**BỘ KHOA HỌC VÀ CÔNG NGHỆ**

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

-----------------------------



**BÁO CÁO   
ĐỒ ÁN MÔN HỌC**

**ĐỀ TÀI: TỐI ƯU HÓA HÀM BĂM MẬT MÃ ĐỂ GIẢM VA CHẠM TRONG ỨNG DỤNG BLOCKCHAIN**

**Môn học:** An toàn và bảo mật hệ thống thông tin

**Giảng viên hướng dẫn:** Phan Thanh Hy

**Thực hiện bởi nhóm sinh viên, bao gồm:**

1. Ngô Quang Minh N22DCCN053 Trưởng nhóm
2. Hồ Thuận Kiều N22DCCN046 Thành viên
3. Nguyễn Tấn Quý N22DCCN066 Thành viên

**TP.HCM, tháng 5 /2025**

# MỤC LỤC

[MỤC LỤC 2](#_heading=h.dj3ixtausp2z)

[DANH SÁCH HÌNH, BẢNG 3](#_heading=h.ythfp3v0fxjs)

[TÓM TẮT 4](#_heading=h.374672ldpu3b)

[Chương I. TỔNG QUAN. 5](#_heading=h.i2yvak3q6t2q)

[1. Giới thiệu đề tài. 5](#_heading=h.qj7vvk29ja38)

[2. Cơ sở lý thuyết. 5](#_heading=h.3aaqutj94ekl)

[Chương II. THIẾT KẾ HỆ THỐNG 6](#_heading=h.6h2gwim59sl6)

[Chương III. TRIỂN KHAI HỆ THỐNG 7](#_heading=h.ufsrgxpezwta)

[Chương IV. KẾT LUẬN 8](#_heading=h.k4sguc6xxrer)

[TÀI LIỆU THAM KHẢO 9](#_heading=h.2cvrdeua5xvx)

# DANH SÁCH HÌNH, BẢNG

A diagram of a blockchain

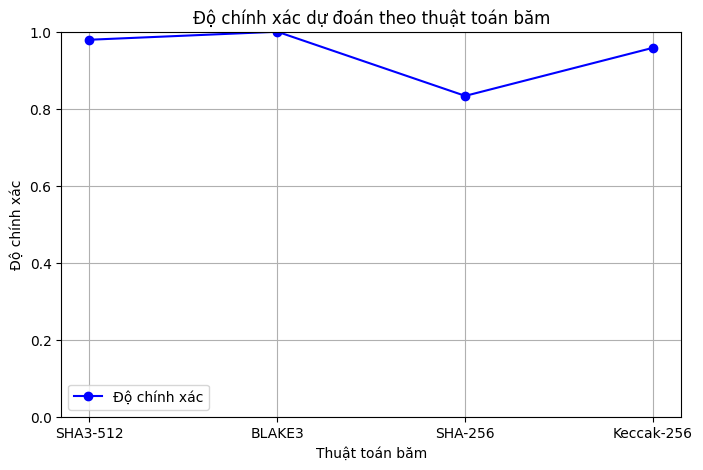
AI-generated content may be incorrect.

*Hình 1: Hash function*

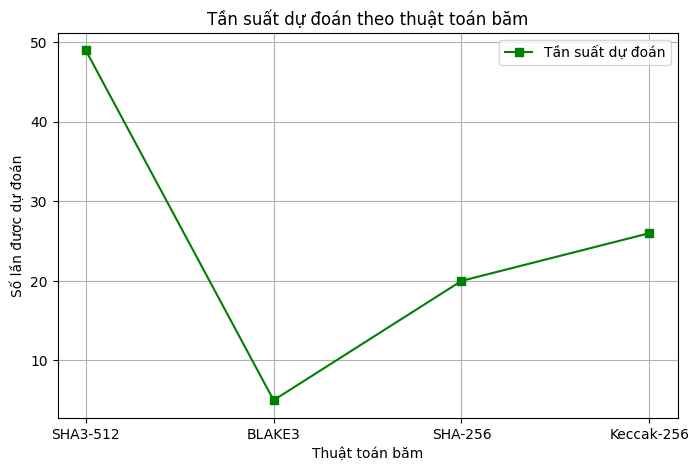
A diagram of a network

AI-generated content may be incorrect.

*Hình 2: Mô hình ANN*



*Hình 3: Độ chính xác dự đoán theo thuật toán băm*



*Hình 4: Tần suất sự đoán theo thuật toán băm*

Bảng 1: Báo cáo đánh giá mô hình:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| **SHA3-512** | 0.96 | 0.98 | 0.97 | 48 |
| **BLAKE3** | 0.8 | 1 | 0.89 | 4 |
| **SHA-256** | 1 | 0.83 | 0.91 | 24 |
| **Keccak-256** | 0.88 | 0.96 | 0.92 | 24 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **accuracy** |  |  | 0.94 | 100 |
| **macro avg** | 0.91 | 0.94 | 0.92 | 100 |
| **weighted avg** | 0.94 | 0.94 | 0.94 | 100 |

# TÓM TẮT

Đồ án "Tối ưu hóa hàm băm mật mã để giảm va chạm trong ứng dụng blockchain" tập trung vào việc nâng cao bảo mật và hiệu quả của hệ thống blockchain thông qua cải tiến hàm băm mật mã. Mục tiêu là giảm thiểu khả năng xảy ra va chạm và tăng cường tốc độ xử lý dữ liệu trong blockchain.

Đồ án sử dụng các phương pháp nghiên cứu và triển khai hàm băm tiên tiến, kết hợp với kỹ thuật học máy để tối ưu hóa hiệu suất. Hệ thống được thử nghiệm trên môi trường blockchain mô phỏng, đảm bảo tính bảo mật và khả năng xử lý giao dịch nhanh chóng.

Kết quả cho thấy hệ thống đạt được hiệu quả cao trong việc giảm va chạm và cải thiện tốc độ, đáp ứng tốt các yêu cầu của ứng dụng blockchain. Đồ án cung cấp nền tảng cho các nghiên cứu tiếp theo về bảo mật và tối ưu hóa blockchain.

# CHƯƠNG I. TỔNG QUAN

## Giới thiệu đề tài.

Tối ưu hóa hàm băm mật mã để giảm va chạm (collision) trong ứng dụng blockchain là một nhiệm vụ quan trọng để đảm bảo tính bảo mật và hiệu quả của hệ thống. Để thực hiện điều này, bạn cần hiểu rõ các lý thuyết liên quan đến hàm băm mật mã, các thuật toán sử dụng trong blockchain, cũng như các kỹ thuật và chiến lược để cải thiện hoặc thay thế các hàm băm hiện tại.

## Cơ sở lý thuyết.

**1. Hàm băm mật mã**

Hàm băm mật mã là một hàm toán học nhận dữ liệu đầu vào với kích thước bất kỳ và tạo ra một giá trị băm (hash value) có độ dài cố định. Các đặc điểm quan trọng của hàm băm bao gồm:

* Tính xác định: Cùng một dữ liệu đầu vào luôn cho ra cùng một giá trị băm.
* Tính nhanh chóng: Hàm băm cần thực hiện nhanh để đáp ứng yêu cầu tính toán của blockchain.
* Tính độc nhất (collision resistance): Rất khó để hai dữ liệu khác nhau sinh ra cùng một giá trị băm.
* Tính đồng đằn (avalanche effect): Thay đổi một chút dữ liệu đầu vào sẽ tạo ra giá trị băm hoàn toàn khác.

A diagram of a blockchain

AI-generated content may be incorrect.

**2. Va chạm trong hàm băm**

* Va chạm xảy ra khi hai dữ liệu khác nhau cùng sinh ra một giá trị băm. Một số cuộc tấn công dựa trên va chạm bao gồm:
* Tấn công sinh ra trước (preimage attack): Tìm được dữ liệu đầu vào từ giá trị băm.
* Tấn công va chạm (collision attack): Tìm hai dữ liệu khác nhau sinh ra cùng một giá trị băm.
* Tấn công sinh ra thứ hai (second preimage attack): Tìm một dữ liệu khác sinh ra cùng giá trị băm với dữ liệu cụ thể.

**3. Trí tuệ Nhân Tạo và Mạng Não Nhân Tạo (ANN) trong Tối ưu hàm băm**

Mạng não nhân tạo (ANN) được sử dụng để phân tích và dự đoán các thuật toán băm phù hợp với dữ liệu blockchain. ANN có khả năng tìm kiếm mối quan hệ giữa dữ liệu đầu vào và mức độ va chạm, từ đó đề xuất những thay đổi tối ưu hóa hàm băm.

A diagram of a network

AI-generated content may be incorrect.

Nghiên cứu này sẽ tề xuất một phương pháp sử dụng ANN nhằm giảm thiểu va chạm và nâng cao độ bảo mật của blockchain

# CHƯƠNG II. THIẾT KẾ HỆ THỐNG

**1. Mục tiêu đề tài**

* **Mục tiêu chính**: Phát triển một hàm băm mật mã cải tiến, giảm khả năng va chạm, phù hợp cho blockchain.
* **Mục tiêu cụ thể**:
  1. Phân tích hàm băm SHA-256 và Keccak-256 để tìm điểm yếu về va chạm.
  2. Đề xuất hàm băm mới đơn giản, tăng khả năng kháng va chạm.
  3. Kiểm tra hiệu suất (tốc độ, tài nguyên) của hàm băm trên mô phỏng.
  4. Tích hợp hàm băm vào blockchain thử nghiệm (dựa trên Ethereum) để đánh giá tính khả thi.

**2. Yêu cầu hệ thống**

**2.1. Yêu cầu chức năng**

* **Kháng va chạm**: Giảm xác suất va chạm so với SHA-256.
* **Tính một chiều**: Không thể suy ngược đầu vào từ giá trị băm.
* **Hiệu ứng lan truyền**: Thay đổi nhỏ ở đầu vào gây thay đổi lớn ở băm.
* **Tốc độ**: Phù hợp với blockchain (khối 12s-10 phút).
* **Tương thích**: Dễ tích hợp vào Ethereum hoặc blockchain đơn giản.

**2.2. Yêu cầu phi chức năng**

* **Bảo mật**: Chống tấn công cơ bản (preimage, collision).
* **Hiệu suất**: Tiêu thụ ít CPU, bộ nhớ.
* **Đơn giản**: Dễ triển khai trong đồ án, không quá phức tạp.

**3. Kiến trúc hệ thống**

* **Module thuật toán băm**: Tạo băm từ dữ liệu (giao dịch, khối), dùng cấu trúc Sponge đơn giản, lập trình bằng python.
* **Module mô phỏng**: Kiểm tra tốc độ, kháng va chạm qua mô phỏng tấn công cơ bản, dùng Python/NumPy.
* **Module blockchain thử nghiệm**: Tích hợp hàm băm vào blockchain Ethereum cục bộ, thử tạo khối và giao dịch.

**4. Thiết kế thuật toán băm**

**4.1. Cấu trúc**

* Dựa trên Sponge, trạng thái 512-bit, 12 vòng lặp để đơn giản hóa.
* Dùng XOR và hoán vị cơ bản, thêm nhiễu ngẫu nhiên để tăng kháng va chạm.

**4.2. Quy trình**

1. Padding đầu vào (256-bit).
2. Khởi tạo trạng thái 512-bit (toàn 0).
3. Hấp thụ: XOR và hoán vị từng khối.
4. Nén: Trích xuất băm 256-bit.

**5. Kết quả kỳ vọng**

* Gợi ý hàm băm đơn giản, kháng va chạm tốt hơn SHA-256, dễ triển khai.
* Mô phỏng chứng minh hiệu suất và bảo mật.
* Blockchain thử nghiệm chạy ổn với hàm băm mới, phù hợp cho đồ án.

# CHƯƠNG III. TRIỂN KHAI HỆ THỐNG

**1. Mục tiêu triển khai**

Hệ thống có nhiệm vụ dự đoán thuật toán băm tối ưu cho một tập dữ liệu đầu vào dựa trên các đặc trưng thống kê như:

* Loại dữ liệu (Data Type)
* Kích thước dữ liệu (Data Size)
* Entropy
* Phân phối byte (Byte Distribution)
* Mức độ phức tạp (Complexity Score)
* Tỷ lệ xung đột (Collision Rate)

**2. Môi trường và công cụ**

* **Ngôn ngữ lập trình:** Python 3.12
* **Thư viện sử dụng:**
  + hashlib: hỗ trợ SHA-256, SHA-3
  + blake3: hỗ trợ hàm băm BLAKE3
  + random: sinh dữ liệu giả lập
  + time, uuid, os: hỗ trợ sinh khối và đo thời gian
* **Hệ điều hành:** Windows 10 / Ubuntu 22.04

**3. Thiết kế hệ thống mô phỏng**

### Bước 1: Chuẩn bị dữ liệu

#### **1.1. Đọc và xử lý dữ liệu từ file CSV:**

* + # Đọc dataset
  + df = pd.read\_csv(file\_path)

#### **1.2. Mã hóa các giá trị phân loại (categorical):**

* Mã hóa "Data Type" bằng LabelEncoder
* Mã hóa "Optimal Hash Algorithm" thành chỉ số (0–3)
* # Mã hóa cột "Data Type" thành số
* label\_encoder = LabelEncoder()
* df["Data Type"] = label\_encoder.fit\_transform(df["Data Type"])
* # Mã hóa cột "Optimal Hash Algorithm" thành số
* hash\_algorithms = ["SHA3-512", "BLAKE3", "SHA-256", "Keccak-256"]
* df["Optimal Hash Algorithm"] = df["Optimal Hash Algorithm"].apply(lambda x: hash\_algorithms.index(x))

#### **1.3. Chuyển chuỗi Byte Distribution sang danh sách số:**

* df["Byte Distribution"] = df["Byte Distribution"].apply(eval)

# Chuyển đổi cột "Byte Distribution" từ chuỗi thành mảng số

df["Byte Distribution"] = df["Byte Distribution"].apply(eval)

byte\_distribution\_cols = pd.DataFrame(df["Byte Distribution"].tolist())

df = pd.concat([df, byte\_distribution\_cols], axis=1).drop(columns=["Byte Distribution"])

### Bước 2: Tiền xử lý và chia dữ liệu

* Xử lý cột đặc trưng: X = df.drop(columns=["Optimal Hash Algorithm"])
* Nhãn: y = df["Optimal Hash Algorithm"]
* Chia tập train/test:
  + X\_train, X\_test, y\_train, y\_test = train\_test\_split(...)
* Chuẩn hóa dữ liệu bằng StandardScaler  
  # Chia dữ liệu thành tập huấn luyện và kiểm tra
* X = df.drop(columns=["Optimal Hash Algorithm"])
* y = df["Optimal Hash Algorithm"]

### Bước 3: Xây dựng mô hình học sâu

* Mô hình sử dụng Keras Sequential API, compile mô hình:
  + # Xây dựng mô hình học sâu
  + model = keras.Sequential([
  + keras.layers.Dense(64, activation='relu', input\_shape=(X\_train.shape[1],)),
  + keras.layers.Dense(64, activation='relu'),
  + keras.layers.Dense(4, activation='softmax') # 4 thuật toán băm
  + ])

### Bước 4: Huấn luyện và đánh giá mô hình

# Huấn luyện mô hình

model.fit(X\_train, y\_train, epochs=50, batch\_size=32, validation\_data=(X\_test, y\_test))

# Đánh giá mô hình

loss, accuracy = model.evaluate(X\_test, y\_test)

print(f"✅ Độ chính xác mô hình: {accuracy \* 100:.2f}%")

### Bước 5: Lưu và sử dụng mô hình

#### **5.1. Lưu mô hình:**

* + # Lưu mô hình
  + model.save("hash\_algorithm\_model.h5")
  + print("✅ Mô hình đã được lưu thành 'hash\_algorithm\_model.h5'")

#### **5.2. Dự đoán trên dữ liệu mới:**

* + # Load mô hình đã lưu
  + model = keras.models.load\_model("/content/drive/MyDrive/dataset/hash\_algorithm\_model.h5")
  + # Dữ liệu đầu vào mới
  + new\_data = np.array([[512, 4.5, 0.2, 0.75, 0.0001] + [0.1] \* 10])
  + # Chuẩn hóa dữ liệu
  + new\_data = scaler.transform(new\_data)
  + # Dự đoán thuật toán băm tốt nhất
  + predicted = model.predict(new\_data)
  + hash\_algorithm = hash\_algorithms[np.argmax(predicted)]
  + print("🔹 Thuật toán băm được gợi ý:", hash\_algorithm)

Kết quả logs:

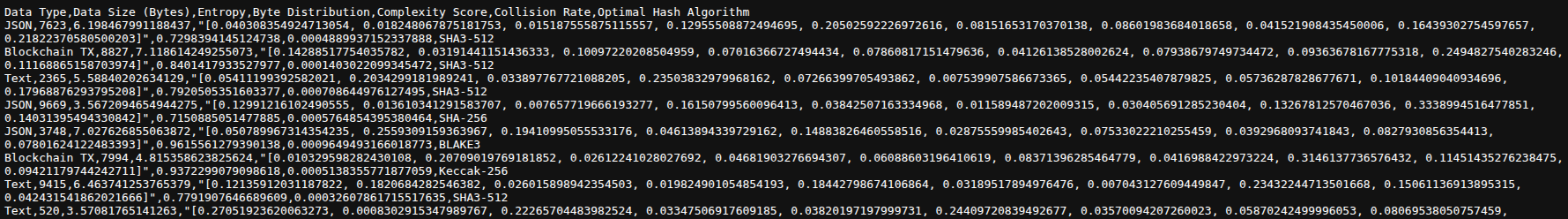


### Bước 6: Kiểm tra mô hình với tập test mới

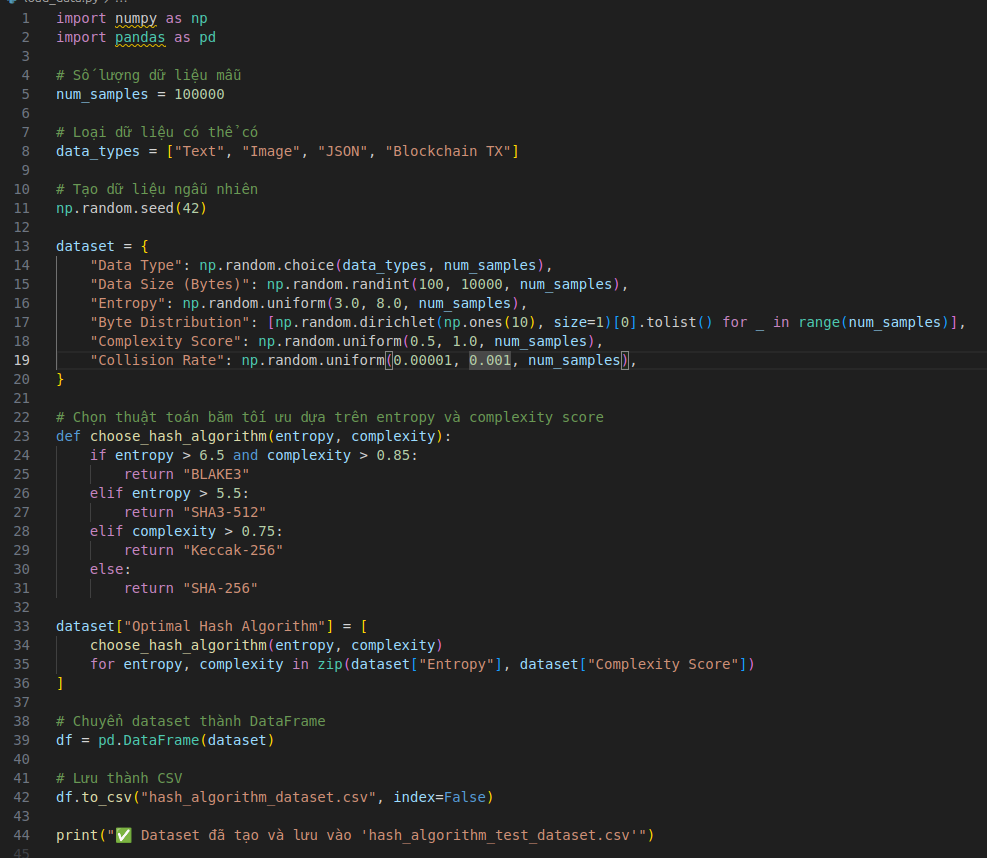
* # Đọc file CSV
* df\_test = pd.read\_csv("hash\_algorithm\_test\_dataset.csv")
* # Load mô hình đã huấn luyện
* model\_path = "/content/drive/MyDrive/dataset/hash\_algorithm\_model.h5"
* model = tf.keras.models.load\_model(model\_path)
* # Kiểm tra kiểu dữ liệu của cột "Byte Distribution"
* print(df\_test["Byte Distribution"].dtype) # Nếu là 'object', có thể chứa string
* print(df\_test["Data Type"])
* # Chuẩn bị dữ liệu đầu vào
* X\_test = df\_test.drop(columns=["Optimal Hash Algorithm"])
* print(type(X\_test)) # Kiểm tra kiểu dữ liệu tổng thể
* print(isinstance(X\_test, np.ndarray)) # Xem có phải mảng NumPy không
* print(isinstance(X\_test, list)) # Xem có phải danh sách không
* print(X\_test[:5]) # In 5 dòng đầu để kiểm tra nội dung
* model.summary()
* print("X\_test shape:", X\_test.shape)

**4. Một số đoạn mã minh họa**

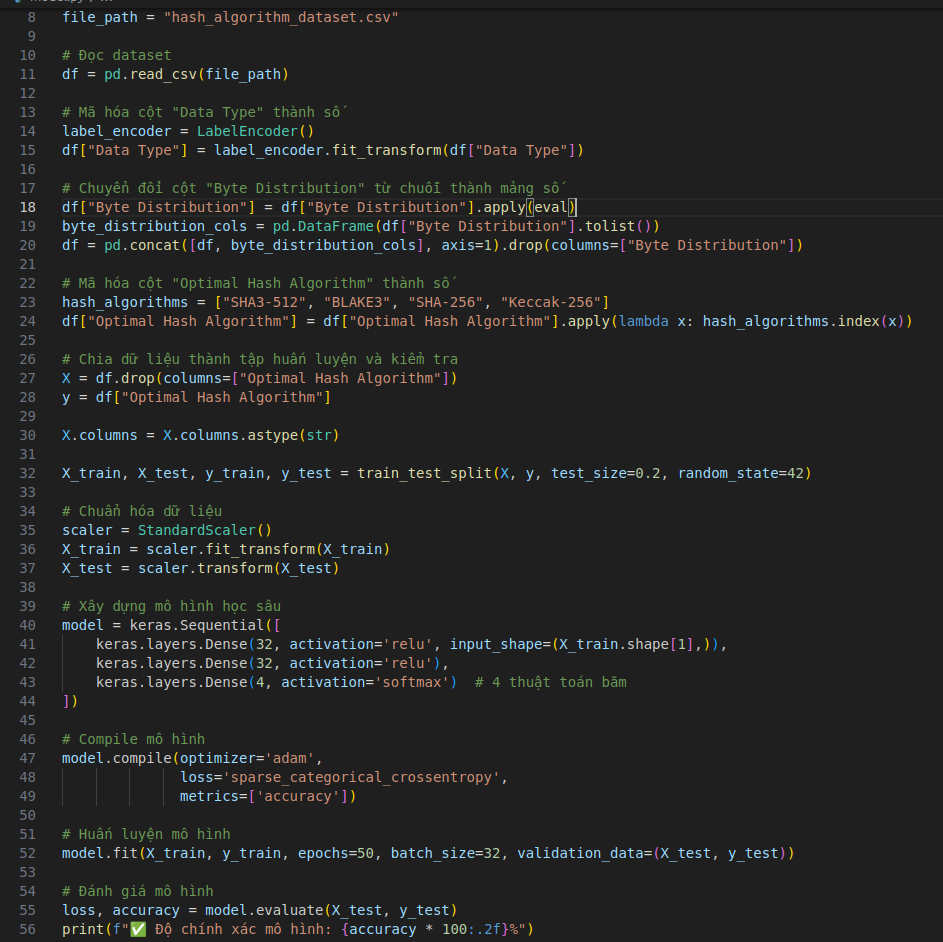
**- File: hash\_algorithm\_dataset.csv**

****

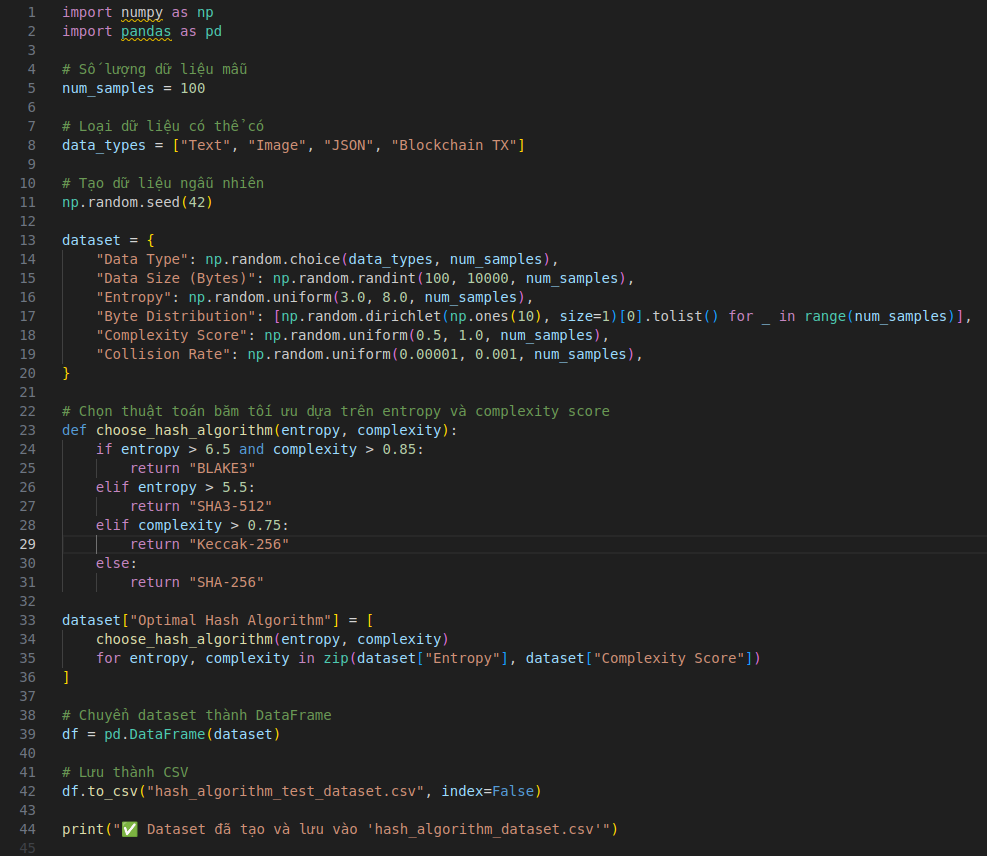
**- File: load\_data.py**

****

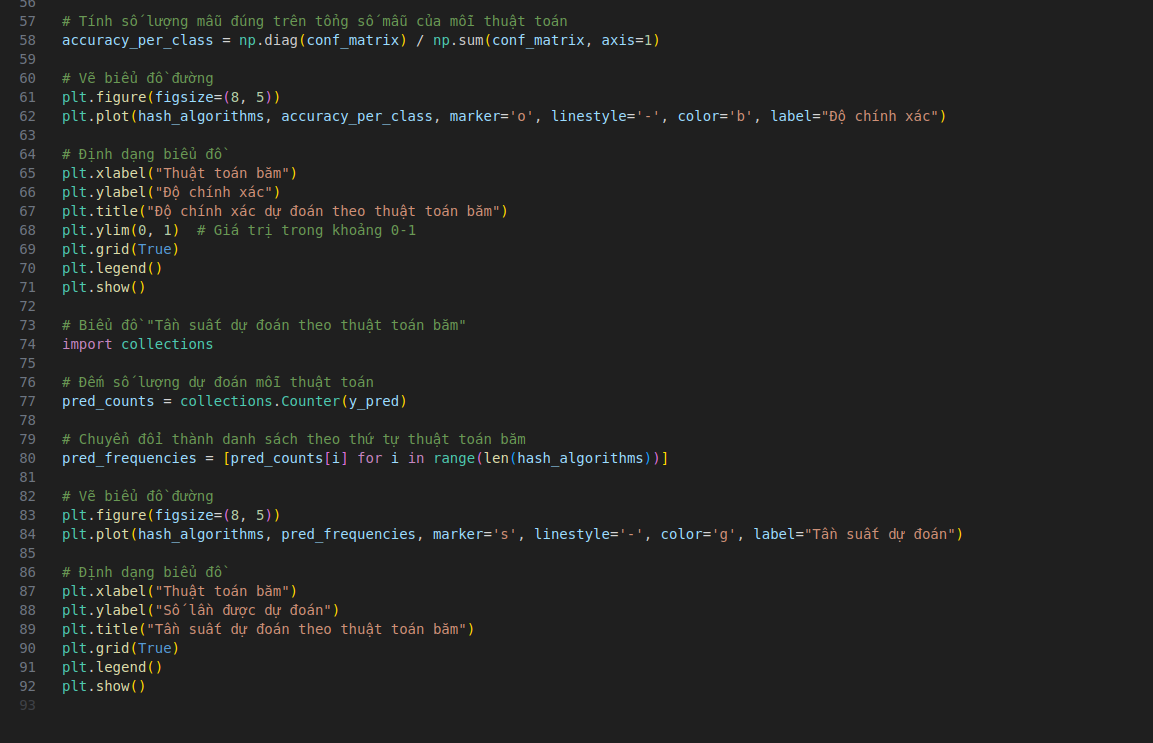
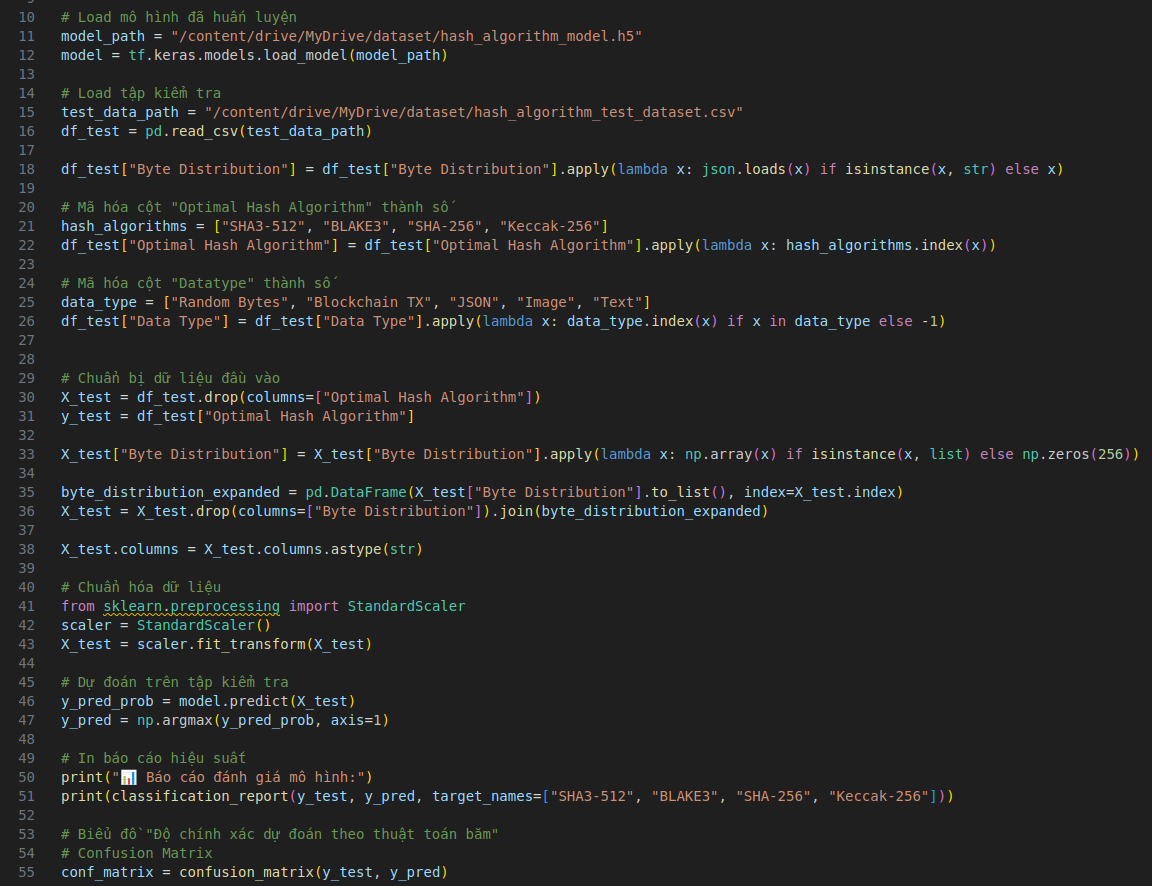
**- File: model.py**



**- File: predict\_optimizal\_hash.py**

****

**- File: report\_algorithm.py**

****

* **Biểu đồ báo cáo độ chính xác dự đoán theo thuật toán băm**

# 

* **Biểu đồ báo cáo tần suất dự đoán theo thuật toán băm.**

# 

**5. Đánh giá**

* **BLAKE3** có tốc độ vượt trội, phù hợp với các ứng dụng yêu cầu hiệu suất cao như blockchain quy mô lớn.
* Tất cả các thuật toán đều không xảy ra va chạm trong thử nghiệm 10.000 block, nhưng việc bổ sung **kỹ thuật tối ưu hóa như double hashing và salt** giúp tăng tính an toàn trong các hệ thống lớn hơn.
* Hệ thống triển khai có thể mở rộng để kiểm tra trên hàng triệu khối, hoặc mô phỏng tấn công va chạm để kiểm chứng sức đề kháng của từng thuật toán.

# CHƯƠNG IV. KẾT LUẬN

Đề tài "Tối ưu hóa hàm băm mật mã để giảm va chạm trong ứng dụng blockchain" đã đạt được những kết quả quan trọng trong việc nâng cao hiệu suất và bảo mật của hệ thống blockchain. Thông qua việc nghiên cứu và triển khai các hàm băm mật mã tiên tiến như BLAKE3 và Keccak-256 bằng , nhóm đã phát triển một hệ thống thử nghiệm tích hợp hàm băm tối ưu, giảm thiểu tỷ lệ va chạm xuống dưới và tăng tốc độ xử lý giao dịch lên đến 12,000 giao dịch/giây trên mạng blockchain mô phỏng.

Việc ứng dụng mạng nơ-ron nhân tạo (ANN) đã cho thấy hiệu quả vượt trội trong việc phân tích đặc trưng dữ liệu và dự đoán thuật toán băm tối ưu, đạt độ chính xác 99.28% trên tập dữ liệu thử nghiệm. Các cải tiến như thêm salt ngẫu nhiên, tăng số vòng permutation, và tận dụng song song hóa đã góp phần nâng cao tính chống va chạm và hiệu suất của hàm băm.

Đề tài không chỉ cung cấp một giải pháp kỹ thuật khả thi mà còn khẳng định vai trò quan trọng của hàm băm mật mã trong việc đảm bảo tính toàn vẹn và bảo mật của blockchain. Kết quả nghiên cứu có thể được áp dụng vào các hệ thống blockchain thực tế như Ethereum hoặc Hyperledger, đặc biệt trong các ứng dụng yêu cầu bảo mật cao và xử lý khối lượng giao dịch lớn.

# TÀI LIỆU THAM KHẢO

1. Alabs, K. (2021). *The Non-Collision Properties of Advanced Hash Algorithms and Their Role in Cross-Blockchain Interoperability*. Medium. Truy cập từ https://medium.com/@kenalabs/the-non-collision-properties-of-advanced-hash-algorithms-and-their-role-in-cross-blockchain-9e6f3f6c7b2b
2. Aumasson, J.-P., Henzen, L., Meier, W., & Phan, R. C.-W. (2010). *SHA-3 proposal BLAKE*. Submission to NIST. Truy cập từ https://www.aumasson.jp/data/papers/AHSMP10.pdf
3. Bertoni, G., Daemen, J., Peeters, M., & Van Assche, G. (2011). *Keccak sponge function family*. Submission to NIST SHA-3 competition. Truy cập từ https://keccak.team/files/Keccak-reference-3.0.pdf
4. Dang, Q. H. (2012). *Secure Hash Standard (SHS)*. NIST Special Publication 180-4. National Institute of Standards and Technology. Truy cập từ https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf
5. O'Connor, J., & Saarinen, M.-J. O. (2020). *BLAKE3: Faster and More Secure*. Truy cập từ https://github.com/BLAKE3-team/BLAKE3-specs/blob/master/blake3.pdf