Giải pháp bảo đảm an toàn cơ sở dữ liệu trong môi trường OUTSOURCE

Nguyễn Hiếu Minh, Phạm Công Thành, Hồ Kim Giàu, Trần Lê Hoàng Tuấn

***Tóm tắt*— Điện toán đám mây đang được ứng dụng rộng rãi nhờ vào những ưu điểm về kinh tế và công nghệ. Tuy nhiên, dữ liệu của người dùng được đưa lên đám mây (outsource cơ sở dữ liệu) sẽ xuất hiện rủi ro khi nhiều thông tin được tập trung vào một nơi. Thậm chí nếu chỉ sử dụng đám mây như một giải pháp sao lưu dữ liệu thì rủi ro vẫn tồn tại. Vì vậy, mã hóa dữ liệu trước khi chuyển lên đám mây để đảm bảo an toàn dữ liệu trở nên cần thiết. Trong bài báo này, chúng tôi đề xuất một giải pháp bảo đảm an toàn khi outsource cơ sở dữ liệu.**

***Abstract*— Cloud computing is being popular because of its advantages in economic and technological aspects. However, due to the user's data will be posted on the cloud (DatabaseOutsourcing), it should be formed much more risk when information is centralized in one place. Even if the cloud is used as a backup solution, risk exist. Data encryption before transferring to cloud to ensure corporate data becomes uncertain. In this paper, we propose a method for security issues to be addressed in database outsourcing.**

Từ khóa— Outsource cơ sở dữ liệu; nhà cung cấp dịch vụ CSDL; mã hóa dữ liệu.

I. ĐẶT VẤN ĐỀ

Ngày nay, điện toán đám mây đang được phát triển rất mạnh mẽ. Việc thuê quản trị bên ngoài cơ sở dữ liệu (outsource CSDL) đang được các tổ chức, doanh nghiệp quan tâm như một giải pháp nhằm giảm bớt chi phí quản lý, và duy trì dữ liệu. Theo đó, các tổ chức hay doanh nghiệp sẽ ủy quyền cho một nhà cung cấp dịch vụ quản trị cơ sở dữ liệu (CSDL - Database Service Provider - DSP) để quản lý và duy trì dữ liệu của mình. Dịch vụ này được gọi là DaaS (Database as a Service) [1]. Trong đó, DSP sẽ cung cấp các phương thức cho phép chủ sở hữu dữ liệu (Data Owner - DO) có thể truy xuất đến các dữ liệu của họ khi đã đưa lên đám mây.

Khi outsource CSDL thì quyền kiểm soát dữ liệu thuộc về DSP và như vậy DO cũng cần phải có những biện pháp thích hợp để bảo vệ CSDL của mình khỏi những cuộc tấn công bên ngoài hay từ chính DSP. CSDL cần phải được bảo đảm an toàn ngay cả với DSP, có nghĩa là DSP cũng không được phép biết nội dung CSDL lưu trữ trên máy chủ của họ, vì thông tin dữ liệu có thể sẽ bị trích xuất hay làm lộ lọt gây tổn hại đến chủ sở hữu CSDL. Bên cạnh đó, DO chỉ cho phép người sử dụng (hoặc khách hàng - Client) được quyền khai thác CSDL và chỉ có thể khai thác được những gì được cấp phép.

Trong bài báo này chúng tôi đề xuất một giải pháp bảo đảm an toàn khi đưa CSDL lên đám mây của các DSP. CSDL trước khi được đưa lên đám mây sẽ được mã hóa (theo chuẩn mã hóa AES/CBC/PKCS5 Padding 128 bit), và sẽ được truy vấn thông qua chỉ mục XML được đặt tại máy chủ web (web server) của DO. Mỗi Client khi khai thác dữ liệu được cấp phép sẽ có các chỉ mục khác nhau.

Giải pháp đề xuất nhằm đảm bảo các yêu cầu về bảo mật, xác thực và toàn vẹn của dữ liệu và các tính chất quan trọng: tính đúng, tính đủ, tính mới. Đóng góp mới của giải pháp bao gồm:

- Sử dụng cấu trúc chỉ mục XML trong phân quyền bảo đảm truy vấn dữ liệu đã mã hóa.
- Sử dụng hàm băm để thực hiện kiểm tra tính toàn vẹn và xác thực. Vì giá trị băm được lưu trữ (lấy dữ liệu ở tất cả các cột trên một dòng cộng lại và lấy giá trị băm của dữ liệu tổng đó) ở cả hai phía DO và DSP, nên DO hoàn toàn có thể kiểm tra tính toàn vẹn và xác thực bằng cách so sánh hai giá trị băm. Hơn nữa giải pháp này tốn ít tài nguyên và thời gian xử lý nhanh hơn cách xác thực bằng chữ ký số.
- Lưu trữ chỉ mục XML ở dạng "cha và con" để giảm bớt dung lượng của các file XML đại diện cho các trường được truy vấn trong CSDL đã mã hóa.
- Chỉ sử dụng ½ độ lớn các giá trị băm, vì vậy đã giảm không gian lưu trữ xuống ½ mà vẫn đảm bảo an toàn dữ liệu.
- Để giảm thời gian truy vấn, dữ liệu sẽ đọc về từng phần thông qua việc phân trang dữ liệu, khi Client chuyển trang dữ liệu sẽ được đọc tiếp; hơn nữa Client có thể biết được tổng số bản ghi cũng như tổng số trang cần xem.
- Tăng tốc độ xử lý thông qua xử lý song song các chỉ mục XML.
- Tối ưu hóa xử lý thông qua lưu trữ những file thường xuyên được đọc vào bộ nhớ đệm (cache).

Bố cục của bài báo bao gồm năm phần như sau: Sau Mục đặt vấn đề, Mục II trình bày một số công trình liên quan đến outsouce CSDL. Mục III mô tả giải pháp đề xuất. Mục IV thảo luận về các kết quả thực nghiệm. Mục cuối là kết luận và hướng phát triển.

II. MỘT SỐ CÔNG TRÌNH LIÊN QUAN

A. Hacigümüş và outsource dữ liệu

Hacigümüş và cộng sự [4] đầu tiên đề cập đến khái niệm outsource dữ liệu. Mô hình đề xuất của họ gồm ba thực thể chính: người sử dụng (hoặc khách hàng), chủ sở hữu CSDL và nhà cung cấp dịch vụ quản trị CSDL.

DO lưu trữ dữ liệu tại máy chủ (server) của DSP. Các dữ liệu được lưu trữ trong định dạng mã hóa ở phía máy chủ DSP tại mọi thời điểm cho các mục đích an toàn. Khi đó, CSDL đã mã hóa cần có thêm thông tin phụ trợ (được gọi là chỉ mục (meta-data)). Đây là những thông tin cho phép thực hiện truy vấn CSDL tại máy chủ DSP mà không cần phải giải mã. DO duy trì meta-data để chuyển đổi các truy vấn của các Client thành một truy vấn khác thích hợp để thực thi trên máy chủ DSP và các Client sẽ nhận được kết quả sau khi thực hiện truy vấn được trả về. Dựa trên các thông tin phụ trợ đã được lưu trữ, truy vấn sẽ được chia thành hai phần: (1) Các truy vấn phía máy chủ DSP về các dữ liệu đã được mã hóa, truy vấn này được thực hiện từ máy chủ DO, (2) Các truy vấn phía DO, truy vấn này được thực hiện từ Client đến DO và kết quả truy vấn sau khi được lọc sẽ trả về từ máy chủ DO cho các Client. Để đạt được các yêu cầu trên, [4] phát triển một mô hình đại số để viết lại truy vấn về hình thức dữ liệu được mã hóa.

Tuy nhiên, phương pháp này có những nhược điểm như tăng chi phí lưu trữ và chi phí để tính toán lại sau các hoạt động cập nhật CSDL.

B. Tìm kiếm dữ liệu mã hóa trên XML

R. Brinkman [7] giới thiệu một cách thức cho phép tìm kiếm so trùng các *tag* của một tài liệu XML đã được mã hóa dựa trên giải thuật *Linear Search Strategy for Full TextDocuments (1)*, gọi là *Tree Search Strategy for XML Documents (2)*.

Giải thuật (1) chia làm 3 giai đoạn: lưu trữ (*storage)*, tìm kiếm (*search*), nhận dữ liệu (*retrieval*). Ở giai đoạn lưu trữ, toàn bộ dữ liệu được chia thành nhiều khối nhỏ cố định, sau đó thực hiện mã hóa các khối này trước khi lưu trữ trên server. DO cần phải ghi nhận một số thông tin về mã hóa để có thể giải mã sau này. Ở giai đoạn tìm kiếm, chuỗi dữ liệu cần tìm sẽ được mã hóa và chuyển đến cho server so trùng trên các khối dữ liệu để xác định vị trí của đoạn dữ liệu mã hóa. Giai đoạn nhận dữ liệu, kết quả mã hóa sẽ được giải mã dựa theo các thông tin mã hóa được ghi nhận tại giai đoạn lưu trữ.

Giải thuật (2) được xây dựng dựa trên giải thuật (1). Tuy nhiên, dữ liệu là một tài liệu XML thay vì *file text* phi cấu trúc. Kích thước của các khối chia ra cũng không đều nhau mà phụ thuộc vào kích thước của từng node (hay mỗi node là một khối), giải thuật (2) chỉ đáp ứng các câu truy vấn dạng tìm kiếm so trùng các *tag name* trong tài liệu XML mà không xử lý đến nội dung dữ liệu bên trong node.

C. Bảo đảm truy vấn

Bảo đảm truy vấn *(Query Assurance)* sẽ đảm bảo kết quả truy vấn trả về từ server là *đúng (correctness), đầy đủ (completeness)* và *mới nhất* (*freshness*) [9].

Einar Mykletun [2] đề 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 (*record*) được lưu kèm theo chữ ký số của bản ghi đó. Kết quả trả về kèm theo với chữ ký số. Client 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 lãng phí thời gian và là một chi phí lớn cho client.

Để giải quyết vấn đề này, [2] 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. [2] 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 có thể tìm (*searchable attribute*) đòi hỏi nhiều cấu trúc tương ứng, điều này có thể làm tăng phí tổn để lưu trữ tại server.

Maithili Narasimha [5] 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 (được sắp xếp theo một thuộc tính cho trước), tạo thành một chuỗi liên tiếp nhau. 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*. Hướng tiếp cận của [5] không đòi hỏi phải tốn thêm nhiều không gian lưu trữ trên server. Mỗi bản ghi dữ liệu chỉ cần lưu thêm một chữ ký.

Tuy nhiên, chi phí xây dựng, tạo các chữ ký và kiểm tra các chữ ký đôi khi cũng lớn đáng kể và tốc độ thường chậm hơn từ 100 - 1000 lần so với việc sử dụng băm (*hashing*). [3] đề xuất giải pháp dựa trên *Embedded Merkle B-tree* (EMB) cho phép đảm bảo *tính đúng, đầy đủ* và *mới*. Việc đảm bảo truy vấn chủ yếu dựa vào các phép băm. Từ đó, có thể giảm bớt thời gian để thực hiện tính toán chữ ký khi CSDL có thay đổi, cũng như thời gian kiểm tra kết quả trả về. [3] đồng thời cũng là giải pháp đầu tiên giải quyết được đầy đủ các vấn đề bảo đảm truy vấn.

Radu Sion [6] đưa ra một hướng tiếp cận mới cho phép đảm bảo tính đầy đủ đối với kết quả trả về từ một tập các câu truy vấn cần được thực hiện (*batch of queries*). Hướng tiếp cận này xây dựng một giao thức dựa trên việc mở rộng giao thức ringer. Dựa trên các challenge-token, gửi kèm theo, một cách ngẫu nhiên, xen kẽ với các câu truy vấn cần thực hiện, Client đã biết trước kết quả của những câu truy vấn này và so sánh nó với kết quả trả về từ server. Nếu trùng khớp thì đảm bảo kết quả trả về từ server đầy đủ.

III. ĐỀ XUẤT MÔ HÌNH DAAS

Mô hình đề xuất bao gồm các thành phần sau:

- Client: những client được phép khai thác dữ liệu của DO, chỉ cần sử dụng bất kỳ trình duyệt web nào có sẵn trên các hệ điều hành.
- Chủ sở hữu CSDL (DO): phải có web server để truy vấn CSDL và quản lý các giao diện người dùng, giao diện hiển thị thông tin cho các người dùng, quản lý các chỉ mục được sinh ra để tìm kiếm trên CSDL đã mã hóa.
- Nhà cung cấp dịch vụ quản trị CSDL DSP: là công ty cung cấp các dịch vụ lưu trữ, quản lý CSDL trên đám mây. DO sẽ mã hóa CSDL của mình và đặt tại các DSP.

A. Cơ chế hoạt động

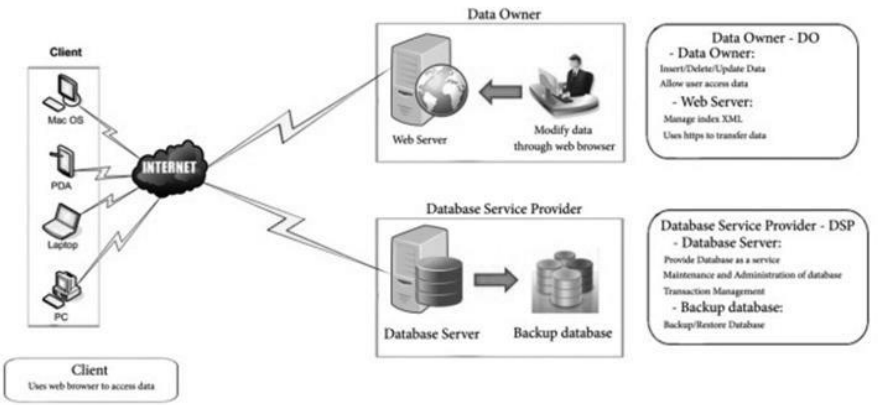
Dựa trên sự kết hợp của mã hóa dữ liệu (theo chuẩn AES). Chỉ mục tìm kiếm (XML) và đọc dữ liệu từng phần để giảm thời gian truy vấn dữ liệu.

Trước tiên, dữ liệu sẽ được mã hóa, với mỗi Client được phép khai thác dữ liệu sẽ có một file XML tương ứng với những trường được phép khai thác được phát sinh. Số lượng kết quả trả về sẽ được thống kê khi duyệt file XML và dữ liệu tương ứng sẽ được truyền về từng phần thông qua việc phân trang dữ liệu.

Khi dữ liệu được truyền từ DO đến Client sẽ sử dụng giao thức SSL (cổng 443) đã hỗ trợ sẵn trên các trình duyệt. Như vậy, phía Client khai thác dữ liệu chỉ cần dùng các trình duyệt web.

B. Mô hình lưu trữ dữ liệu

Dữ liệu được lưu trữ trên đám mây sẽ được mã hóa ở dạng chuỗi. Tại web server của DO sẽ lưu trữ các chỉ mục XML để truy vấn dữ liệu được mã hóa. Đối với mỗi Client được phép khai thác dữ liệu với các trường khác nhau sẽ có file XML khác nhau.



Hình 1. Mô hình DaaS đề xuất

Dữ liệu mã hóa được tạo ra theo cách sau: Mã hóa dữ liệu của từng cột. Đối với những cột có giá trị giống nhau dữ liệu sẽ được cộng thêm một chuỗi ngẫu nhiên *"-xxxxxx"* trong đó *x* là dạng chữ số, số lượng x là không cố định. Hiện tại, chuỗi ngẫu nhiên được sử dụng là 6 ký số *x*, các số x này được sinh ra ngẫu nhiên. Ví dụ, với dữ liệu là TRUE chuỗi sau khi thêm vào sẽ trở thành "TRUE-123456" sau đó sẽ mã hóa toàn bộ chuỗi và lưu trữ xuống DB tại DSP. Khi khôi phục dữ liệu, chỉ cần cắt chuỗi dựa trên ký tự "-" và chỉ cần lấy giá trị đầu tiên sẽ nhận được dữ liệu gốc là TRUE.

File XML được phát sinh bao gồm: Để giảm bớt dung lượng của mỗi file, sẽ đặt thêm 1 file XML để chứa giá trị băm của thông tin Client và với tất cả các giá trị băm trong chỉ mục sẽ chỉ sử dụng ½ giá trị băm đó. Với việc sử dụng 64 bit thì khả năng trùng giá trị khoảng 1 tỷ bản ghi mới xảy ra một lần.

C. Đánh giá mô hình đề xuất

Đảm bảo được tính bí mật: vì CSDL đã được mã hóa nên chính bản thân DSP cũng không thể biết nó chứa dữ liệu gì.

Đảm bảo truy vấn với các tính chất:

- Tính đúng: Có thể kiểm tra với 3 kiểu tấn công máy chủ DSP: Chỉnh sửa một bộ dữ liệu trong CSDL outsource; Thêm một bộ dữ liệu giả vào CSDL outsource; Xóa trực tiếp các bộ dữ liệu trong CSDL outsource hoặc xóa các bộ dữ liệu trong kết quả trả về từ máy chủ.
- Tính đủ: Có thể kiểm tra bằng cách duyệt chỉ mục tại web server, số lượng chỉ mục được truy vấn phải tương ứng với số lượng bản ghi được trả về từ máy chủ DSP.
- Tính mới: Chỉ có duy nhất DO là được phép thay đổi dữ liệu. Khi DO thay đổi/thêm dữ liệu mới vào CSDL outsource, dữ liệu sẽ được cập nhật và đồng thời chỉ mục của bản ghi đó tại web server sẽ được đặt ở cuối cùng, như vậy có thể nhận biết được bản ghi nào vừa mới được DO thay đổi/thêm mới.

D. Về tốc độ và khả năng xử lý chỉ mục XML

XML được chọn để làm chỉ mục vì nó hỗ trợ trên tất cả các hệ điều hành hiện nay thông qua các bộ phân tích (parser) như SAX hay DOM… và cũng có rất nhiều cách đọc XML. Khi đọc XML sẽ sử dụng SAX Parser để đọc vì thời gian xử lý của SAX nhanh hơn cho việc tìm kiếm so với XPath hay XQuery, còn XPath và XQuery chỉ thực thi truy vấn trên cây DOM. Để tạo ra được cây DOM, cần một quá trình phân tích file XML, sau đó chuyển thành cây DOM rồi mới tiến hành truy vấn được, trong khi SAX vừa phân tích XML vừa xử lý. Khi quá trình phân tích XML kết thúc thì đã có kết quả trả về. Hơn nữa SAX thực thi không lưu lại trên vùng nhớ nên rất thuận lợi khi xử lý trên web. Tại một thời điểm, số lượng người dùng tìm kiếm có thể sẽ rất lớn, nếu dùng XPath hay XQuery sẽ tạo ra rất nhiều cây DOM trên vùng nhớ, có thể sẽ gây tràn vùng nhớ.

Khi đọc, các chỉ mục XML sẽ gọi các xử lý song song để đọc nhiều chỉ mục XML cùng một lúc. Khi Client gọi câu lệnh tìm kiếm kết hợp nhiều trường hay khi DO cập nhật lại dữ liệu sẽ có nhiều file XML được xử lý, nếu xử lý tuần tự sẽ tốn rất nhiều thời gian, do vậy, giải pháp đề xuất cách xử lý song song các chỉ mục XML, đọc nhiều file XML cùng một lúc giúp tiết kiệm thời gian.

Ngoài ra, đối với file XML cha, mỗi khi xử lý, đều phải đọc file này để xác định bản ghi trong CSDL. Giải pháp đề xuất hướng xử lý đối với những file thường xuyên được đọc là sẽ nạp vào vùng cache, như vậy tốc độ đọc sẽ tăng lên do được truy xuất trực tiếp từ cache.

E. Về tốc độ và khả năng xử lý CSDL

Dữ liệu sẽ được đọc về từng phần, số lượng bản ghi có thể thay đổi, trong giải pháp đưa ra là thử nghiệm với 10 bản ghi cho một lần đọc, khi Client chuyển sang xem trang tiếp theo, lúc đó dữ liệu mới được đọc về tiếp (gần giống cách xử lý của Google), hơn nữa, Client có thể biết được tổng số bản ghi đã truy vấn và tổng số trang cần phải xem. Ưu điểm của cách xử lý này là tránh được việc đọc quá nhiều dữ liệu trong cùng một thời điểm. Khi ứng dụng được viết ở dạng web có thể xuất hiện tình trạng quá tải của server do tại một thời điểm, số lượng Client tăng đột ngột và cùng lúc truy vấn CSDL, nếu mỗi Client đều lấy về một lượng lớn bản ghi sẽ làm server không xử lý kịp, dẫn đến bị treo.

F. Về không gian lưu trữ chỉ mục XML

Mỗi file chỉ mục XML đại diện cho một trường trong CSDL chứa các thông tin: Giá trị băm của bản ghi (kiểm tra tính đúng của dữ liệu) dùng làm định danh để truy vấn trên CSDL đã mã hóa; Các giá trị băm dùng để tìm kiếm thông tin trên CSDL đã mã hóa.

Giải pháp để giảm dung lượng lưu trữ XML:

- Xây dựng 1 file chỉ mục tổng chứa toàn bộ các giá trị băm của từng bản ghi mà không đặt trên từng file để giảm dung lượng của chỉ mục.
- Các giá trị băm chỉ lấy ½ giá trị: như vậy

dung lượng của các chỉ mục sẽ giảm, với giá trị băm 128 bit (sử dụng hàm băm MD5) khi giảm ½ sẽ còn 64 bit tương ứng với 2^{-32} giá trị, như vậy xác suất va chạm trong CSDL vào khoảng 1 tỷ bản ghi.

Cách Thực thi truy vấn như sau:

Ví dụ, với bảng dữ liệu: *Employee (id, FirstName, LastName, Address, Email, Phone, salary, role, experience)* tất cả dữ liệu đều đã được mã hóa trước khi đưa lên đám mây. Để thực thi truy vấn, cần xử lý thông tin trong file chỉ mục XML.

Khi xử lý chỉ mục XML xong, có thể biết được ngay số lượng bản ghi cần phải truy vấn trên CSDL outsource. Lúc này giảm thời gian truy vấn trên CSDL cũng như giảm vùng nhớ khi phải đọc quá nhiều dữ liệu là điều hết sức cần thiết.

Dựa trên giá trị băm được tìm thấy khi đọc chỉ mục XML, server sẽ thực thi truy vấn với giá trị băm tương ứng trong CSDL outsource đã được mã hóa. Bên cạnh việc sử dụng XML như chỉ mục để tìm kiếm trong CSDL đã mã hóa, còn có thể đặt vào XML mức độ ưu tiên đọc dữ liệu hoặc quyền được phép hay không được phép đọc một số bản ghi nào đó mà hoàn toàn không cần phải cài đặt thêm thuộc tính vào CSDL đã có sẵn.

Số lượng bản ghi được tìm thấy sẽ được thông báo cho Client, web server sẽ tiến hành phân trang dữ liệu, lấy một phần trong tổng số bản ghi được tìm thấy, lọc dữ liệu cần thiết cho từng Client, sau đó giải mã rồi mới in ra giao diện và truyền về phía Client. Khi Client chuyển sang trang tiếp theo để xem, dữ liệu sẽ tiếp tục được lấy về và xử lý như trên trước khi truyền về phía Client.

Khi DO tiến hành các truy vấn xóa (delete), cập nhật (update) hoặc thêm mới (insert) các bản ghi trong CSDL thì file XML chỉ mục sẽ được cập nhật lại giá trị mới tương ứng với bản ghi bị thay đổi. Việc cập nhật dữ liệu thực hiện bằng cách đọc toàn bộ đoạn dữ liệu có chứa bản ghi cần update (hay sẽ chứa hàng insert) sau đó thực hiện cập nhật dữ liệu, tính toán lại giá trị băm của bản ghi rồi cập nhật trở lại server.

IV. KẾT QUẢ THỰC NGHIỆM

Để đánh giá mô hình thử nghiệm được đề xuất, nhóm tác giả thực hiện một số trường hợp thử nghiệm với thời gian thực thi là thời gian khi bắt đầu truy vấn đến khi dữ liệu được đọc từ ResultSet (con trỏ đọc dữ liệu của Microsoft SQL Server) và nạp vào các object để sẵn sàng truyền đi trên mạng (không tính thời gian hiển thị các object) như sau:

TH 1. Thời gian thực thi truy vấn thô trên CSDL không mã hóa *(tính bằng giây)*.

TH 2. Thời gian thực thi truy vấn trên chỉ mục và tải CSDL nạp một lần lên vùng nhớ rồi phân trang dữ liệu *(tính bằng giây)*.

TH 2.1 Thời gian thực thi truy vấn trên chỉ mục XML *(tính bằng giây)*.

TH 3. Thời gian thực thi truy vấn trên chỉ mục phân trang và tải CSDL trên từng trang, khi chuyển trang mới tải tiếp *(tính bằng giây)*.

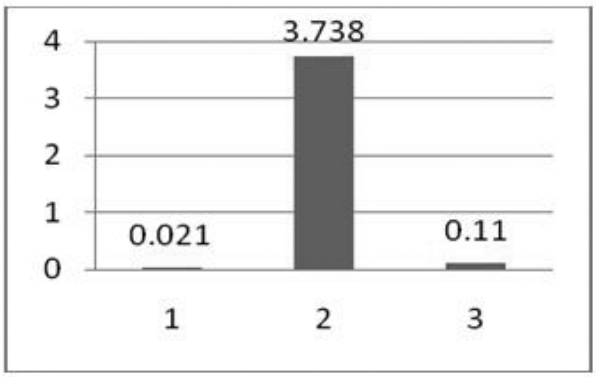
TH 3.1 Thời gian thực thi truy vấn trên chỉ mục XML *(tính bằng giây)*.

Thử nghiệm được tiến hành trên CSDL Mondial-3.0.xml [8] đã chuyển CSDL sang dạng bảng trong SQL Server 2005 và tiến hành mã hóa bảng thành phố (city) với 3.051 bản ghi được mã hóa. Các số liệu đo đạc dựa trên câu truy vấn tìm kiếm những thành phố có chỉ số dân số (population) > 500000, tương ứng với câu truy vấn trong XML là /mondial/country/city [population > 500000] (Q1).

Kết quả thực hiện câu truy vấn Q1 được ghi trong Bảng 1 như sau:

BẢNG 1. KẾT QUẢ THỰC NGHIỆM CHO TRUY VẤN Q1 (TÍNH BẰNG GIÂY)

Tổng số bản ghi	3051
Số bản ghi được tìm thấy	466
Trường hợp 1	0,021
Trường hợp 2	3,738
Trường hợp 2.1	0,016
Trường hợp 3	0,11
Trường hợp 3.1	0,018



Hình 2. Thời gian thực thi truy vấn

Hình 2 cho thấy thời gian thực thi câu truy vấn ở dạng không mã hóa ở cột 1 là tối ưu nhất, thời gian để đọc toàn bộ dữ liệu trong một lần ở cột 2

là quá lớn, mô hình đưa ra có thời gian xử lý ở cột 3 tuy không tối ưu bằng cột 1 nhưng chấp nhận được.

Vì số lượng bản ghi trong bảng city không đủ lớn, nên thử nghiệm sẽ tạo thêm một CSDL khác sử dụng SQL Server 2005 gồm có 1 bảng với các trường như sau: *Employee (id, FirstName, LastName, Address, Email, Phone, Salary, Role, Experience)* với số lượng bản ghi được thử nghiệm là 10.000, 20.000, 30.000, 40.000, 50.000, 60.000 và 70.000 bản ghi. Mỗi trường trong CSDL sinh ra 1 file XML chỉ mục tương ứng. Như vậy, quá trình tìm kiếm tương ứng với dữ liệu trên mỗi trường trong CSDL chính là quá trình duyệt file XML con. Dựa trên kết quả trả về khi duyệt file XML con, hệ thống sẽ đọc XML cha để lấy các định danh, quyền được phép đọc dữ liệu và các câu truy vấn tương ứng sẽ được phát sinh để lấy dữ liệu từ CSDL outsourced đã được mã hóa. Sau khi lấy được kết quả trả về, dữ liệu sẽ được giải mã và gửi cho Client.

Việc lấy đọc từ CSDL outsourced mỗi lần chỉ lấy về 1 lượng nhỏ bản ghi nên thời gian thực thi truy vấn trên SQL Server là không đáng kể, do đó việc tìm kiếm dữ liệu thời gian thực thi chủ yếu nằm ở việc xử lý file XML của SAX Parser.

BẢNG 2. KẾT QUẢ THỰC NGHIỆM CHO TRUY VẤN Q2 (TÍNH BẰNG GIÂY)

Tổng số bản ghi	Số bản ghi được tìm thấy	Trường hợp 1	Trường hợp 2	Trường hợp 2.1	Trường hợp 3	Trường hợp 3.1
11.000	149	0,062	1,972	0,07	0,181	0,067
20.695	249	0,104	3,838	0,118	0,268	0,115
30.940	349	0,142	7,084	0,168	0,364	0,173
40.962	449	0,169	11,127	0,227	0,479	0,224
51.000	549	0,204	15,757	0,29	0,559	0,267
60.640	649	0,237	21,93	0,333	0,782	0,356
70.851	749	0,26	28,911	0,388	0,741	0,374

Hai câu truy vấn khác nhau đã được xem xét từ CSDL trên là:

- Câu truy vấn đầu tiên thực thi truy vấn gần đúng với giá trị được lưu trữ trong cột Lastname. Tương ứng với câu lệnh trong SQL Server là:

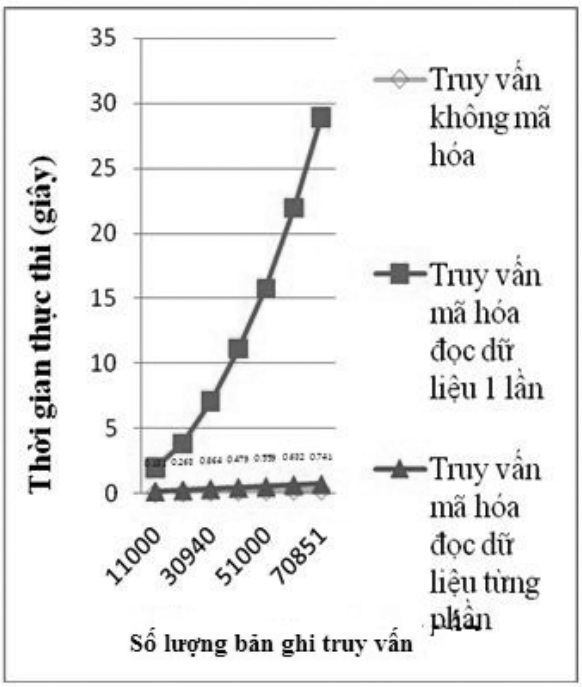
*SELECT * FROM Employee WHERE LastName LIKE?* (Q2)

- Câu truy vấn thứ hai thực thi truy vấn lấy về những giá trị trong một khoảng cho trước trong cột Salary. Tương ứng với câu lệnh trong SQL Server là:

*SELECT * FROM Employee WHERE Salary >= ? and Salary <= ?* (Q3)

Kết quả thực nghiệm cho 2 câu truy vấn Q2 và Q3 được ghi trong Bảng 2 và Bảng 3.

Đồ thị Hình 3 cho thấy, khi dữ liệu càng lớn đối với dạng truy vấn dữ liệu chỉ đọc 1 lần, độ dốc của đường biểu diễn thời gian thực thi càng lớn nên không có tính khả thi. Đối với dạng truy vấn đọc dữ liệu từng phần thời gian tăng do dung lượng file chỉ mục tăng, nhưng thời gian đọc file chỉ mục rất nhỏ, còn truy vấn CSDL mỗi lần chỉ đọc một lượng bản ghi nhất định nên thời gian là như nhau.



Hình 3. Đồ thị thời gian thực thi truy vấn Q2

BẢNG 3. KẾT QUẢ THỰC NGHIỆM CHO TRUY VẤN Q3 (TÍNH BẰNG GIÂY)

Tổng số bản ghi	Số bản ghi được tìm thấy	TH 1	TH 2	TH 2.1	TH 3	TH 3.1
11.000	149	0,025	1,783	0,037	0,144	0,035
20.695	249	0,039	3,882	0,067	0,208	0,061
30.940	349	0,049	6,953	0,094	0,281	0,081
40.962	449	0,059	11,015	0,111	0,382	0,104
51.000	549	0,063	15,738	0,135	0,42	0,133
60.640	649	0,073	21,889	0,158	0,49	0,162
70.851	749	0,089	28,591	0,284	0,568	0,194

Đồ thị Hình 4 cho thấy, khi lượng bản ghi tăng, việc đọc dữ liệu về trong một lần để xử lý là điều không thể thực hiện, hơn nữa trong môi trường web, số lượng Client tại một thời điểm có thể tăng rất nhanh. Với giải pháp dữ liệu đọc từng phần, thời gian thực thi thấp, nhưng so với đồ thị tại hình 3 thì do dung lượng thông tin trong file chỉ mục nhỏ hơn nên thời gian đọc ngắn hơn.

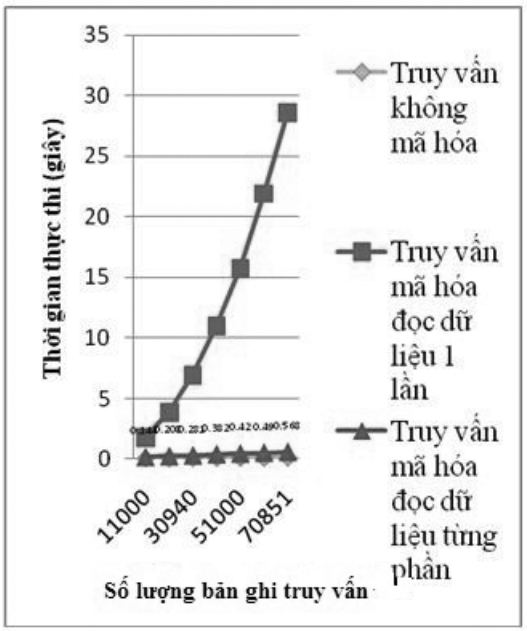
Từ kết quả thực nghiệm, có thể nhận thấy rằng, truy vấn trên dữ liệu thô (trường hợp 1) cho kết quả tối ưu nhất. Khi dữ liệu được outsource, sẽ không truy vấn được với bản rõ của dữ liệu mà phải truy vấn trên bản dữ liệu đã mã hóa.

Trường hợp 2 có chi phí cao hơn do phải lấy cùng một lúc toàn bộ dữ liệu về, như vậy, dữ liệu càng lớn thời gian thực thi càng lâu.

Trường hợp 3 có chi phí thấp nhất do dữ liệu được tải về từng phần, Client cũng không thể xem hết tất cả dữ liệu cùng một lúc, dựa vào cơ chế phân trang và đọc dữ liệu từng phần, khi Client nhấn chuyển trang, dữ liệu mới được đọc về tiếp.

Trong cả trường hợp 2 và 3, có thể nhận thấy, tốc độ tìm kiếm dữ liệu nhanh hay chậm là do việc xử lý chỉ mục XML, còn tốc độ xử lý chỉ mục XML là bằng nhau.

- Đối với chức năng SEARCH: với những Client có quyền khai thác dữ liệu của DO, chỉ khai thác được những trường họ được phép. Chỉ có DO mới có quyền khai thác toàn bộ dữ liệu bao gồm cả những bản ghi không được phép khai thác. Việc đánh dấu những bản ghi không được quyền khai thác còn có thể cho biết bản ghi nào mới được DO thêm vào hoặc mới được cập nhật lại.



Hình 4. Đồ thị thời gian thực thi truy vấn Q3

- Đối với CSDL có nhiều trường, nhiều bảng, sẽ xuất hiện nhiều chỉ mục XML được sinh ra, khi Client gọi chức năng SEARCH kết hợp nhiều thuộc tính của bảng hay nhiều bảng, hoặc khi DO cập nhật lại dữ liệu sẽ có rất nhiều file XML cần được xử lý, việc xử lý tuần tự nhiều file XML với dung lượng lớn là không khả thi vì thời gian xử lý quá lâu. Bài báo đưa ra giải pháp xử lý song song, tại một thời điểm xử lý nhiều file XML cùng một lúc, các kết quả trả về sẽ được tính toán tiếp.

Ngoài ra, đối với file XML cha thường xuyên được xử lý, sẽ đưa ra giải pháp nạp vào cache. Như vậy, khi được gọi xử lý các file này sẽ được đọc trực tiếp từ cache, nhanh hơn rất nhiều so với việc phải đọc lại file XML từ đầu, đây cũng là cách tối ưu hóa xử lý của các hệ điều hành.

- Đối với chức năng INSERT: khi DO thêm mới dữ liệu, dữ liệu sẽ được mã hóa sau đó được thêm vào CSDL của DSP, đồng thời tại Web server của DO sẽ phát sinh chỉ mục mới trên file XML của dữ liệu vừa mới được thêm vào.
- Đối với chức năng DELETE: khi DO gọi lệnh xóa dữ liệu, dữ liệu sẽ được xóa trên DSP đồng thời xóa thông tin của dữ liệu đó trên chỉ mục XML.
- Đối với chức năng UPDATE: khi DO cập nhật lại thông tin của một bản tin nào đó, dữ liệu sẽ được mã hóa và cập nhật tại CSDL của DSP, đồng thời tại Web server hệ thống sẽ xóa chỉ mục cũ và ghi lại chỉ mục mới với thông tin của bản ghi mới được cập nhật.
- Đối với hàm tính toán gộp như SUM, AVERAGE… [2] đề ra giải pháp giải quyết được tính đúng cho các câu truy vấn dạng chỉ đọc không tính toán gộp (SUM, AVERAGE) được bằng cách mỗi bản ghi kèm theo chữ ký số của bản ghi đó. Nhưng để giải quyết được tính đúng phải kiểm tra một lượng lớn chữ ký số là không khả thi. [2] cũng nêu giải pháp khác là Merkle Hash Tree (MHT) để đảm bảo kết quả trả về là đầy đủ, tuy nhiên mỗi cấu trúc MHT này chỉ dùng cho 1 thuộc tính của bảng, điều này sẽ làm tăng phí lưu trữ tại server. Với giải pháp được đề xuất, khi Client gọi chức năng tính toán gộp như SUM, AVERAGE… Web server sẽ tiến hành tìm kiếm trên chỉ mục XML những giá trị tương ứng và tính toán giá trị tổng của những bản ghi thỏa mãn điều kiện. Khi XML được duyệt xong cũng là lúc tổng được tính toán xong, giá trị trung bình chỉ cần lấy tổng chia cho số bản ghi thỏa mãn điều kiện được tìm thấy trong chỉ mục, sau đó các bản ghi sẽ được đọc

về từng phần thông qua phân trang dữ liệu. Bên cạnh đó, việc kiểm tra tính đúng, tính đủ và tính mới của dữ liệu bằng cách sử dụng ½ giá trị băm cũng giúp tiết kiệm không gian lưu trữ.

- Đối với dung lượng chỉ mục XML: với CSDL test Employee có 9 trường sẽ có 9 file chỉ mục cho từng trường, và với cách lấy ½ giá trị băm, thử nghiệm với 10.000, 20.000, 30.000, 40.000, 50.000, 60.000 và 70.000 bản ghi:

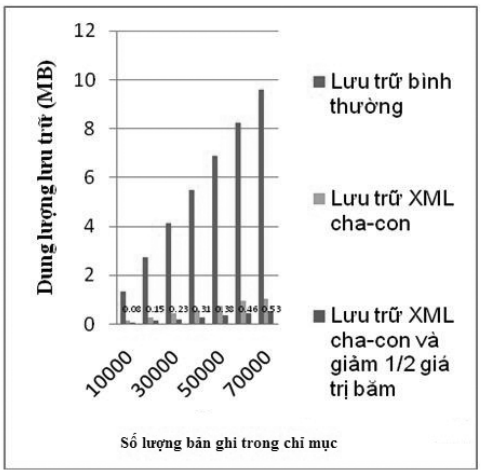
Trường hợp 1: lưu trữ bình thường chi phí cho chỉ mục, chỉ tính giá trị băm của khóa chưa tính các giá trị băm khác.

Trường hợp 2: thêm 1 file XML tổng, lưu toàn bộ giá trị băm của khóa.

Trường hợp 3: thêm 1 file XML tổng kết hợp với giá trị băm giảm ½

BẢNG 4: DUNG LƯỢNG CHỈ MỤC XML CHỈ TÍNH GIÁ TRỊ KHÓA (TÍNH BẰNG MB)

Tổng số bản ghi	Trường hợp 1	Trường hợp 2	Trường hợp 3
10000	1.37	0.15	0.08
20000	2.75	0.31	0.15
30000	4.12	0.46	0.23
40000	5.49	0.61	0.31
50000	6.87	0.76	0.38
60000	8.24	0.96	0.46
70000	9.61	1.07	0.53



Hình 5. Dung lượng lưu trữ của chỉ mục XML

Đồ thị trong Hình 5 cho thấy, chỉ với thuộc tính khóa dung lượng trong các file chỉ mục đã giảm rất nhiều khi tách khóa nằm ở 1 file riêng. Hơn nữa, với các giá trị băm khác trong các chỉ mục đều giảm ½ độ dài, sẽ tiết kiệm được rất nhiều không gian lưu trữ chỉ mục.

V. KẾT LUẬN

Mô hình outsource CSDL mang lại cho DO rất nhiều lợi ích. Tuy nhiên, vấn đề bảo đảm an toàn cho dữ liệu cần phải hết sức quan tâm. Trong bài báo này, chúng tôi đã đề xuất một giải pháp kết hợp việc mã hóa dữ liệu (sử dụng AES), tìm kiếm dữ liệu đã mã hóa (thông qua chỉ mục XML) và truyền dữ liệu về phía đối tượng được phép khai thác dữ liệu (thông qua SSL).

Ưu điểm của giải pháp được đề xuất:

- Sử dụng các giao thức chạy trên web, chỉ cần cấu hình Web Server, các Client được phép khai thác dữ liệu có thể sử dụng bất kỳ trình duyệt web nào để có thể truy vấn và lấy dữ liệu mà không cần phải cài đặt thêm bất kỳ thư viện/phần mềm nào;
- Đảm bảo tốc độ xử lý tìm kiếm: Giải pháp đề xuất là lấy dữ liệu về từng phần như vậy tránh cho server phải đọc cùng lúc quá nhiều dữ liệu;
- Sử dụng hàm băm để kiểm tra tính đúng của dữ liệu, tốc độ xử lý nhanh hơn nhiều so với phương pháp sử dụng chữ ký số;
- Có thể chống lại tấn công kiểu suy diễn, với những bản ghi có nội dung giống nhau có thể biến đổi để sinh ra các bộ mã hóa khác nhau;
- Sử dụng chỉ mục là XML với phân tích cú pháp SAX để tìm kiếm làm giảm thời gian và vùng nhớ để xử lý hơn XPath hay Xquery;
- Tăng tốc xử lý các file XML thông qua lập trình xử lý song song;
- Tối ưu hóa xử lý thông qua việc đặt file XML thường xuyên được xử lý vào cache tương tự như cách xử lý của các hệ điều hành;
- Giải quyết được một số hàm tính toán gộp như SUM, AVERAGE;
- Tỷ lệ giãn tin do việc mã hóa sinh ra khi so sánh giữa CSDL mã hóa và CSDL không mã hóa là 1:1,86;
- Khi có nhiều người dùng truy cập lên cùng lúc vẫn truy vấn được dữ liệu và đảm bảo an toàn và đủ dữ liệu, tuy nhiên việc xử lý nhiều người dùng tại cùng 1 thời điểm còn phụ thuộc vào khả năng đáp ứng của server, trong trường hợp số lượng người dùng quá lớn có thể mở rộng cấu hình server hoặc sử dụng mô hình Clustering.

Ngoài các vấn đề đã nêu, còn một vấn đề khác cần được giải quyết là đảm bảo tính riêng tư (User Privacy) của các Client. Bên cạnh đó, đối với các khu vực cần đảm bảo an toàn cao (an ninh, quốc phòng) không nên sử dụng giao thức thương mại

SSL. Nhóm tác giả đang hướng đến giải pháp truyền dữ liệu về phía Client rồi mới giải mã. Như vậy, dữ liệu vẫn đang được mã hóa trên đường truyền nên sẽ không cần dùng đến giao thức SSL.

TÀI LIỆU THAM KHẢO

[1]. E. Mykletun and G. Tsudik, "Aggregation Queries in the Database-As-a-Service Model", IFIP WG 11.3 Working Conference on Data and Applications Security, July 2006.

[2]. Einar Mykletun, Maithili Narasimha, Gene Tsudik, "Authentication andIntegrity in Outsourced Databases", In ISOC Symposium on Network and Distributed System Security NDSS, 2004

[3]. Feifei Li, Marios Hadjieleftheriou, George Kollios, Leonid Reyzin, "Dynamic Authenticated Index Structures for Outsourced Databases", Sigmod 2006, June 27-29, 2006, Chicago, Illinois, USA.

[4]. Hacigümüş H., Iyer B. R., Li C., Mehrotra S., "Executing SQL over encrypted data in the database service provider model", Proc. of the ACM SIGMOD, 2002, pp. 216-227.

[5]. Maithili Narasimha, Gene Tsudik, "DSAC: An Approach to EnsureIntegrity of Outsourced Databases using Signature Aggregation and Chaining", ACM Conference on Information and Knowledge Management(CIKM'05), November 2005.

[6]. Radu Sion, "Query Executing Assurance for Outsourced Databases", Proceedings of the 31st VLDB Conference, Trondheim, Norway 2005.

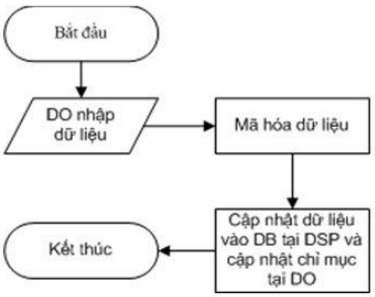
[7]. R. Brinkman, L. Feng, J. Doumen, P.H. Hartel, and W. Jonker, "EfficientTree Search in Encrypted Data", Information System Security Journal, vol.13, pp. 14-21, July 2004.

[8]. Sample dataset: World geographic database integrated from the CIA WorldFactbook, the International Atlas, and the TERRA database among othersources. http://www.cs.washington.edu/research/xmldatasets/data/mondial/mondial-3.0.xml

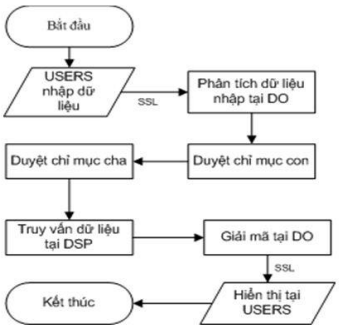
[9]. Tran Khanh Dang, "Ensuring correctness, completeness, and freshness for Outsourced Tree-Indexed Data", Information Resource Management Journal, 2008.

PHỤ LỤC
MỘT SỐ LƯU ĐỒ THUẬT TOÁN

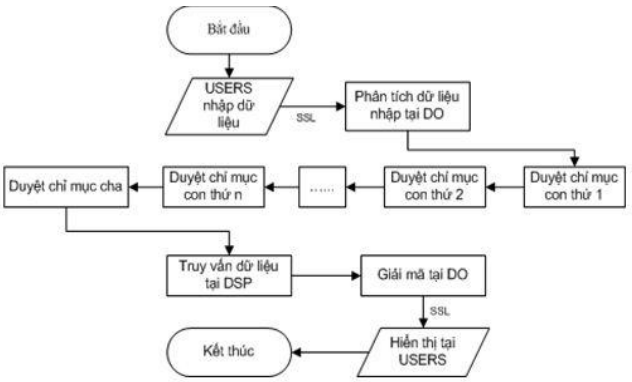
1. Quá trình nhập liệu



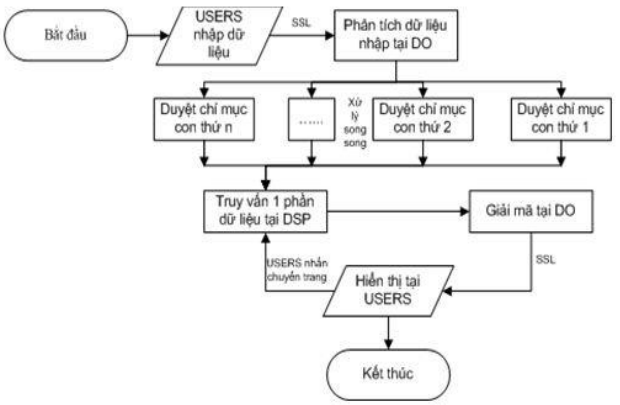
2. Quá trình tìm kiếm trên 1 trường



3. Quá trình tìm kiếm trên n trường



4. Cải tiến quá trình tìm kiếm trên n trường



SƠ LƯỢC VỀ TÁC GIẢ

PGS. TS. Nguyễn Hiếu Minh

Đơn vị công tác: Học viện Kỹ thuật quân sự, Bộ Quốc phòng, Hà Nội.

E-mail: hieuminhmta@gmail.com

Tốt nghiệp đại học và Thạc sĩ chuyên ngành Vô tuyến Điện tử, Học viện Kỹ thuật Quân sự năm 1993 và 1999. Nhận bằng Tiến sĩ Công nghệ Thông tin - Đại học Kỹ thuật Điện Saint -Peterburg, Liên bang Nga năm 2006.

Hướng nghiên cứu hiện nay: An toàn mạng, mật mã, công nghệ mạng.

ThS. Phạm Công Thành

Đơn vị công tác: Viện đào tạo quốc tế FPT TP. Hồ Chí Minh, TP. Hồ Chí Minh.

E-mail: phamcongthanh1983@gmail.com

Tốt nghiệp Đại học Mở TP. Hồ Chí Minh năm 2006. Nhận bằng Thạc sĩ ngành Khoa học Máy tính, Học viện Kỹ thuật Quân sự năm 2014.

Hướng nghiên cứu hiện nay: Lập trình Web với JAVA và lập trình di động với Android.

ThS. Hồ Kim Giàu

Đơn vị công tác: Đại học Thông tin liên lạc, Khánh Hòa.

Email: hkgiau@gmail.com

Tốt nghiệp đại học Khoa học -Tự nhiên, TP. Hồ Chí Minh năm 2005. Nhận bằng Thạc sỹ tại Học viện Bưu chính Viễn thông TP. Hồ Chí Minh năm 2011.

Hướng nghiên cứu hiện nay: An toàn mạng.

Trần Lê Hoàng Tuấn

Đơn vị công tác: Ngân hàng TMCP Việt Nam Thương tín

E-mail: hoangtuan1187@yahoo.com