Elements RPC

@DG Lab Nakamura

Agenda

- 準備
- Confidential Transactions を使ってみる。
- Confidential Assets を使ってみる。

準備

準備

elementsのノード2つ立ち上げます。

1:データフォルダと設定ファイルの作成

2:aliasの設定

3:デーモンの起動

準備　1:データフォルダと設定ファイル(1/2)

このセッション用のデータフォルダを作成します。

```
$ mkdir ~/bc2_elementsdir1
$ mkdir ~/bc2_elementsdir2
```

設定ファイルのあるフォルダに移動します。(1行で入力します)

```
$ cd
~/bc2/elements/conference/rpc_tutorial
```

準備　1:データフォルダと設定ファイル(2/2)

このセッション用の設定ファイルを配置します。
(コピー先のファイル名は**数字がつきません！**)

```
$ cp ./elements1.conf
~/bc2_elementsdir1/elements.conf
$ cp ./elements2.conf
~/bc2_elementsdir2/elements.conf
```

準備　2:aliasの設定(1/3)

入力操作を簡単にするためaliasを設定します。

```
$ cd ~/bc2/elements/src
$ pwd
```

カレントディレクトリを確認してください。

準備　2:aliasの設定(2/3)

```
$ alias e1-cli="$PWD/elements-cli
-datadir=$HOME/bc2_elementsdir1"
$ alias e1-dae="$PWD/elementsd
-datadir=$HOME/bc2_elementsdir1"
```

準備　2:aliasの設定(3/3)

```
$ alias e2-cli="$PWD/elements-cli
-datadir=$HOME/bc2_elementsdir2"
$ alias e2-dae="$PWD/elementsd
-datadir=$HOME/bc2_elementsdir2"
```

以下、e1をalice、e2をbobと呼びます。

準備　3:デーモンの起動(1/12)

```
$ e1-dae
$ e2-dae
$ e1-cli getwalletinfo
error code: -28
error message:
Loading wallet...
```

準備　3:デーモンの起動(2/12)

→起動には時間がかかるため、最初はエラーになります。

しばらく待って再実行し初期状態をみてみましょう。
(20秒程度かかる場合があります。)

準備　3:デーモンの起動(3/12)

```
$ el-cli getwalletinfo
{
  "walletversion": 130000,
  "balance": {
  },
…
```

準備　3:デーモンの起動(4/12)

```
  "unconfirmed_balance": {
  },
  "immature_balance": {
    "maincoin": 21000000.00000000
  },
…
```

準備　3:デーモンの起動(5/12)

```
  "unconf
  },
  "immature_balance": {
    "maincoin": 21000000.00000000
  },
...
```

regtestネットワークでは、最初は全ての資金が"immature balance" に見えます。

準備　3:デーモンの起動(6/12)

```
$ el-cli generate 101
[
  "325fefeb......
  "485fcc8c.....
  "10cf5ab4.......27749472",
. . .
```

generateしてimmatureな状態を解消します。

準備　3:デーモンの起動(7/12)

```
$ e1-cli getwalletinfo
{
  "walletversion"
  "balance": {
    "maincoin": 21000000.00000000
. . .
```

maincoinはaliceが使える状態となりました。

準備　3:デーモンの起動(8/12)

```
$ e2-cli getwalletinfo
{
   "walletversion"
   "balance": {
      "maincoin": 21000000.00000000
. . .
```

maincoinはbobからも使える状態となります。

準備　3:デーモンの起動(9/12)

aliceに半分maincoinを引き渡します。

```
$ e1-cli sendtoaddress $(e1-cli
getnewaddress) 10500000 "" "" true
XXXXX...
$ e1-cli generate 101
. . .
```

準備　3:デーモンの起動(10/12)

bobにも半分maincoinを引き渡します。

```
$ e2-cli sendtoaddress $(e2-cli
getnewaddress) 10500000 "" "" true
XXXXX...
$ e2-cli generate 101
. . .
```

準備　3:デーモンの起動(11/12)

```
$ e1-cli getbalance
{
   "maincoin": 10500000.00000000
}
```

mainoinの半分がaliceに分配されました。

準備　3:デーモンの起動(12/12)

```
$ e2-cli getbalance
{
  "maincoin": 10500000.00000000
}
```

残り半分はbobに分配されました。これで準備は終わりです。

Confidential Transactions

Confidential Transactions (1/20)

bob(e2)が自分自身にElementsの(ブラインド)アドレスを使って送金してみます。まず、受信用アドレスを生成します。

```
$ A0=$(e2-cli getnewaddress)
$ echo $A0
CTEvb3mDterNwwr...iY1ntMggcLcS
```

ブラインドする場合のアドレスは`CTE`で始まます。

Confidential Transactions (2/20)

ブラインドする場合のアドレスは`CTE`で始まります。

```
$ e2-cli validateaddress $A0
{
   "isvalid": true,
   "address": "CTEvb3m...cLcS",
   "scriptPubKey": "76a9...88ac",
....
```

Confidential Transactions (3/20)

```
...
  "address": "CTEvb3m...cLcS",
  "scriptPubKey": "76a9...88ac",
  "confidential_key": "0392...7b92",
  "unconfidential": "2dha2p...qkCv",
  "ismine": true,
...
```

Confidential Transactions (3/20)

addressが検証対象の
(入力した)アドレスです。

```
...
  "address": "CTEvb3m...cLcS",
  "scriptPubKey": "76a9...88ac",
  "confidential_key": "0392...7b92",
  "unconfidential": "2dha2p...qkCv",
  "ismine": true,
...
```

Confidential Transactions (3/20)

addressが検証対象の
(入力した)アドレスです。

```
...
  "address": "CTEvb3m...cLcS",
  "scriptPubKey": "76a9...88ac",
  "confidential_key": "0392...7b92",
  "unconfidential": "2dha2p...qkCv",
  "ismine": true
...
```

unconfidentialがブラインドしない場合のアドレス(2～)です。

Confidential Transactions (3/20)

addressが検証対象の
(入力した)アドレスです。

```
...
   "address": "CTEvb3m...cLcS",
   "scriptPubKey": "76a9...88ac",
   "confidential_key": "0392...7b92",
   "unconfidential": "2dha2p...qkCv",
```

confidential_keyが
CTのkeyです。

unconfidentialがブラインドしない場合のアドレス(2～)です。

Confidential Transactions (4/20)

rpcコマンドの詳細はヘルプがあります。

```
$ e2-cli help validateaddress
validateaddress "address"
Return information about the given
bitcoin address.
Arguments:
1. "address" (string, required)...
```

Confidential Transactions (4/20)

rpcコマンドの詳細はヘルプがあります。

```
$ e2-cli help validateaddress
validat
Return                                    he given
bitcoin address.
Arguments:
1. "address" (string, required)...
```

ただ不備もそこそこあります。

Confidential Transactions (5/20)

bobがこのアドレスに(bobからbobに)送金します。

```
$ TXID=$(e2-cli sendtoaddress $A0 1)
$ echo $TXID
52fdb9379fa8d628bad617383ad16...4159
$ e2-cli generate 1
[
...
```

Confidential Transactions (6/20)

bobのウォレットでブラインド情報をみてみましょう。

```
$ e2-cli gettransaction $TXID
{
   "amount": {
      "maincoin": 0.00000000
   },
   "fee": -0.00042720,
```

Confidential Transactions (7/20)

```
"details": [
  {
    "account": "",
    "address": "2dha2p...qkCv",
    "category": "send",
    "amount": -1.00000000,
...
```

Confidential Transactions (7/20)

```
"details": [
  {
    "account": ""
    "address": "2dha2p...qkCv",
    "category": "send",
    "amount": -1.00000000,
...
```

中身が見えているのでブラインドなしのアドレスで表示されます。

Confidential Transactions (8/20)

aliceのウォレットからは見えません。

```
$ e1-cli gettransaction $TXID
error code: -5
error message:
Invalid or non-wallet transaction id
```

Confidential Transactions (9/20)

aliceはトランザクションの公開情報は見えます。

```
$ e1-cli getrawtransaction $TXID 1
{
   "hex": "0200...",
   "txid": "52fdb9379fa8d628...4159",
   "hash": "0df6b158f7017e20...3f3d",
   "withash": "283433d7f4803...70ac",
```

Confidential Transactions (10/20)

```
"vout": [
   {
      "value-minimum": 0.00000001,
      "value-maximum": 11258999.068426
      "ct-exponent": 0,
      "ct-bits": 50,
      "amountcommitment": "095...84c",
```

Confidential Transactions (11/20)

```
"vout": [
  {
    "value-minimum": 0.00000001,
    "value-maximum": 11258999.068426
    "ct-exponent": 0,
    "ct-bits": 50,
    "amountcommitment": "095...84c",
```

値が正の範囲であることしかわかりません。

Confidential Transactions (12/20)

値が正の範囲であることしかわかりません。

```
{
  "value-minimum": 0.00000001,
  "value-maximum": 42.94967296,
  "ct-exponent": 0,
  "ct-bits": 32,
  "amountcommitment": "08c...0dd",
```

Confidential Transactions (13/20)

feeは、値がわかります。

```
{
      "value": 0.00042720,
      "asset": "09f663de96be7...6621",
      "n": 2,
      "scriptPubKey": {
…
         "type": "fee"
```

Confidential Transactions (14/20)

aliceに送信先アドレスの秘密鍵をインポートしましょう。

```
$ PR0=$(e2-cli dumpprivkey $A0)
$ echo $PR0
cQ...
$ e1-cli importprivkey $PR0
```

Confidential Transactions (15/20)

aliceのウォレットからも見えるようになります。

```
$ e1-cli gettransaction $TXID
{
...
  "details": [
  ],
```

しかしdetailsは空で、額などがわかりません。

Confidential Transactions (16/20)

```
$ e1-cli getbalance
{
  "maincoin": 10500000.00000000
}
$ e1-cli listunspent 1 1
[
]
```

Confidential Transactions (16/20)

```
$ e1-cli getbalance
{
  "maincoin": 10500000.00000000
}
$ e1-cli listunspent 1 1
[
]
```

額が不明なので残高やUTXOに反映されません。

Confidential Transactions (17/20)

aliceにブラインディングキーもインポートしましょう。

```
$ B0=$(e2-cli dumpblindingkey $A0)
$ echo $B0
a...
$ e1-cli importblindingkey $A0 $B0
```

Confidential Transactions (18/20)

これでaliceからもトランザクションの額が見えるようになりました。

```
$ e1-cli getbalance
{
  "maincoin": 10500001.00000000
}
```

Confidential Transactions (19/20)

```
$ e1-cli listunspent 1 1
[
  {
    "txid": "52fdb9379fa8d6...4159",
    "vout": 1,
    "address": "2dha2p...qkCv",
...
```

Confidential Transactions (20/20)

```
...
    "amount": 1.00000000,
    "asset": "09f663de96be7...6621",
    "assetcommitment": "0a4...314a",
    "assetlabel": "maincoin",
...
```

これでConfidential Transactionsの章は終わりです。

Confidential Assets

Confidential Assets (1/24)

RPCコマンドにassetの引数や結果が追加されています。

```
$ elements-cli help getwalletinfo
getwalletinfo ( assetlabel )
Returns an object containing various wallet
state info.
1. "assetlabel" (string, optional) Hex asset
id or asset label for balance. "*" retrieves
all known asset balances.
```

Confidential Assets (1/24)

RPCコマンドにassetの引数や結果が追加されています。

```
$ e1-cli help getwalletinfo
getwalletinfo ( assetlabel )
Returns an object containing various wallet
                                    onal) Hex asset
id or asset label for balance. "*" retrieves
all known asset balances.
```

“*”で全てのassetを指定する構文は廃止されました。これはその残骸です。

Confidential Assets (2/24)

指定しなければ全てのassetを出力します。

```
$ el-cli getwalletinfo
{
  "walletversion": 130000,
  "balance": {
    "maincoin": 10500001.00000000
...
```

Confidential Assets (2/24)

指定しなければ全てのassetを出力します。

```
$ e1-cli getwalletinfo
{
   "walletversi
   "balance": {
      "maincoin": 10500001.00000000
...
```

maincoinも一つのassetとして扱われます。

Confidential Assets (3/24)

assetは16進数のIDで識別されます。

```
$ el-cli dumpassetlabels
{
"maincoin":"09f663de96be771f50cab5ded
00256ffe63773e2eaa9a604092951cc3d7c66
21"
}
```

Confidential Assets (3/24)

assetは16進数のIDで識別されます。

```
$ el-cli dumpasset
{
"maincoin":"09f663de96be771f50cab5ded
00256ffe63773e2eaa9a604092951cc3d7c66
21"
}
```

“maincoin”の部分は、
ローカルなラベルです。

Confidential Assets (4/24)

assetの指定はIDでもラベルでも可能です。
[ラベルを指定した場合]

```
$ e1-cli getwalletinfo maincoin
{
   "walletversion": 130000,
   "balance": 10500001.00000000,
   "unconfirmed_balance": 0.00000000,
```

Confidential Assets (5/24)

[IDを指定した場合]

```
$ e1-cli getwalletinfo
09f663de96be771f50cab5ded00256ffe6377
3e2eaa9a604092951cc3d7c6621
{
  "walletversion": 130000,
  "balance": 10500001.00000000,
```

Confidential Assets (6/24)

独自のアセットも発行できます。aliceがassetを1、再発行トークンを1として発行してみます。

```
$ ISSUE=$(e1-cli issueasset 1 1)
$ ASSET=$(echo $ISSUE | jq '.asset' | tr -d '"')
$ echo $ASSET
```

1b4e...75ce

Confidential Assets (6/24)

独自のアセットも発行できます。aliceがassetを1、再発行トークンを1として発行してみます。

```
$ ISSUE=$(e1-cli issueasset 1 1)
$ ASSET=$(echo $ISSUE | jq '.asset' | tr -d '"')
$ echo $ASSET
1b4e...75ce
```

うまくいかない場合は echo $ISSUE してassetの値をコピーしてASSETに設定しましょう。

Confidential Assets (7/24)

発行したアセットはaliceのウォレットで見えます。

```
$ el-cli listissuances
[
  {
      "isreissuance": false,
      "token": "121d18df414676...92a",
      "tokenamount": 1.00000000,
      "tokenblinds": "6f730c3...3d95",
```

Confidential Assets (8/24)

アセットの発行時に、再発行トークンに正の値を指定しておくと、元のアセットを再発行できます。

```
$ RI=$(e1-cli reissueasset $ASSET 1)
$ echo $RI
{ "txid": "b0ef48...227b",  "vin": 1 }
```

Confidential Assets (9/24)

再発行トークンのみやブラインドしないassetも発行できます。

```
$ el-cli issueasset 0 1 false
{
   "txid": "713f5d0aa0945fdc83...8fda",
…
   "asset": "5a8cfd0d6dec88e26...4abb",
   "token": "8f27476d6fe9b4c81...442b"
}
```

Confidential Assets (10/24)

最初に発行したassetを指定すると2つの発行履歴が見えます。

```
$ el-cli listissuances $ASSET
[
  {
      "isreissuance": true,
...
      "txid": "b0ef4817992d...1d0f227b",
...
      "assetamount": 1.00000000,
```

Confidential Assets (11/24)

```
  },
  {
      "isreissuance": false,
...
      "txid": "9015f2583c6b...14b724ae",
...
      "asset": "38f9cf2e5ba...6328b36a",
      "assetamount": 1.00000000,
...
```

Confidential Assets (12/24)

ここまでのコマンドのヘルプを確認しましょう。

```
$ el-cli help dumpassetlabels
dumpassetlabels
Lists all known asset id/label pairs
in this wallet. This list can be
modified with `-assetdir`
configuration argument.
```

Confidential Assets (13/24)

```
$ el-cli help issueasset
issueasset assetamount tokenamount (
blind )
Create an asset. Must have funds in
wallet to do so. Returns asset hex id.
Arguments:
1. "assetamount" (numeric or string,
```

Confidential Assets (14/24)

```
$ elements-cli help reissueasset
reissueasset asset assetamount
Create more of an already issued
asset. Must have reissuance token in
wallet to do so. Reissuing does not
affect your reissuance token balance,
only asset.
```

Confidential Assets (15/24)

```
$ e1-cli help listissuances
listissuances ( asset )
List all issuances known to the wallet
for the given asset, or for all issued
assets if none provided.
Arguments:
1. "asset" (string, optional) The...
```

Confidential Assets (16/24)

aliceからbobにアセットを送ってみましょう。。

```
$ BADDR=$(e2-cli getnewaddress)
$ echo $BADDR
$ e1-cli sendtoaddress $BADDR 1 "" ""
false $ASSET
$ e1-cli generate 1
```

Confidential Assets (16/24)

aliceからbobにアセットを送ってみましょう。。

```
$ BADDR=$(e2-cli getnewaddress)
$ echo $BADDR
$ e1-cli sendtoaddress $BADDR 1 "" "" false $ASSET
$ e1-cli generate 1
```

6つめのパラメータがassetの指定です。

Confidential Assets (17/24)

assetを受け取ったことを確認します。

```
$ e2-cli getwalletinfo
{
   "walletversion": 130000,
   "balance": {
     "maincoin": 10499999.99957280,
     "1b4e...75ce": 1.00000000
```

Confidential Assets (18/24)

assetを指定するとこんな感じになります。

```
$ e2-cli getwalletinfo $ASSET
{
  "walletversion": 130000,
  "balance": 1.00000000,
  "unconfirmed_balance": 0.00000000,
...
```

Confidential Assets (19/24)

bobには、いま受信したassetの発行履歴の額/再発行トークンは見えません。

```
$ e2-cli listissuances
[
]
```

Confidential Assets (19/24)

bobには、いま受信したassetの発行履歴の額/再発行トークンは見えません。

```
$ e2-cli listissuances
[
  {
    "isreissuance": true,
...
```

再発行分が見える場合あり。
ただしassetamount=-1

Confidential Assets (20/24)

asset発行に関連づいたアドレスをインポートします。

```
$ T1=$(echo $ISSUE | jq '.txid' | tr -d '"')
$ A1=$(e1-cli gettransaction $T1 | jq '.details[0].address' | tr -d '"')
$ e2-cli importaddress $A1
```

Confidential Assets (21/24)

asset発行履歴は見えますが発行量は不明(-1)のままです。

```
$ e2-cli listissuances
[
  {
...
    "tokenamount": -1,
...
    "assetamount": -1,
...
```

Confidential Assets (22/24)

isuuanceblindingkeyもインポートしましょう。

```
$ VIN=$(echo $ISSUE | jq '.vin' | tr -d '"')
$ ISK=$(e1-cli dumpissuanceblindingkey $T1 $VIN)
$ e2-cli importissuanceblindingkey $T1 $VIN $ISK
```

Confidential Assets (23/24)

```
$ e2-cli listissuances
[
  {
...
    "tokenamount": 1.00000000,
...
    "assetamount": 1.00000000,
...
```

Confidential Assets (23/24)

```
$ e2-cli listissuances
[
  {
...
      "tokenamount": 1.00000000,
...
      "assetamount": 1.00000000,
...
```

発行量も見えるようになりました。

Confidential Assets (24/24)

<宿題>
aliceが再発行したassetの発行量をbobから見えるようにしましょう。
<ヒント>
#20/24のスライドで、issueassetの出力から関連するトランザクション、アドレスを取出しました。

まとめ

- **a**sset関連のコマンド/オプションが増えている
- **b**lind関連のオプションも増えている
- **c**onfidentialなアドレスを使う

おまけ

・bitcoin 14からRPCコマンドでnamed形式が利用可能です。(位置ではなく名前で指定。)

例） `bitcoin-cli -named help command="generate"`

※elements側は、対応がまだ不十分です。

おまけ

daemonを停止しましょう。

```
$ el-cli stop
Elements server stopping
$ e2-cli stop
Elements server stopping
```

おまけ

復習するには、~/bc2/elementsディレクトリで

```
$./conference/rpc_tutorial/rpc_tutorial.sh
```

を実行してください。

参考資料

・APIリファレンス

https://github.com/ElementsProject/elementsbp-api-reference/blob/master/api.md

・Range Proof
https://blockstream.com/bitcoin17-final41.pdf#page=5

nakamura@dglab.com