Implémentation d'une Blockchain en Python

Cryptographie appliquée et Blockchain

Master 1

Systèmes Distribués et Intelligence Artificielle

Préparé par : **Yahya Ghallali**
Institution : **ENSET**
May 5, 2025

Contents

1 Introduction

Ce rapport détaille l'implémentation d'une blockchain en Python, en expliquant les différents composants et leurs fonctionnalités. La blockchain implémentée comprend un système complet de gestion de transactions, de minage de blocs, et de validation de la chaîne.

2 Architecture du Système

2.1 Structure du Projet

Le projet est organisé en deux dossiers principaux :

- **Models**/ : Contient les classes principales de la blockchain
  - `blockchain.py` : Gestion de la chaîne de blocs
  - `block.py` : Structure des blocs
  - `transaction.py` : Gestion des transactions
  - `transaction_pool.py` : Pool de transactions en attente
- **Utils**/ : Contient les utilitaires et fonctions d'aide
  - `crypto_utils.py` : Fonctions cryptographiques
  - `utils.py` : Utilitaires généraux
  - `crypto_constants.py` : Constantes cryptographiques

3 Composants Principaux

3.1 Blockchain

La classe principale qui gère la chaîne de blocs. Elle est responsable de :

- La création du bloc genesis
- L'ajout de nouveaux blocs
- La validation de l'intégrité de la chaîne
- L'ajustement de la difficulté de minage

3.1.1 Initialisation et Bloc Genesis

```
def __init__(self, difficulty: int = 2):
    self.chain: List[Block] = []
    self.difficulty = difficulty
    self._create_genesis_block()

def _create_genesis_block(self):
    genesis_transaction = Transaction(
        sender="network",
        recipient="genesis-address",
        amount=50,
        timestamp=get_timestamp(),
    )
    genesis_block = Block(
        index=0,
```

```
        previous_hash="0" * CryptoConstants.HASH_LEN,
        transactions=[genesis_transaction],
        timestamp=get_timestamp(),
    )
    self.chain.append(genesis_block)
```

Listing 1: Initialisation de la Blockchain

3.1.2 Validation de la Chaîne

```
def is_valid_chain(self) -> bool:
    for i in range(1, len(self.chain)):
        current_block = self.chain[i]
        previous_block = self.chain[i - 1]

        if current_block.previous_hash != previous_block.hash:
            return False

        if current_block.hash != current_block._calculate_hash():
            return False

        if not current_block.has_valid_transactions():
            return False

    return True
```

Listing 2: Validation de la Blockchain

3.2 Block

La structure de base d'un bloc dans la chaîne. Chaque bloc contient :

- Un index unique
- Un horodatage
- Le hash du bloc précédent
- Une liste de transactions
- Un nonce pour le minage
- Son propre hash

3.2.1 Calcul du Hash

```
def _calculate_hash(self) -> str:
    block_header = {
        "index": self.index,
        "timestamp": self.timestamp,
        "previous_hash": self.previous_hash,
        "transactions": [tx.transaction_hash for tx in self.transactions],
        "nonce": self.nonce,
    }
    return generate_hash(block_header)
```

Listing 3: Calcul du Hash d'un Bloc

3.2.2 Mining du Bloc

```python
def mine_block(self, difficulty: int) -> None:
    target = "0" * difficulty
    while self.hash[:difficulty] != target:
        self.nonce += 1
        self.hash = self._calculate_hash()
```

Listing 4: Mining d'un Bloc

3.3 Transaction

La représentation d'une transaction dans le système. Chaque transaction contient :

- L'expéditeur (clé publique)
- Le destinataire (clé publique)
- Le montant
- Un horodatage
- Une signature numérique
- Un hash de transaction

3.3.1 Validation des Transactions

```python
def is_valid(self) -> bool:
    if self.sender == "network":
        return True
    return verify_signature(self.transaction_hash, self.signature, self.
        sender)
```

Listing 5: Validation d'une Transaction

3.4 Transaction Pool

La gestion du pool de transactions en attente. Cette classe :

- Stocke les transactions en attente d'être incluses dans un bloc
- Vérifie la validité des transactions
- Gère l'ajout et la suppression des transactions

3.5 Blockchain Application

L'application principale qui utilise la blockchain. Elle fournit :

- Une interface pour créer des transactions
- La gestion du minage de blocs
- La validation de la chaîne
- La persistance des données

4 Cryptographie et Sécurité

4.1 Fonctions Cryptographiques

Le système utilise plusieurs mécanismes cryptographiques :

4.1.1 Génération de Hash

```python
def generate_hash(data: Any) -> str:
    if not isinstance(data, str):
        data = dump_data(data)
    encoded_data = data.encode()
    hash_result = hashlib.sha512(encoded_data).hexdigest()
    return hash_result
```

Listing 6: Génération de Hash

4.1.2 Signatures Numériques

Le système utilise ECDSA (Elliptic Curve Digital Signature Algorithm) pour les signatures :

```python
def generate_signature(data: Any, private_key_str: str) -> str:
    private_key_str = base64.b64decode(private_key_str)
    private_key = ecdsa.SigningKey.from_string(private_key_str, curve=
        ecdsa.SECP256k1)
    signature = private_key.sign(data.encode())
    return base64.b64encode(signature).decode()
```

Listing 7: Génération de Signature

4.2 Mécanismes de Sécurité

La blockchain implémentée inclut plusieurs mécanismes de sécurité :

- **Preuve de travail (Proof of Work)** : Utilise un système de difficulté ajustable pour le minage
- **Signatures numériques** : Utilise ECDSA avec la courbe SECP256k1
- **Validation de l'intégrité** : Vérifie les liens entre les blocs et la validité des transactions
- **Protection contre la double dépense** : Validation des transactions avant inclusion dans un bloc
- **Hachage sécurisé** : Utilise SHA-512 pour la génération des hashs

5 Performance et Optimisations

5.1 Optimisations de Performance

Les optimisations de performance incluent :

- **Gestion efficace de la mémoire** : Utilisation de structures de données optimisées
- **Validation optimisée** : Vérification rapide des transactions et des blocs
- **Pool de transactions** : Gestion efficace des transactions en attente
- **Logging intelligent** : Système de logging configurable pour le débogage

6 Conclusion

Ce projet implémente une blockchain complète avec toutes les fonctionnalités essentielles, incluant la création de blocs, la gestion des transactions, et la validation de la chaîne.