing in Database Systems", in: *2014 19th International Conference on Engineering of Complex Computer Systems (ICECCS)*, 73–76.

[90] Zhang Yupeng, Katz Jonathan, Papamanthou Charalampos, (2015), "Integridb: Verifiable SQL for outsourced databases", in: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ACM, 1480–1491.

[91] Zych Anna, Petković Milan, Jonker Willem, (2008), "Efficient key management for cryptographically enforced access control", *Computer Standards & Interfaces*, 30 (6), 410–417.