UCSB CS 291D: Blockchains and Cryptocurrencies **Fall 2020**

Lecture 14: ZCash

Lecturer: Shumo Chu *Scribes: Aarti Jivrajani*

In this class, we are going to take a look at ZCash which is an application of zk-SNARKs.

Bitcoin tries to give a solution to the problem of decentralized payment. But Bitcoin has a privacy issue - unlike conventional payment systems where the transactions are private. Bitcoin is called *pseudo-anonymous* but that is not synonymous to privacy. It is possible to link the actual identity of the user to the public key using transactional graph analysis and machine learning techniques.

14.1 Predecessors of ZCash

14.1.1 Mixer

- It is a kind of centralized party. Each mixer has a *pool* into which money can be deposited and from which money can be withdrawn.
- Money can be deposited and withdrawn multiple times. And this amount need not be the same.
- Let the deposit amount be v_1, v_2 and v_3 and withdrawal amount be v'_1, v'_2 and v'_3. Mixer expects the following property to satisfy:
$$v_1 + v_2 + v_3 = v'_1 + v'_2 + v'_3$$
- For each v_i and v'_i, different addresses can be used. Let these addresses be $PK_1, PK_2, PK_3....$ and PK'_1, PK'_2, PK'_3
- From an adversary point of view, the mixer pool acts like a shield and the addresses (especially during periods of high volume) are not visible. At the least, tracking the transactions becomes very difficult in this case.

Drawbacks

- A centralized party - in this case the pool - needs to be trusted. This voids the point of using a cryptocurrency altogether.
- The pool also acts like a centralized target for attacks.
- A *time latency* needs to be introduced between transaction *in* and transaction *out.* This is to prevent the transactions from being decoded using the time information.

Consider a bitcoin called *basecoin.* In case of mixer, we get the basecoin *in* and basecoin *out.* In order to obfuscate the transaction graph, we need some kind of *processing bench.*

Herein lies the motivation for ZCash - instead of a centralized trusted party, can we implement a *mixer* kind of pool which embeds consensus within it. Can we also convince every participant in the consensus round, that a particular transaction is a valid transaction and at the same time, there should be no information leakage.

14.2 ZCash

After taking a closer look at the system described above, using zk-SNARK seems to be the right choice to implement such a payment system.

14.2.1 Design Goal

- Decentralized Anonymous Payment Scheme.
- At any given time, there is a unique snapshot of the ledger state and is available to everyone in the consensus. Here, a *ledger* is a sequence of append-only transactions.
- ZCash supports 2 types of transactions - base transactions and ZCash transactions. At the outset, ZCash can be considered to be a ***fork of Bitcoin.***

14.2.2 Toy ZCash

For the sake of simplicity, consider *Toy ZCash.* The difference between toy ZCash and real ZCash is that it assumes that all the coins have a **fixed nomination**. Toy ZCash uses crypto-primitives like zk-SNARK and commitment.

Commitment Let's say we have a random sample r.

$$r \overset{\$}{\leftarrow} D$$

$$C := \mathrm{COMM}_r(m)$$

Where m is a message.
Lets say Bob sends a commitment to Alice. The point of a commitment is that at any point in time, Bob has a secret r. If Bob wants to prove that he *committed*, he just needs to review r and m and verify it.

How is a new coin created by Toy ZCash?
Here, we assume that there is an underlying bitcoin like ledger. Coin minting essentially means that the base coin is converted into private coin. Consider a user u:

$$r \overset{\$}{\leftarrow} D$$

The user also needs to sample a random number which would be a serial number sn of the coin.

$$sn \overset{\$}{\leftarrow} D$$

, After which a coin commitment needs to be computed.

$$cm := \mathrm{COMM}_r(sn)$$

Here, the coin is a triple as follows:

$$c := (r, sn, cm)$$

Where cm is the commitment and r and sn act like a secret. User u sends TX_{mint} (which contains cm) as well as one basecoin to the ledger. When the ledger receives this cm and one basecoin, TX_{mint} is appended to the ledger. Note that, the user need to disclose their identity/address in this case. The transaction can take place over the underlying peer-to-peer network and thus a simple message relay would suffice.

The ledger contains CM_{list} which is a list of all coin commitments. To increase efficiency, a merkle tree can be used instead of a simple list.

What is the same secret is used by another user u'?
Lets say u' has the same secret $c := (r, sn, cm)$ and wants to *spend* the coin. This is equivalent to *claiming a base coin from private coin.* u' could send c by posting TX_{spend}. TX_{spend} has the following parts to it:

- Serial number sn.
- zk-Proof of NP Statement π_i. u' needs to prove the NP Statement having 2 parts:
  - An r is known, such that $cm' = \text{COMM}_r(sn)$
  - $cm' \in CM_{list}$

  Here, both r and cm' are private.

Both the sender and the receiver shouldn't reveal the secret r. If the secret r is revealed, everyone can track the transactions. The $\underline{key}$ is here that r ***should always be secret*** and can only be shared between the sender and receiver.

14.2.3 Optimizations

Optimization 1:
As part of the NP Statement, $cm' \in CM_{list}$ needs to be proved. The size of the CM_{list} grows linearly. This lowers the look-up performance as the list grows. The use of a ***Merkle Tree*** helps solve this problem. So now, we only need to prove that $cm' \in R_t$ where R_t is the merkle root. This reduces the look-up time $O(n)$ to $O(logn)$.

Optimization 2:
Direct payment: Lets say we have 2 new requirements:

- Support arbitrary value (unlike toy ZCash).
- Sender shouldn't be able to track the coin after sending, i.e., the sender shouldn't know when the receiver spends the coin using the serial number sn that the sender sampled.

We can **redesign** the ***commitment scheme*** in order to meet these requirements as follows:
Derive 3 PRFs from a single random seed: x

- $PRF_x^{addr}(\cdot)$ is used to compute the address
- $PRF_x^{sn}(\cdot)$ is used to compute the serial number
- $PRF_x^{PK}(\cdot)$ is used to compute the public key, ensuring that this function is collision resistant.

Address Scheme: User u generates a key-pair (a_{pk}, a_{sk}) with a constraint the a coin designated to a_{pk} can only be spent by someone having a_{sk}. This coin could be user u's own coin which they got by minting or a coin sent by some other user using the public key a_pk. The key-pair is generated as follows:

$$a_{sk} \overset{\$}{\leftarrow} A$$

$$a_{pk} := PRF^{addr}_{a_{sk}}(\cdot)$$

Redesign of Minting scheme
Goal here is to mint a coin c with value v. User has a secret a_{sk}. Let ρ be the coin specific random secret.

$$\rho \overset{\$}{\leftarrow} D$$

$$sn := PRF^{sn}_{a_{sk}}(\rho)$$

This coin is signed by both the coin specific secret(ρ) and user-specific secret a_{sk}. Now this is a 2-stage commitment.

- $k := \mathrm{COMM}_r(a_{pk}||\rho)$, $r \overset{\$}{\leftarrow} D$
- $cm := \mathrm{COMM}_s(v||k)$, $s \overset{\$}{\leftarrow} D$, commit the value as well, since at a certain point in time, this value has to be revealed.

To formally define the coin c,

$$c := (a_{pk}, v, \rho, r, s, cm)$$

The coin has public key, value, random serial number, secret of first stage commitment, secret of second stage commitment and the final commitment.

$$TX_{mint} := (v, k, s, cm)$$

Observations

- In the final mint transaction, it has cm, s and v. Everyone can verify whether $cm =? \mathrm{COMM}_s(v||k)$
- ρ, a_{pk} are hidden

Note: Mint transaction is only valid if user u deposits v BTC.

14.2.4 Pour Protocol

The spending protocol is called the pour protocol in this case. In order to upkeep the privacy guarantee of the transaction, linking any 2 coins just by observing their values should be avoided. This is done by maintaining the following relation:

$$v_{old} = v_1 + v_2$$

Here, v_{old} is the *coin-in* and v_1 and v_2 are the coins out. Effectively, we now have 3 coins such that:

$$c^{old} = (a^{old}_{pk}, v^{old}, \rho^{old}, r^{old}, s^{old}, cm^{old})$$

$$c_1^{new} = (a_{pk,1}^{new}, v_1^{new}, \rho_1^{new}, r_1^{new}, s_1^{new}, cm_1^{old})$$
$$c_2^{new} = (a_{pk,2}^{new}, v_2^{new}, \rho_2^{new}, r_2^{new}, s_2^{new}, cm_2^{old})$$

User wishes to pour c^{old} to c_1^{new}, c_2^{new}. Following the 2 stage commitment rules, ρ is generated as follows:

$$\rho_i^{new} \overset{\$}{\leftarrow} D, \text{ where } i \in 1, 2$$

- $r_i^{new} \overset{\$}{\leftarrow} D$, $k_i^{new} = \text{COMM}_{r_i}^{new}(a_{pk,i}^{new} || \rho_i^{new})$
- $s_i^{new} \overset{\$}{\leftarrow} D$, $cm_i^{new} := \text{COMM}_{s_i}^{new}(v_i^{new} || k_i^{new})$

The serial number of the old coin (sn_{old}) while pouring, is **public**. Essentially, serial number is used to prevent double spending; recall that the ledger is checked if sn was spent previously. cm_1^{new} and cm_2^{new} is public since they are written to the ledger. Everything else is kept private.

zk-Statement: I know $c_{old}, c_1^{new}, c_2^{new}, \rho^{old}$ and a_{sk}^{old} such that:

- $cm^{old} \in \text{TREE}(root)$
- $v_{old} = v_1^{new} + v_2^{new}$
- $sn^{old} = PRF_{a_{sk^{old}}}^{sn}(\rho)$, this step verifies the serial number sn^{old}
- cm_i^{new} is well formed, where $i \in 1, 2$
- $a_{pk}^{old} = PRF_{a_{sk}^{old}}^{addr}(\cdot)$ Intuition: Before spending the coin, the user needs to prove that they are the owners of the old coin.

14.2.5 Summary

To summarize, ZCash is the first payment system which truly provides privacy to the transactions in a distributed system. By the scheme of the ZCash protocol, the sender has no way to track the transaction and the coin through the ZCash system.