Private Zcash payments on Ethereum

Ariel Gabizon, 

First we need to understand a little about how Zcash private payments work.

(In Ethereum meetup later, code demonstration of (simpler) atomic non-private trades between Zcash & Ethereum)

Recall Bitcoin's set of unspent transaction outputs.

(PK_1,2.3BTC), (PK_2, 0.4BTC),..

To spend money Alice signs a message with a secret key corresponding to a public address in an output.

For simplicity, assume each output/note is exactly 1 BTC.

Each node stores:

$Note_1=(PK_1)$
$Note_2=(PK_2)$
$Note_3=(PK_3)$

Now think of each note as containing a randomly picked ``serial number'' $\mathbf{r_i}$.

$\mathbf{Note_1=(PK_1,r_1)}$
$\mathbf{Note_2=(PK_2,r_2)}$
$\mathbf{Note_3=(PK_3,r_3)}$

For privacy..the node database will only contain *hashes* of the notes

$\mathbf{H_1}=\mathrm{HASH}(\mathbf{PK_1},\mathbf{r_1})$
$\mathbf{H_2}=\mathrm{HASH}(\mathbf{PK_2},\mathbf{r_2})$
$\mathbf{H_3}=\mathrm{HASH}(\mathbf{PK_3},\mathbf{r_3})$

For privacy, the node will continue to store $\mathbf{H_i}$ *even after* ***Note***$_i$ *has been spent*.

The node also stores a *nullifier set* that contains the hashes of all serial numbers of notes previously spent

$\mathbf{H_1}=\mathrm{HASH}(\mathbf{PK_1},\mathbf{r_1})$

$\mathbf{H_2}=\mathrm{HASH}(\mathbf{PK_2},\mathbf{r_2})$

$\mathbf{H_3}=\mathrm{HASH}(\mathbf{PK_3},\mathbf{r_3})$

Nullifier set after ***Note***$_2$ *has been spent:*

$\mathbf{nf_1}=\mathrm{HASH}(\mathbf{r_2})$

To spend a note, Alice sends a **ZK-proof** that she knows a secret key controlling a note s.t.

-It's hash is in the table

-The hash of its serial number is ***not*** in the nullifier set

$\mathbf{H_1}=\mathrm{HASH}(\mathbf{PK_1},\mathbf{r_1})$

$\mathbf{H_2}=\mathrm{HASH}(\mathbf{PK_2},\mathbf{r_2})$

$\mathbf{H_3}=\mathrm{HASH}(\mathbf{PK_3},\mathbf{r_3})$

Nullifier set after $\mathbf{Note_2}$ *has been spent:*

$\mathbf{nf_1}=\mathrm{HASH}(\mathbf{r_2})$

Zero-Knowledge proofs

Program **F(x,w)**, **x**-public input, **w**-hidden input

Given **x**, Alice can prove to Bob there exists **w** s.t.

F(x,w)=`acc`

*proof doesn't leak information about **w***

In our case:

***x* =** Table of note hashes, $\mathbf{nf_2}$

$\mathbf{w = sk_i, r_i}$

A protocol for private Zcash payments in Ethereum

Someone please implement!

To summarize, Zcash transaction consists of

- New note hash: **H=**HASH(**PK**,**r**)
- Old note nullifier: **nf=**HASH(**r`**)
- zk-proof: π

Private Zcash payments in Ethereum

1. Zcash recipient chooses **Note**=(**PK**,**r**) they would like to receive. Let **H**=HASH(**PK**,**r**).
2. Zcash sender commits in to **nf** Zcash transaction will use.
3. Ethereum contract logic: If legitimate Zcash transaction (**H,nf,**π) submitted then invoke desired contract method.
4. Double spend prevention: Have timelock. If second distinct Zcash transaction (**H',nf,**π') submitted before timelock lifted, invokation is canceled.

..but are there good zk-SNARK constructions?

There are since the breakthrough work of Gennaro,Gentry,Parno and Raykova

“Quadratic Span Programs and Succinct NIZKs without PCPs”

simple (and wrong) SNARK examples

1.want to prove know a,b with a+b=7 mod p

g – generator of group of order p where DL is hard.

Prover: send $A=g^a$, $B= g^b$

Verifier: Check that

$A*B=g^{a+b} = g^7$

2. Prove we know a,b,c with (a+b)*c = 7 mod p

Need: Bi-linear pairings:

Map $e:G\times G \rightarrow G_T$ such that $e(g^a,g^b)=g_T^{a*b}$

(Exists for some elliptic curve groups)

more detailed SNARK example, leading to QAPs

Prover: Send $A=g^a$, $B=g^b$, $C=g^c$

Verifier: Check that

$e(A*B,C) = (g_T)^7$

$e(A*B,C) = e(g^{a+b},g^c) = g_T^{(a+b)c}$

3. Prove you know a,b,c,d with $(a+b)*bc = 7 \bmod p$

Label multiplication gates: