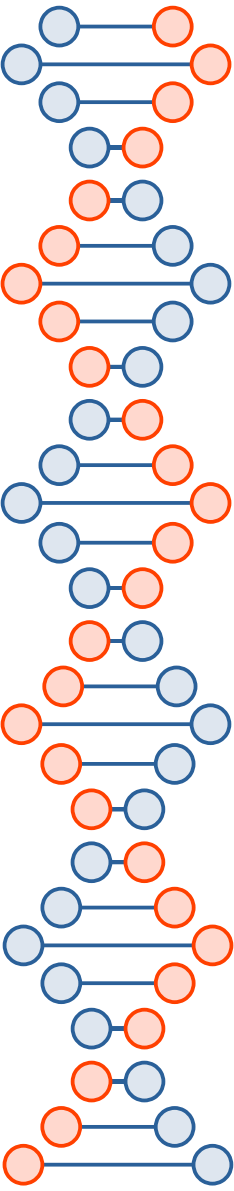




# **La Blockchain, effet de mode ou réel évolution**

DAUDIT Ruddy – M1 Informatique

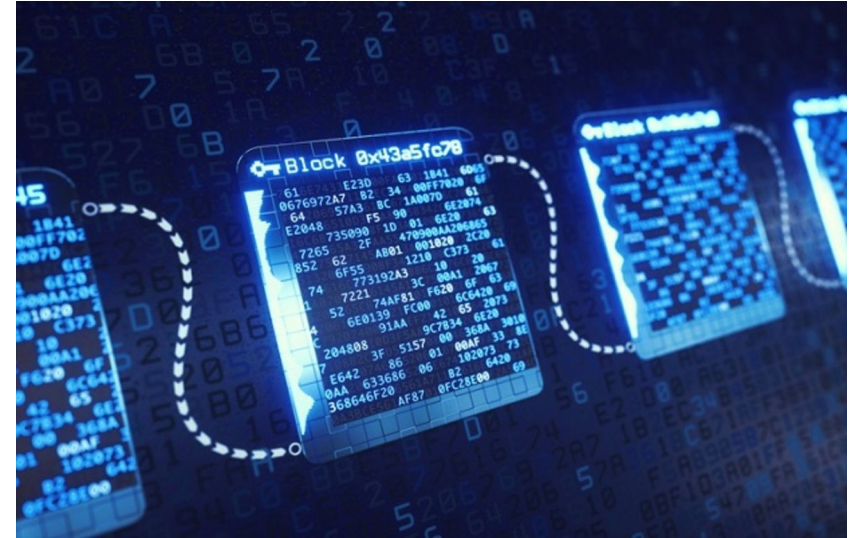
# Sommaire

- 
- Introduction
  - Problématique
  - Proposition
  - Résultats
  - Conclusion



# Introduction

- Fonction de hachage
- Fonction de hachage cryptographique



# Introduction - Définition

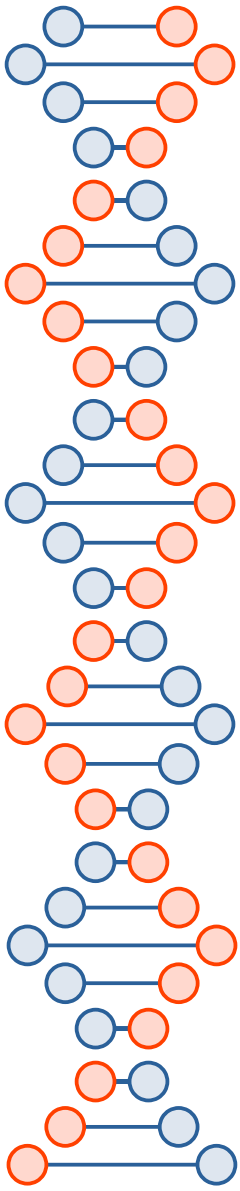
La blockchain est une base de données distribuée dont les informations envoyées par les utilisateurs et les liens internes à la base sont vérifiés et groupés à intervalles de temps réguliers, formant des blocs ainsi qu'une chaîne.





# Problématique

La blockchain, est elle le moyen le plus sûr de  
sécurisé des données ?



# Problématique

- État de l'art

Utilisations d'articles de  
chercheurs

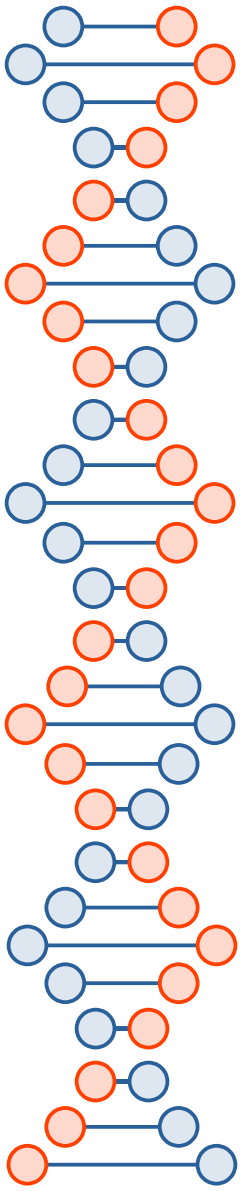
Vidéos

Recherches

- Programmation

Recherches de codes  
existant

Différents langages de  
programmation



# Proposition

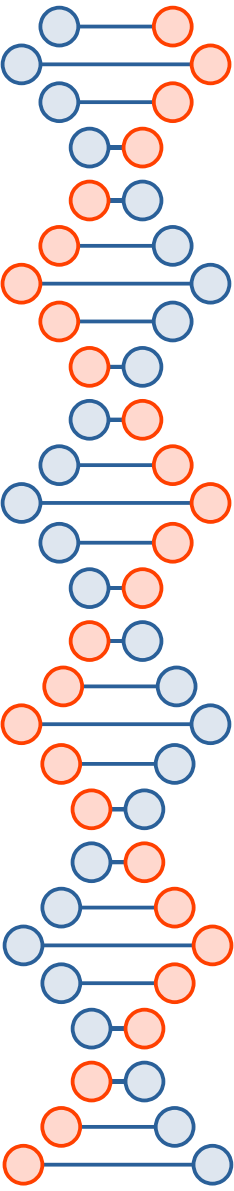
Altérer des blocs de la chaîne afin de prouver si  
le système est sécurisé

# Proposition - Expériences

- Nous utilisons principalement l'algorithme SutirthaDey car celui-ci permet de pouvoir étudier les blocs de plus près par leur signatures, de pouvoir vérifier tout changement, d'ajouter de nouveaux blocs à la chaîne.
- De là, nous pourrions étudier toutes les variations des signatures et savoir si on peut accéder aux informations d'un bloc tel un utilisateur malveillant.



# Propositions - Étapes



```
root@ruddy-VirtualBox:/home/ruddy/Documents/Ophelie# ./a.out
1)addNewNumber
2)alterNthBlock
3)printAllBlocks
4)verifyChain
Choice: █
```

- Terminal

# Proposition - Algorithmes

```
void addBlock(int data){
    if(head==NULL)
    {
        head=malloc(sizeof(struct block));
        sha256("", sizeof(""), head->prevHash);
        head->blockData=data;
        return;
    }
    struct block *currentBlock=head;
    while(currentBlock->link)
        currentBlock=currentBlock->link;
    struct block *newBlock=malloc(sizeof(struct block));
    currentBlock->link=newBlock;
    newBlock->blockData=data;
    sha256(tostring(*currentBlock), sizeof(*currentBlock), newBlock->prevHash);
}
```

```
void printBlock(struct block *b){
    printf("%p\\t", b);
    hashPrinter(b->prevHash, sizeof(b->prevHash));
    printf("\\t[%d]\\t", b->blockData);
    printf("%p\\n", b->link);
}

void printAllBlocks(){
    printf("block address\\t|\\t\\t\\tPreviousHash\\t\\t\\t|\\t\\tNumber\\t|\\tNextAddress\\n");
    struct block *curr=head;
    int count=0;
    while(curr)
    {
        printBlock(curr);
        curr=curr->link;
    }
}
```

# Proposition - Altération

```
void alterNthBlock(int n,int newData){
    struct block *curr=head;
    int count=1;
    if(curr==NULL)
    {
        printf("Nth block does not exists!\n");
        return;
    }
    while(count!=n){
        if(curr->link==NULL && count!=n){
            printf("Nth block does not exists!\n");
            return;
        }
        else if(count==n)
            break;
        curr = curr->link;
        count++;
    }
    printf("Before: ");
    printBlock(curr);
    curr->blockData=newData;
    printf("\nAfter: ");
    printBlock(curr);
    printf("\n");
}
```

## Résultats - Echec

```
Choice: 1
Enter Number: 7
Choice: 1
Enter Number: 3
Choice: 1
Enter Number: 4
Choice: 3
block address || PreviousHash
| Number ||
0x5636902a2ac0| 6e340b9cffb37a989ca544e6bb780a2c78901d3fb33738768511a30617afa01
d| [7]| 0x5636902a2b00
0x5636902a2b00| 250d3060b8659b2b4b251ddb0f65ddabb58770723f6219927e40abdf97b4e4
f| [3]| 0x5636902a2b00
0x5636902a2b00| ae254eaac19479be9204727e8f2630f72ef0bba602f91c89cbd1cf937cbae5a
d| [4]| (nil)
Choice: 1
```

```
Choice: 2
Which block to alter?: 5
Enter the value: 7
Nth block does not exists!
```

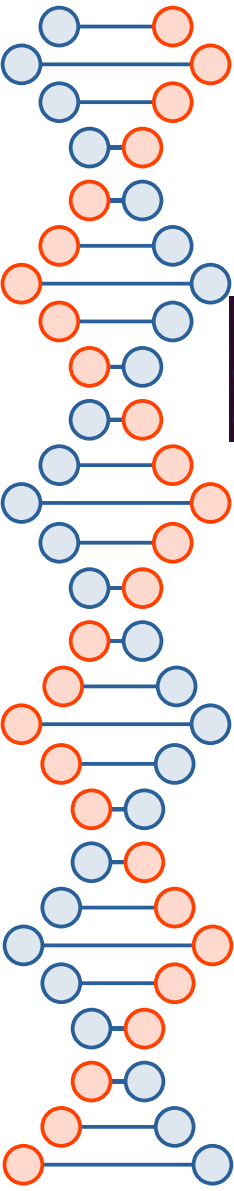


# Résultat - Réussite

```
Which block to alter?: 1
Enter the value: 0
Before: 0x5636902a2ac0] 6e340b9cffb37a989ca544e6bb780a2c78901d3fb33738768511a30
617afa01d          [7]      0x5636902a2b00

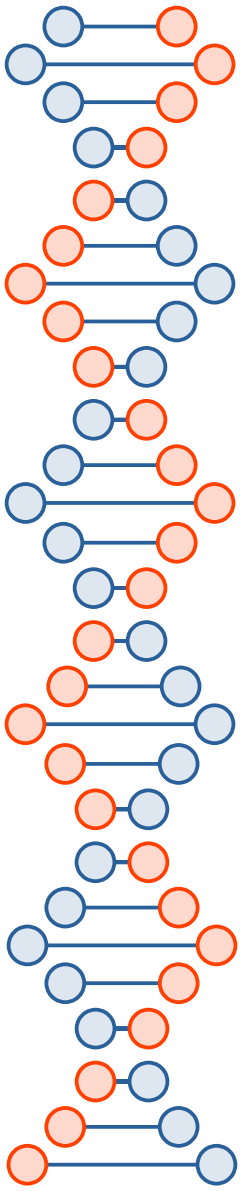
After: 0x5636902a2ac0] 6e340b9cffb37a989ca544e6bb780a2c78901d3fb33738768511a30
617afa01d          [0]      0x5636902a2b00
```

# Résultat - Réussite



```
3      [4]      ae254eaac19479be9204727e8f2630f72ef0bba602f91c89cbd1cf937cbae5ad - ae254eaac19479be9204727e8f2630f72ef0bba602f91c89cbd1cf937cbae5adPrevious Block Verified!
```

```
Choice: 4
2      [3]      7f7d33e8b571e199f1937ce4cbd506ab4ee9349d4cfb77979fb7aa03d3405c65 - 250d3060b8659b2b4b251ddb0f65ddabb58770723f6219927e40abdfd97b4e4fPrevious Block Verification Failed!
3      [11]     ae254eaac19479be9204727e8f2630f72ef0bba602f91c89cbd1cf937cbae5ad - ae254eaac19479be9204727e8f2630f72ef0bba602f91c89cbd1cf937cbae5adPrevious Block Verified!
4      [7]      9c6366d2df0f3a440a6d5f552e0b6a873f4bc0dfe3806629ab38a2c388b81d30 - dbc5462abef23ca0c2013ca74d142aa43b003a61458091f1f1db778d6a17fe60Previous Block Verification Failed!
```



# Conclusion

- Positifs

Décentralisation

Signatures

Route

Tiers de confiance

- Négatifs

Consomme énormément  
d'énergie

Nombre de mineurs

Confiance