

# Blockchains & Cryptocurrencies

## **Anonymity - II**



Image from [cryptonomad.info](https://cryptonomad.info)

Instructor: Abhishek Jain  
Johns Hopkins University - Spring 2021

# 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
  - CryptoNote (recap), Confidential Transactions, ZeroCoin (and maybe ZCash)
  - Along the way: Homomorphic Commitments, Zero-Knowledge Proofs

# Recall: CryptoNote & RingCT

- 2012: CryptoNote (“Nicolas van Saberhagen”)
  - Originally launched as part of the ByteCoin currency
  - Anonymous creator, did a pre-mine
  - Was forked multiple times into many different currencies, including bitmonero -> Monero
  - Protocol ideas later improved into RingCT, which hid amounts as well as inputs (used in Monero today)

# CryptoNote idea

- I want to make a transaction with (e.g.,) one input
  - But I don't want to reveal which 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

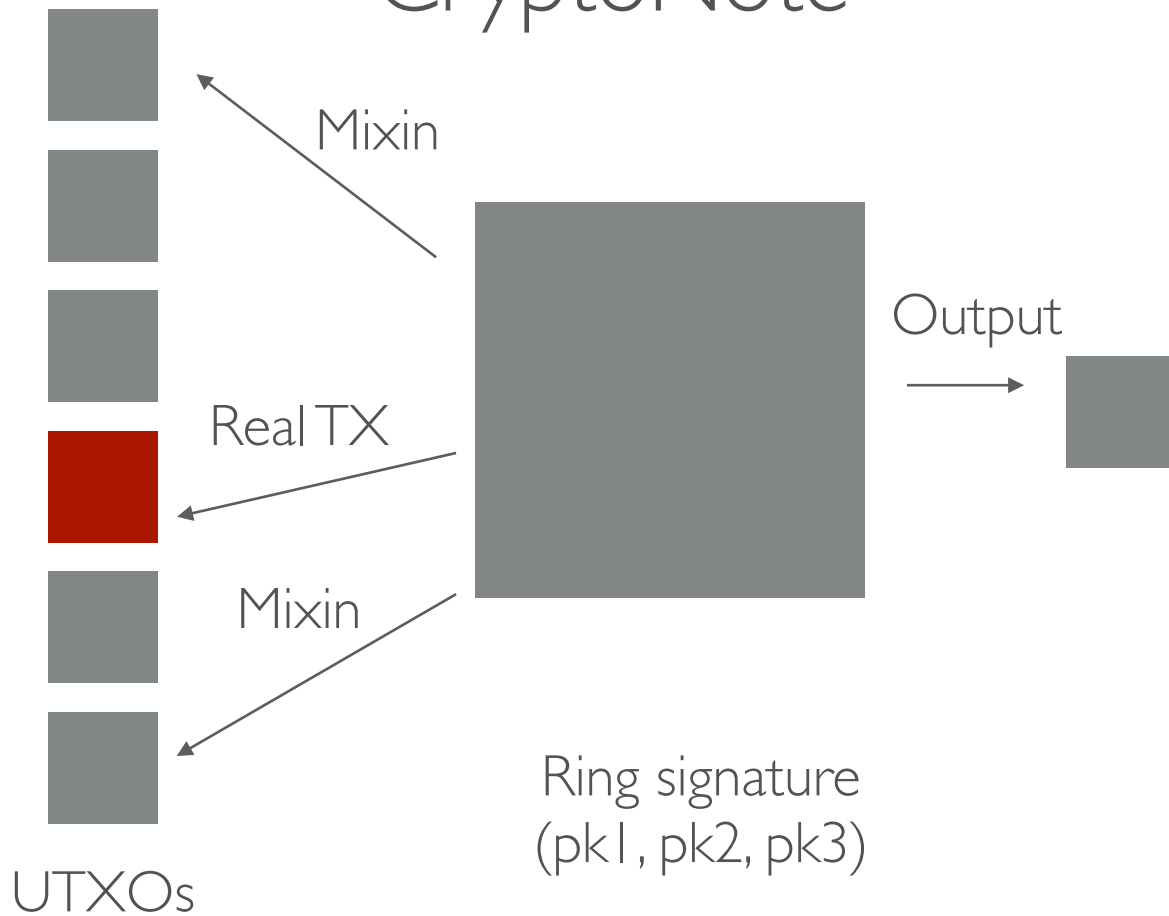
# Ingredient: Ring Signatures

- Normal signature: sign with  $sk$ , verify with  $pk$
- **Ring signature:**
  - Sign with my secret key +  $N-1$  other people's public keys  
(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 which signer actually made the signature! (It could be any of the key owners!)

# CryptoNote idea

- Make all transactions the same value (e.g., 1 ByteCoin)
- Make all addresses single-use (auto-generated)
- Assume (for simplicity) that spender has one “real” input
  1. Identify  $N-1$  unrelated “cover” transactions from the UTXO set, get those public keys (called “mixins”)
  2. Make a ring signature on her transaction, using her secret key and the  $N-1$  public keys for the mixing
  3. Post signature plus a “key image” (function of the real secret key) to prevent the real transaction being spent twice

# CryptoNote



# CryptoNote Limitations

- CryptoNote ring signatures grow as  $O(N)$  where  $N$  is number of inputs (including Mixins)
  - Ditto signing time and verification time
  - In practice this limits Mixin number to something modestly small (1-7)
  - Think: How does “small”  $N$  affect anonymity guarantee?
- Original CryptoNote required all input transactions be the same value, requiring multiple “denominations”
  - Think: Why? How to overcome this?



# *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  $\geq$  “total” output.
- **Main Challenge**: How to verify that a transaction is valid when the values are hidden?

# *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  $(\text{Sum of inputs}) - (\text{Sum of outputs})$  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
  - $C = \text{Commit}(\text{message}; \text{randomness})$
  - **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  $C = \text{Hash}(\text{message} \parallel r)$
- **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  $\{1, \dots, p-1\}$  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  $r \in \{0, \dots, groupOrder - 1\}$ , 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  $(m, r) \neq (m', r')$  such that  $g^m h^r = g^{m'} h^{r'}$ . Then it holds that:

$$g^m g^{xr} = g^{m'} g^{xr'} \quad \text{and thus,}$$
$$m + xr = m' + xr' \pmod{q}$$

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  $C_3$  is a commitment to the sum  $m_1+m_2$   
(under randomness  $r_1+r_2$ )



# Zero-Knowledge (ZK) Proofs

- Invented by Goldwasser, Micali, Rackoff in 1980s
  - Prove a statement without revealing any other information
  - 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
- Builds on ideas from Maxwell's Confidential 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
  - Requires changes to the Bitcoin consensus protocol!
- **Main Advantage:** Huge anonymity set (potentially, all transactions)
  - How to do this without performance penalty?



# Zerocoin (MGGR14)

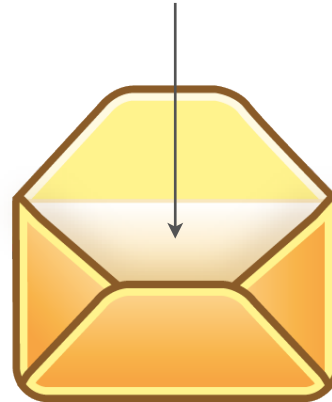
- I can take Bitcoin from my wallet
  - Turn them into 'Zerocoins'
  - Where they get 'mixed up' with many other users' coins
  - I can redeem them to a new fresh Wallet



# Zerocoin

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

823848273471012983



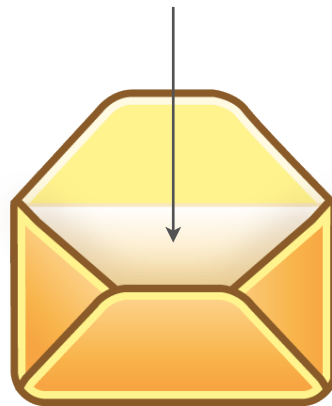
# Minting Zerocoin

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

$$C = \text{Commit}(SN; r)$$

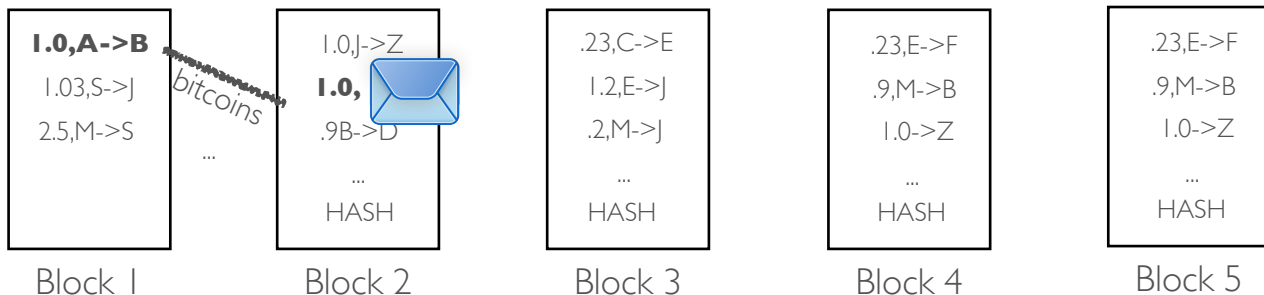
$$C = g^{SN} h^r \bmod p$$

823848273471012983



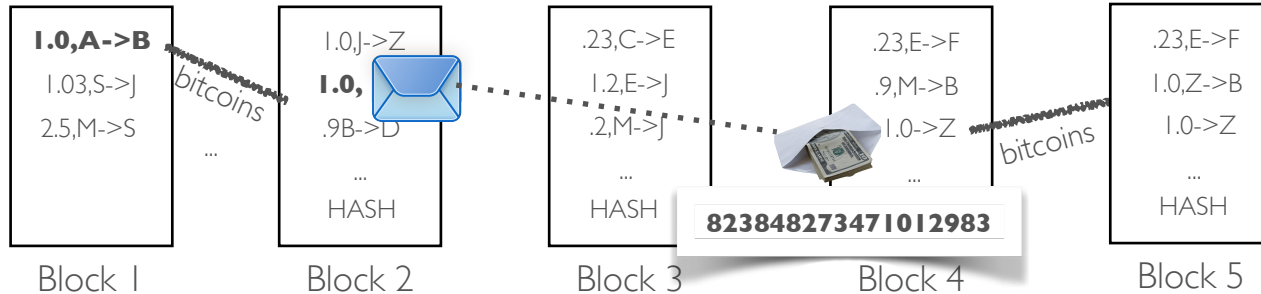
# Minting Zerocoin

- 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

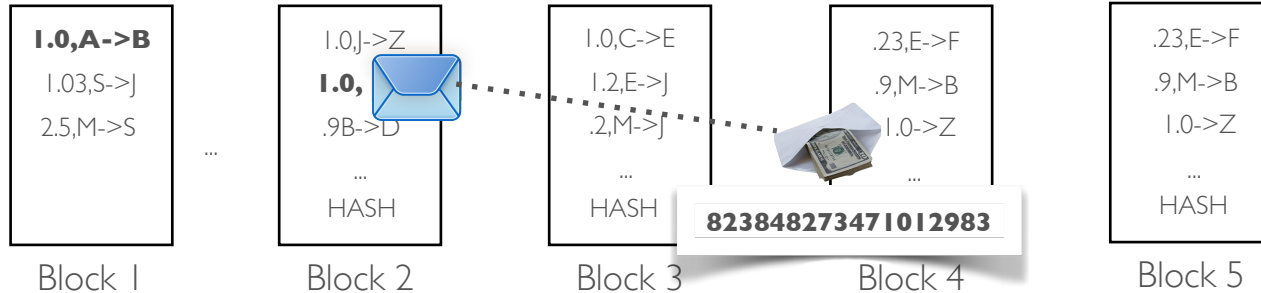
- You can redeem zerocoins back into bitcoins
  - Reveal the serial number & Prove that it corresponds to some Zerocoin on the chain
  - In exchange you get one bitcoin (if SN is not already used)





# 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  $C_1, C_2, \dots, C_N$   
Revealed serial number  $SN$

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

$$C = C_1 \vee C = C_2 \vee \dots \vee C = C_N \\ \wedge C = \textit{Commit}(SN; r)$$

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

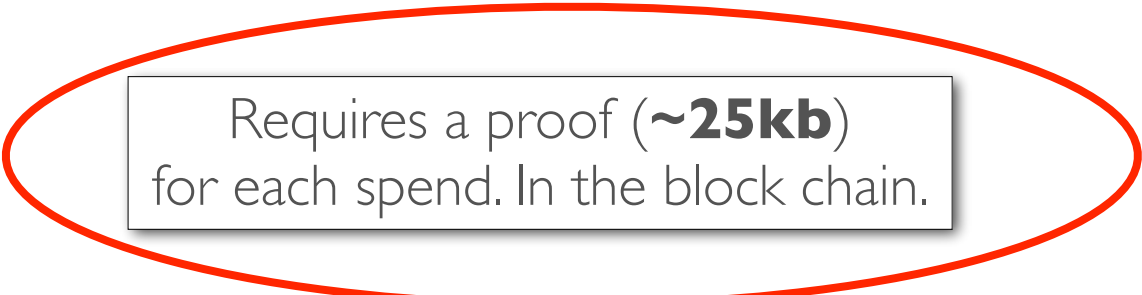
# Spending Zerocoin

- Zerocoin (actual protocol)
  - Use an efficient RSA one-way accumulator
  - Accumulate  $C_1, C_2, \dots, C_N$  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  $C_1, C_2, \dots, C_N$  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

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