

A LA DECOUVERTE DE NBITCOIN C#



MEETUP  DOCDOKU
digital enterprise applications

> Introduction

Au programme,

- ▶ Généralement, les présentations autour de Bitcoin commencent par une approche théorique des mécanismes permettant de sécuriser le réseau.
- ▶ Dans cette présentation nous allons faire abstraction de tout cela → .
- ▶ Nous allons considérer la **blockchain Bitcoin** comme **un service de stockage décentralisé** que l'on va exploiter en mode Blockchain As a Service.
- ▶ Nous consommerons ce service à l'aide de la librairie [NBitcoin.net](#) et de l'api [QBitNinja](#) → 
- ▶ Nous auront une approche pratique, qui consistera à **programmer et signer un transfert Bitcoin** entre deux adresses que nous allons créer.
- ▶ La notion de « **transfert** » et de « **vérification de propriété** » est fondamentale pour le réseau Bitcoin puisqu'il a été spécialement conçu pour cela !
- ▶ Tout en construisant notre transfert, nous découvrirons à quoi servent une **adresse**, une **clé privée**, une **clé publique** ou une **transaction** ...

> Introduction

Généralités sur Bitcoin

- ▶ Un système de **transfert et de vérification** de propriété,
- ▶ Repose sur un **réseau pair à pair**,
- ▶ Pas d'autorité **centrale**,

- ▶ L'application initiale et l'innovation principale de ce réseau est **un système de monnaie numérique décentralisé**
- ▶ L'unité de compte de ce système est le **Bitcoin**,

- ▶ Bitcoin fonctionne avec des logiciels et un protocole,
- ▶ Permet à ses participant d'émettre et de gérer des **transactions** de façon collective et automatique,

- ▶ Un protocole **libre et ouvert** dont le code source est publié sous **licence MIT**,

- ▶ Bitcoin est conçu pour s'**auto-réguler**,
- ▶ Son bon fonctionnement est garanti par une organisation générale que **tout le monde peut examiner**,
- ▶ Tout y est **public** :
 - les protocoles de base,
 - les algorithmes cryptographiques utilisés,
 - les programmes opérationnels,
 - les données de comptes,
 - les débats des développeurs.
 - ...

> Réaliser un Transfert Bitcoin

▶ Problématique :

- Des fonds initiaux ont été envoyés à Bob,
- Bob a réalisé un transfert de Bitcoin à Alice en lui laissant un message,
- Nous allons recréer la transaction entre Bob et Alice à l'aide de NBitcoin,



> Réaliser un Transfert Bitcoin

► Configuration du projet :

```
> mkdir btcMeetup
> cd btcMeetup
> dotnet new console                         // nouvelle application de type console
> dotnet add package NBitcoin                 // la fameuse librairie NBitcoin (écrite par Nicolas Dorier)
> dotnet add package QBitNinja.Client          // la librairie QBitNinja (également développée par Nicolas Dorier, Metaco SA)
> dotnet restore
```



The screenshot shows the Visual Studio code editor with two tabs open: `Program.cs` and `btcMeetup.csproj`.

Program.cs:

```
1 using System;
2
3 namespace btcMeetup
4 {
5     class Program
6     {
7         static void Main(string[] args)
8         {
9             Console.WriteLine("Hello World!");
10        }
11    }
12 }
```

btcMeetup.csproj:

```
1 <Project Sdk="Microsoft.NET.Sdk">
2
3   <PropertyGroup>
4     <OutputType>Exe</OutputType>
5     <TargetFramework>netcoreapp2.2</TargetFramework>
6   </PropertyGroup>
7
8   <ItemGroup>
9
10    <PackageReference Include="NBitcoin" Version="4.1.2.36" />
11
12    <PackageReference Include="QBitNinja.Client" Version="1.0.3.49" />
13
14  </ItemGroup>
15
16 </Project>
```

A red arrow points from the line `<PackageReference Include="NBitcoin" Version="4.1.2.36" />` in the `btcMeetup.csproj` file to the `NBitcoin` library usage in the `Console.WriteLine` statement of `Program.cs`. A blue arrow points from the line `<PackageReference Include="QBitNinja.Client" Version="1.0.3.49" />` in the `btcMeetup.csproj` file to the `QBitNinja.Client` library usage in the `Console.WriteLine` statement of `Program.cs`.

Pour manipuler les différents éléments du protocole Bitcoin.

Pour diffuser et récupérer des données stockées dans la blockchain Bitcoin.

> Réaliser un Transfert Bitcoin

▶ Pour réaliser un transfert sur le protocole Bitcoin, on s'appuie sur 5 éléments fondamentaux :

Private Key

- Permet de signer une transaction,
- Donne le droit de dépenser les fonds liés à une adresse,
- Ne doit pas être partagé, doit être conservé en lieu sûr,

Public Key

- Permet de s'assurer que vous êtes le propriétaire d'une adresse pouvant recevoir des fonds.
- Est générée à partir de la clé privée (l'inverse étant « impossible »)
- La clé public permet de générer une adresse bitcoin et un scriptPubKey

ScriptPubKey

- C'est un peu l'équivalent d'une BitcoinAddress mais au niveau du protocole.
- Vous n'envoyez donc pas des fonds à une adresse mais à un ScriptPubKey.
- N'est pas facilement partageable contrairement à une adresse.

Bitcoin Address

- Pour recevoir un transfert de fonds,
- Information facilement encodable en QR Code,
- Peut être communiqué à tous le monde,

Transaction

- Structure de données permettant d'encoder un transfert de valeur entre un ou plusieurs participants au réseau Bitcoin.

> Réaliser un Transfert Bitcoin > Créer une adresse

▶ Générer une adresse pour Bob et une adresse pour Alice :



- ▶ Une adresse Bitcoin est une information que vous allez partager avec les autres utilisateurs du réseaux pour recevoir des fonds.



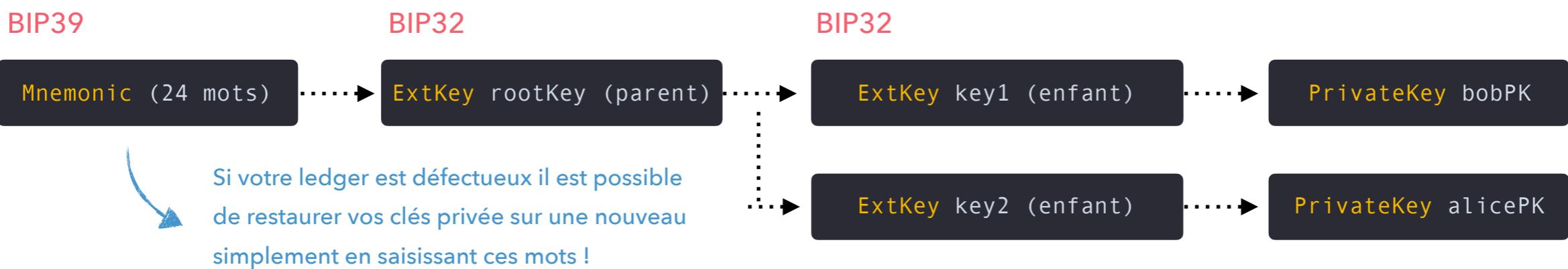
- ▶ Pour obtenir une adresse vous devez d'abord générer une clé privée.
- ▶ Elle est le seul moyen de dépenser les bitcoins envoyés à votre adresse.
- ▶ Les clés privées sont personnelle et ne sont pas stockées sur le réseau.
- ▶ Elles peuvent être générées sans être connecté à internet.

NBitcoin supporte plusieurs standards pour générer des clés privées.

Nous allons utiliser les standards **BIP32** et **BIP39** qui permettent de générer une clé racine à partir d'une **wordlist**.

C'est sur ce principe que fonctionne les cold-wallets tel que Ledger Nano ou Trezor.

Cette **wordlist** ou **seed** ou **mnemonic** permet de re-générer indéfiniment la même suite de clé.



> Réaliser un Transfert Bitcoin > Créer une clé privée

- ▶ Générer une nouvelle mnémonique :

```
Mnemonic mnemonic = new Mnemonic(Wordlist.English, WordCount.TwentyFour);  
Console.WriteLine(mnemonic);  
  
// Output :  
inmate tide belt huge spirit hip language arctic slice decide street foil express asthma vehicle  
donkey term drive finger exotic still grant sample ski
```

- ▶ Restaurer une mnémonique existante :

```
var wordlist = "inmate tide belt huge spirit hip language arctic slice decide street foil express asthma vehicle  
donkey term drive finger exotic still grant sample ski";  
Mnemonic mnemonic = new Mnemonic(wordlist);  
  
// Output :  
inmate tide belt huge spirit hip language arctic slice decide street foil express asthma vehicle  
donkey term drive finger exotic still grant sample ski
```



(fausse wordlist pour l'exemple)

> Réaliser un Transfert Bitcoin > Créer une clé privée

- ▶ On génère la clé racine qui dérive de notre « Mnémonique » et d'un mot de passe :

```
var wordlist = "inmate tide belt huge spirit hip language arctic slice decide street foil express asthma vehicle  
donkey term drive finger exotic still grant sample ski";  
  
Mnemonic mnemo = new Mnemonic(wordlist);  
  
ExtKey hdRootKey = mnemo.DeriveExtKey("danang_Sky36");
```

- ▶ On peut maintenant générer deux « sous-clés », une pour Bob, une pour Alice, en dérivant de notre clé racine :

```
ExtKey hdKey1 = hdRootKey.Derive(0);  
ExtKey hdKey2 = hdRootKey.Derive(1);  
  
var bobPK = hdKey1.PrivateKey;  
var alicePK = hdKey2.PrivateKey;
```



- ▶ Remarque :

- Grâce à ce mécanisme de dérivation il est possible de générer et régénérer des arborescences complexes de clés.
- On peut par exemple construire une arborescence de clés basées sur l'organigramme d'une entreprise.

> Réaliser un Transfert Bitcoin > Créer une clé privée > Créer une clé publique

- ▶ A partir de nos deux clés privées, nous allons pouvoir obtenir leurs clés publiques respectives :

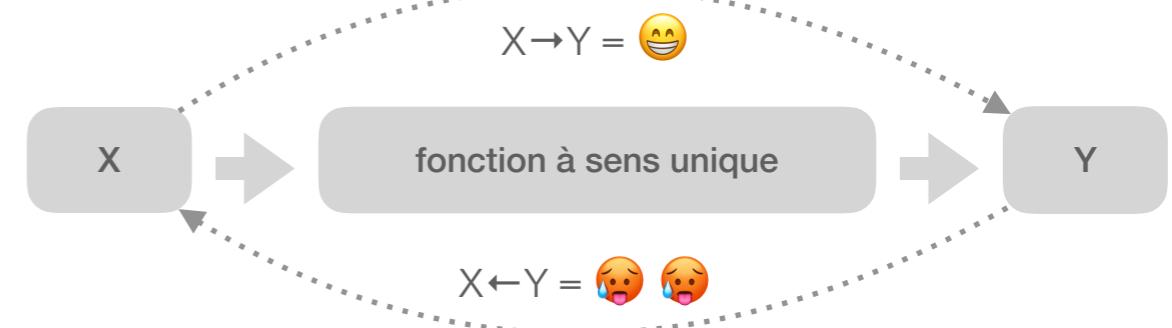


- ▶ On génère une clé publique à partir d'une clé privée au moyen d'une fonction à sens unique.
- ▶ C'est à dire une fonction qui peut aisément être calculé mais difficilement inversée.
- ▶ La clé publique permet de recevoir des fonds et d'attester que vous êtes le propriétaire d'une adresse.
- ▶ En revanche, elle ne permet pas de dépenser les fonds d'une adresse.

```
ExtKey hdKey1 = hdRootKey.Derive(0);
ExtKey hdKey2 = hdRootKey.Derive(1);

var bobPK    = hdKey1.PrivateKey;
var alicePK = hdKey2.PrivateKey;

var bobPubKey  = bobPK.PubKey;      Clé publique d'Alice
var alicePubKey = alicePK.PubKey;
```



> Réaliser un Transfert Bitcoin > Créer une clé privée > Créer une clé publique > Créer une adresse

- Après avoir généré les clés publiques d'Alice et Bob, nous allons pouvoir obtenir leurs adresses :

```
ExtKey hdKey1 = hdRootKey.Derive(0);
ExtKey hdKey2 = hdRootKey.Derive(1);

var bobPK    = hdKey1.PrivateKey;
var alicePK = hdKey2.PrivateKey;

var bobPubKey = bobPK.PubKey;
var alicePubKey = alicePK.PubKey;

var bobAddress = bobPK.PubKey.GetAddress(ScriptPubKeyType.Legacy, Network.Main);
var aliceAddress = alicePK.PubKey.GetAddress(ScriptPubKeyType.Legacy, Network.Main);

// Output :
Adresse de Bob : 16joUFCSaVDfacYsDxCm2oF5mRyLTq6DvY
Adresse d'Alice : 13hA6HP6W4BCwmtr89CKh6fTra19Exm4bL
```

Pour générer une adresse testNet ou mainNet

Par défaut !



Public Key



Bitcoin Address



Network

- Il existe 2 réseaux Bitcoins : **MainNet** et **TestNet**
- On obtient une adresse Bitcoin à partir de sa clé publique, et en précisant le réseau sur lequel on souhaite opérer.
- Remarque: sur le MainNet, les erreurs peuvent coûter cher 😱 !

> Réaliser un Transfert Bitcoin > Créer une clé privée > Créer une clé publique > Créer une adresse

- ▶ On peut consulter le solde des adresses d'Alice et de Bob à l'aide d'un explorateur de blockchain tel que « BlockCypher » ou « QBitNinja » :

BOB 🧑

16joUFCsaVDfacYsDxCm2oF5mRyLTq6DvY

ALICE 🧑

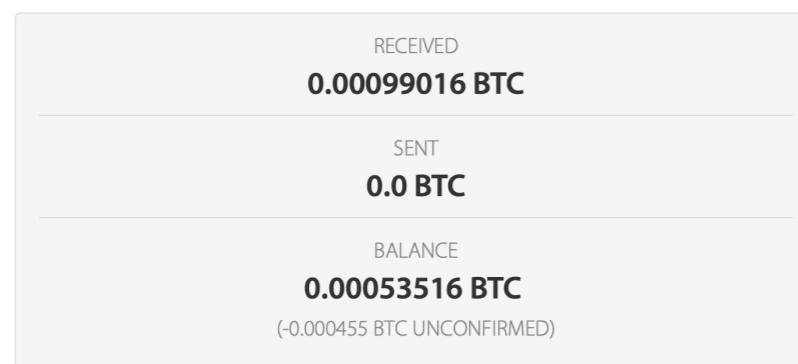
13hA6HP6W4BCwmtr89CKh6fTra19Exm4bL



The screenshot shows the BlockCypher website interface. At the top, there are dropdown menus for 'BTC' and 'Address, transact' along with a search bar. Below the header, the text 'Bitcoin Address' is displayed next to the address '16joUFCsaVDfacYsDxCm2oF5mRyLTq6DvY'. The background features a stylized 3D network graph.

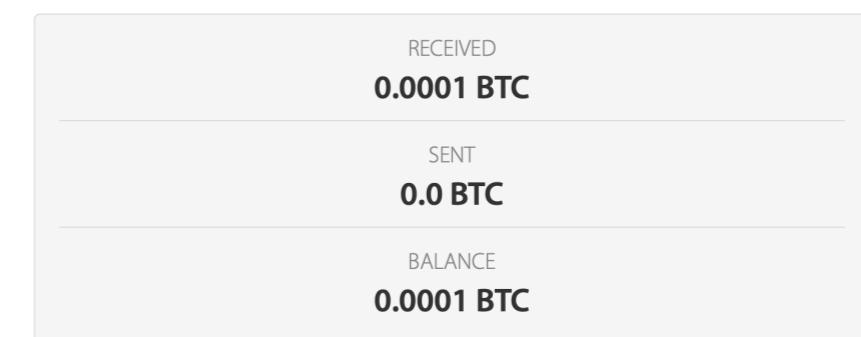


The screenshot shows the BlockCypher website interface. At the top, there are dropdown menus for 'BTC' and 'Address, transact' along with a search bar. Below the header, the text 'Bitcoin Address' is displayed next to the address '13hA6HP6W4BCwmtr89CKh6fTra19Exm4bL'. The background features a stylized 3D network graph.



The screenshot shows the transaction history for Bob's address. It includes a summary table:

RECEIVED	0.00099016 BTC
SENT	0.0 BTC
BALANCE	0.00053516 BTC (-0.000455 BTC UNCONFIRMED)

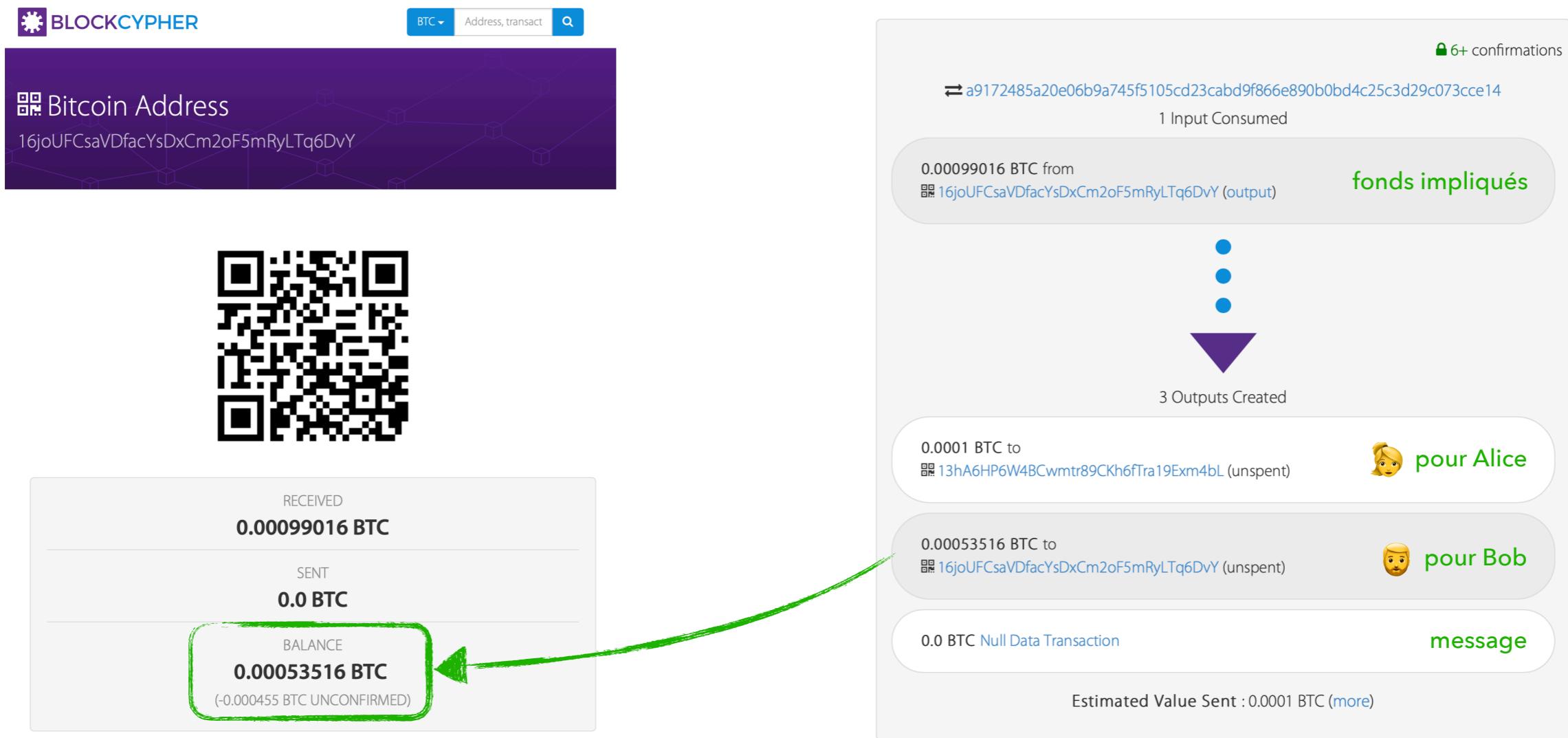


The screenshot shows the transaction history for Alice's address. It includes a summary table:

RECEIVED	0.0001 BTC
SENT	0.0 BTC
BALANCE	0.0001 BTC

> Réaliser un Transfert Bitcoin > Créer une clé privée > Créer une clé publique > Créer une adresse

- ▶ Si on regarde d'un peu plus près les informations fournis par BlockCypher concernant l'adresse de Bob, on peut remarquer deux choses plutôt curieuses au premier abords :
 - la différence entre le total des inputs et des outputs (0.000355 BTC) → **Frais de minage**.
 - Bob se renvoie à lui même une partie des fonds impliqués dans la transaction → **Tous les fonds impliqués doivent être dépensés.**



The screenshot illustrates a Bitcoin transaction on the BlockCypher platform. On the left, a balance summary shows:

- RECEIVED: 0.00099016 BTC
- SENT: 0.0 BTC
- BALANCE: 0.00053516 BTC (-0.000455 BTC UNCONFIRMED)

A green arrow points from the "BALANCE" section to the transaction details on the right. The transaction details show:

- Input: a9172485a20e06b9a745f5105cd23cabd9f866e890b0bd4c25c3d29c073cce14 (1 Input Consumed)
- Output 1: 0.00099016 BTC to 16joUFCsaVDfacYsDxCm2oF5mRyLTq6DvY (output) (pour Alice)
- Output 2: 0.00053516 BTC to 16joUFCsaVDfacYsDxCm2oF5mRyLTq6DvY (unspent) (pour Bob)
- Output 3: 0.0 BTC Null Data Transaction (message)

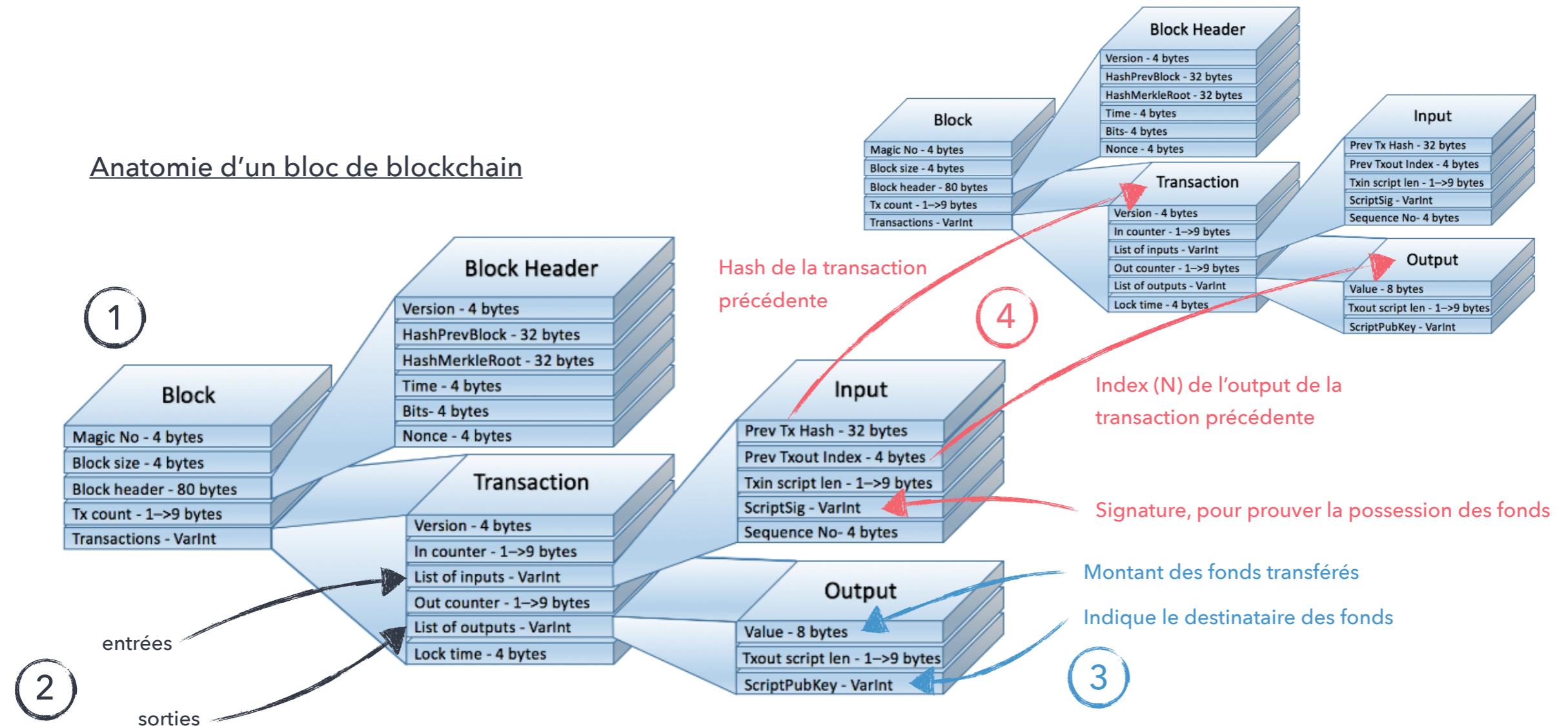
Annotations highlight specific parts of the transaction details:

- "fonds impliqués" (involved funds) is written next to the input amount.
- "pour Alice" and "pour Bob" are written next to their respective outputs.
- "message" is written next to the null data transaction output.

> Réaliser un Transfert Bitcoin > Construire une transaction Bitcoin

- ▶ Pour envoyer des fonds de Bob vers Alice il va donc nous falloir construire une transaction et la soumettre au réseau Bitcoin.
- ▶ Les transactions, sont au coeur du système Bitcoin, elles contiennent les informations relatives aux transferts de valeur entre les participants du réseau.

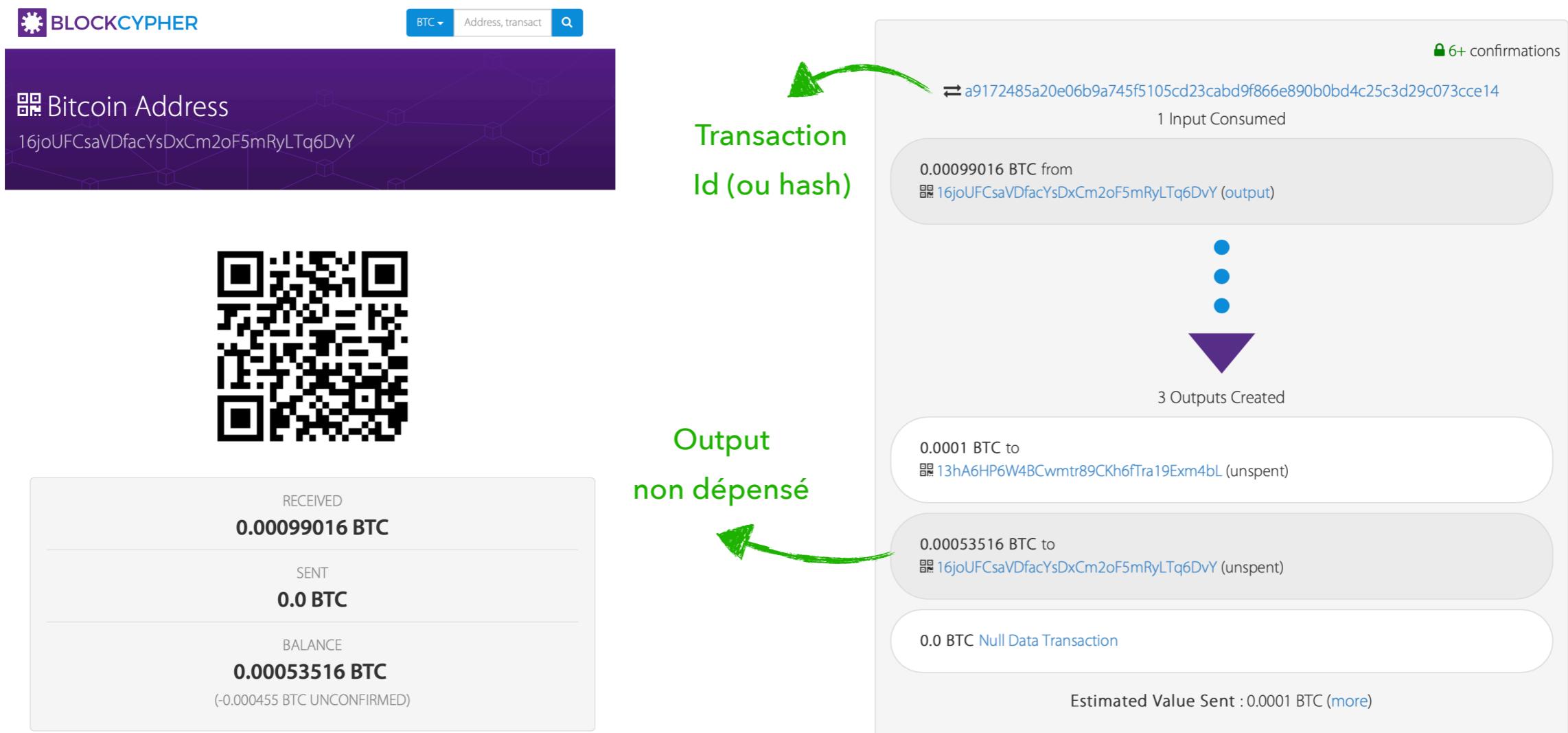
Anatomie d'un bloc de blockchain



- ▶ Pour envoyer des fonds à Alice, il va donc falloir faire référence à l'output d'une transaction qui a transféré des fonds initiaux à Bob et que ce dernier n'a pas dépensé ...

> Réaliser un Transfert Bitcoin > Construire une transaction Bitcoin

- En consultant la dernière transaction liées à l'adresse de Bob, on retrouve rapidement cette information :



The screenshot shows two main parts of the BlockCypher interface:

- Left Side (Transaction Details):** A QR code for the address **16joUFCsaVdfacYsDxCm2oF5mRyLTq6DvY**. Below it is a table with the following data:

RECEIVED	0.00099016 BTC
SENT	0.0 BTC
BALANCE	0.00053516 BTC (-0.000455 BTC UNCONFIRMED)
- Right Side (Transaction History):** A list of transactions for the same address:
 - Transaction Id (ou hash):** [a9172485a20e06b9a745f5105cd23cabd9f866e890b0bd4c25c3d29c073cce14](#) (6+ confirmations).
 - 1 Input Consumed
 - 0.00099016 BTC from [16joUFCsaVdfacYsDxCm2oF5mRyLTq6DvY](#) (output)
 - Output non dépensé:** [13hA6HP6W4BCwmtr89CKh6fTra19Exm4bL](#) (unspent) - 0.0001 BTC
 - [16joUFCsaVdfacYsDxCm2oF5mRyLTq6DvY](#) (unspent) - 0.00053516 BTC
 - 0.0 BTC Null Data Transaction

Annotations with arrows highlight the **Transaction Id** and the **Output non dépensé**.

- Notre nouvelle transaction va donc « s'accrocher » à l'output non dépensé de celle-ci !
- QBitNinja va nous permettre de récupérer ces informations afin de les manipuler dans notre application console.

> Réaliser un Transfert Bitcoin > Construire une transaction Bitcoin

- ▶ Récupérer une transaction stockée sur la blockchain avec le client QBitNinja :

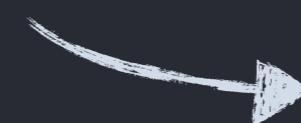
```
var txId = uint256.Parse("a9172485a20e06b9a745f5105cd23cabd9f866e890b0bd4c25c3d29c073cce14");

var client = new QBitNinjaClient(Network.Main);
var transactionResponse = client.GetTransaction(txId).Result;

Console.WriteLine($"Transaction ID: { transactionResponse.TransactionId }");
Console.WriteLine($"Transaction Confirmation(s): { transactionResponse.Block Confirmations } »);

// Output :

Transaction ID: a9172485a20e06b9a745f5105cd23cabd9f866e890b0bd4c25c3d29c073cce14
Transaction Confirmation(s) : 97
```



Hash ou Id de la transaction

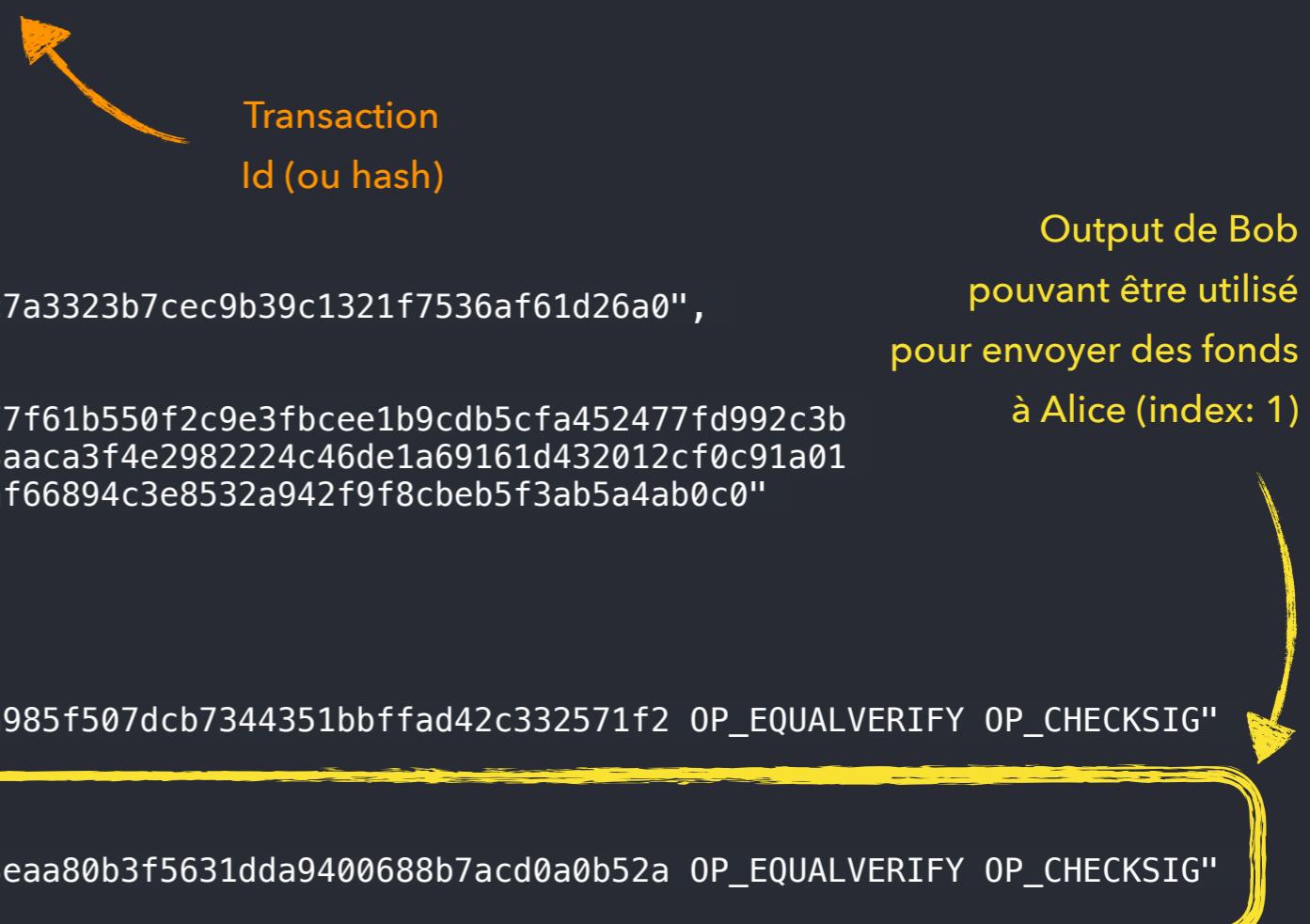
- ▶ Affichons l'intégralité de la transaction :

```
Console.WriteLine( transactionResponse.Transaction );
```

> Réaliser un Transfert Bitcoin > Construire une transaction Bitcoin

- ▶ Récupérer une transaction stockée sur la blockchain avec le client QBitNinja :

```
{  
  "hash": "a9172485a20e06b9a745f5105cd23cabd9f866e890b0bd4c25c3d29c073cce14",  
  "ver": 1,  
  "vin_sz": 1,  
  "vout_sz": 3,  
  "lock_time": 0,  
  "size": 248,  
  "in": [  
    {  
      "prev_out": {  
        "hash": "e1bc9d694c055e0ff32a105a33b2ac7a3323b7cec9b39c1321f7536af61d26a0",  
        "n": 1  
      },  
      "scriptSig": "304402200324b4e474f985c38f677f61b550f2c9e3fbcee1b9cdb5cfa452477fd992c3b  
d0220573662bcd393729719efda3aaca3f4e2982224c46de1a69161d432012cf0c91a01  
022f924ece773d140333652d3b5af66894c3e8532a942f9f8cbeb5f3ab5a4ab0c0"  
    }  
  ],  
  "out": [  
    {  
      "value": "0.00010000",  
      "scriptPubKey": "OP_DUP OP_HASH160 1d887a985f507dcb7344351bbffad42c332571f2 OP_EQUALVERIFY OP_CHECKSIG"  
    },  
    {  
      "value": "0.00053516",  
      "scriptPubKey": "OP_DUP OP_HASH160 3ef0e3eaa80b3f5631dda9400688b7acd0a0b52a OP_EQUALVERIFY OP_CHECKSIG"  
    },  
    {  
      "value": "0.00000000",  
      "scriptPubKey": "OP_RETURN 486920446f63646f6b752021"  
    }  
  ]  
}
```



The diagram includes several annotations:

- An orange arrow points from the text "Transaction Id (ou hash)" to the "hash" field in the JSON.
- A yellow bracket highlights the entire "out" array, which contains three output objects. A yellow arrow points from this bracket to the text "Output de Bob pouvant être utilisé pour envoyer des fonds à Alice (index: 1)".
- A yellow bracket highlights the second output object in the "out" array, which has its "scriptPubKey" field highlighted with a yellow underline.



> Réaliser un Transfert Bitcoin > Construire une transaction Bitcoin > Inputs

- ▶ Maintenant que l'ont sait quoi dépenser, on peut commencer à construire notre nouvelle transaction :

```
Transaction aliceFundingTransaction = Transaction.Create(Network.Main);
```

- ▶ Et ajouter un nouvel input à cette nouvelle transaction :

```
TxOut txOutToSpend = null;
OutPoint outPointToSpend = null;

foreach(var txOut in bobFundingTransaction.Outputs)
{
    if (txOut.ScriptPubKey == bobPK.ScriptPubKey)
    {
        outPointToSpend = new OutPoint(txId, bobFundingTransaction.Outputs.IndexOf(txOut));
        txOutToSpend = txOut;
    }
}

aliceFundingTransaction.Inputs.Add(outPointToSpend);
```

Parcours tous les outputs de la transaction qui a transmis les fonds initiaux à Bob

ScriptPubKey → Adresse

Si l'adresse pointée est celle de Bob alors on sélectionne cet OutPoint.

Outpoint :

TxID: a9172485a20e06b9a745f5105cd23cabd9f866e890b0bd4c25c3d29c073cce14
N: 1

> Réaliser un Transfert Bitcoin > Construire une transaction Bitcoin > Inputs

- ▶ Si on visualise notre transaction à ce stade, on obtient :

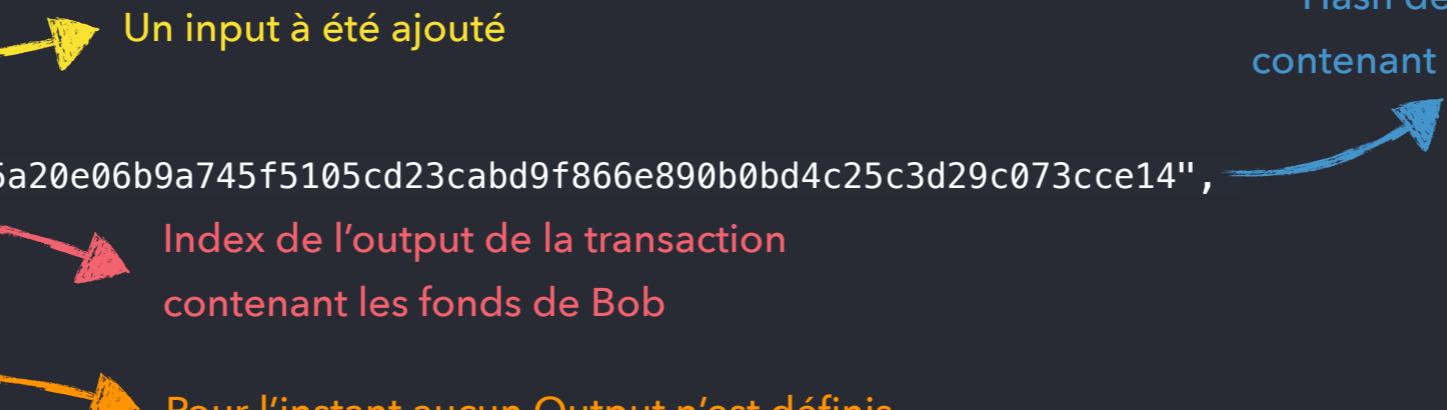
```
{  
    "hash": "8a6c71659df1f561eb072a36d698084b4c53f58f7a63e74046d12c609eb62137",  
    "ver": 1,  
    "vin_sz": 1,  
    "vout_sz": 0,  
    "lock_time": 0,  
    "size": 51,  
    "in": [  
        {  
            "prev_out": {  
                "hash": "a9172485a20e06b9a745f5105cd23cabd9f866e890b0bd4c25c3d29c073cce14",  
                "n": 1  
            },  
            "scriptSig": ""  
        }  
    ],  
    "out": []  
}
```

Un input a été ajouté

Index de l'output de la transaction
contenant les fonds de Bob

Pour l'instant aucun Output n'est définis

Hash de la transaction
contenant les fonds de Bob



- ▶ Voyons maintenant comment dépenser notre input ...

> Réaliser un Transfert Bitcoin > Construire une transaction Bitcoin > Répartition des fonds

- On calcule la répartition des fonds entre les différents Outputs de notre future transaction en respectant le fait que :

- tous les fonds en input de la transaction doivent être dépensés,
- la différence entre le montant total des inputs et des outputs sera reversé au mineur,
- des frais de minages trop faible entraînent un traitement plus long de la transaction (voire elle n'est jamais traitée),

```

var bobBalance = bobFundingTransaction.Outputs[outPointToSpend.N].Value.ToDecimal(MoneyUnit.BTC); → Montant à répartir
var minerFees = new Money(0.000355m, MoneyUnit.BTC); → Frais de minage
var aliceAmount = new Money(0.000100m, MoneyUnit.BTC); → Montant que Bob souhaite envoyer à Alice → 🧑 → 🧔
var bobChangeAmount = bobBalance - (aliceAmount + minerFees).ToString("0.00000000", CultureInfo.InvariantCulture); → Montant que Bob souhaite conserver → 🧑 → 🧑
    
```

- Pour calculer les frais de minages on peut directement chercher sur google :

Google

Date	Next Block Fee	3 Blocks Fee
2019-06-26	3.57 USD/tx	3.51 USD/tx
2019-06-25	2.34 USD/tx	2.20 USD/tx
2019-06-24	2.18 USD/tx	1.88 USD/tx
2019-06-23	1.74 USD/tx	1.44 USD/tx

- plus vous payez, plus vite est traitée votre transaction,
- il s'agit d'une moyenne, le prix est fixé en fonction du poids de votre transaction.
- si vous ajoutez un message, vous devrez augmenter les frais.
- pour calculer les frais adéquats :

<https://bitcoinfees.earn.com/> (cost per byte)

+

```
var txSize = aliceFundingTransaction.ToBytes();
```

```

TxOut aliceTxOut = new TxOut
{
    Value = aliceAmount,
    ScriptPubKey = alicePK.ScriptPubKey,
};

TxOut bobChangeBackTxOut = new TxOut
{
    Value = new Money(bobChangeBackAmount, MoneyUnit.BTC),
    ScriptPubKey = bobPK.ScriptPubKey,
};

aliceFundingTransaction.Outputs.Add(aliceTxOut);
aliceFundingTransaction.Outputs.Add(bobChangeBackTxOut);

```

Ajout des Outputs à la transaction



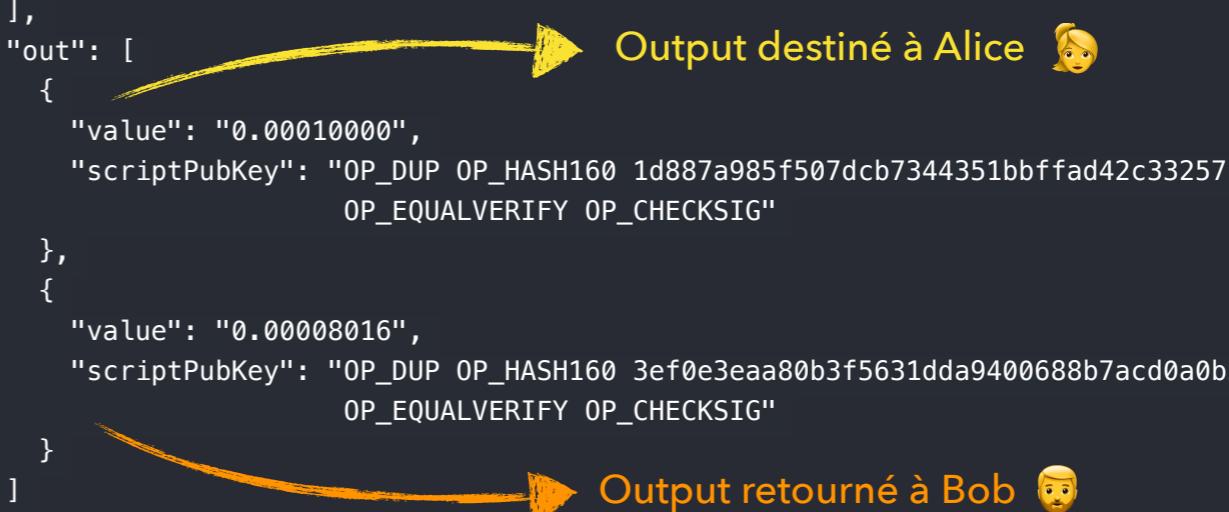
```

{
    "hash": "88f979bcb051e52762e5c1c34a753f439e39f61b4c924a4d4e8179671a80860b",
    "ver": 1,
    "vin_sz": 1,
    "vout_sz": 2,
    "lock_time": 0,
    "size": 119,
    "in": [
        {
            "prev_out": {
                "hash": "a9172485a20e06b9a745f5105cd23cabd9f866e890b0bd4c25c3d29c073cce14",
                "n": 1
            },
            "scriptSig": ""
        }
    ],
    "out": [
        {
            "value": "0.00010000",
            "scriptPubKey": "OP_DUP OP_HASH160 1d887a985f507dcb7344351bbffad42c332571f2
                           OP_EQUALVERIFY OP_CHECKSIG"
        },
        {
            "value": "0.00008016",
            "scriptPubKey": "OP_DUP OP_HASH160 3ef0e3eaa80b3f5631dda9400688b7acd0a0b52a
                           OP_EQUALVERIFY OP_CHECKSIG"
        }
    ]
}

```

Output destiné à Alice 

Output retourné à Bob 



```
{
  "hash": "acaa5c9350dcf4d5cd5af877a1fa01e1aea985d8418673fb0317a00c6a46956",
  "ver": 1,
  "vin_sz": 1,
  "vout_sz": 3,
  "lock_time": 0,
  "size": 142,
  "in": [
    {
      "prev_out": {
        "hash": "a9172485a20e06b9a745f5105cd23cabd9f
          866e890b0bd4c25c3d29c073cce14",
        "n": 1
      },
      "scriptSig": ""
    }
  ],
  "out": [
    {
      "value": "0.00010000",
      "scriptPubKey": "OP_DUP OP_HASH160
        1d887a985f507dcb7344351bbffad42c332571f2
        OP_EQUALVERIFY OP_CHECKSIG"
    },
    {
      "value": "0.00008016",
      "scriptPubKey": "OP_DUP OP_HASH160
        3ef0e3eaa80b3f5631dda9400688b7acd0a0b52a
        OP_EQUALVERIFY OP_CHECKSIG"
    },
    {
      "value": "0.00000000",
      "scriptPubKey": "OP_RETURN 486920446f63646f6b752021"
    }
  ]
}
```



Hi Docdoku !

Output contenant le message

- ▶ On peut donc stocker une information sans forcément transférer de la valeur, du moment que les frais de minage sont couverts !
- ▶ Ce message est inaltérable et incensurable
- ▶ Vous payez pour écrire ce message, mais sa lecture est « gratuite », par n'importe qui ou n'importe quoi ayant accès au réseau Bitcoin.

> Réaliser un Transfert Bitcoin > Construire une transaction Bitcoin > Signer la transaction

- Avant de pouvoir diffuser votre transaction, vous devez d'abords la signer à l'aide de votre clé privée :

```
ICoin coinToSign = new Coin(outPointToSpend, txOutToSpend);  
  
aliceFundingTransaction.Sign(bobPK, coinToSign);
```

Output de la transaction contenant
les fonds d'Input de notre transaction

Signature

Clé privée de Bob

AVANT

```
{  
    "hash": "acaa5c9350dcf4d5cd5af877a1fa  
          01e1aea985d8418673fb0317a00c6a46956",  
    "ver": 1,  
    "vin_sz": 1,  
    "vout_sz": 3,  
    "lock_time": 0,  
    "size": 142,  
    "in": [  
        {  
            "prev_out": {  
                "hash": "a9172485a20e06b9a745f5105cd23cabd9f  
                  866e890b0bd4c25c3d29c073cce14",  
                "n": 1  
            },  
            "scriptSig": ""  
        }  
    ],  
    "out": [  
        ...  
    ]  
}
```

Pas de signature

APRES

```
{  
    "hash": "acaa5c9350dcf4d5cd5af877a1fa  
          01e1aea985d8418673fb0317a00c6a46956",  
    "ver": 1,  
    "vin_sz": 1,  
    "vout_sz": 3,  
    "lock_time": 0,  
    "size": 142,  
    "in": [  
        {  
            "prev_out": {  
                "hash": "a9172485a20e06b9a745f5105cd23cabd9f  
                  866e890b0bd4c25c3d29c073cce14",  
                "n": 1  
            },  
            "scriptSig": "304402200324b4e474f985c38f677f61b550f2c9e3fbcee1b9c  
              db5cf452477fd992c3bd0220573662bcd393729719efda3aac  
              a3f4e2982224c46de1a69161d432012cf0c91a01"  
        }  
    ],  
    "out": [  
        ...  
    ]  
}
```

Signature attestant que vous êtes
bien le propriétaire des fonds

> Réaliser un Transfert Bitcoin > Diffuser une transaction sur le réseau Bitcoin

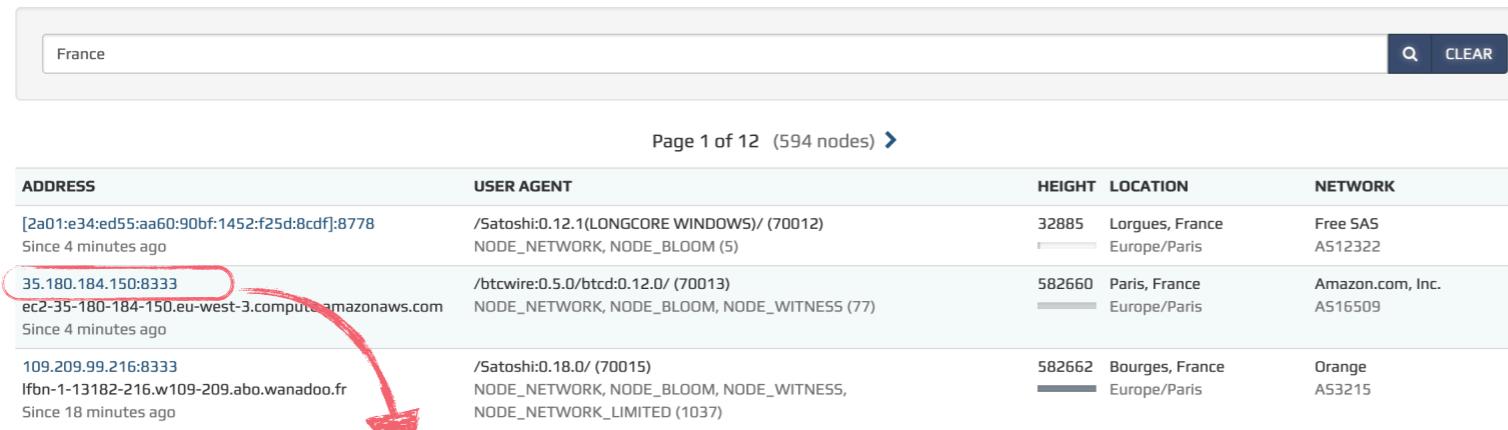
- ▶ Il est désormais temps de diffuser notre transaction au réseau !
- ▶ Pour cela, il faut la transmettre à un noeuds du réseau,
- ▶ Plusieurs options s'offre à vous :

- Héberger votre propre noeuds et lui soumettre la transaction,
- Vous connecter à un noeuds existant (<https://bitnodes.earn.com/>),
- Copier le code Hexadécimal de votre transaction et le pusher via un explorateur de blocks tel que BlockCypher.
- Passer par l'API QBitNinja !

BITNODES

Bitnodes is currently being developed to estimate the size of the Bitcoin network by finding all the reachable nodes in the network.

SUPPORTED BY [EARN.COM](#)



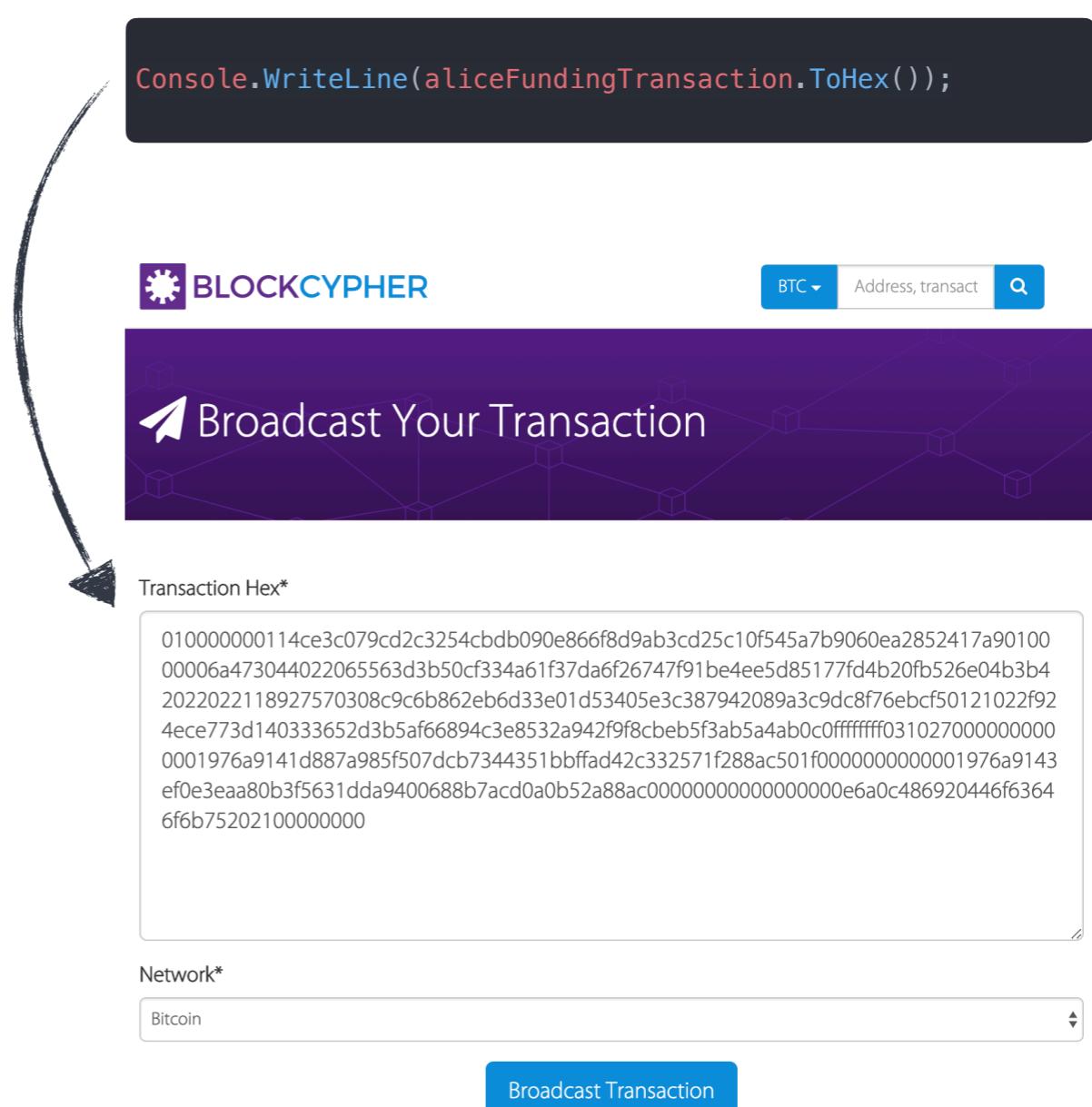
ADDRESS	USER AGENT	HEIGHT	LOCATION	NETWORK
[2a01:e34:ed55:aa60:90bf:1452:f25d:8cdf]:8778 Since 4 minutes ago	/Satoshi:0.12.1(LONGCORE WINDOWS)/ (70012) NODE_NETWORK, NODE_BLOOM (5)	32885	Lorgues, France Europe/Paris	Free SAS AS12322
35.180.184.150:8333 ec2-35-180-184-150.eu-west-3.compute.amazonaws.com Since 4 minutes ago	/btcwire:0.5.0/btcd:0.12.0/ (70013) NODE_NETWORK, NODE_BLOOM, NODE_WITNESS (77)	582660	Paris, France Europe/Paris	Amazon.com, Inc. AS16509
109.209.99.216:8333 Ifbn-1-13182-216.w109-209.abo.wanadoo.fr Since 18 minutes ago	/Satoshi:0.18.0/ (70015) NODE_NETWORK, NODE_BLOOM, NODE_WITNESS, NODE_NETWORK_LIMITED (1037)	582662	Bourges, France Europe/Paris	Orange AS3215

Adresse IP + Port du noeuds

> Réaliser un Transfert Bitcoin > Diffuser une transaction sur le réseau Bitcoin

- ▶ Il est désormais temps de diffuser notre transaction au réseau !
- ▶ Pour cela, il faut la transmettre à un noeuds du réseau,
- ▶ Plusieurs options s'offre à vous :

- Héberger votre propre noeuds et lui soumettre la transaction,
- Vous connecter à un noeuds existant (<https://bitnodes.earn.com/>),
- Copier le code Hexadécimal de votre transaction et le pusher via un explorateur de blocks tel que BlockCypher.
- Passer par l'API QBitNinja !



The screenshot shows the BlockCypher website's "Broadcast Your Transaction" page. At the top, there is a code block containing the C# code: `Console.WriteLine(aliceFundingTransaction.ToHex());`. Below this, the BlockCypher logo is visible. The main area features a purple background with a network graph and the text "Broadcast Your Transaction". A large text input field labeled "Transaction Hex*" contains the long hexidecimal string of the transaction. Below the input field, a dropdown menu labeled "Network*" is set to "Bitcoin". At the bottom right, a blue button labeled "Broadcast Transaction" is visible.

> Réaliser un Transfert Bitcoin > Diffuser une transaction sur le réseau Bitcoin

- ▶ Il est désormais temps de diffuser notre transaction au réseau !
- ▶ Pour cela, il faut la transmettre à un noeuds du réseau,
- ▶ Plusieurs options s'offre à vous :
 - Héberger votre propre noeuds et lui soumettre la transaction,
 - Vous connecter à un noeuds existant (<https://bitnodes.earn.com/>),
 - Copier le code Hexadécimal de votre transaction et le pusher via un explorateur de blocks tel que BlockCypher.
 - Passer par l'API QBitNinja !

```
BroadcastResponse broadcastResponse = client.Broadcast(aliceFundingTransaction).Result;

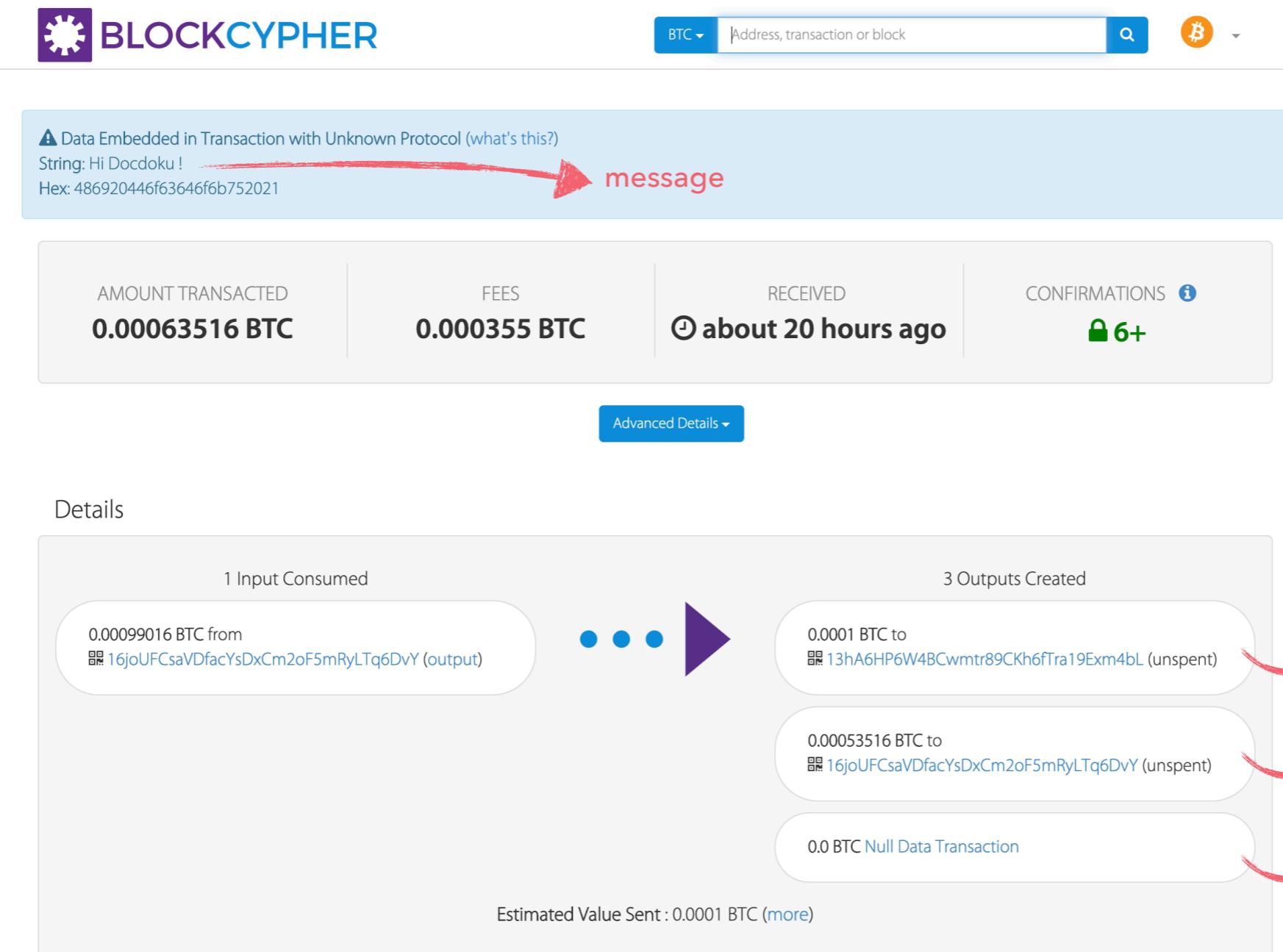
if (!broadcastResponse.Success)
{
    Console.Error.WriteLine("Error Code: " + broadcastResponse.Error.ErrorCode);
    Console.Error.WriteLine("Error Message: " + broadcastResponse.Error.Reason);
}
else
{
    Console.WriteLine("Success ! You can check out the hash of the transaction in any block explorer :");
    Console.WriteLine($"> Tx ID : { aliceFundingTransaction.GetHash() }");
    Console.WriteLine($"> Tx HEX : { aliceFundingTransaction.ToHex() }");
}

// Output

Success ! You can check out the hash of the transaction in any block explorer :
> Tx ID : a9172485a20e06b9a745f5105cd23cabd9f866e890b0bd4c25c3d29c073cce14
> Tx HEX : 01000000114ce3c079cd2c3254cbdb090e866f8d9ab3cd25c10f545a7b9060ea2852417a9010000006a473044022065563d3b50cf334a61f37da6f26747f91be4e
e5d85177fd4b20fb526e04b3b42022022118927570308c9c6b862eb6d33e01d53405e3c387942089a3c9dc8f76ebcf50121022f924ece773d140333652d3b5af668
c3e8532a942f9f8cbeb5f3ab5a4ab0c0ffffffff0310270000000000001976a9141d887a985f507dc7344351bbffad42c332571f288ac501f0000000000001976a
9143ef0e943eaa80b3f5631dda9400688b7acd0a0b52a88ac0000000000000000e6a0c486920446f63646f6b7520210000000
```

> Réaliser un Transfert Bitcoin > Diffuser une transaction sur le réseau Bitcoin

- ▶ Pour consulter l'état de notre transaction diffusée, rendez-vous sur un explorateur de Block tel que BlockCypher :



The screenshot shows a transaction detail page from BlockCypher. At the top, there's a search bar with "BTC" selected and a message input field containing "Hi Docdoku!". Below the search bar, the transaction summary includes:

- AMOUNT TRANSACTED: 0.00063516 BTC
- FEES: 0.000355 BTC
- RECEIVED: about 20 hours ago
- CONFIRMATIONS: 6+

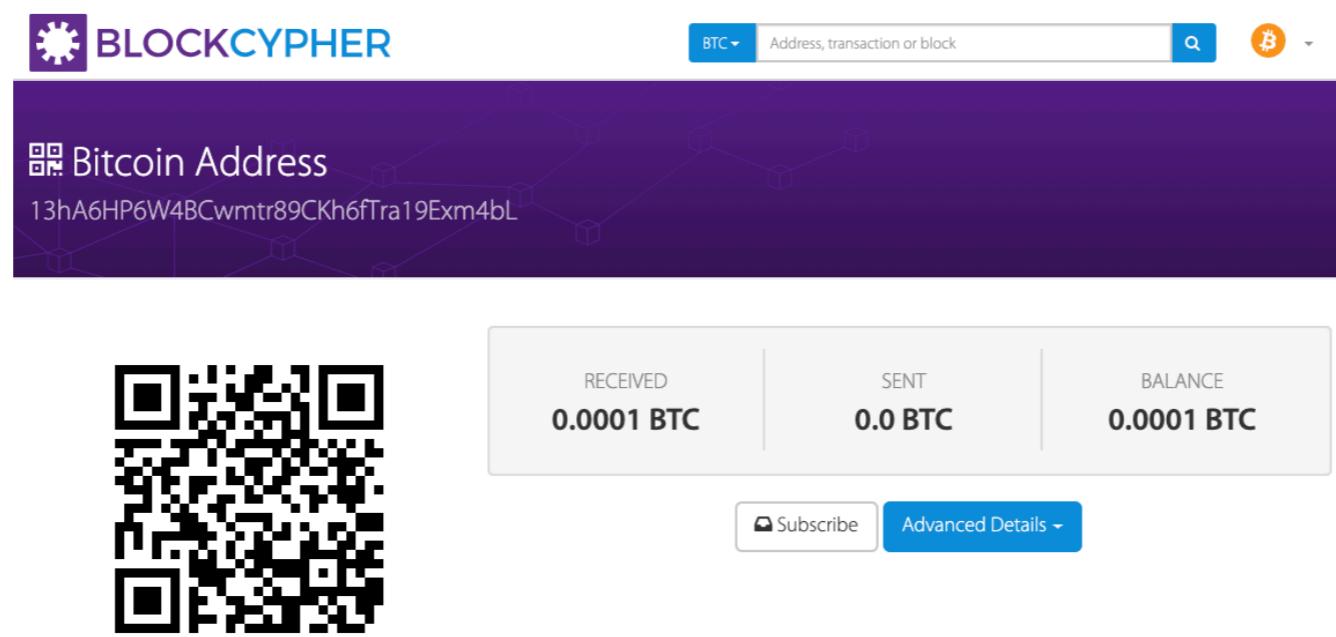
Below the summary, the "Details" section shows the transaction flow:

- 1 Input Consumed: 0.00099016 BTC from address 16joUFCsaVDfacYsDxCm2oF5mRyLTq6DvY (output).
- 3 Outputs Created:
 - 0.0001 BTC to address 13hA6HP6W4BCwmtr89CKh6fTra19Exm4bL (unspent) - labeled "Fonds reçus par Alice"
 - 0.00053516 BTC to address 16joUFCsaVDfacYsDxCm2oF5mRyLTq6DvY (unspent) - labeled "Bob change back"
 - 0.0 BTC Null Data Transaction - labeled "message"

At the bottom, it says "Estimated Value Sent : 0.0001 BTC (more)".

> Réaliser un Transfert Bitcoin > Diffuser une transaction sur le réseau Bitcoin

- ▶ Si on consulte l'adresse d'Alice :



The screenshot shows a BlockCypher interface for the Bitcoin address `13hA6HP6W4BCwmtr89CKh6fTra19Exm4bL`. The address is displayed prominently at the top left. Below it is a QR code. In the center, there's a summary box with the following details:

RECEIVED 0.0001 BTC	SENT 0.0 BTC	BALANCE 0.0001 BTC
-------------------------------	------------------------	------------------------------

Below this summary is a "Subscribe" button and an "Advanced Details" button. To the right of the summary box, a red arrow points downwards with the text "Solde mis à jour" (Balance updated).

At the bottom, a large box displays a single transaction. The transaction has 1 Input Consumed and 3 Outputs Created. The inputs and outputs are as follows:

- Input Consumed: 0.00099016 BTC from [16joUFCSavDfacYsDxCm2oF5mRyLTq6DvY](#) (output)
- Outputs Created:
 - 0.0001 BTC to [13hA6HP6W4BCwmtr89CKh6fTra19Exm4bL](#) (unspent)
 - 0.00053516 BTC to [16joUFCSavDfacYsDxCm2oF5mRyLTq6DvY](#) (unspent)
 - 0.0 BTC Null Data Transaction

A red arrow points from the "Transaction" text to the transaction box.

1 Transaction

6+ confirmations

a9172485a20e06b9a745f5105cd23cabd9f866e890b0bd4c25c3d29c073cce14

1 Input Consumed

0.00099016 BTC from [16joUFCSavDfacYsDxCm2oF5mRyLTq6DvY](#) (output)

3 Outputs Created

0.0001 BTC to [13hA6HP6W4BCwmtr89CKh6fTra19Exm4bL](#) (unspent)

0.00053516 BTC to [16joUFCSavDfacYsDxCm2oF5mRyLTq6DvY](#) (unspent)

0.0 BTC Null Data Transaction

Estimated Value Sent : 0.0001 BTC ([more](#))

Solde mis à jour

Transaction

> Conclusion & Perspectives

- ▶ L'application première du réseau Bitcoin est l'**échange de la cryptomonnaie** du même nom,
- ▶ D'ailleurs sur ce réseau **tout se paye en Bitcoins**,
- ▶ Mais il peut également servir à **autre chose** (et NBitcoin supporte tout cela):
 - stocker des données de façon **indélébile et non-censurable**,
 - **émettre et détruire** votre propre « **tokens** » pour représenter les parts d'une entreprise, des actions ou des votes,
 - attacher un « **contrat ricardien** » à un token (une sorte de smart contract lisible par l'homme et destiné au monde **juridique**)
- ▶ Le protocole est en perpétuelle évolution et vise à devenir de plus en plus efficient (plus rapide, moins coûteux),
- ▶ Si aujourd'hui la **lenteur** et les **frais de transaction élevés** limite l'usage du réseaux, des mises à jour comme le « **Lightning Network** » ont le potentiel de changer la donne et **maintenir Bitcoin sur le devant de la scène**.
 - Si vous avez des questions ou des idées d'applications, n'hésitez pas ! -