# Blockchains & Cryptocurrencies

## Anonymity - II

Image from cryptonomad.info

Instructor: Matthew Green & Abhishek Jain
Johns Hopkins University - Spring 2023

# Agenda

- **Last Time:** Started new thread on anonymity
  - Pseudonymity vs anonymity
  - Why Bitcoin does not guarantee anonymity
  - Older approaches to anonymity: Blind Signatures (in E-Cash), Mixers (centralized vs de-centralized), CryptoNote
- **Today:** Continue the thread
  - Confidential Transactions, ZeroCoin (and maybe ZCash)
  - Homomorphic Commitments, Zero-Knowledge Proofs

# Recall: CryptoNote idea

- I want to make a transaction with (e.g.,) one input
  - But I don't want to reveal <u>which</u> transaction is my input
  - Standard Bitcoin transactions do reveal this, and it leads to privacy problems
  - I could mix with other people (e.g., CoinJoin) but they would have to participate with me online, and that's annoying

# Key Ingredient: **Ring Signatures**

- Normal signature: sign with *sk*, verify with *pk*
- **Ring signature**:
  - Sign with my secret key + N-1 <u>other people's public keys</u> (Signer does not have to know the other secret keys!)
  - Verifier verifies with all *N* public keys (she must know them)
  - **Privacy**: verifier does not learn <u>which</u> signer actually made the signature! (It could be any of the key owners!)

# CryptoNote Limitations

- CryptoNote ring signatures grow as O(N) where N is number of inputs
    - Ditto signing time and verification time
    - In practice this limits N to something modestly small (1-7)
- Original CryptoNote required all input transactions be the same value

# *Confidential* Transactions [Maxwell]

# *Confidential* Transactions [Maxwell]

- "Hide" transaction value using commitments

# *Confidential* Transactions [Maxwell]

- "Hide" transaction value using commitments
  - Think: Why would this be beneficial?

# *Confidential* Transactions [Maxwell]

- "Hide" transaction value using commitments
  - Think: Why would this be beneficial?
- What if we want to support multiple inputs and outputs?

# *Confidential* Transactions [Maxwell]

- “Hide” transaction value using commitments
  - Think: Why would this be beneficial?
- What if we want to support multiple inputs and outputs?
  - Need to establish that “total” input >= “total” output.

# *Confidential* Transactions [Maxwell]

- "Hide" transaction value using commitments
  - <u>Think</u>: Why would this be beneficial?
- What if we want to support multiple inputs and outputs?
  - Need to establish that "total" input >= "total" output.
- **<u>Main Challenge:</u>** How to verify that a transaction is valid when the values are hidden?

# *Confidential* Transactions [Maxwell]

- Two Ideas:

# *Confidential* Transactions [Maxwell]

- Two Ideas:
  - **(Additively) Homomorphic commitments**: There is an operation that can be performed on commitments that will result in addition of underlying values

# *Confidential* Transactions [Maxwell]

- Two Ideas:
  - **(Additively) Homomorphic commitments**: There is an operation that can be performed on commitments that will result in addition of underlying values
    - Now, need to establish that (Sum of inputs) - (Sum of outputs) is non-negative.

# *Confidential* Transactions [Maxwell]

- Two Ideas:
  - **(Additively) Homomorphic commitments**: There is an operation that can be performed on commitments that will result in addition of underlying values
    - Now, need to establish that (Sum of inputs) - (Sum of outputs) is non-negative.
  - **Zero-Knowledge Proofs**: Prove something about committed values **without revealing the values**!

# Commitments

- Like a digital "envelope": allows you to commit to a message value, without revealing what it is
    - C = Commit(message; randomness)
    - **Hiding**: given a commitment, can't see what message it is, until I "open" the commitment and reveal it to you
    - **Binding**: giving you a commitment "binds" me to a specific message/value. I can't change my mind when I open it.

# Recall: Hash commitments

- **Commit Procedure**:
  - Pick some random "salt" (e.g., 256 bits) **r**
  - Compute C = Hash(message || r)
- **Open Procedure**: Reveal (message, r), verifier checks hash
- **Additive Homomorphism**: Not known for general hash functions :-(

# Pedersen Commitments

- Let $G = \langle g \rangle$ be a "cyclic" group where it is hard to find *x* given $(g, g^x)$ — AKA the **discrete log problem** (DLP) is hard
    - E.g., *G* can be a subgroup of a finite field $\{1, \ldots, p-1\}$ where exponentiation/multiplication are modulo *p*
    - We also need two public **generators**: $g, h$ *such that nobody knows the discrete log of h w.r.t. g*
- Commitment to message *m*: Pick random $r \in \{0, \ldots, groupOrder - 1\}$, compute: $C = g^m \cdot h^r$
- To open the commitment, simply reveal $(m, r)$

# Pedersen Commitments

- Why is this secure?
  - **Hiding:** If g, h are generators, then $h^r$ is a random element of the group, so. $C = g^m \cdot h^r$ is too

# Pedersen Commitments

- Why is this secure?
  - **Hiding:** If g, h are generators, then $h^r$ is a random element of the group, so $C = g^m \cdot h^r$ is too
  - **Binding:** Let q be the group order. Let $h = g^x$ for some unknown *x*. Assume an attacker can find (m, r) != (m', r') such that .Then it holds that: $g^m h^r = g^{m'} h^{r'}$

$$g^m g^{xr} = g^{m'} g^{xr'}$$

and thus,

$$m + xr = m' + xr' \ mod \ q$$

We can solve for *x*, which means solving DLP, which is contradiction!

# Pederson Commitments

- Pedersen commitments are additively homomorphic:
  - Commit to "m1": $C_1 = g^{m_1} h^{r_1}$
    Commit to "m2": $C_2 = g^{m_2} h^{r_2}$
  - Now multiply the two commitments together:

$$\begin{aligned} C_3 &= C_1 \cdot C_2 \\ &= g^{m_1} h^{r_1} \cdot g^{m_2} h^{r_2} \\ &= g^{m_1+m_2} h^{r_1+r_2} \end{aligned}$$

Notice that C3 is a commitment to the sum m1+m2 (under randomness r1+r2)

# Zero-Knowledge (ZK) Proofs

# Zero-Knowledge (ZK) Proofs

- Invented by Goldwasser, Micali, Rackoff in 1980s

# Zero-Knowledge (ZK) Proofs

- Invented by Goldwasser, Micali, Rackoff in 1980s
  - Prove a statement <u>without revealing any other information</u>

# Zero-Knowledge (ZK) Proofs

- Invented by Goldwasser, Micali, Rackoff in 1980s
  - Prove a statement <u>without revealing any other information</u>
  - What does this mean?

# Zero-Knowledge (ZK) Proofs

- Invented by Goldwasser, Micali, Rackoff in 1980s
    - Prove a statement <u>without revealing any other information</u>
    - What does this mean?
- Powerful Theorem by Goldreich-Micali-Wigderson from 1980s:
  **Anything in NP can be proven in zero-knowledge**

# Zero-Knowledge (ZK) Proofs

- Invented by Goldwasser, Micali, Rackoff in 1980s
    - Prove a statement <u>without revealing any other information</u>
    - What does this mean?
- Powerful Theorem by Goldreich-Micali-Wigderson from 1980s:
  **Anything in NP can be proven in zero-knowledge**
- What is NP?

# Zero-Knowledge (ZK) Proofs

- Invented by Goldwasser, Micali, Rackoff in 1980s
  - Prove a statement <u>without revealing any other information</u>
  - What does this mean?
- Powerful Theorem by Goldreich-Micali-Wigderson from 1980s:
  **Anything in NP can be proven in zero-knowledge**
- What is NP?
  - Class of languages where membership can be efficiently verified using a "witness" (a.k.a certificate of validity)

# Zero-Knowledge (ZK) Proofs

- Invented by Goldwasser, Micali, Rackoff in 1980s
  - Prove a statement <u>without revealing any other information</u>
  - What does this mean?
- Powerful Theorem by Goldreich-Micali-Wigderson from 1980s:
  **Anything in NP can be proven in zero-knowledge**
- What is NP?
  - Class of languages where membership can be efficiently verified using a "witness" (a.k.a certificate of validity)

# RingCT Extension to CryptoNote

- Uses these tools to achieve variable-value, hidden transactions
- Proofs of transaction validity used in RingCT are special-purpose, not general-purpose (we will later discuss how using general-purpose proofs can simplify design)

# Zerocoin (MGGR14)

- Proposed as an extension to Bitcoin in 2014
  - <u>Requires changes to the Bitcoin consensus protocol!</u>
- **Main Advantage**: Huge anonymity set (potentially, all transactions)
  - How to do this without performance penalty?

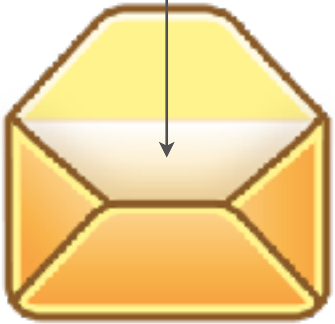
# Zerocoin (MGGR14)

- I can take Bitcoin from my wallet
    - Turn them into ‘Zerocoins’
    - Where they get ‘mixed up’ with many other users’ coins
    - I can redeem them to a new fresh Wallet

# Zerocoin

- Zerocoins are just numbers
  - Each is a digital commitment to a random serial number
  - Anyone can make one!

823848273471012983

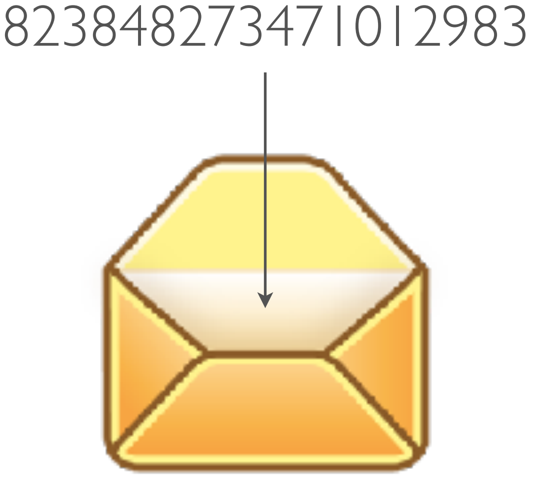


# Minting Zerocoin

- Zerocoins are just numbers
  - Each is a digital commitment to a random serial number *SN*
  - Anyone can make one!

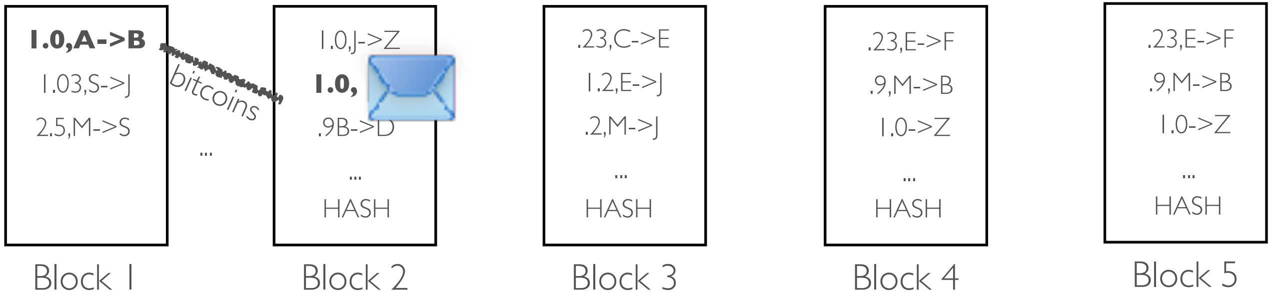
$$C = Commit(SN; r)$$

$$C = g^{SN} h^r \ mod \ p$$



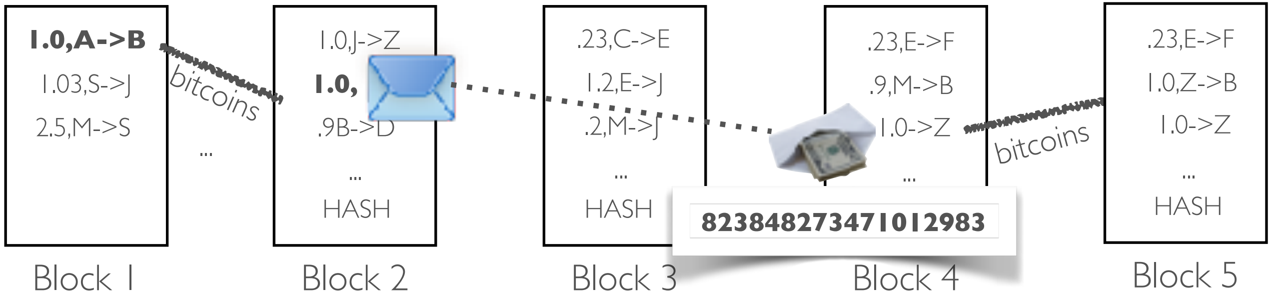
# Minting Zerocoin

- Zerocoins are just numbers
  - They have value once you write them into a valid transaction on the blockchain
  - Valid: has inputs totaling some value e.g., 1 bitcoin



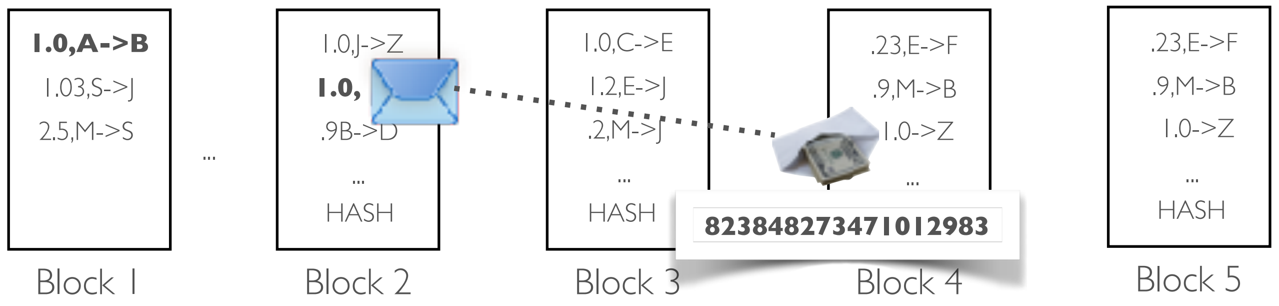
# Redeeming Zerocoin

- You can redeem zerocoins back into bitcoins
  - Reveal the serial number &
    Prove that it corresponds to some Zerocoin on the chain
  - In exchange you get one bitcoin (if SN is not already used)



# Spending Zerocoin

- Why is spending anonymous?
  - It's all in the way we 'prove' we have a Zerocoin
  - This is done using a zero knowledge proof



# Spending Zerocoin

- Here we prove that:
  (a) there exists a Zerocoin in the block chain
  (b) we just revealed the actual serial number inside of it

- Revealing the serial number prevents double spending

- The trick is doing this efficiently!

# Spending Zerocoin

- Possible proof statement (not efficient, see CryptoNote):

  - Public values: list of Zerocoin commitments $C_1, C_2, \ldots, C_N$
    Revealed serial number *SN*

  - Prove you know a coin *C* and randomness *r* such that:

$$C = C_1 \ \vee \ C = C_2 \ \vee \ \ldots \ \vee \ C = C_N$$
$$\wedge \ C = Commit(SN; r)$$

  - Problem: using standard techniques, this ZK proof has cost/size O(*N*)

# Spending Zerocoin

- Zerocoin (actual protocol)
  - Use an efficient RSA one-way accumulator
  - Accumulate $C_1, C_2, \ldots, C_N$ to produce a short value $A$
  - Then prove knowledge of a short witness s.t. $C \in inputs(A)$
  - And prove knowledge that $C$ opens to the serial number

Requires a proof (**~25kb**)
for each spend. In the block chain.

# Spending Zerocoin

- Zerocoin (actual protocol)
  - Use an efficient RSA one-way accumulator
  - Accumulate $C_1, C_2, \ldots, C_N$ to produce a short value $A$
  - Then prove knowledge of a short witness s.t. $C \in inputs(A)$
  - And prove knowledge that $C$ opens to the serial number

Requires a proof (**~25kb**)
for each spend. In the block chain.

# Anonymity set comparison

- Anonymity set in CoinJoin:
  - **M**: where **M** is number of inputs in the transaction (bounded by TX size)
- Anonymity set in ByteCoin/RingCT:
  - **N**: where **N** is the number of inputs allowed in a transaction (bounded by TX size, 7-11 historically)
- Anonymity set in Zerocoin:
  - **P**: where **P** is number of total Zerocoins minted on the blockchain thus far* (independent of TX size)

# Zero-Knowledge Proofs for NP

# Zero-Knowledge Proofs for NP

- Powerful Theorem by Goldreich-Micali-Wigderson from 1980s:
  **Anything in NP can be proven in zero-knowledge**

# Zero-Knowledge Proofs for NP

- Powerful Theorem by Goldreich-Micali-Wigderson from 1980s:
  **Anything in NP can be proven in zero-knowledge**

- How do we show this?

# Zero-Knowledge Proofs for NP

- Powerful Theorem by Goldreich-Micali-Wigderson from 1980s:
  **Anything in NP can be proven in zero-knowledge**

- How do we show this?

  - Design a ZK proof for an "NP-Complete" Language (e.g., CircuitSAT)

# Zero-Knowledge Proofs for NP

- Powerful Theorem by Goldreich-Micali-Wigderson from 1980s:
  **Anything in NP can be proven in zero-knowledge**

- How do we show this?

  - Design a ZK proof for an "NP-Complete" Language (e.g., CircuitSAT)

  - <u>On the whiteboard</u>: ZK Proof for Sudoku puzzles (generalized Sudoku is NP-Complete)