The Theory Behind Blockchains (Spring 19)
Recitation 10

Eliad Tsfadia

1 Interactive Zero-Knowledge Proof (Reminder)

An interactive proof (P, V) for a language $L \subseteq \{0,1\}^*$ is a protocol between an efficient prover P and an efficient verifier V with the following properties:

- Completeness: On $x \in L$ and a witness w: $\Pr[(P(w), V)(x) = 1] \geq 1 - \text{negl}(|x|)$.
- Soundness: On $x \notin L$ and for any efficient cheating prover P^*: $\Pr[(P^*, V)(x) = 1] \leq \text{negl}(|x|)$.

Such a proof is ZK if on true statement $x \in L$, whatever the verifier learns from the interaction $(P(w), V)(x)$, it could have learned on its own. Formally: There exists an efficient simulator S such that for any efficient strategy V^*,

$$(P(w), V^*)(x) \approx S(x)$$

Fact from class: Assuming OWF, there exists an interactive ZK proof for any $L \in NP$.

2 Non-Interactive Zero-Knowledge proofs (NIZK)

In a blockchain we would like to use proofs which are non-interactive: a prover can produce a proof π which is ZK but also publicly verifiable. A naive attemp is to define it as follows:

- Completeness: On $x \in L$ and a witness w: $\Pr[V(x, \pi) = 1 \colon \pi \leftarrow P(x, w)] \geq 1 - \text{negl}(|x|)$.
- Soundness: On $x \notin L$ and for any efficient cheating prover P^*: $\Pr[V(x, \pi^*) = 1 \colon \pi^* \leftarrow P^*(x)] \leq \text{negl}(|x|)$.
- Zero Knowledge: There exists a simulator S such that on $x \in L$: $P(x, w) \approx S(x)$.

Bad news: Such proofs only exists for trivial languages $L \in BPP$ (why?).

Therefore, we look at a slightly different model in which it possible to create non-interactive proofs for non-trivial languages. The model is Common Random String (CSR), meaning that when generating a proof for a statement x, we assume the existence of a random string $R \in \{0,1\}^{\text{poly}(|x|)}$ in the sky that all parties has access to:

- Completeness: On $x \in L$ and a witness w:

$$\Pr[V(x, \pi, R) = 1 \colon R \leftarrow \{0,1\}^{\text{poly}(|x|)}, \pi \leftarrow P(x, w, R)] \geq 1 - \text{negl}(|x|)$$

- Soundness: On $x \notin L$ and for any efficient cheating prover P^*:

$$\Pr[V(x, \pi^*, R) = 1 \colon R \leftarrow \{0,1\}^{\text{poly}(|x|)}, \pi^* \leftarrow P^*(x, R)] \leq \text{negl}(|x|)$$

- Zero Knowledge: There exists an efficient simulator S such that on $x \in L$:

$$(x, R, P(x, w, R))_{R \leftarrow \{0,1\}^{\text{poly}|x|}} \approx (x, S(x))$$

Note that in the ZK part, the CRS string R is chosen by the simulator, and therefore such a proof does not imply that L is trivial (why?)

Fact: Assuming TDP, there exists a NIZK proof in the CRS model for any $L \in NP$.

3 NIZK in Zerocoin (Reminder)

Recall that in the Zerocoin system, there are two types of coins: a basecoin and a zerocoin. The key feature that provides anonymity is that you can convert basecoins into zerocoins and back again, and when you do that, it breaks the link between the original basecoin and the new basecoin. In this system, Basecoin is the currency that you transact in, and Zerocoin just provides a mechanism to trade your basecoins in for new ones that are unlinkable to the old ones.

Zerocoins come into existence by minting, and anybody can mint a zerocoin. For simplicity, we assumed that there is only one denomination worth 1.0 zerocoins, and that each zerocoin is worth one basecoin. Just minting one doesn't automatically give it any value — you can't get free money. It acquires value only when you put it onto the block chain, and doing that will require giving up one basecoin.

Minting a zerocoin is done in three steps:

1. Generate serial number S and a random secret r.

2. Compute $Commit(S, r)$, a commitment to the serial number.

3. Publish the commitment onto the block chain.

To spend a zerocoin and redeem a new basecoin, you need to prove that you previously minted a zerocoin, without revealing both S and r. This is where NIZK proof comes in. At any point, there will be many commitments on the block chain — let's call them $C_1, \ldots, C_n$. Then in order to spend a zerocoin with serial number S to redeem a new basecoin: Create a special "spend" transaction that contains S along with a NIZK proof of the statement: "I know r such that $Commit(S, r)$ is in the set $\{C_1, \ldots, C_n\}$".

Observe that r is kept secret throughout; neither the mint nor the spend transaction reveals it. That means nobody knows which serial number corresponds to which zerocoin.

4 Zerocoins with different worth values

The above solution assumed that each zerocoin is worth one basecoin. Say that I have 30 basecoins and I want to mint from them one zerocoin that worth 15 basecoins. Later, I might want to spend only 10 of the basecoins inside the zerocoin and stay with a zerocoin that worth 5 basecoins. How can we enable such a flexibility?

The mint process of zerocoin with value V should consist of the following steps:

1. Generate serial number S and a random secret r (as before).

2. Compute $C = Commit(V, S, r)$, a commitment to the value and the serial number.

3. Generate a NIZK proof $\pi_{(V,C)}$ for the statement: ”I know (S, r) such that $C = Commit(V, S, r)$”.
4. Publish $(V, C, \pi_{(V,C)})$ in the blockchain.

In words, we publish a proof that the new zerocoin indeed has value V. Now, given a zerocoin which has value V and serial number S, if its owner wants to spend $V' < V$ of it and put the rest inside a new zerocoin, it should do the following steps:

1. Generate a serial number S' and a random secret r' (for the new zerocoin).
2. Compute $C' = Commit(V - V', S', r')$.
3. Generate a NIZK proof $\pi'_{(S,V',C')}$ for the statement: ”I know $(\tilde{V} > 0, r, S', r')$ such that $Commit(V' + \tilde{V}, S, r) \in \{C_1, \ldots, C_n\}$ and $C' = Commit(\tilde{V}, S', r')$”.
4. Publish $(S, V', C', \pi'_{(S,V',C')})$ in the blockchain to redeem V' basecoins.

Namely, $\pi'_{(S,V',C')}$ proves that we took value V' from a zerocoin of serial number S and created a new zerocoin C' for rest of the unspent $V - V'$ basecoins.

What is the level of anonymity that this system achieves?