

**ĐẠI HỌC QUỐC GIA HÀ NỘI  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**



**Trần Thế Lâm**

**ỨNG DỤNG CÔNG NGHỆ CHUỖI KHỎI  
ĐỂ TĂNG CƯỜNG AN TOÀN  
CHO CÁC HỆ THỐNG SCADA**

**TÓM TẮT LUẬN VĂN TỐT NGHIỆP THẠC SĨ**  
**Chuyên ngành: Kỹ thuật phần mềm**

**HÀ NỘI - 2024**

ĐẠI HỌC QUỐC GIA HÀ NỘI  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Trần Thế Lâm

ỨNG DỤNG CÔNG NGHỆ CHUỖI KHỎI  
ĐỂ TĂNG CƯỜNG AN TOÀN  
CHO CÁC HỆ THỐNG SCADA

TÓM TẮT LUẬN VĂN TỐT NGHIỆP THẠC SĨ

Ngành: Kỹ thuật phần mềm

Chuyên ngành: Kỹ thuật phần mềm

Giảng viên hướng dẫn: PGS. TS. Trương Anh Hoàng

HÀ NỘI - 2024

# Mục lục

<b>Chương 1</b>	<b>Giới thiệu</b>	<b>1</b>
1.1.	Đặt vấn đề . . . . .	1
1.2.	Mục tiêu nghiên cứu . . . . .	2
<b>Chương 2</b>	<b>Kiến thức nền tảng</b>	<b>4</b>
2.1.	Giới thiệu SCADA . . . . .	4
2.1.1.	Kiến trúc hệ thống . . . . .	5
2.1.2.	Tầm quan trọng của dữ liệu và thực trạng . . . . .	6
2.2.	Giới thiệu về chuỗi khối . . . . .	6
2.2.1.	Khái niệm và cơ sở lý thuyết chuỗi khối . . . . .	6
2.2.2.	Nền tảng Hyperledger Fabric . . . . .	8
<b>Chương 3</b>	<b>Phân tích yêu cầu tăng cường an toàn cho hệ thống SCADA</b>	<b>11</b>
3.1.	Yêu cầu chi tiết . . . . .	11
3.2.	Kiến trúc hệ thống . . . . .	12
3.3.	Thiết kế ứng dụng minh họa . . . . .	13
<b>Chương 4</b>	<b>Lập trình và thực nghiệm</b>	<b>16</b>
4.1.	Tổng quan . . . . .	16
4.2.	Triển khai mạng thử nghiệm . . . . .	17
4.3.	Xây dựng chuỗi mã . . . . .	17
4.4.	Phát triển phụ trợ . . . . .	18
4.5.	Thực thi . . . . .	19

<b>Chương 5</b>	<b>Đảm bảo chất lượng mã</b>	<b>20</b>
5.1.	Kiểm chứng . . . . .	20
5.2.	Kiểm thử tính bảo mật trên mạng thử nghiệm . . . . .	21
<b>Chương 6</b>	<b>Kết luận</b>	<b>22</b>
<b>Tài liệu tham khảo</b>		<b>25</b>

# Chương 1

## Giới thiệu

### 1.1. Đặt vấn đề

Với sự phát triển liên tục của các ngành công nghiệp, SCADA đã trở nên phổ biến được áp dụng trong các hệ thống phức tạp như điện lực, dầu khí[1]. Do đó, việc đưa ra quyết định của người vận hành cần được thực hiện một cách cẩn thận để tránh gây ra những tổn hại nghiêm trọng. Để có cái nhìn tổng quan về thực tế của hệ thống, người vận hành cần dựa vào thông số và dữ liệu lịch sử[2]. Vì vậy, dữ liệu đóng vai trò rất quan trọng và cần thiết để thực hiện các chức năng quản lý, kiểm soát và giám sát. Ngoài ra, dữ liệu trong các hệ thống SCADA cũng có vai trò quan trọng trong việc tạo ra các báo cáo, đánh giá và dự đoán về hoạt động của quy trình[3]. Dữ liệu cung cấp thông tin quan trọng để ra quyết định, định hướng cho việc tối ưu hóa quy trình và nâng cao hiệu suất.

Hiện nay, dữ liệu trên hệ thống SCADA được lưu trữ theo nhiều phương pháp truyền thống khác nhau như lưu trữ dưới dạng tệp, cơ sở dữ liệu, hệ thống lưu trữ đám mây[2]. Cơ sở dữ liệu có thể bị tấn công bằng cách sử dụng SQL injection và việc lưu trữ dữ liệu tập trung tại một nơi rủi ro khiến toàn bộ hệ thống SCADA có thể bị ảnh hưởng[4]. Lưu trữ dữ liệu trên đám mây cũng có nguy cơ rủi ro bảo mật như rò rỉ thông tin hoặc mất mát dữ liệu[5]. Để giảm thiểu những điểm yếu này, hệ thống SCADA cần triển khai các biện pháp bảo mật mạnh mẽ hơn để giảm thiểu các lỗ hổng và tạo ra một hệ thống lưu trữ dữ liệu an toàn và tin cậy.

Chuỗi khối là công nghệ mang lại sự tin cậy và đảm bảo tính toàn vẹn của dữ liệu. Khi có ít nhất hai bên tham gia vào mạng lưới, không có bên nào có quyền kiểm soát hoàn toàn thông tin. Điều này đồng nghĩa với việc các bản ghi không thể bị thay đổi, và không có bên nào trong hệ thống có thể xóa bỏ các dấu vết. Với việc sử dụng công nghệ chuỗi khối, không ai trong hệ thống có thể thay đổi hoặc xóa bỏ các giao dịch đã được ghi lại trước đó. Điều này đảm bảo rằng mọi bên trong hệ thống phải tuân thủ các quy tắc và quy định đã được thiết lập, và không thể che giấu hoặc thay đổi thông tin một cách bất hợp pháp.

Đảm bảo chất lượng mã là một trong những yếu tố then chốt để xây dựng và phát triển phần mềm có độ tin cậy. Để đảm bảo chất lượng mã, việc áp dụng một quy trình kiểm tra và sửa lỗi bài bản là điều cần thiết. Quy trình này bao gồm kiểm chứng và kiểm thử. Qua đó, chất lượng mã được đảm bảo và phần mềm hoạt động một cách chính xác, đáp ứng yêu cầu và tin cậy. Việc sử dụng các công cụ hỗ trợ đảm bảo chất lượng mã là vô cùng cần thiết để phát hiện lỗi hổng và ngăn ngừa thiệt hại tiềm ẩn. Hiện nay, với sự phát triển của AI, nhà phát triển có thể nhận được những gợi ý sửa lỗi hiệu quả, từ đó nâng cao chất lượng và an toàn của mã nguồn.

## **1.2. Mục tiêu nghiên cứu**

Luận văn hướng tới hai mục tiêu chính:

- Ứng dụng công nghệ chuỗi khối để tăng cường an toàn, bảo mật cho hệ thống SCADA. Chuỗi khối là một công nghệ tiên tiến giúp giảm bớt sự phụ thuộc của các bên trung gian, đảm bảo tính toàn vẹn dữ liệu và tăng cường tính bảo mật. Một trong những nền tảng chuỗi khối phổ biến được ứng dụng rộng rãi trong doanh nghiệp hiện nay là Hyperledger Fabric. Nghiên cứu này sẽ tiến hành thử nghiệm lưu trữ dữ liệu trên nền tảng Hyperledger Fabric nhằm đánh giá các ưu điểm và nhược điểm của công nghệ chuỗi khối so với các phương pháp lưu trữ truyền thống.
- Nghiên cứu các công cụ kiểm chứng kiểm thử để nâng cao chất lượng chuỗi mã.. Mục tiêu của nghiên cứu là tìm hiểu và đánh giá các công cụ hỗ trợ

kiểm chứng, kiểm thử mã nguồn viết bằng ngôn ngữ Go. Từ đó, đề xuất những công cụ thích hợp và áp dụng chúng vào quá trình phát triển. Trong quá trình nghiên cứu, tính năng *đề xuất sửa lỗi* đã được bổ sung vào một công cụ phát hiện lỗi, nhằm hỗ trợ quá trình sửa lỗi diễn ra nhanh chóng và hiệu quả hơn.

## Chương 2

# Kiến thức nền tảng

Chương này trình bày các kiến thức nền tảng liên quan đến đề tài nghiên cứu, tập trung vào việc giới thiệu kiến trúc và mô hình hoạt động của hệ thống SCADA kết hợp với công nghệ chuỗi khối. Ngoài ra, nền tảng chuỗi khối Hyperledger Fabric, được áp dụng trong các thí nghiệm của luận văn, cũng được mô tả chi tiết.

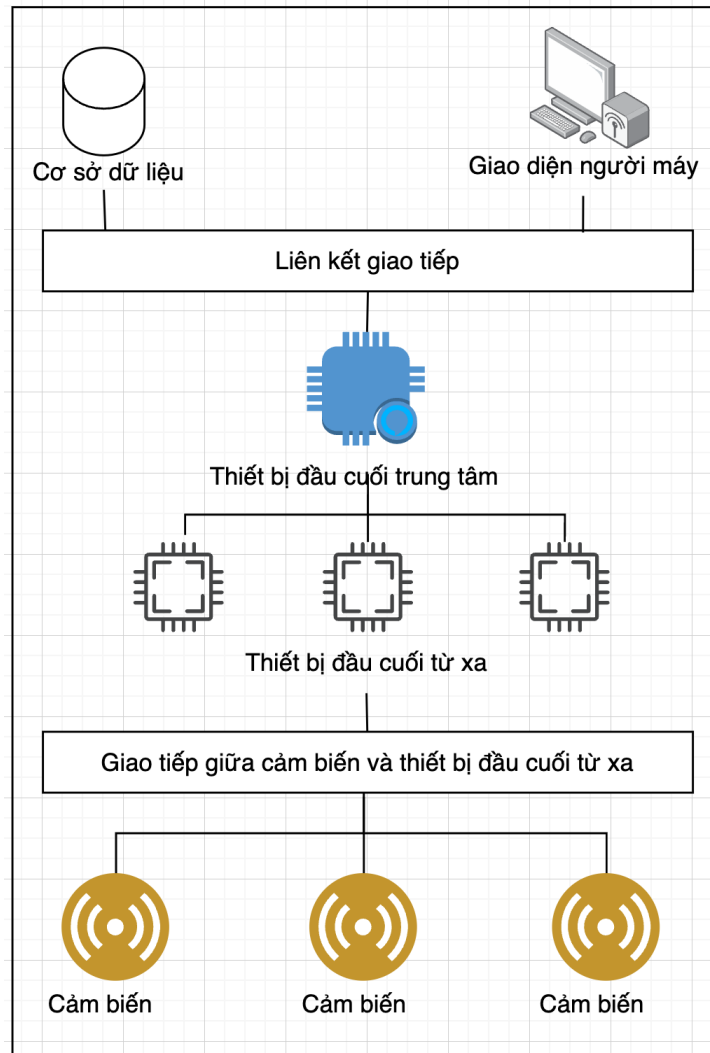
### 2.1. Giới thiệu SCADA

Hệ thống SCADA (Supervisory Control and Data Acquisition) là hệ thống điều khiển giám sát và thu thập dữ liệu. Hệ thống này bao gồm phần mềm và phần cứng, đồng thời cho phép thu thập dữ liệu từ xa và tại chỗ từ thiết bị công nghiệp. SCADA ra đời từ nỗ lực của các tổ chức công nghiệp nhằm giải quyết vấn đề dựa vào người vận hành để giám sát và điều khiển thiết bị theo cách thủ công bằng cách sử dụng nút bấm, công tắc chọn và quay số tương tự[1]. SCADA được triển khai ở trong nhiều ngành công nghiệp như: dầu khí, điện, vận tải, năng lượng tái tạo. Với nền công nghiệp phát triển tiên tiến, hệ thống SCADA đóng vai trò vô cùng quan trọng trong việc tối ưu hóa hoạt động của các nhà máy công nghiệp. Những lợi ích mà hệ thống SCADA mang lại là không thể phủ nhận. Tuy nhiên, SCADA cũng phải đối mặt với khó khăn, điển hình là việc triển khai SCADA[6]. Hệ thống SCADA có khả năng tích hợp với một số công nghệ khác như IoT, AI, phân tích dữ liệu lớn[6].



### 2.1.1. Kiến trúc hệ thống

Kiến trúc SCADA là một hệ thống phức tạp được thiết kế để điều khiển giám sát và thu thập dữ liệu từ các thiết bị trong một hệ thống công nghiệp được thể hiện ở hình 2.1.



Hình 2.1: Kiến trúc các thành phần trong hệ thống SCADA[7].

Hệ thống SCADA được xây dựng dựa trên cơ sở các thành phần chính như: *Thiết bị đầu cuối từ xa (RTU)*, *Thiết bị đầu cuối trung tâm (MTU)*, bộ chuyển động và cảm biến. Phần mềm bao gồm: *Giao diện người máy(HMI)*, một cơ sở dữ liệu trung tâm và phần mềm người dùng khác[8].

### 2.1.2. Tầm quan trọng của dữ liệu và thực trạng

Trong môi trường công nghiệp, việc thu thập và lưu trữ dữ liệu là một yếu tố quan trọng để đảm bảo sự vận hành hiệu quả và an toàn. Dữ liệu cho phép người vận hành có cái nhìn toàn diện về các hoạt động được diễn ra trong quá khứ. Điều này rất quan trọng để đưa ra các quyết định thông minh và đúng đắn trong quá trình vận hành. Hiện nay, dữ liệu lịch sử thường được lưu vào các hệ quản trị cơ sở dữ liệu như MySQL, SQLite, ... [9]. Khi các cơ sở dữ liệu tập trung này không được bảo vệ đúng cách sẽ dễ dàng bị tin tặc xâm nhập vào hệ thống và truy cập, sửa đổi hoặc xóa dữ liệu lịch sử. Vì lý những do đó, cần tìm một giải pháp mang tính an toàn và bảo mật cao hơn cho việc xử lý dữ liệu.

## 2.2. Giới thiệu về chuỗi khối

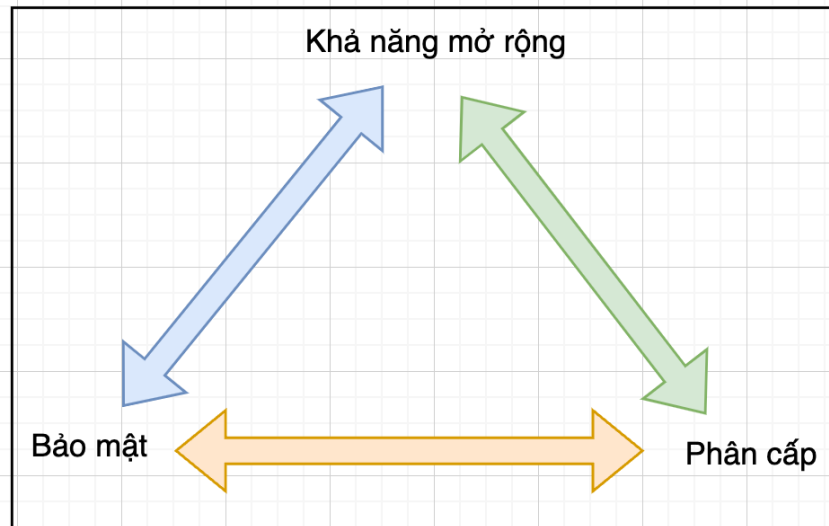
### 2.2.1. Khái niệm và cơ sở lý thuyết chuỗi khối

#### Định nghĩa

Chuỗi khối là một cơ sở dữ liệu phân cấp lưu trữ các khối thông tin được liên kết với nhau bằng mã hóa, mở rộng theo thời gian với mục tiêu để chống lại sự thay đổi của dữ liệu [10]. Những đặc điểm cốt lõi của chuỗi khối bao gồm *không thể thay đổi, phi tập trung, thúc đẩy sự đồng thuận, minh bạch* [11]:

#### Bộ ba bất khả thi của chuỗi khối

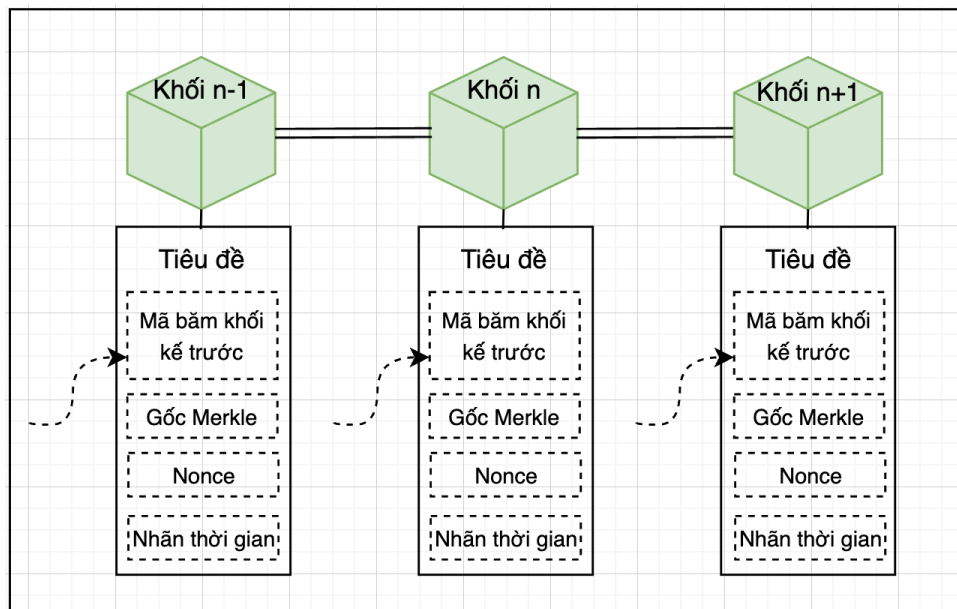
Chuỗi khối đang được sử dụng để lưu trữ dữ liệu đáng tin cậy trong nhiều hệ thống phần mềm [12]. Tuy nhiên, việc thiết kế và triển khai chuỗi khối luôn gặp phải những thách thức. Bộ ba bất khả thi của chuỗi khối là một trong những thách thức nổi bật nhất mà bất cứ nền tảng chuỗi khối nào đều muốn giải quyết. Nó được phát sinh từ số lượng giao dịch ngày càng tăng nhưng thông lượng hạn chế trong các nền tảng chuỗi khối hiện nay [12]. Bộ ba bất khả thi tuyên bố rằng một nền tảng chuỗi khối chỉ có thể đáp ứng hai trong số ba thuộc tính: khả năng mở rộng, phân cấp và bảo mật [12].



Hình 2.2: Bộ ba bất khả thi của chuỗi khối[13].

### Kiến trúc chuỗi khối

Chuỗi khối bao gồm các khối được liên kết với nhau, được minh họa trong hình 2.3. Mỗi khối bao gồm các giao dịch. Một khối sẽ bao gồm một tiêu đề khối và danh sách các giao dịch. Trong đó, tiêu đề khối bao gồm *mã băm khối kế trước*, *gốc merkle*, *nonce*, *nhãn thời gian*.



Hình 2.3: Kiến trúc chuỗi khối[14].

## **Cách thức hoạt động**

Khi một giao dịch được thực hiện, nó sẽ được gửi lên mạng lưới chuỗi khối để tiến hành xác thực và được thêm vào chuỗi. Một khối mới, đại diện cho các giao dịch mới, sẽ được khởi tạo và bao gồm thông tin về các giao dịch, dữ liệu của khối trước đó cùng với thời gian tạo khối. Sau đó, khối này sẽ được phân phối đến tất cả các nút trong mạng lưới. Các nút nhận khối mới sẽ bắt đầu quá trình xác thực các giao dịch bên trong khối. Sau khi các giao dịch được xác thực thành công, các nút sẽ thêm khối vào chuỗi bằng cách tính toán một giá trị băm cho khối mới, dựa trên dữ liệu trong khối và định danh của khối trước đó.

### **2.2.2. Nền tảng Hyperledger Fabric**

#### **Giới thiệu chung**

Hyperledger Fabric là nền tảng công nghệ sổ cái phân tán được cấp phép mã nguồn mở và được thiết kế để sử dụng trong bối cảnh doanh nghiệp, mang lại một số khả năng khác biệt chính so với các nền tảng sổ cái phân tán hoặc chuỗi khối phổ biến khác[15]. Hyperledger Fabric là chuỗi khối đầu tiên hỗ trợ các hợp đồng thông minh được viết bằng các ngôn ngữ lập trình có mục đích chung như Java, Go và NodeJS thay vì việc sử dụng những ngôn ngữ đặc thù[15].

#### **Lưu trữ**

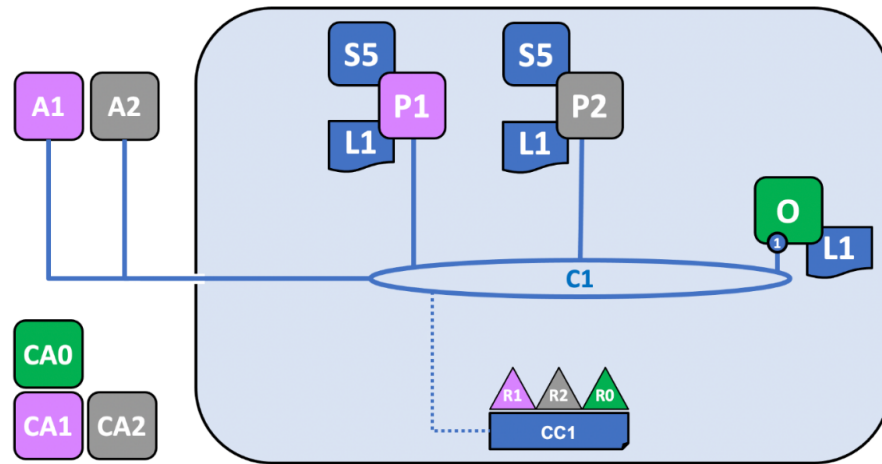
Hyperledger Fabric sử dụng một kiến trúc lưu trữ kết hợp giữa tệp và cơ sở dữ liệu nhằm đảm bảo tính bất biến, hiệu quả và linh hoạt trong quản lý dữ liệu[16]. Trạng thái hiện tại của mạng được lưu trữ trong các cơ sở dữ liệu như LevelDB hoặc CouchDB. Trong khi đó, chuỗi khối được lưu trữ dưới dạng các khối liên kết với nhau thông qua hàm băm và được ghi nhận trong các tệp.

#### **Mô hình**

Hyperledger Fabric được thiết kế để đáp ứng tính linh hoạt của từng nghiệp vụ đặc thù trong môi trường doanh nghiệp. Các thành phần cấu thành mô hình này bao

gồm tài sản, chuỗi mã, tính năng số cái, tính riêng tư, dịch vụ thành viên và bảo mật, đồng thuận [17]:

### Cấu trúc mạng



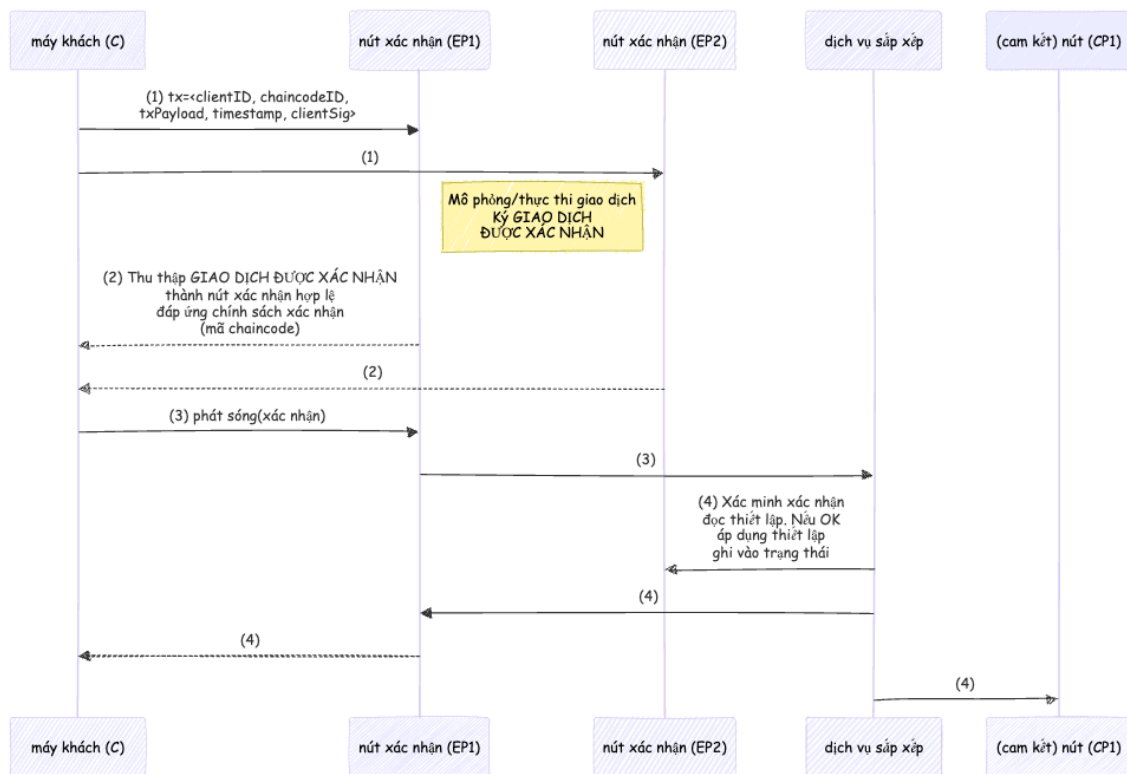
Hình 2.4: Cấu trúc mạng cơ bản[18].

Mạng của Hyperledger Fabric bao gồm các tổ chức, các nút, các kênh và các dịch vụ sắp xếp. Mỗi tổ chức đại diện cho một thực thể hoạt động độc lập, và có thể có nhiều nút để thực hiện các giao dịch và chia sẻ dữ liệu. Các kênh cho phép các thành viên trong mạng tạo ra các kênh riêng tư để trao đổi thông tin và thực hiện giao dịch chỉ giới hạn cho những thành viên tham gia trong kênh đó. Các dịch vụ sắp xếp giúp đảm bảo rằng các giao dịch được xác nhận và thực hiện theo đúng thứ tự.

### Luồng giao dịch

Luồng giao dịch của Hyperledger Fabric bao gồm các bước quan trọng để đảm bảo tính toàn vẹn và bảo mật của dữ liệu trong mạng. Trong Hyperledger, một giao dịch thường đi qua 6 bước [19]:

- ① Ứng dụng khách khởi tạo một giao dịch.
- ② Các nút trong mạng chứng thực, xác minh chữ ký và thực hiện giao dịch.



Hình 2.5: Luồng thực hiện một giao dịch Hyperledger Fabric[19].

- ③ Ứng dụng sẽ kiểm tra các phản hồi đề xuất, xác minh chữ ký nút xác nhận và so sánh tính đồng nhất của các phản hồi đề xuất.
- ④ Ứng dụng khách tập hợp các xác nhận thành một giao dịch.
- ⑤ Giao dịch được xác nhận và cam kết.
- ⑥ Sổ cái được cập nhật.

### Trường hợp sử dụng

Hyperledger Fabric cung cấp một môi trường phát triển linh hoạt, mạnh mẽ cho việc xây dựng các hệ thống chuỗi khối tùy chỉnh, đáng tin cậy và có tính mở rộng. Vì vậy, Hyperledger được áp dụng trong rất nhiều lĩnh vực như: chăm sóc sức khỏe, giáo dục và đào tạo, quản lý chuỗi cung ứng, bảo hiểm[20].

## Chương 3

# Phân tích yêu cầu tăng cường an toàn cho hệ thống SCADA

Trong chương này, nghiên cứu tập trung vào việc thiết kế và triển khai ứng dụng Hyperledger Fabric trong hệ thống SCADA, nhằm mục tiêu phát triển một hệ thống SCADA với tính bảo mật cao và độ tin cậy vượt trội. Quá trình thiết kế và triển khai bao gồm việc xác định các thành phần và chức năng thiết yếu của hệ thống SCADA, bao gồm quản lý, xử lý dữ liệu, xác thực người dùng và thiết bị, cũng như quản lý quyền truy cập. Các biện pháp bảo mật như mã hóa được áp dụng để đảm bảo sự an toàn và đáng tin cậy của dữ liệu trong hệ thống.

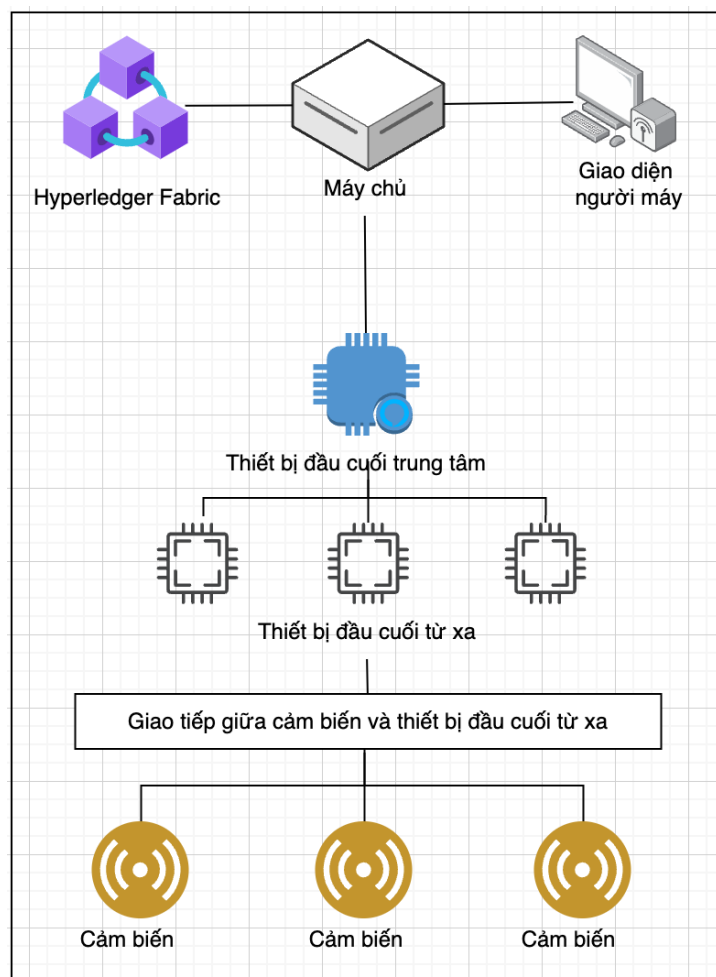
### 3.1. Yêu cầu chi tiết

Trong luận văn này, một hệ thống lưu trữ thông tin cảm biến khi phát hiện sự cố và thông tin người dùng trong hệ thống SCADA được xây dựng dựa trên công nghệ Hyperledger Fabric. Mỗi cảm biến sẽ gửi dữ liệu lên chuỗi khối thông qua API ngay khi phát hiện sự thay đổi bất thường trong môi trường hoạt động. Đồng thời, thông tin đăng nhập của người dùng sẽ được xác thực và lưu trữ trên chuỗi khối nhằm đảm bảo tính bảo mật và tính toàn vẹn của dữ liệu. Những đặc tả yêu cầu bao gồm:

- *Ghi nhận dữ liệu từ cảm biến.*

- Đăng nhập và xác thực người dùng.
- Xem danh sách và tìm kiếm lịch sử của cảm biến.
- Thêm/sửa/xóa người dùng.
- Xem lịch sử truy cập.
- Đổi mật khẩu.

## 3.2. Kiến trúc hệ thống



Hình 3.1: Tổng quan hệ thống SCADA kết hợp Fabric.

Trong nghiên cứu này, một kiến trúc hệ thống mới được đề xuất, dựa trên kiến trúc đã được trình bày tại 2.1.1.. Hình 3.1 minh họa kiến trúc tổng quan hệ thống

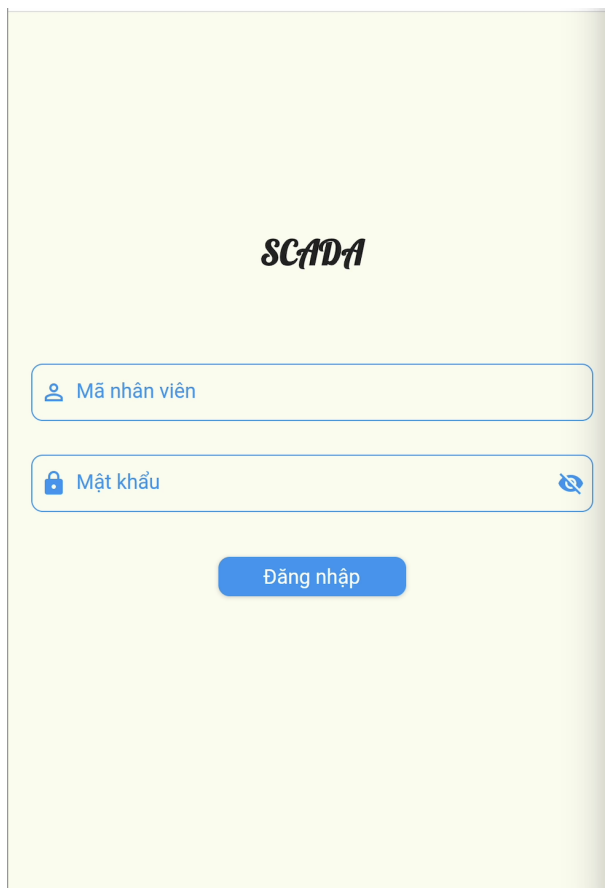


SCADA kết hợp Hyperledger Fabric. Bên cạnh các thành phần đã được giới thiệu, nghiên cứu thực hiện thay thế cơ sở dữ liệu bằng Hyperledger Fabric và bổ sung thêm một máy chủ. Máy chủ chịu trách nhiệm xử lý và phân tích dữ liệu được thu thập từ các thiết bị đầu cuối. Trong khi đó, Hyperledger Fabric chịu trách nhiệm lưu trữ dữ liệu các bản ghi quan trọng của hệ thống SCADA.

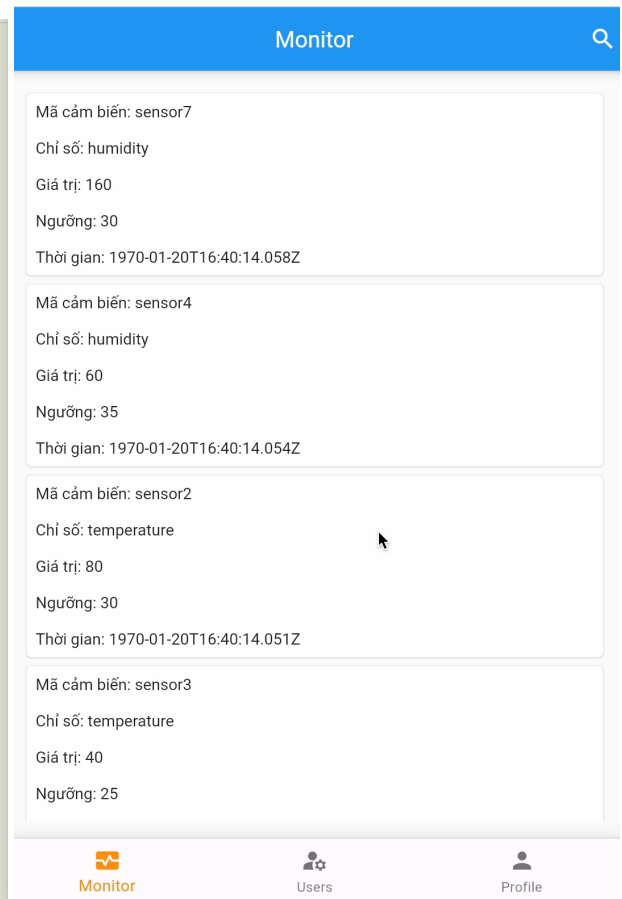
### 3.3. Thiết kế ứng dụng minh họa

Trong phần thiết kế ứng dụng minh họa, luận văn sẽ trình bày tổng quan về các trường hợp sử dụng chính của hệ thống, cùng với các trường hợp sử dụng chi tiết nhằm đáp ứng các yêu cầu đã được định nghĩa trong mục 3.1.. Các trường hợp sử dụng chính bao gồm:

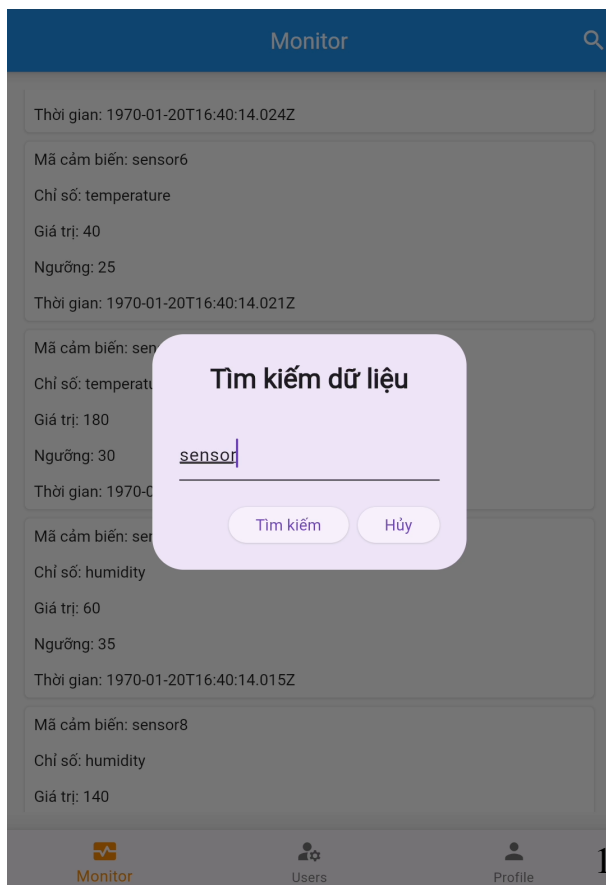
- *Ghi nhận dữ liệu từ cảm biến.*
- *Đăng nhập và xác thực người dùng.* Hình 3.2 trình bày giao diện màn hình đăng nhập.
- *Xem và tìm kiếm lịch sử của sự kiện.* Hình 3.3 trình bày giao diện màn hình danh sách sự kiện vượt ngưỡng. Hình 3.4 trình bày giao diện màn hình tìm kiếm sự kiện vượt ngưỡng theo mã cảm biến.
- *Quản lý người dùng.* Hình 3.5 trình bày giao diện màn hình danh sách người dùng. Hình 3.6 trình bày giao diện màn hình thêm mới người dùng.
- *Theo dõi lịch sử đăng nhập.* Hình 3.7 trình bày giao diện lịch sử đăng nhập của tài khoản.
- *Thay đổi mật khẩu.* Hình 3.8 trình bày giao diện đổi mật khẩu của tài khoản.



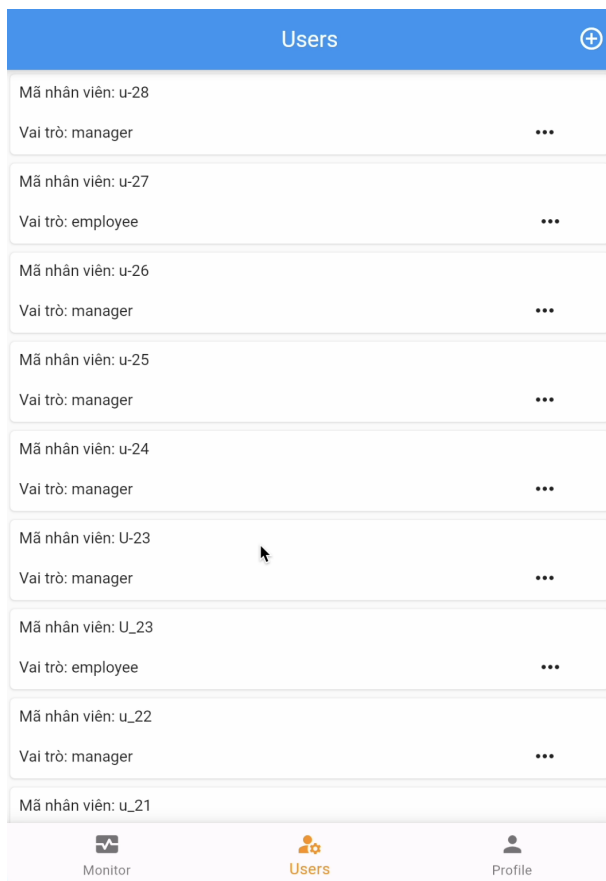
Hình 3.2: Màn hình đăng nhập.



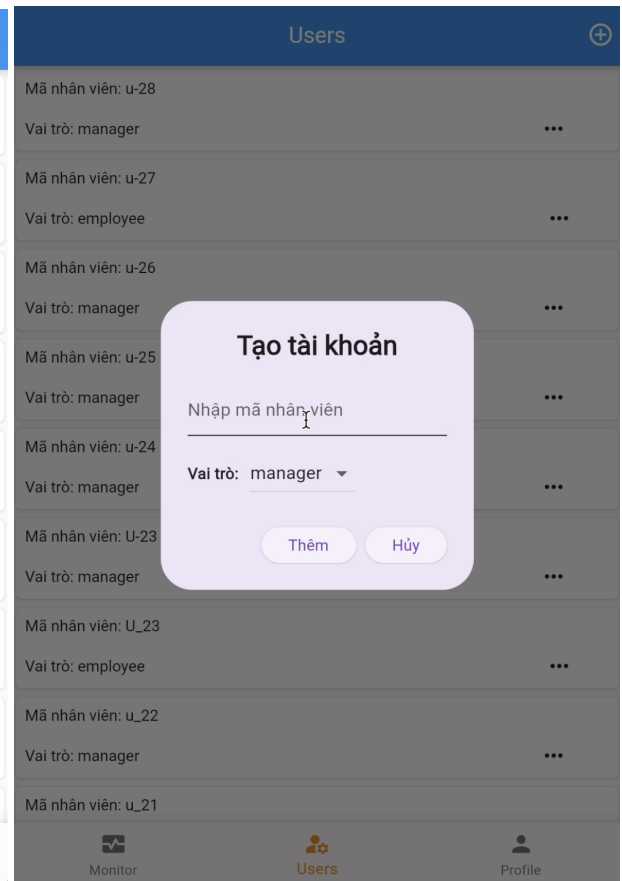
Hình 3.3: Màn hình hiển thị thông số của cảm biến khi vượt ngưỡng.



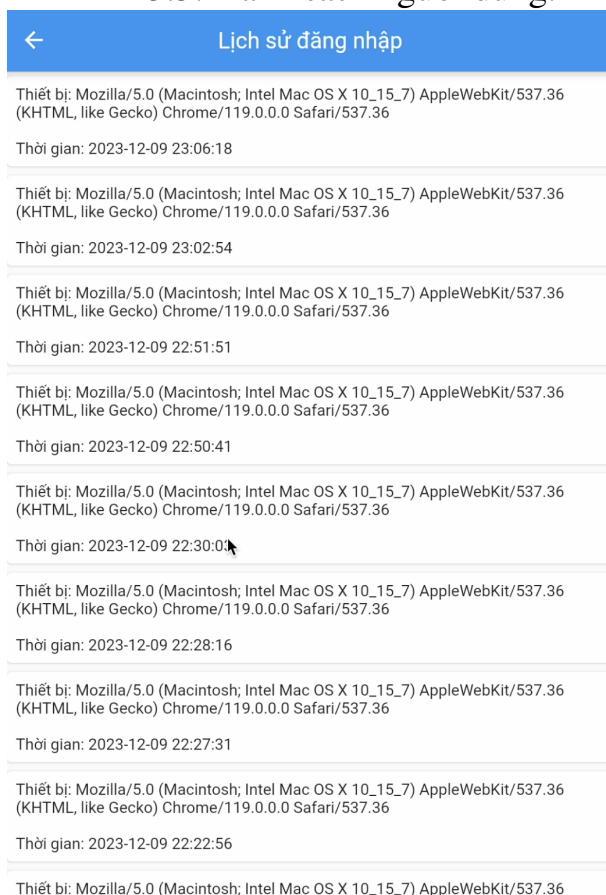
Hình 3.4: Màn hình tìm kiếm sự kiện.



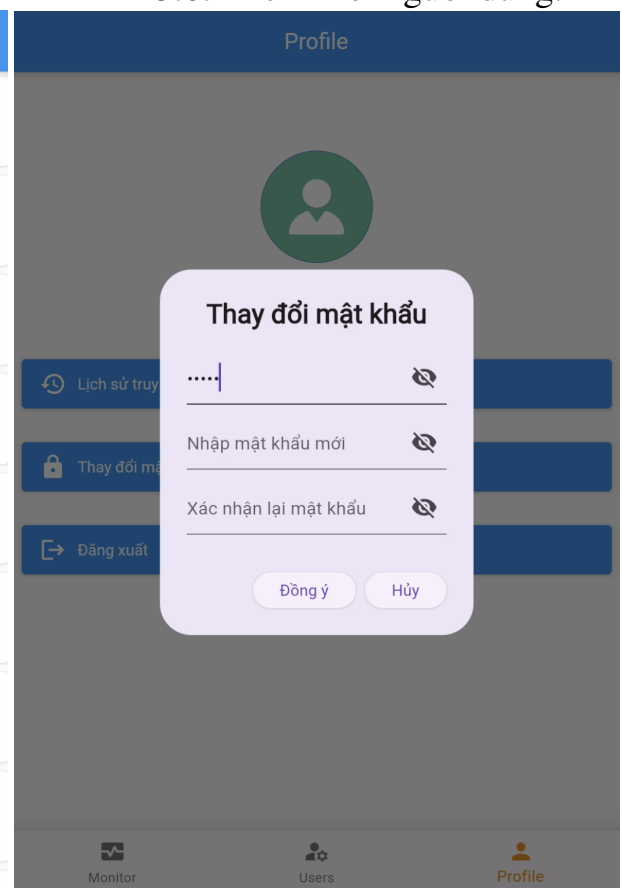
Hình 3.5: Danh sách người dùng.



Hình 3.6: Thêm mới người dùng.



Hình 3.7: Lịch sử đăng nhập



Hình 3.8: Thay đổi mật khẩu.

## Chương 4

# Lập trình và thực nghiệm

Để thực hiện các yêu cầu đã được nêu trong chương 3, tác giả tiến hành quá trình lập trình và phát triển ứng dụng. Ngôn ngữ lập trình Go được chọn lựa và áp dụng xuyên suốt toàn bộ quá trình triển khai nhằm đảm bảo tính hiệu quả và sự nhất quán của hệ thống.

### 4.1. Tổng quan

Phần tổng quan cung cấp một cái nhìn toàn diện về các bước thực nghiệm, cấu trúc thư mục mã và cấu trúc dữ liệu, nhấn mạnh tầm quan trọng của việc tổ chức và quản lý hiệu quả trong quá trình phát triển hệ thống. Những yếu tố này không chỉ đảm bảo tính chính xác và hiệu quả của nghiên cứu mà còn tạo nền tảng vững chắc cho việc mở rộng và bảo trì hệ thống trong tương lai. Quá trình lập trình và thực nghiệm bao gồm bốn bước chính sau:

- *Thiết lập mạng thử nghiệm.*
- *Phát triển chuỗi mã.*
- *Phát triển phụ trợ.*
- *Thực thi.*

## 4.2. Triển khai mạng thử nghiệm

Trong phần này, tác giả sẽ tiến hành các bước cần thiết để thiết lập mạng thử nghiệm của Hyperledger Fabric. Các công cụ cần thiết để vận hành mạng thử nghiệm bao gồm docker<sup>1</sup>, git<sup>2</sup> và curl<sup>3</sup>. Các bước để chạy thực nghiệm:

- Tải bản sao của dự án hyperledger/fabric-samples:  
*git clone github.com/hyperledger/fabric-samples*
- Truy cập vào thư mục test-network:  
*cd fabric-samples/test-network*
- Tạo kênh giao tiếp và khởi tạo thử nghiệm:  
*./network.sh up createChannel*

## 4.3. Xây dựng chuỗi mã

Ngôn ngữ lập trình Go được lựa chọn để phát triển chaincode trong thử nghiệm này. Mặc dù các lựa chọn khác như Java và NodeJS cũng khả thi, Go được ưu tiên do tính hiệu quả và khả năng tương thích tốt với môi trường Hyperledger Fabric. Các bước thực hiện bao gồm:

- *Xây dựng cấu trúc lưu trữ* đáp ứng các yêu cầu hệ thống, đồng thời đảm bảo tính toàn vẹn và bảo mật dữ liệu. Quá trình thiết kế bao gồm việc xác định các trường thông tin, quy định chặt chẽ về định dạng và kiểu dữ liệu tương ứng.
- *Xây dựng hàm khởi tạo* đóng vai trò thiết lập trạng thái ban đầu của hệ thống. Cụ thể, hàm này tạo một người dùng có vai trò là quản trị viên và lưu thông tin tương ứng vào chuỗi khối.

---

<sup>1</sup>Docker là một nền tảng mã nguồn mở giúp ta đóng gói và chạy các ứng dụng trong các môi trường cô lập gọi là containers.

<sup>2</sup>Git là một hệ thống quản lý phiên bản phân tán được sử dụng rộng rãi trong phát triển phần mềm.

<sup>3</sup>Curl là một công cụ dòng lệnh được sử dụng để gửi và nhận dữ liệu qua mạng.

- *Xây dựng hàm thêm mới người dùng.*
- *Xây dựng hàm thêm mới sự kiện.*
- *Xây dựng hàm lấy danh sách sự kiện.*
- *Xây dựng hàm xác thực người dùng.*

## 4.4. Phát triển phụ trợ

Để tối ưu hóa khả năng tương thích với nền tảng Hyperledger Fabric, việc phát triển các thành phần phụ trợ nên sử dụng một trong ba ngôn ngữ lập trình chính: Go, Java hoặc NodeJS. Trong nghiên cứu này, ngôn ngữ Go kết hợp với framework Fiber<sup>4</sup> phiên bản thứ hai, đã được lựa chọn để phát triển các tính năng cần thiết. Sự lựa chọn này dựa trên hiệu năng và khả năng tích hợp mạnh mẽ của Go với Hyperledger Fabric. Quá trình phát triển phụ trợ bao gồm các bước sau:

- *Xây dựng tệp thực thi chính:* Phần này tập trung vào việc thiết lập cấu hình ban đầu, khởi tạo các dịch vụ cần thiết, định nghĩa các điều hướng và thiết lập cơ chế lắng nghe cho các yêu cầu đến.
- *Xây dựng tệp cấu hình:* Mục tiêu chính là tạo một cổng giao tiếp giữa thành phần phụ trợ và Hyperledger Fabric, đảm bảo việc trao đổi dữ liệu diễn ra thông suốt và an toàn.
- *Xây dựng xử lý nghiệp vụ:* Các phương thức cốt lõi được triển khai bao gồm: đăng nhập, thay đổi mật khẩu, thêm người dùng, truy xuất danh sách người dùng, thu thập lịch sử đăng nhập, thêm sự kiện, tìm kiếm sự kiện và truy xuất danh sách sự kiện.
- *Xây dựng lớp xác thực trung gian:* Quá trình xác thực này giúp đảm bảo rằng chỉ các thông tin hợp lệ và chính xác mới được chấp nhận và tiếp tục xử lý.
- *Xây dựng tệp thực thi:* Mọi yêu cầu của máy khách gửi đến đều được xác thực trước khi đưa vào xử lý nghiệp vụ

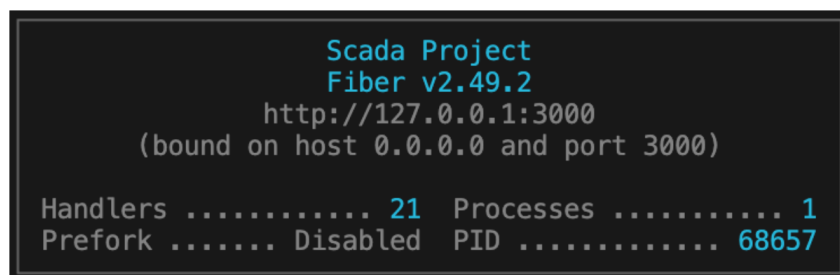
---

<sup>4</sup><https://docs.gofiber.io/>

- *Xây dựng định tuyến*: Cơ chế định tuyến giúp quản lý và điều phối các yêu cầu từ máy khách đến các hàm xử lý tương ứng, đồng thời tạo nên một cấu trúc rõ ràng và dễ quản lý cho ứng dụng.

## 4.5. Thực thi

Trong giai đoạn triển khai cuối cùng, chương trình được thực thi bằng lệnh **go run main.go**. Khi chương trình đang được thực thi, màn hình console sẽ hiển thị các thông tin như sau:



```
Scada Project
Fiber v2.49.2
http://127.0.0.1:3000
(bound on host 0.0.0.0 and port 3000)

Handlers ..... 21  Processes ..... 1
Prefork ..... Disabled  PID ..... 68657
```

Hình 4.1: Khởi chạy chương trình phụ trợ.

## Chương 5

# Đảm bảo chất lượng mã

Chương 4 trình bày quá trình phát triển ứng dụng tích hợp Hyperledger Fabric với hệ thống SCADA, đáp ứng các yêu cầu bài toán đã được xác định trong chương 3. Do mã nguồn được phát triển bằng ngôn ngữ Go, các công cụ phân tích và kiểm tra mã nguồn dành riêng cho Go sẽ được sử dụng để phát hiện và xử lý các lỗi tiềm ẩn. Quá trình này đảm bảo chất lượng và độ tin cậy của ứng dụng.

### 5.1. Kiểm chứng

Trong quá trình khảo sát và nghiên cứu, tám công cụ hỗ trợ đã được đánh giá và phân tích. Dựa trên các tiêu chí đánh giá về chức năng, số sao trên github và thời điểm cập nhật cuối cùng thì *testing*, *golangci-lint* và *gosec* được lựa chọn để áp dụng trong dự án. Công cụ *testing* được lựa chọn cho kiểm thử đơn vị vì được tích hợp sẵn với Go. Việc viết kiểm thử đơn vị cũng giúp ích quá trình bảo trì và mở rộng mã nguồn, đảm bảo rằng các thay đổi không làm ảnh hưởng đến các phần khác của hệ thống. Công cụ *golangci-lint*<sup>1</sup> được lựa chọn để thực hiện kiểm tra và tối ưu hóa chất lượng mã nguồn. Công cụ này có khả năng phát hiện và báo cáo đa dạng các vấn đề, bao gồm lỗi cú pháp, vi phạm quy tắc lập trình cũng như các vấn đề tiềm ẩn liên quan đến hiệu suất và bảo mật. Công cụ *gosec*<sup>2</sup> phổ biến trong phân tích mã nguồn nhằm phát hiện các lỗ hổng bảo mật tiềm ẩn dựa trên một tập

---

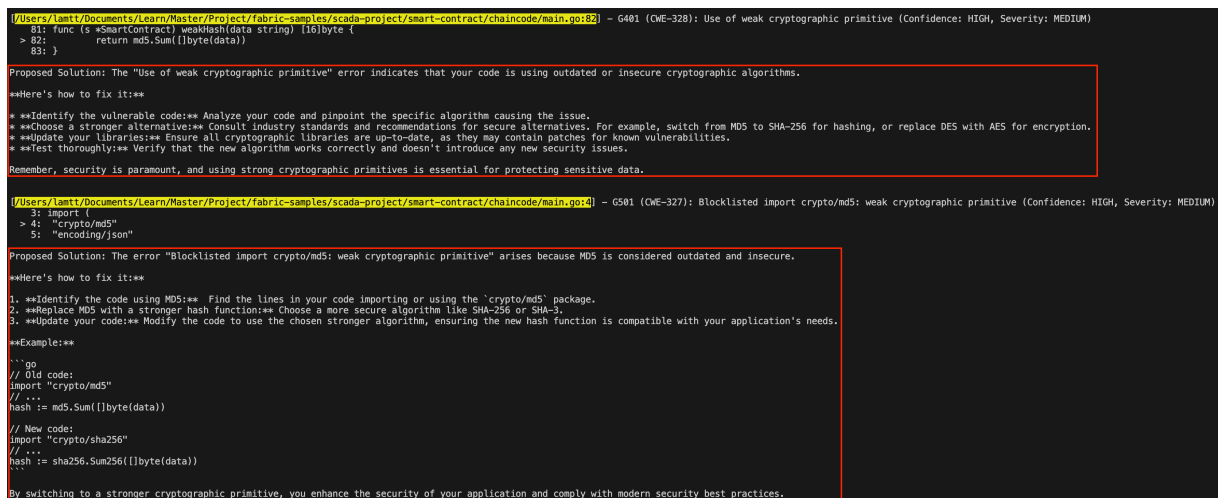
<sup>1</sup><https://github.com/golangci/golangci-lint>

<sup>2</sup><https://github.com/securego/gosec>



hợp các quy tắc bảo mật chuẩn<sup>3</sup>.

Các công cụ được giới thiệu trong luận văn chủ yếu tập trung vào việc phát hiện lỗi, nhưng chưa hỗ trợ tự động sửa lỗi hoặc đề xuất giải pháp. Trong quá trình nghiên cứu, một cải tiến đã được thực hiện nhằm bổ sung tính năng *đề xuất giải pháp* cho công cụ *gosec*. Cụ thể, sau khi *gosec* phát hiện lỗ hổng bảo mật, thông tin chi tiết về lỗ hổng được cung cấp cho Gemini để tạo gợi ý sửa lỗi được minh họa ở hình 5.1. Chi tiết về các thay đổi trong mã nguồn được trình bày tại <https://github.com/securego/gosec/pull/1177/files>



```
[/Users/lanth/Documents/learn/master/project/fabric-samples/scada-project/smart-contract/chaincode/main.go:82] - G401 (CWE-328): Use of weak cryptographic primitive (Confidence: HIGH, Severity: MEDIUM)
81: func (s *SmartContract) weakhash(data string) [16]byte {
82:     return md5.Sum([]byte(data))
83: }

Proposed Solution: The "Use of weak cryptographic primitive" error indicates that your code is using outdated or insecure cryptographic algorithms.
**Here's how to fix it:**
* **Identify the vulnerable code:** Analyze your code and pinpoint the specific algorithm causing the issue.
* **Choose a stronger alternative:** Consult industry standards and recommendations for secure alternatives. For example, switch from MD5 to SHA-256 for hashing, or replace DES with AES for encryption.
* **Update your libraries:** Ensure all cryptographic libraries are up-to-date, as they may contain patches for known vulnerabilities.
* **Test thoroughly:** Verify that the new algorithm works correctly and doesn't introduce any new security issues.
Remember, security is paramount, and using strong cryptographic primitives is essential for protecting sensitive data.

[/Users/lanth/Documents/learn/master/project/fabric-samples/scada-project/smart-contract/chaincode/main.go:4] - G501 (CWE-327): Blocklisted import crypto/md5: weak cryptographic primitive (Confidence: HIGH, Severity: MEDIUM)
5: import (
> 4: "crypto/md5"
5: "encoding/json"

Proposed Solution: The error "Blocklisted import crypto/md5: weak cryptographic primitive" arises because MD5 is considered outdated and insecure.
**Here's how to fix it:**
1. **Identify the code using MD5:** Find the lines in your code importing or using the 'crypto/md5' package.
2. **Replace MD5 with a stronger hash function:** Choose a more secure algorithm like SHA-256 or SHA-3.
3. **Update your code:** Modify the code to use the chosen stronger algorithm, ensuring the new hash function is compatible with your application's needs.
**Example:**
// Old code:
import "crypto/md5"
// ...
hash := md5.Sum([]byte(data))
// New code:
import "crypto/sha256"
// ...
hash := sha256.Sum256([]byte(data))
By switching to a stronger cryptographic primitive, you enhance the security of your application and comply with modern security best practices.
```

Hình 5.1: Kết quả chạy *gosec* cùng với tính năng *đề xuất giải pháp*.

## 5.2. Kiểm thử tính bảo mật trên mạng thử nghiệm

Đánh giá tính bảo mật của mạng Hyperledger Fabric được thực hiện thông qua việc triển khai một mạng thử nghiệm gồm hai máy chủ. Mọi giao dịch trên mạng đều phải trải qua quy trình xác thực bởi cả hai máy chủ, đảm bảo tính nhất quán và chống lại các hành vi giả mạo. Các bước kiểm thử bao gồm:

- Chuẩn bị chuỗi mã
- Triển khai chuỗi mã
- Kiểm thử giao dịch

<sup>3</sup><https://github.com/securego/gosec?tab=readme-ov-file#available-rules>

## Chương 6

# Kết luận

Hiện nay, việc lưu trữ dữ liệu trên hệ thống SCADA thường được thực hiện thông qua các hệ quản trị cơ sở dữ liệu truyền thống, tuy nhiên phương pháp này có thể dễ dàng bị tấn công bởi các kẻ xâm nhập hoặc có thể bị thay đổi để tạo ra thông tin sai lệch, từ đó dẫn đến những quyết định không chính xác từ phía người vận hành và gây ra những hậu quả nghiêm trọng. Vì vậy, luận văn đề xuất sử dụng công nghệ chuỗi khối để lưu trữ dữ liệu thay vì sử dụng cơ sở dữ liệu truyền thống. Công nghệ chuỗi khối, với những ưu điểm như tính không thể thay đổi, tính phi tập trung và tính minh bạch, sẽ mang lại một môi trường lưu trữ dữ liệu an toàn hơn và đáng tin cậy hơn.

Môi trường của SCADA thường tồn tại trong một hệ thống doanh nghiệp, vì vậy việc chọn một loại chuỗi khối phù hợp với môi trường doanh nghiệp là rất quan trọng. Luận văn đề xuất sử dụng Hyperledger Fabric như một nền tảng thử nghiệm để lưu trữ dữ liệu trên hệ thống SCADA. Hyperledger Fabric được chọn vì có những ưu điểm nổi trội được mô tả ở mục 2.2.2.. Tuy nhiên, cần lưu ý rằng Hyperledger Fabric cũng đối mặt với một số khó khăn và hạn chế so với các cơ sở dữ liệu truyền thống. Trong việc mở rộng và xử lý tải lớn, Hyperledger Fabric có thể gặp khó khăn đặc biệt khi số lượng thành viên trong mạng lưới tăng lên. Khả năng mở rộng của nền tảng này có thể bị hạn chế, và việc xử lý một lượng lớn giao dịch có thể trở nên khó khăn.

Một trong những khó khăn của Hyperledger Fabric là khả năng tìm kiếm và truy vấn dữ liệu. Do tập trung vào tính bảo mật và quyền riêng tư, Hyperledger

Fabric thiếu một công cụ truy vấn mạnh mẽ như ngôn ngữ truy vấn SQL trong các cơ sở dữ liệu truyền thống. Điều này gây khó khăn trong việc thực hiện các truy vấn phức tạp và tìm kiếm dữ liệu linh hoạt. Trong tương lai, luận văn hướng đến giải quyết khó khăn này là sử dụng một cơ sở dữ liệu phụ để lưu trữ thông tin hỗ trợ cho việc tìm kiếm và truy vấn. Trong hệ thống này, dữ liệu liên quan đến tìm kiếm và truy vấn sẽ được lưu trữ trong cơ sở dữ liệu phụ, trong khi Hyperledger Fabric vẫn giữ vai trò lưu trữ dữ liệu chính dưới dạng khóa - giá trị. Khi có yêu cầu tìm kiếm hoặc truy vấn, hệ thống sẽ truy xuất thông tin từ cơ sở dữ liệu phụ để tìm kiếm và truy vấn dữ liệu liên quan. Kết quả thu được sẽ được sử dụng để thực hiện tìm kiếm thông tin cụ thể trên Hyperledger Fabric. Bằng cách kết hợp sử dụng cả Hyperledger Fabric và cơ sở dữ liệu phụ có thể vượt qua hạn chế về khả năng tìm kiếm và truy vấn dữ liệu của Hyperledger Fabric. Hệ thống sẽ trở nên linh hoạt hơn và có khả năng thực hiện các truy vấn phức tạp và tìm kiếm dữ liệu theo yêu cầu.

Để đảm bảo chất lượng mã, luận văn áp dụng một quy trình bao gồm kiểm chứng và kiểm thử. Kiểm chứng giúp phát hiện và sửa chữa những lỗi cú pháp, lỗi hỏng bảo mật trong mã nguồn. Việc này, đảm bảo rằng mã tuân thủ các tiêu chuẩn lập trình và ngăn ngừa các lỗi cơ bản. Kiểm thử được thực hiện để đảm bảo tính đúng đắn của hệ thống. Thông qua quy trình kiểm chứng và kiểm thử, chất lượng mã được đảm bảo và phần mềm có thể hoạt động một cách chính xác, đáp ứng đầy đủ các yêu cầu của hệ thống và đáng tin cậy. Luận văn tiến hành đánh giá hiệu quả của các công cụ kiểm tra như *gosec* và *golangci-lint* trong việc phát hiện các lỗi và lỗi hỏng bảo mật. Các công cụ đã được sử dụng để xác định các vấn đề trong mã nguồn, tuy nhiên chúng chỉ giúp phát hiện các lỗi mà chưa thực sự cung cấp các giải pháp sửa chữa. Để giải quyết hạn chế trên, luận văn đã cải tiến công cụ *gosec* bằng cách bổ sung thêm tính năng *đề xuất giải pháp sửa lỗi*. Qua đó, giúp quá trình sửa lỗi trở nên hiệu quả hơn, thay vì chỉ dừng lại ở việc phát hiện các lỗi. Tính năng *đề xuất giải pháp sửa lỗi* đã được tích hợp và xuất bản chính thức trong mã nguồn của công cụ *gosec*.

Mã nguồn tại chương 4 được sử dụng trong đề tài “Nghiên cứu xây dựng phần mềm nền tảng để phát triển các ứng dụng cho hệ thống SCADA có sử dụng công nghệ chủ chốt của cách mạng công nghiệp lần thứ 4 (CMCN 4)” thuộc “Công

việc 3.14.1 Nghiên cứu tổng quan về các phương pháp xây dựng ứng dụng cho di động”.

# Tài liệu tham khảo

- [1] Next what is scada? supervisory control and data acquisition – defining the system, 12 2021. URL <https://premioinc.com/blogs/blog/what-is-scada-supervisory-control-and-data-acquisition-defining-the-system>.
- [2] Is scada a data historian?, 12 2022. URL <https://www.empoweredautomation.com/is-scada-a-data-historian>.
- [3] Cole Wangsness. What is a scada system and how does it work?, 09 2023. URL <https://www.onlogic.com/company/io-hub/what-is-a-scada-system-and-how-does-it-work/#>.
- [4] Lorenzo Stella. Looking where not allowed — navigating databases with sql injection, 08 2023. URL <https://medium.com/@starlaurentius/not-allowed-to-know-navigating-databases-with-sql-injection-2777a40e2e4>.
- [5] Steve. The dangers of storing files on the cloud, 04 2023. URL <https://ghostvolt.com/blog/the-dangers-of-storing-files-on-the-cloud.html>.
- [6] Scada system - scada info. URL <https://www.scadainfo.com/scada-system/>.
- [7] Kolin Paul Geeta Yadav. Architecture and security of scada systems: A review. 2020.
- [8] A. Aborujilah M. Irfan Shahzad, S. Musa. The scada review: System

components, architecture, protocols and future security trends. *Science Publications*, 2014.

- [9] Data historian. URL <https://www.empoweredautomation.com/data-historian>.
- [10] Blockchain – wikipedia. URL <https://vi.wikipedia.org/wiki/Blockchain>.
- [11] Rubina Lakhani Karim Sultan, Umar Ruhi. Conceptualizing blockchains: Characteristics applications. 2018.
- [12] Hamid R. Barzegar Nabil El Ioini Claus Pahl Jan Werth, Mohammad Hajian Berenjestanaki. A review of blockchain platforms based on the scalability, security and decentralization trilemma. *SN Computer Science*, 03 2023. doi: 10.5220/0011837200003467.
- [13] Bộ ba bất khả thi của blockchain (blockchain trilemma) là gì?, 05 2023. URL <https://www.okx.com/vi/learn/blockchain-trilemma-guide>.
- [14] Blockchain structure, 11 2022. URL <https://www.geeksforgeeks.org/blockchain-structure/>.
- [15] Introduction. URL <https://hyperledger-fabric.readthedocs.io/en/latest/blockchain.html>.
- [16] Ledger, . URL <https://hyperledger-fabric.readthedocs.io/en/release-1.3/ledger/ledger.html>.
- [17] Hyperledger fabric model, . URL [https://hyperledger-fabric.readthedocs.io/en/release-2.5/fabric\\_model.html#assets](https://hyperledger-fabric.readthedocs.io/en/release-2.5/fabric_model.html#assets).
- [18] Fabric sample network, . URL <https://hyperledger-fabric.readthedocs.io/en/release-2.5/network/network.html>.
- [19] Fabric transaction flow, . URL <https://hyperledger-fabric.readthedocs.io/en/release-2.5/txflow.html>.
- [20] Hyperledger fabric use cases and case studies, . URL <https://101blockchains.com/hyperledger-fabric-use-cases/>.