KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN CỦA LUẬN ÁN

Một số 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 và tính toán 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.
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. 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.

Hướng phát triển của luận án

1. 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.
2. 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.
3. Nghiên cứu các thuật toán xác thực dữ liệu đảm bảo tính đủ và tính mới thích hợp với CSDL động.

MỞ ĐẦU

1. Động lực nghiên cứu:

Điện toán đám mây phát triển mạnh mẽ giúp 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. Như vậy, DO sẽ giảm được chi phí trong việc đầu tư hạ tầngp và nhân công để quản lý khi sử dụng CSDL. Ngoài ra, 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 hơn.

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. Ngoài ra, 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.

Để bảo vệ "tài sản" quý giá của mình, DO mã hoá dữ liệu trước khi lưu trữ lên ODBS. Khi mã hoá dữ liệu, 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 cho chi phí về thời gian, tài nguyên hệ 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. Ngoài ra, bài toán xác thực dữ liệu trên CSDL mã

khi truy vấn ngẫu nhiên từ nhiều bảng hoặc CSDL động 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. Các đóng góp của luận án:

1. 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.
2. Mô hình và thuật toán xác thực ODBS có ý nghĩa kiểm tra tính đúng đắn của dữ liệu khi truy vấn trên CSDL mã. Ngoài ra, các thuật toán xác thực lô được đề xuất dựa trên hai bài toán khó cũng 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.
3. Mô hình 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.
4. Mô hình 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.

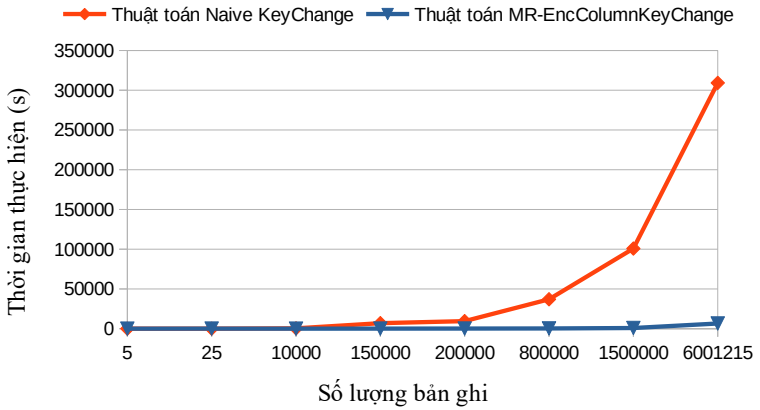
3. Bố cục luận án:

Luận án được tổ chức như sau: Mở đầu; 3 chương nội dung; Kết luận và hướng nghiên cứu tương lai; Danh mục các công trình nghiên cứu.

3.4 Phân tích, thử nghiệm các phương pháp đề xuất

Luận án chưa tìm 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 3.5. Luận án sử dụng Hadoop 2.7.0 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 x 8, Ram 8GB, ổ cứng HDD 1TB. Hệ điều hành Ubuntu 16.10. Sử dụng sqoop-1.4.4 để chuyển đổi dữ liệu giữa CSDL và HDFS.

Thời gian đổi khoá của phương pháp đề xuất được so sánh với phương pháp NaiveKeyChange như hình 3.2. Kết quả thực nghiệm cho thấy khi dữ liệu càng lớn, hiệu quả phương pháp đề xuất càng cao.



Hình 3.2: Kết quả thời gian thực hiện đổi khoá

3.5 Kết luận

Chương 3 giới thiệu về quản lý khoá của người dùng và bài toán thay đổi khóa cho ODBS. 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. Ngoài ra, 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 khóa. Vì vậy, phương pháp này đảm bảo cho CSDL luôn sẵn sàng hoạt động khi thay khóa.

3.3.2 Đề xuất phương pháp đổi khoá dựa trên MapReduce

Thuật toán đổi khoá của dữ liệu mã hoá ở mức cột thực hiện trên MapReduce theo 3 bước sau đây:

- Bước 1: Chuyển 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 3.5)
- Bước 3: Cập nhập dữ liệu trong các tệp tin HDFS của hadoop lên ODBS.

Thuật toán 3.5: 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
```

Giai đoạn thực hiện đổi khoá mức cột của bảng CSDL được đề xuất trong thuật toán 3.5.

Chương 1

Những vấn đề chung về an toàn ODBS

1.1 Tổng quan về an toàn ODBS

ODBS là một dịch vụ trên nền tảng như một dịch vụ (Platform as a Service) của điện toán đám mây. Trong mô hình ODBS, DO thuê một nhà cung cấp dịch vụ (Database Service Provider – DSP) 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 và DO không có quyền truy cập vào các dịch vụ khác của máy chủ DSP.

Mô hình ODBS thường có ba đối tượng chính:

- Nhà cung cấp dịch vụ: 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.
- Chủ sở hữu dữ liệu: Người tạo ra và toàn quyền quyết định dữ liệu. DO là người sẽ thuê dịch vụ của DSP để tổ chức 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.

1.2 Những nguy cơ mất an toàn ODBS

Một số vấn đề về an toàn ODBS:

- 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". Ngoài ra, 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.

1.3 Các bài toán bảo đảm an toàn ODBS

Nhiều bài toán bảo đảm an toàn cho ODBS được đặt ra, trong đó có các bài toán như:

- Tính bí mật của dữ liệu (Data confidentiality): DO thường mã hóa dữ liệu bởi một hàm mật mã với khóa bí mật trước khi lưu trữ trên đám mây. Trong bài toán này, việc quản lý và thay đổi khoá mã là một bài toán quan trọng. Thay đổi khoá là quá trình giải mã và mã hoá lại toàn bộ dữ liệu của DO đã lưu trữ ở máy chủ của DSP.
- Tính riêng tư của dữ liệu (Data privacy): 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): Kết quả trả về từ server phải đảm bảo tính đúng (correctness), đầy đủ (completeness) và mới nhất (freshness).

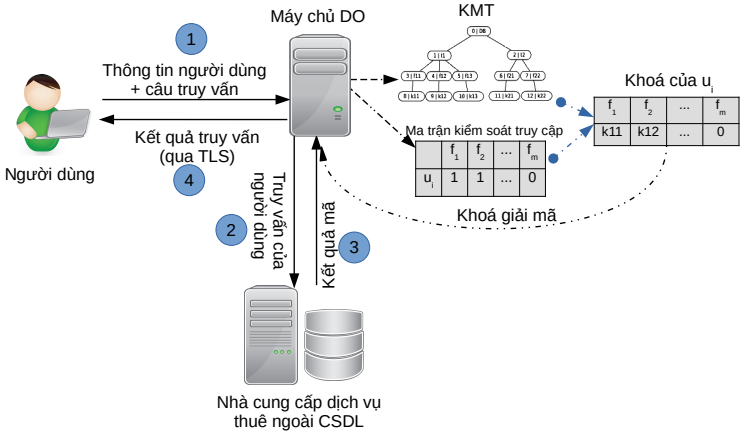
Ngoài ra còn có các bài toán bảo đảm an toàn cho ODBS khác như: xác thực người dùng, bảo vệ tính riêng tư người dùng, ghi nhật ký hệ thống và bảo vệ siêu dữ liệu...

1.4 Kết luận chương 1

Chương 1 trình bày những kiến thức chung về ODBS, các bài toán đảm bảo an toàn cho ODBS làm cơ sở để đưa ra các định hướng, các bài toán nghiên cứu và giải quyết. 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 và phương pháp đổi khoá của CSDL mã hoá. Cuối cùng là kết luận của luận án và đề nghị hướng phát triển nghiên cứu tiếp theo.

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. Mô hình quản lý truy cập dữ liệu mức cột của người dùng được mô tả như hình 3.1.



Hình 3.1: Mô hình truy cập dữ liệu mức cột của người dùng

3.3 Đề xuất phương pháp thay đổi khóa mã cho ODBS

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 hoá, 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. Một phương pháp ngây thơ (naive method) có thể giải quyết bài toán thay khóa là:

- 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 hoá lên ODBS.

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
```

Quản lý khóa mã hoá tệp

Quá trình mã hoá/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.

Chương 2

Giảm thời gian truy vấn và xác thực khi truy vấn CSDL mã trên ODBS

2.1 Giới thiệu chung

Để đảm bảo được tính bí mật dữ liệu, DO mã hóa dữ liệu trước khi lưu trữ lên DSP. Tuy nhiên, khi truy vấn trên dữ liệu mã thì thường làm cho máy chủ xử lý chậm hơn so với truy vấn trên dữ liệu rõ. Vì vậy cần có phương pháp giảm thời gian cho truy vấn mã. Mặt khác, khi truy vấn CSDL từ ODBS, người dùng muốn kết quả trả về là chính xác. Do đó, khi có kết quả trả về, DO cần phải xác thực dữ liệu trước khi giải mã trả kết quả rõ cho người dùng.

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

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. 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:
  $R = R_1 \cup R_2 ... \cup R_x$

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

2.3.1 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 $\text{len}(p) = \text{L}$.
- 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).

2.3.2 Lược đồ xác thực lô Rabin-Schnorr

Lược đồ ký, kiểm tra chữ ký và xác thực lô Rabin-Schnorr lần lượt là các thuật toán 2.1, 2.2, 2.3.

Thuật toán 2.1: 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)

tin mã hoá và dữ liệu về cấu trúc tệp và thư mục.

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 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*. Thuật toán đọc, ghi file được mô tả như trong thuật toán 3.1, 3.2.

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**

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 hoá-giải mã nội dung tệp từ kho lưu trữ khoá-giá trị. Các hàm mã hoá-giải mã nội dung tệp được thực hiện như thuật toán 3.3, 3.4.

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, 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 $\mathscr{K}_1(k11, k12, k13)$, $\mathscr{K}_2(k21, k22)$.

Ngoài ra, ta có thể biểu diễn KMT bằng một cấu trúc mảng quản lý khoá như bảng 3.1. 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.

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 3.1: 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 hoá, 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.

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

Thuật toán 2.2: 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 2.3: 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 $\sigma_i(r_i, s_i)$, $1 \le i \le k$
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** $(\prod_{i=1}^{k} r_i = g^u y^v \bmod p)$ **then return** *"Accept"*
5 **else return** *"Reject"*

2.3.3 Lược đồ xác thực lô RSA-Schnorr

Lược đồ ký, kiểm tra chữ ký và xác thực lô RSA-Schnorr lần lượt là thuật toán 2.4, 2.5, 2.6.

Thuật toán 2.4: 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 2.5: 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"*

Thuật toán 2.6: 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 $\sigma_i(r_i, s_i)$, $1 \le i \le k$
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** $(\prod_{i=1}^{k} r_i = g^u y^v \bmod p)$ **then return** *"Accept"*
5 **else return** *"Reject"*

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

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

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 2.1 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** $(s \ge n/2)$ **then** $s = n - s$.

Tương ứng với việc bổ sung ở bước 6 trong thuật toán 2.1 thì thuật toán 2.2 và 2.3 phải thêm điều kiện $(s < n/2)$.

Lược đồ RSA-Schnorr cũng thêm điều kiện $gcd(s, n) \neq 1$ ở bước 3 trong thuật toán 2.4.

Chương 3

Quản lý, thay đổi khoá mã của ODBS

3.1 Giới thiệu chung

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. Để bảo đảm tính bí mật dữ liệu, DO cần có biện pháp thay đổi khoá mã (rekey). Tuy nhiên, hiện nay vẫn chưa có công trình nghiên cứu liên quan đến bài toán thay đổi khóa.

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 đáp ứng được tính sẵn sàng của dữ liệu.

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 $1 \le i \le n$, $1 \le j \le m$.

Định nghĩa 3.2.1. *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:

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

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.

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

Phân tích đánh giá các trường hợp tấn công

- 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 khoá k và thực hiện $\mu = E_k(r)$.
- 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.
  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 2.11 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.
  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 2.12 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.

2.6 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. Chương 2 cũng đề xuất mô hình kiểm tra tính đúng đắn của CSDL mã dựa trên xác thực lô. 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 không tốn thời gian giải mã sau khi xác thực như các đề xuất trước đây.

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

Để nâng cao tính hiệu quả, lược đồ lấy 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".

□ **Các tham số** Các tham số cho lược đồ Rabin-Schnorr và RSA-Schnorr giống như nêu trong mục 2.3.1 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 2.7, 2.8, 2.9.

Thuật toán 2.7: 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)

□ **Lược đồ RSA-Schnorr cải tiến** Thuật toán 2.10.

2.4 Xác thực dữ liệu mã hóa thuê ngoài

2.4.1 Quá trình hoạt động

Luận án chọn thuật toán Rabin-Schnorr để xác thực dữ liệu. Các tham số dùng cho hệ thống: như mục 2.3.5, 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

Thuật toán 2.8: 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"*

Thuật toán 2.9: 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 2.10: 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)

Thử nghiệm các phương pháp đề xuất

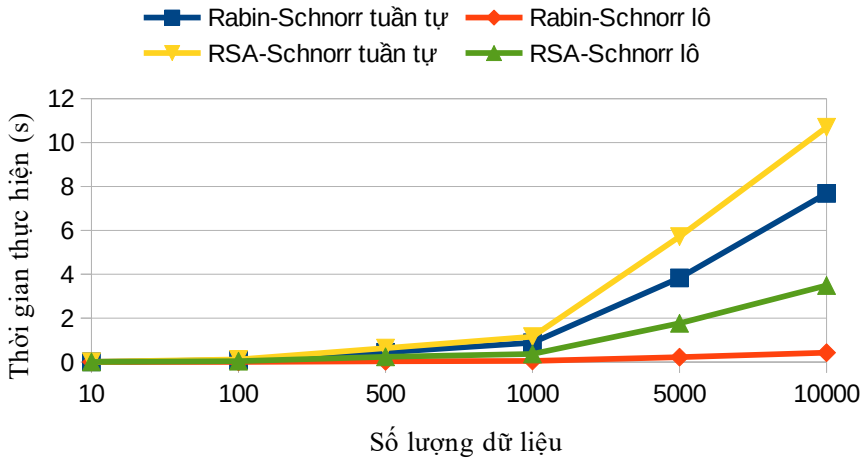
Để thử nghiệm các phương pháp được đề xuất, luận án tiến hành cài đặt các thuật toán bằng ngôn ngữ lập trình Python và thực hiện trên máy tính Core™ i5-4310U CPU @ 2.0GHz x 4, Ram 8GB, hệ điều hành Ubntu 20.04.

Thời gian thực hiện tạo chữ ký với trên số lượng các dữ liệu tương ứng như bảng 2.4.

Bảng 2.4: Thời gian thực hiện tạo chữ ký (s)

Số lượng dữ liệu	Thuật toán 2.7	Thuật toán 2.10
10	0,25614	0,01634
100	1,535692	0,150983
1000	14,061793	1,228292
10000	136,160404	10,059431

Luận án tiến hành thử nghiệm thời gian thực hiện trên hai phương pháp: xác thực tuần tự k chữ ký và phương pháp xác thực lô cho từng thuật toán, kết quả như hình 2.2.



Hình 2.2: Thời gian xác thực chữ ký

Bảng 2.1: Chi phí tiết kiệm được của lược đồ cải tiến tương ứng với cặp (L, N)

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

Chi phí thực hiện các thuật toán đề xuất

Chi phí các công việc của các thuật toán đề xuất như bảng 2.2. Trong đó, k là số lượng chữ ký cần xác thực lô.

Bảng 2.2: 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	$(1,5.(L+N)+2.k+2).t_m(L)$ $+k(t_H+t_m(N))$	$(3.L+1,5.N+2k+1).t_m(L)$ $+k(t_H+t_m(N))$

Chi phí thực hiện tính toán của các giai đoạn xác thực CSDL mã trên ODBS được mô tả như bảng 2.3.

Bảng 2.3: 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	$h_T(k_T(2.T_m(L)+t_m(L)+t_m(N)+t_H+T_D)$ $+(1,5.(L+N)+1)t_m(L)$

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 $j(1\le i\le n_T, 1\le j\le m_T)$.
Các giai đoạn hoạt động xác thực ODBS được thực hiện lần lượt như thuật toán 2.11, 2.12.

Thuật toán 2.11: Thuật toán lưu CSDL mã hoá

Input: Bảng T trong CSDL, khoá bí mật
Output: Lưu bảng dữ liệu mã hoá 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 $(s\ge n/2)$ **then** $s\leftarrow n-s$
    d.append$(\mu_{ij},(r,s))$
  end
  T'.addrow(d)
end
Lưu trữ bảng T' lên máy chủ DSP

2.4.2 Một số trường hợp truy vấn CSDL

Cập nhập dữ liệu

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(μ'_1, $\sigma'_1,\mu'_2,\sigma'_2,...,\mu'_{m_T},\sigma'_{m_T}$);".

Thuật toán 2.12: 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

Khi thực hiện sửa dữ liệu thì người dùng tính toán các giá trị mã và chữ 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 $c3 = \mu_3$;"

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 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.

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

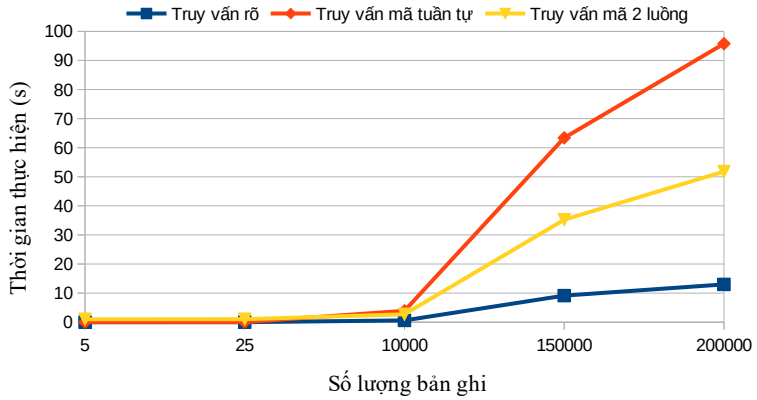
* FROM T_1 INNER JOIN T_2 ON $T_1.id1 = T_2.id2$;" đến DSP. Máy chủ DSP xử lý và trả kết quả $T_r = \{AutoNum1, id1, \sigma_{id1}, \mu1_{ij}, \sigma1_{ij}, AutoNum2, id2, \sigma_{id2}, \mu2_{ij}, \sigma2_{ij} | i = 1, 2, ...l_T; j = 1, 2, ..., p_T\}$ về máy chủ trung gian. Do $\sigma_{id1}, \sigma_{id2}, \sigma1_{ij}, \sigma2_{ij}$ 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.

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

Để đánh giá kết quả của phương pháp đề xuất, luận án thử nghiệm trên máy tính Core™ i5-4310U CPU @ 2.0GHz x 4, Ram 8GB, hệ điều hành Ubntu 20.04. Thuật toán mã hoá dữ liệu là AES, hệ quản trị CSDL MySQL. Thực hiện câu truy vấn *"SELECT $f_1, f_2, \ldots, f_n$ FROM t"* , trong đó $f_1, f_2, \ldots, f_n$ là các trường dữ liệu trong bảng TPC-H.

Kết quả thực hiện được mô tả ở hình 2.1.



Hình 2.1: Thời gian thực hiện truy vấn

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

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

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. Chi phí (theo đơn vị $t_m(L)$) tiết kiệm được khi sử dụng các lược đồ cải tiến so với lược đồ ban đầu như trong bảng 2.1.