Healthcare BlockChain

Presented by Maria Afara and Ali Fakih
Supervised by Dr. Mohammad Chaito

Contents

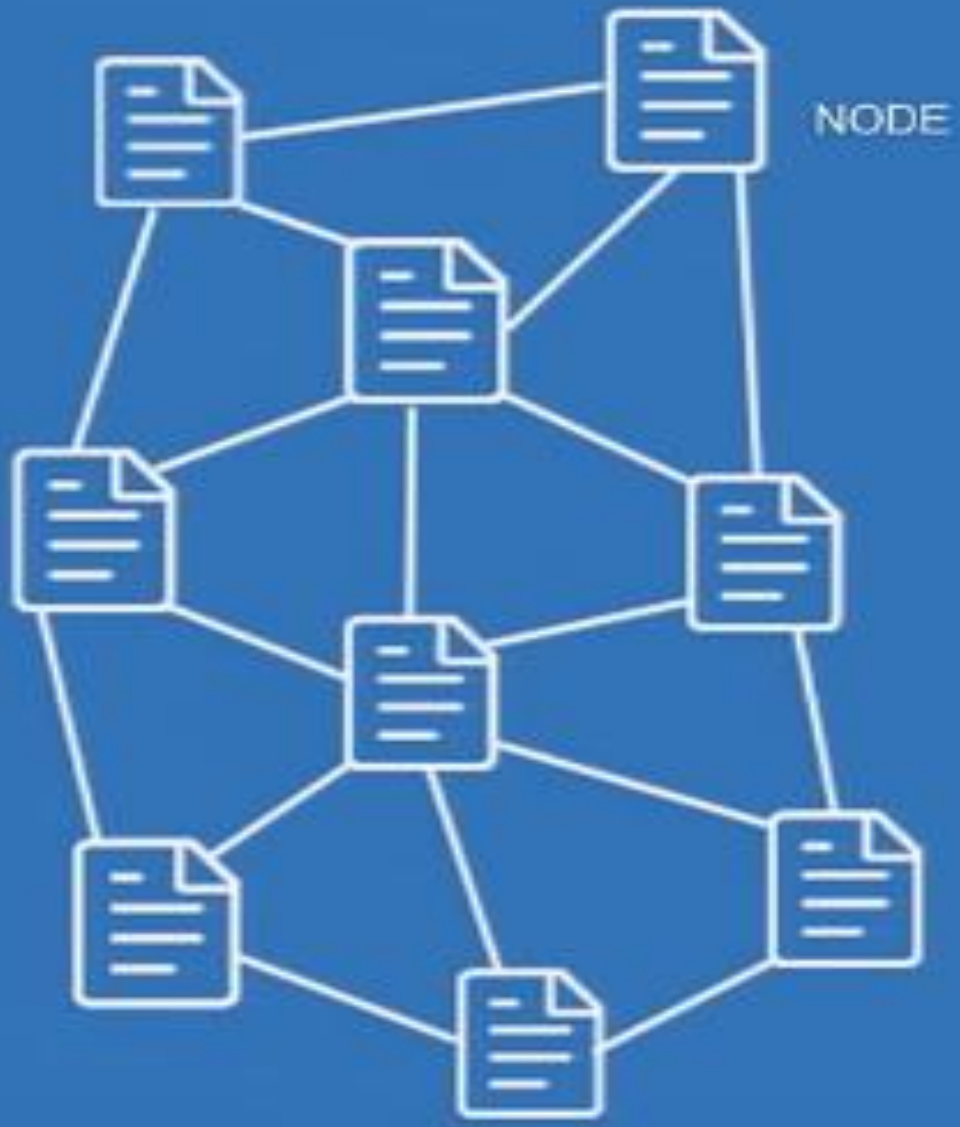
- Reason BlockChain exits.
- BlockChain
- Health Sector problem
- HealthCare BlockChain
- Implementation

Trust is like blood pressure. It's silent, vital to good health, and if abused it can be deadly.

2

— Frank K. Sonnenberg —

BlockChain


NODE

Transactions Information

↓

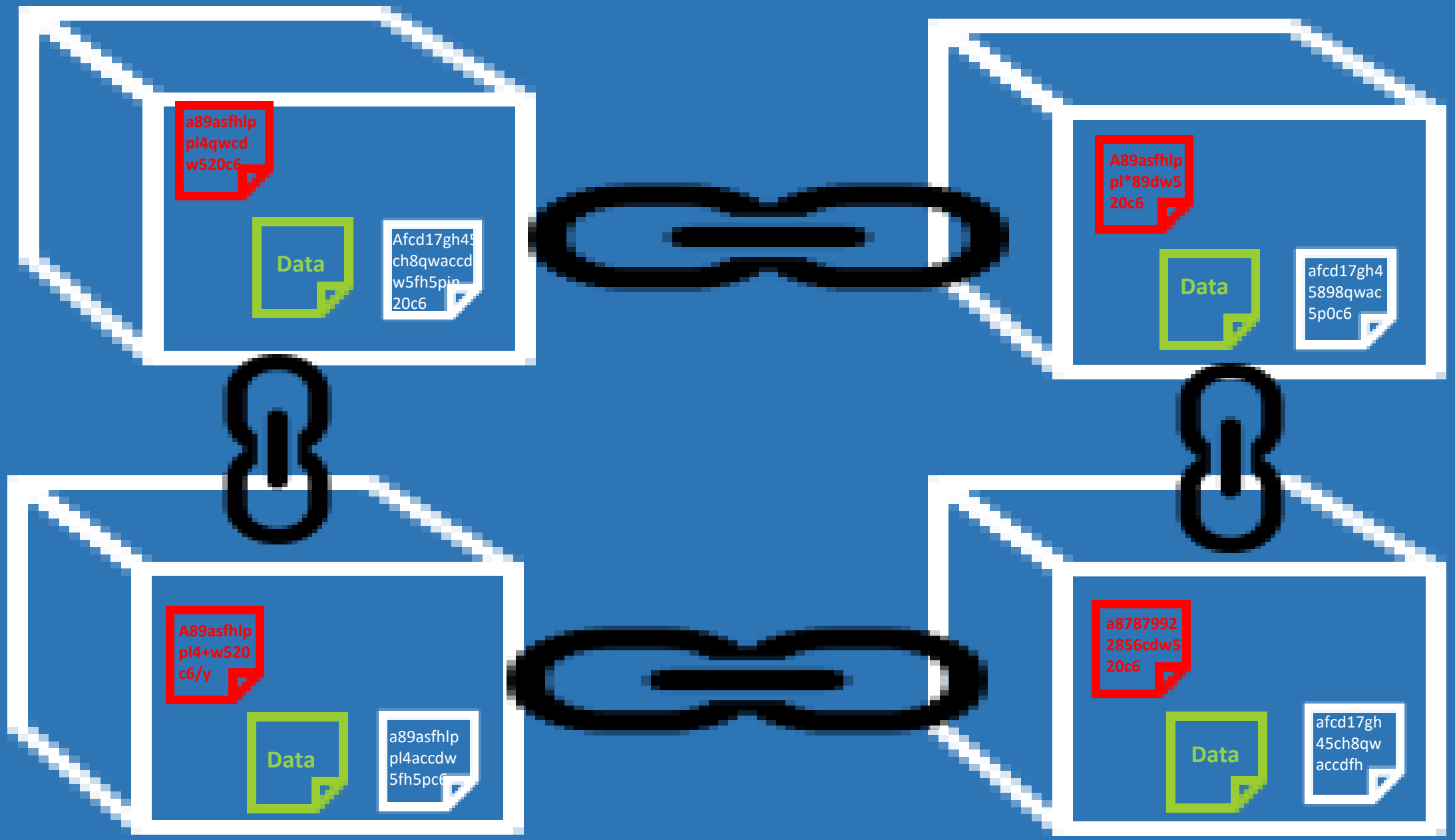
Secure Hash Algorithm

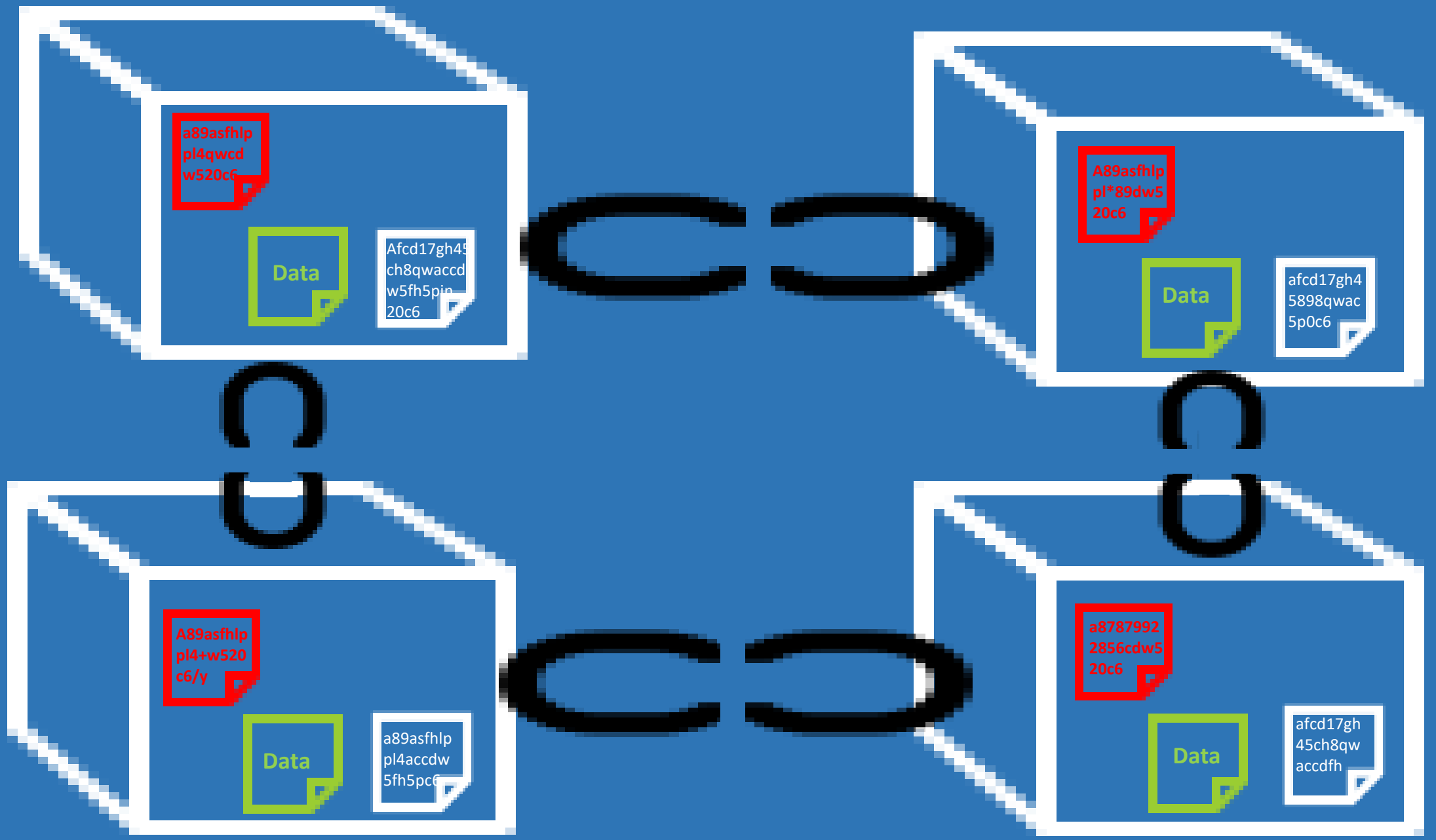
↓

Hash

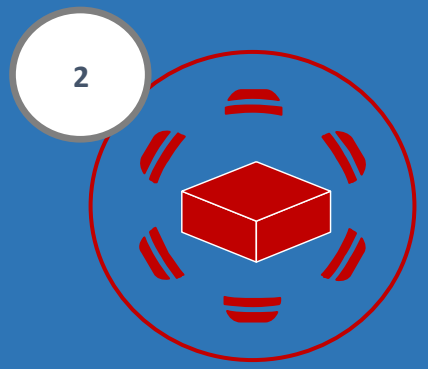
Transactions Information

Hash


a89asfhlp
pl4qwcd
w520c6
Data
Afcd17gh45
ch8qwaccd
w5fh5pip
20c6
A89asfhlp
pl*89dw5
20c6
Data
afcd17gh4
5898qwac
5p0c6
A89asfhlp
pl4+w520
c6/y
Data
a89asfhlp
pl4accdw
5fh5pc6
a8787992
2856cdw5
20c6
Data
afcd17gh
45ch8qw
accdfh


a89asfhlp
pl4qwcd
w520c6
Data
Afcd17gh45
ch8qwaccd
w5fh5pip
20c6
A89asfhlp
pl*89dw5
20c6
Data
afcd17gh4
5898qwac
5p0c6
A89asfhlp
pl4+w520
c6/y
Data
a89asfhlp
pl4accdw
5fh5pc6
a8787992
2856cdw5
20c6
Data
afcd17gh
45ch8qw
accdfh

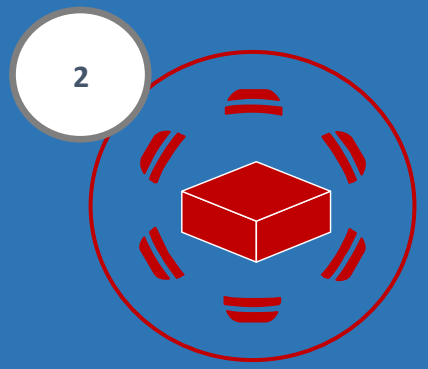
- **Someone creates a transaction**



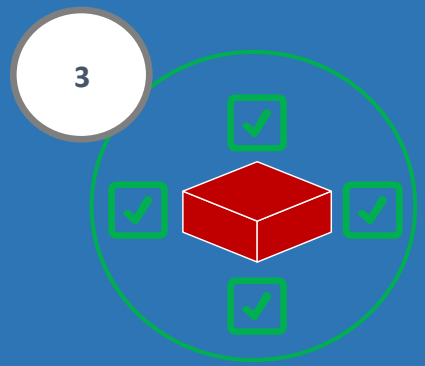
- Someone creates a transaction

- The requested transaction is broadcast to a P2P network consisting of computers, known as Miners

- Someone creates a transaction

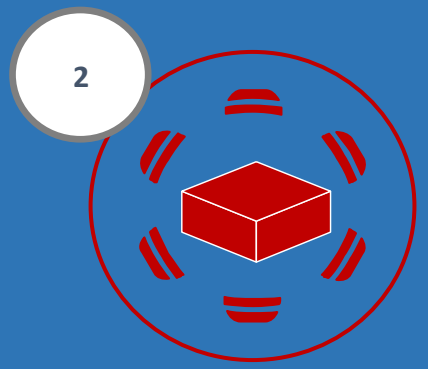


- The requested transaction is broadcast to a P2P network consisting of computers, known as Miners

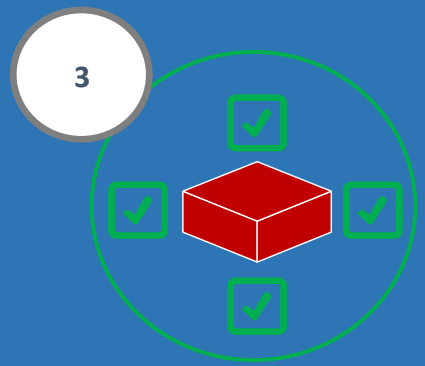


- The miner validate the transaction, put it into a block and finally create a hash for the block by solving a hard mathematical problem.

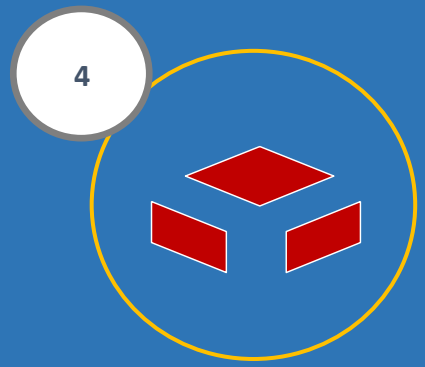
- Someone creates a transaction



- The requested transaction is broadcast to a P2P network consisting of computers, known as Miners

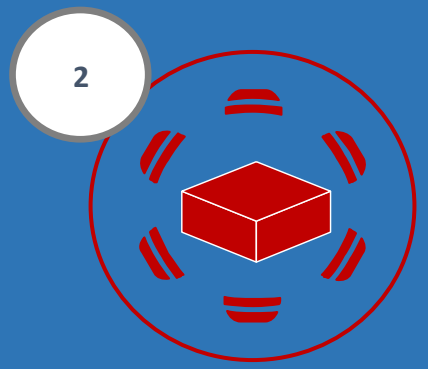


- The miner validate the transaction, put it into a block and finally create a hash for the block by solving a hard mathematical problem.

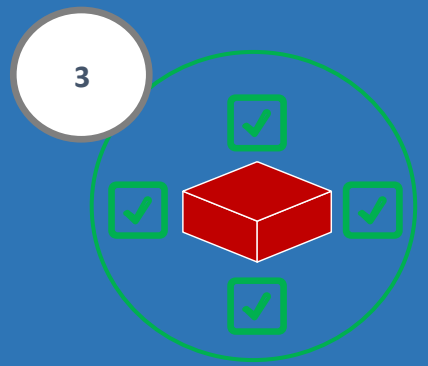


- Once verified, this transaction is represented as a new block. And then published to the other miners.

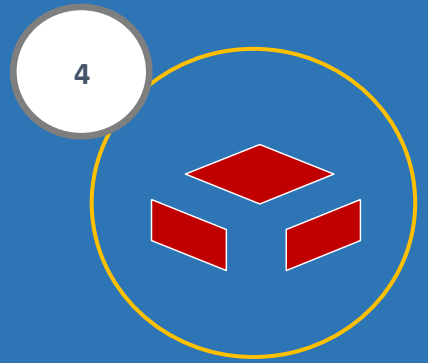
- Someone creates a transaction



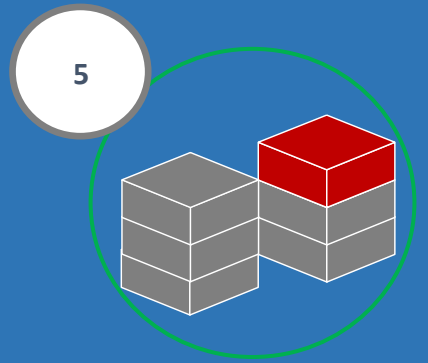
- The requested transaction is broadcast to a P2P network consisting of computers, known as Miners



- The miner validate the transaction, put it into a block and finally create a hash for the block by solving a hard mathematical problem.



- Once verified, this transaction is represented as a new block. And then published to the other miners.

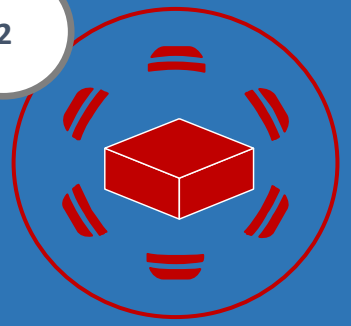


- The new block is then added to the existing block chain. After the miners have validated each transaction

1

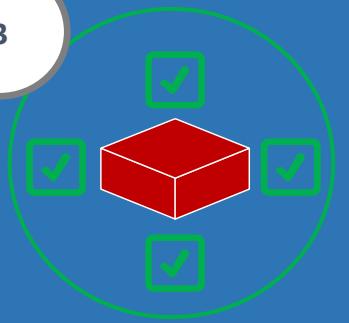
- Someone creates a transaction

2



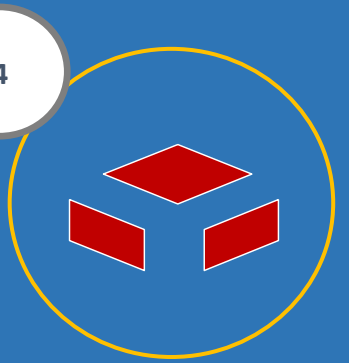
- The requested transaction is broadcast to a P2P network consisting of computers, known as Miners

3



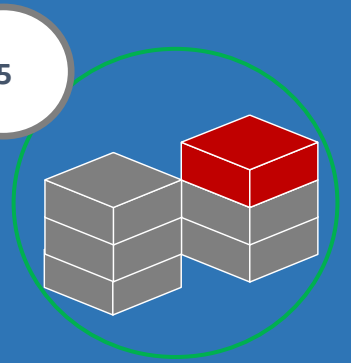
- The miner validate the transaction, put it into a block and finally create a hash for the block by solving a hard mathematical problem.

4



- Once verified, this transaction is represented as a new block. And then published to the other miners.

5



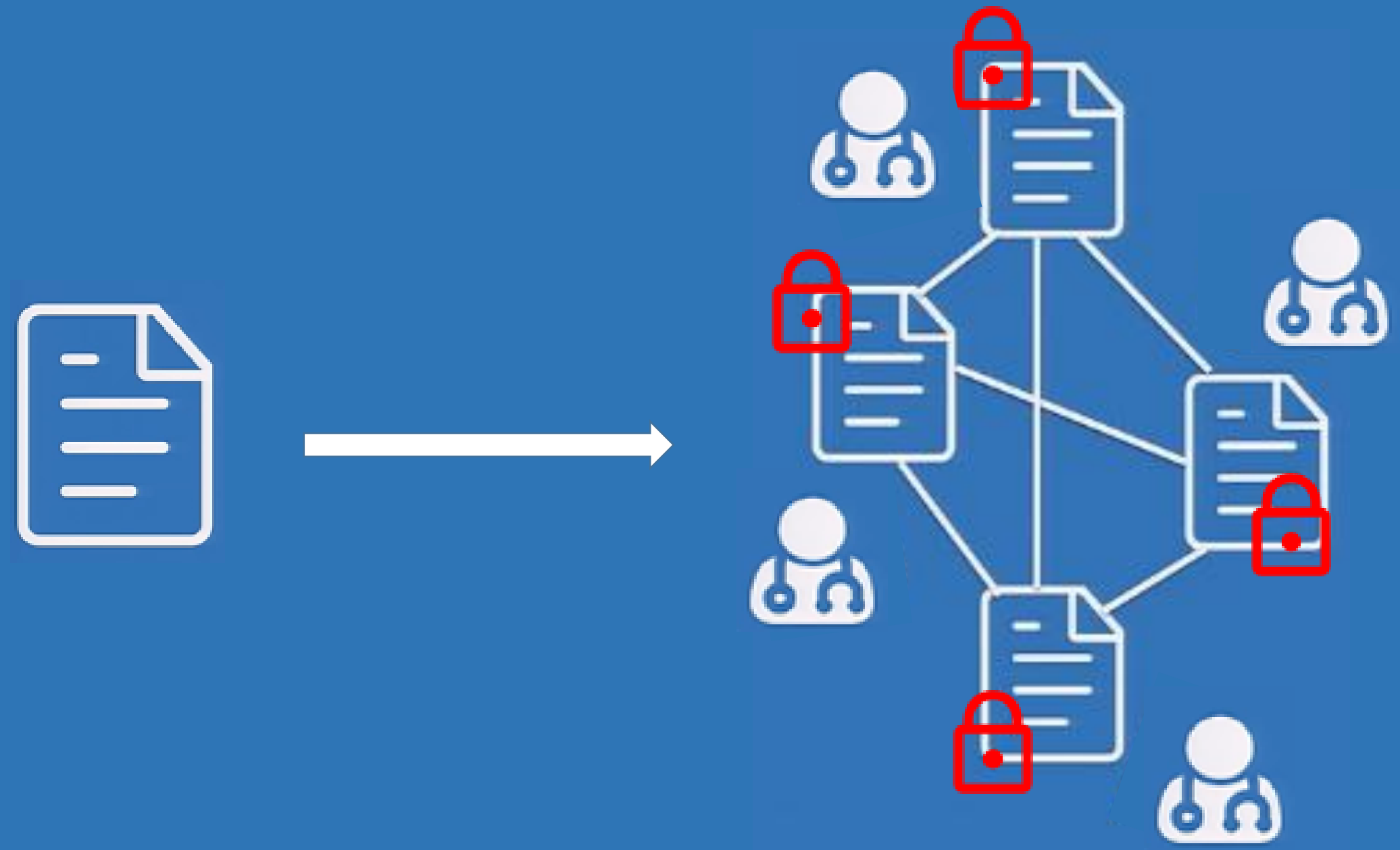
- The new block is then added to the existing block chain. After the miners have validated each transaction

6

- The transaction is complete

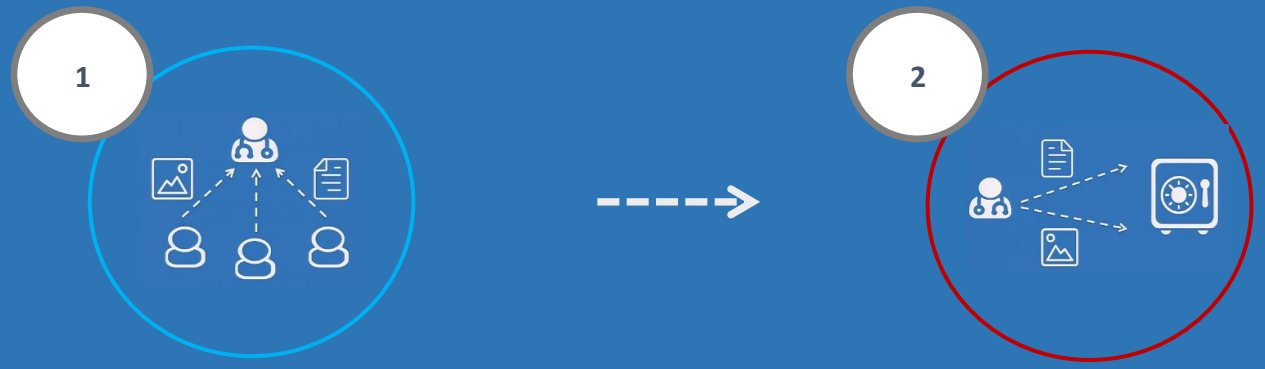
afcd17gh4
5ch8qwac
cdw5fh5pi
n20c6

Lack of Interoperability



HealthCare BlockChain

- 1.Health Care providers collect information from the patients



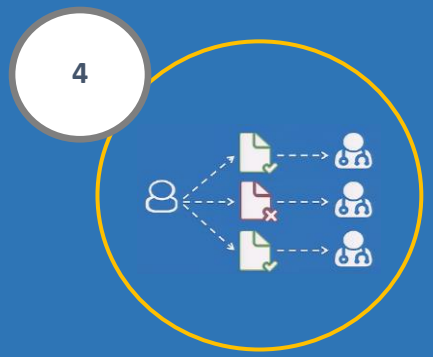
- 1.Health Care providers collect information from the patients
- 2. The data is stored in existing databases

- 1.Health Care providers collect information from the patients
- 2. The data is stored in existing databases
- 3. The data is secured then redirected to the block chain

- 1.Health Care providers collect information from the patients

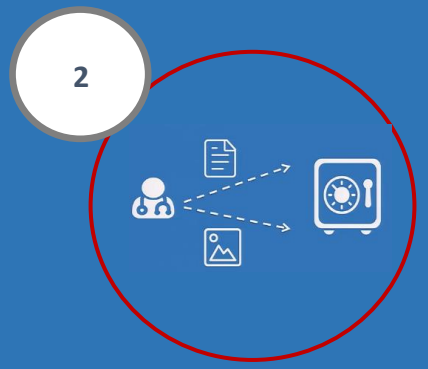
- 2. The data is stored in existing databases

- 3. The data is secured then redirected to the block chain

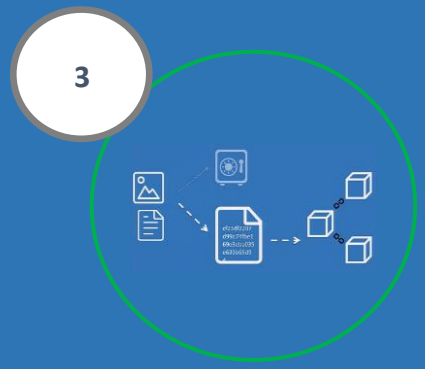


- 4. The patient decides who has access to his medical records

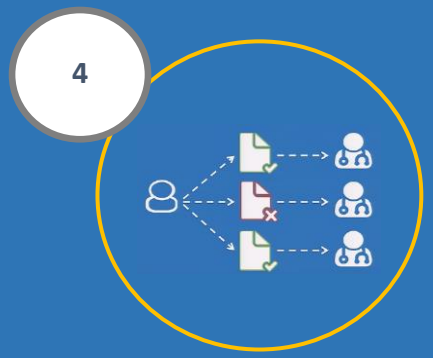
- 1.Health Care providers collect information from the patients



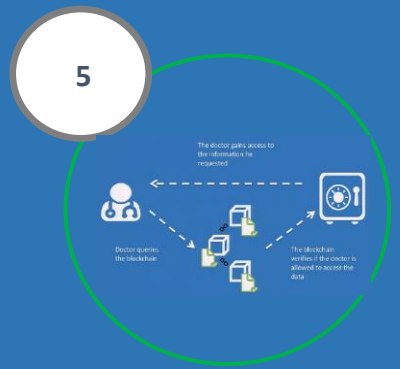
- 2. The data is stored in existing databases



- 3. The data is secured then redirected to the block chain



- 4. The patient decides who has access to his medical records



- 5. Healthcare stakeholders can query the block chain to obtain access to the information

1

Data Generation

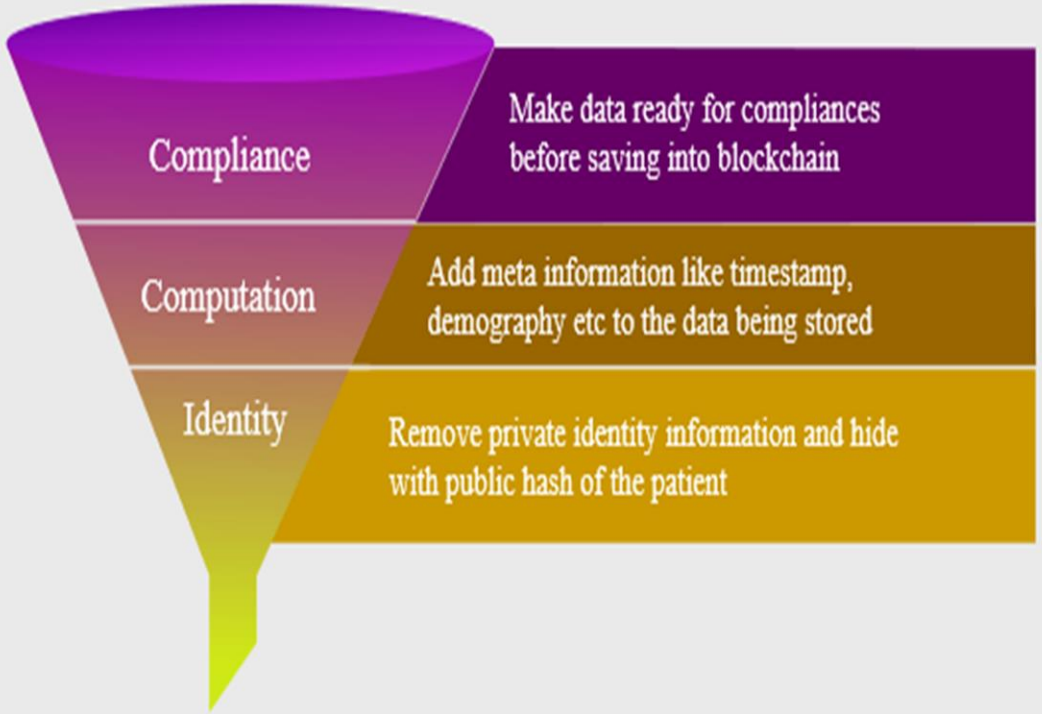
Data being generated by various sources including Smart devices, diagnostic tools, health care providers and legacy systems

Patient Health Data

2

Data Cleaning and Enrichement

Preparation before saving data into blockchain



3

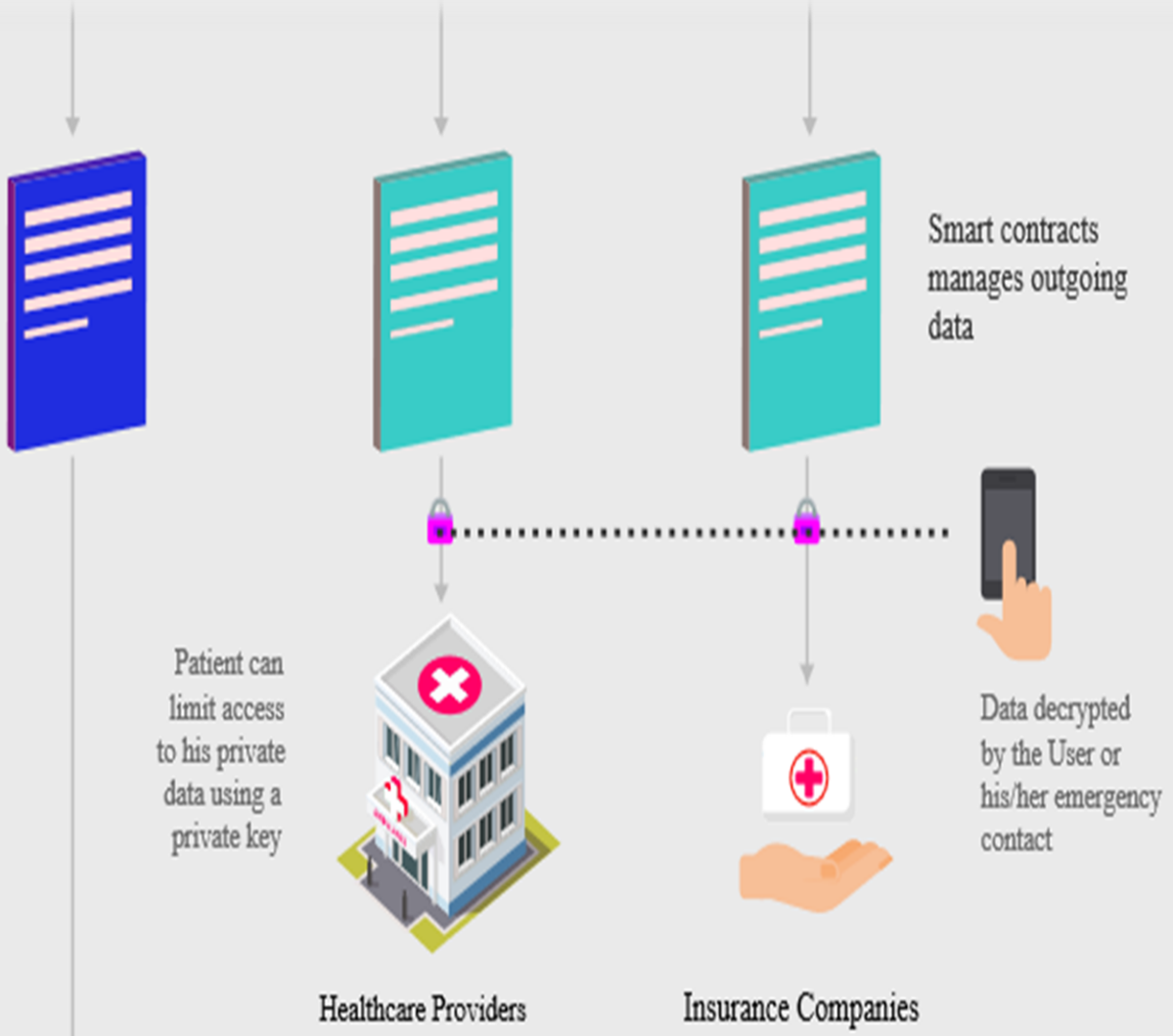
Data Capturing

Save data to blockchain

4

Data Consumption

Using patient health information for the treatment



5

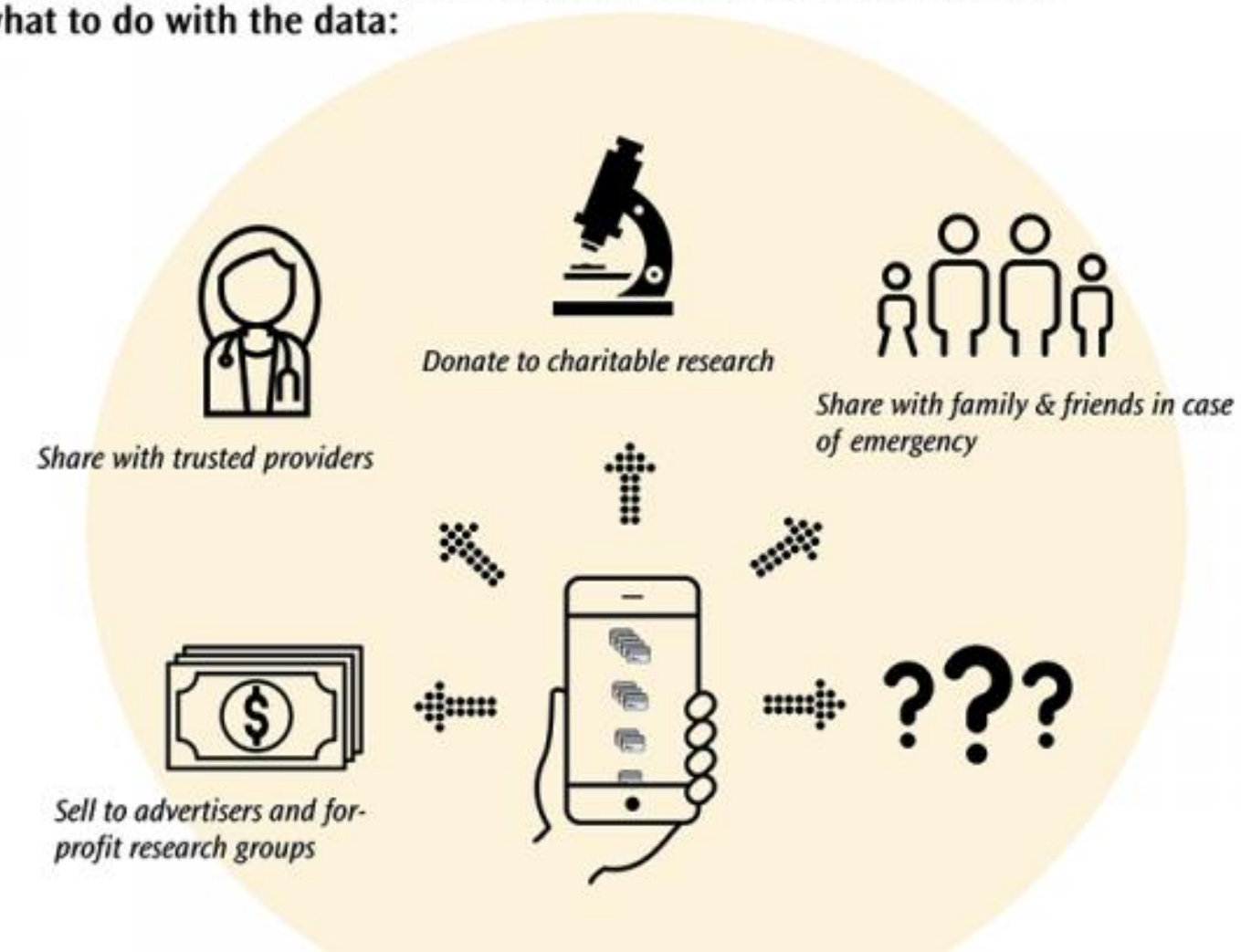
Data Mining

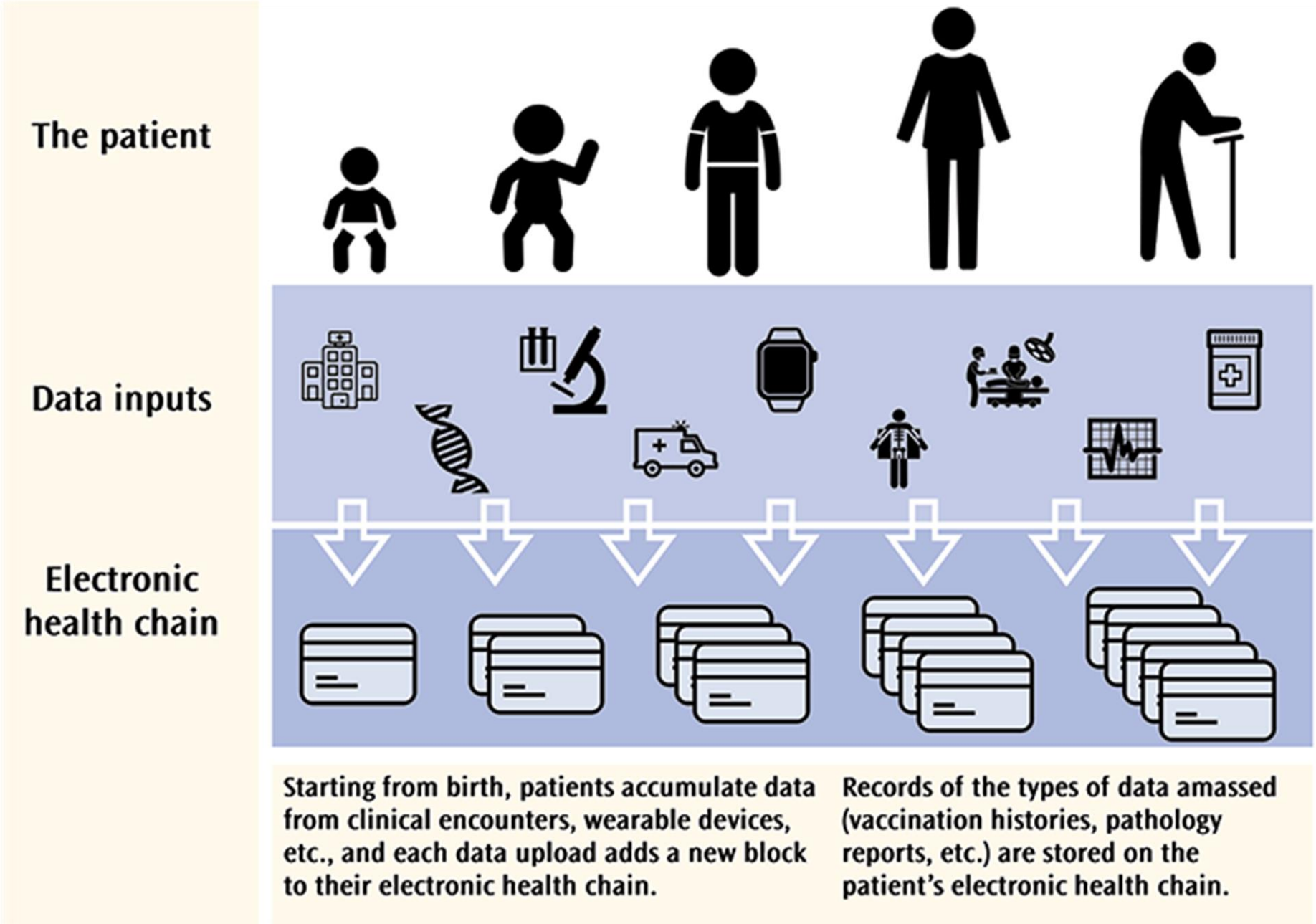
Research and further discovery

Clinical Trials and Research

Non-identifiable patient data (e.g. age, gender, illness) is publically available

Each *patient* owns his or her electronic health chain and decides what to do with the data:




The patient
Data inputs
Electronic health chain
Starting from birth, patients accumulate data from clinical encounters, wearable devices, etc., and each data upload adds a new block to their electronic health chain.
Records of the types of data amassed (vaccination histories, pathology reports, etc.) are stored on the patient's electronic health chain.

Implementation

1) How the health record will be added to the blockchain?

2) What happen if someone tries to impersonate someone else?

3) What happen if someone alter the information that are already in blockchain?

4) How the private data of patient can be accessed from blockchain?

```java
public class Profile {

    public PrivateKey privateKey;
    public PublicKey publicKey;
    public String name;
    public int id;
    public String key = "Bar12345Bar12345"; // 128 bit key
```

```java
public class Transaction {
    public String transactionId; //Contains a hash of transaction*
    public PublicKey sender; //Patient address/public key.
    public String name; //patient name
    public int id;  //pateint id
    public Boolean Ispub; //Type of information (public or private)
    public String value; //Contains the informations.
    public byte[] enc; //contains the hashed informations
    public byte[] signature;
```

```java
public class Block {

    public String hash;
    public String previousHash;
    public String merkleRoot;
    public ArrayList<Transaction> transactions = new ArrayList<Transaction>(); //our data will be a simple message.
    public long timeStamp; //as number of milliseconds since 1/1/1970.
    public int nonce;
```

```
//Create Profiles:
profileA = new Profile("Ali",1);
profileB = new Profile("Bob",2);
profileC = new Profile("Maria",3);


//Ali creates HEALTH Record
transaction = new Transaction(profileA.publicKey, "Grippe" ,profileA.name ,profileA.id ,false ,profileA.key);
transaction.generateSignature(profileA.privateKey);  //manually sign the genesis transaction

//Bob creates fake HEALTH Record cause he generates the signature using Maria's Public key & his private key
transaction1 = new Transaction(profileC.publicKey, "FAKE INFORMATION" ,profileC.name ,profileC.id ,true ,profileC.key);
transaction1.generateSignature(profileB.privateKey);     //manually sign the genesis transaction

//Maria Creates Transaction
transaction2 = new Transaction(profileC.publicKey, "Grippe" ,profileC.name ,profileC.id ,true ,profileC.key);
transaction2.generateSignature(profileC.privateKey);     //manually sign the genesis transaction


Block block = new Block("0");
block.addTransaction(transaction);
block.addTransaction(transaction1);
addBlock(block);


Block block1 = new Block(block.hash);
block1.addTransaction(transaction2);
addBlock(block1);


Block block2 = new Block(block1.hash);
block2.addTransaction(transaction2);
addBlock(block2);
block2.transactions.get(0).value="Other disease!";
isChainValid();
```

```
public Profile( String name , int id) {
    this.name=name;
    this.id=id;
    generateKeyPair();
}

public void generateKeyPair() {
    try {
        KeyPairGenerator keyGen = KeyPairGenerator.getInstance("ECDSA","BC");
        SecureRandom random = SecureRandom.getInstance("SHA1PRNG");
        ECGenParameterSpec ecSpec = new ECGenParameterSpec("prime192v1");
        // Initialize the key generator and generate a KeyPair
        keyGen.initialize(ecSpec, random); //256
        KeyPair keyPair = keyGen.generateKeyPair();
        // Set the public and private keys from the keyPair
        privateKey = keyPair.getPrivate();
        publicKey = keyPair.getPublic();
     //    System.out.println("pKey:"+privateKey);
     //   System.out.println("puKey:"+publicKey);
    }catch(Exception e) {
        throw new RuntimeException(e);
    }
}
```

```java
public void generateSignature(PrivateKey privateKey) {
    String data = StringUtil.getStringFromKey(sender)  +value +id + name +pub   ;
    signature = StringUtil.applyECDSASig(privateKey,data);
}
```

```java
//Add transactions to this block
public boolean addTransaction(Transaction transaction) {
    //process transaction and check if valid, unless block is genesis block then ignore.
    if(transaction == null) return false;
    if((!"0".equals(previousHash))) {
        if((transaction.processTransaction() != true)) {
            System.out.println("Transaction failed to process. Discarded.");
            return false;
        }
    }
    if((transaction.processTransaction() != true)) {
        System.out.println("Transaction failed to process. Discarded.");
        return false;
    }

    transactions.add(transaction);
    System.out.println("Transaction Successfully added to Block");
    return true;
}
```

```java
public boolean verifySignature(PublicKey p) {
    String data = StringUtil.getStringFromKey(p)  + value + id + name + pub ;
    return StringUtil.verifyECDSASig(p, data, signature);
}
```

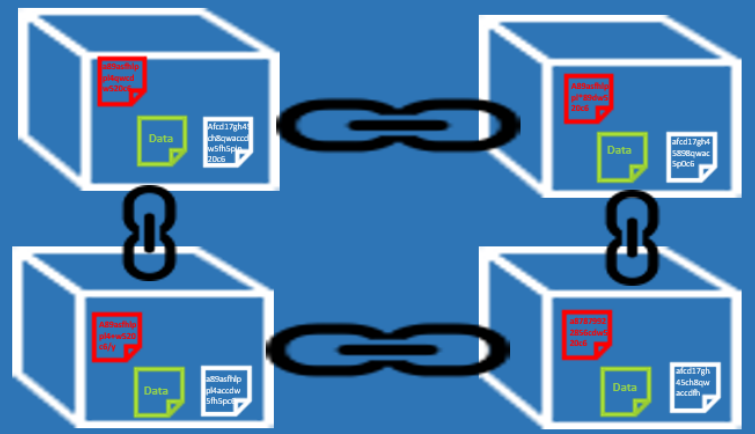
```
//Increases nonce value until hash target is reached.
    public void mineBlock(int difficulty) {
        merkleRoot = StringUtil.getMerkleRoot(transactions);
        String target = StringUtil.getDificultyString(difficulty); //Create a string with difficulty * "0"
        while(!hash.substring( 0, difficulty).equals(target)) {
            nonce ++;
            hash = calculateHash();
        }
        System.out.println("Block Mined!!! : " + hash);
    }

//Calculate new hash based on blocks contents
public String calculateHash() {
    String s="";
    for(int i=0;i < transactions.size() ; i++ ){
        s+=transactions.get(i).transactionId + " " + transactions.get(i).value +  " " +  transactions.get(i).signature + " " + transactions.get(i).id  +

    }

    String calculatedhash = StringUtil.applySha256(
            previousHash +
            Long.toString(timeStamp) +
            Integer.toString(nonce) +
            merkleRoot + s
            );
    return calculatedhash;
}
```

```java
public static Boolean isChainValid() {
    Block currentBlock;
    Block previousBlock;
    String hashTarget = new String(new char[difficulty]).replace('\0', '0');
    //loop through blockchain to check hashes:
    for(int i=1; i < blockchain.size(); i++) {
        currentBlock = blockchain.get(i);
        previousBlock = blockchain.get(i-1);
        //compare registered hash and calculated hash:
        if(!currentBlock.hash.equals(currentBlock.calculateHash()) ){
            System.out.println("#Current Hashes not equal");
            return false;
        }
        if(i== 1 ){
            if(!previousBlock.hash.equals(previousBlock.calculateHash()) ){
                System.out.println("Current Hashes not equal");
                return false;
            }
        }
        //compare previous hash and registered previous hash
        if(!previousBlock.hash.equals(currentBlock.previousHash) ) {
            System.out.println("#Previous Hashes not equal");
            System.out.println("#Blockchain is NOT valid");
            return false;
        }
        //check if hash is solved
        if(!currentBlock.hash.substring( 0, difficulty).equals(hashTarget)) {
            System.out.println("#This block hasn't been mined");
            System.out.println("#Blockchain is NOT valid");
            return false;
        }
    }
    System.out.println("Blockchain is valid");
    return true;
}
```



Output:

```
Transaction Successfully added to Block
#Transaction Signature failed to verify
Transaction failed to process. Discarded.
Block Mined!!! : 000af70a6f166dbab25fb01402448aa9e3a2a15e2caa6479ce9e32913b12135f
Transaction Successfully added to Block
Block Mined!!! : 0008bbf7f1641c4eca038d716baae754d9862e142e43a515435d8a7101ae6fea
Transaction Successfully added to Block
Block Mined!!! : 000f1f50ac1e7bbf0b8717d3019289a53c0dee3252785eb8d8fecf990ab56245
#Current Hashes not equal
#Blockchain is NOT valid
```

```java
//all the medical record in the blockchain
System.out.println("\n\n\n");
System.out.println("All Transactions");
if(isChainValid())
    for(int i=0; i<blockchain.size() ; i++){
        System.out.println("Block#" + i + ":");
        for(int j=0; j<  blockchain.get(i).transactions.size() ; j++){
            System.out.println("\t Transaction#" + j + ":");
            Transaction tr=blockchain.get(i).transactions.get(j);

                System.out.println("\t\t Name: " + blockchain.get(i).transactions.get(j).name);
                System.out.println("\t\t Id: " + blockchain.get(i).transactions.get(j).id);
                System.out.println("\t\t isPublicData: " + blockchain.get(i).transactions.get(j).pub);
                System.out.println("\t\t Content: "  +blockchain.get(i).transactions.get(j).value);
                System.out.println("-----------------------------------");



        }
    }
```

Output:

```
All Transactions
Blockchain is valid
Block#0:
        Transaction#0:
                Name: Ali
                Id: 1
                isPublicData: false
                Content: iÚY˜¸H|⊠ÊtðY  ”Šo
-----------------------------------
Block#1:
        Transaction#0:
                Name: Maria
                Id: 3
                isPublicData: true
                Content: Grippe
-----------------------------------
Block#2:
        Transaction#0:
                Name: Maria
                Id: 3
                isPublicData: true
                Content: Grippe
-----------------------------------
```

```java
//all the medical record for profile A!
System.out.println("\n\n\n");
System.out.println("Transaction for profile A + We know his secret key");
if(isChainValid())
    for(int i=0; i<blockchain.size() ; i++){
        System.out.println("Block#" + i + ":");
        for(int j=0; j<  blockchain.get(i).transactions.size() ; j++){
            Transaction tr=blockchain.get(i).transactions.get(j);
            if( tr.verifySignature(profileA.publicKey) ){
                System.out.println("\t Transaction#" + j + ":");

                if(blockchain.get(i).transactions.get(j).pub){
                    System.out.println("\t\t Name: " + blockchain.get(i).transactions.get(j).name);
                    System.out.println("\t\t Id: " + blockchain.get(i).transactions.get(j).id);
                    System.out.println("\t\t isPublicData: " + blockchain.get(i).transactions.get(j).pub);
                    System.out.println("\t\t Content: "  +blockchain.get(i).transactions.get(j).value);
                    System.out.println("-----------------------------------");
                }
                else{
                    String content=blockchain.get(i).transactions.get(j).value;
                    SecretKeySpec aesKey = new SecretKeySpec(profileA.key.getBytes(), "AES");
                    Cipher cipher = Cipher.getInstance("AES");
                    // decrypt the text
                    cipher.init(Cipher.DECRYPT_MODE, aesKey);
                    String decrypted = new String(cipher.doFinal(blockchain.get(i).transactions.get(j).enc));
                    System.out.println("\t\t Name: " + blockchain.get(i).transactions.get(j).name);
                    System.out.println("\t\t Id: " + blockchain.get(i).transactions.get(j).id);
                    System.out.println("\t\t isPublicData: " + blockchain.get(i).transactions.get(j).pub);
                    System.out.println("\t\t Content: "  +decrypted);
                    System.out.println("-----------------------------------");
                }
            }

        }
    }
```

Output:

```
Blockchain is valid
Block#0:
        Transaction#0:
                Name: Ali
                Id: 1
                isPublicData: false
                Content: Grippe
-----------------------------------
```

Thank you