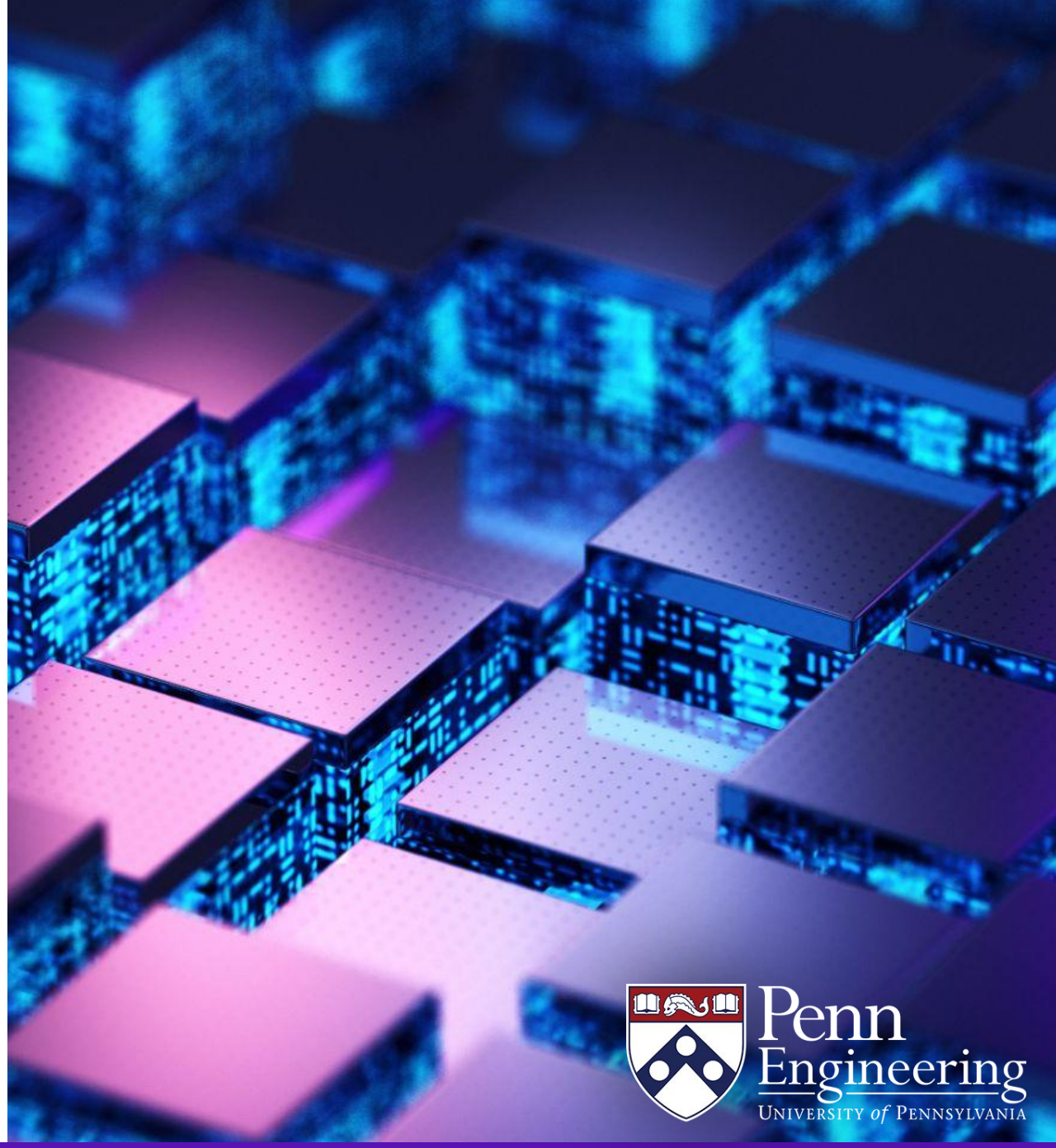
EAS 5830: BLOCKCHAINS

# ZCash

Professor Brett Hemenway Falk



# Zcash

- Founded in 2016
- Based on ZeroCoin (2013)
- Based on Bitcoin
  - Proof of Work
  - UTXO model
- Peak market cap $2.5 B (January 2018)
- Market cap $475 M (November 2023)

# Zcash protocol upgrades

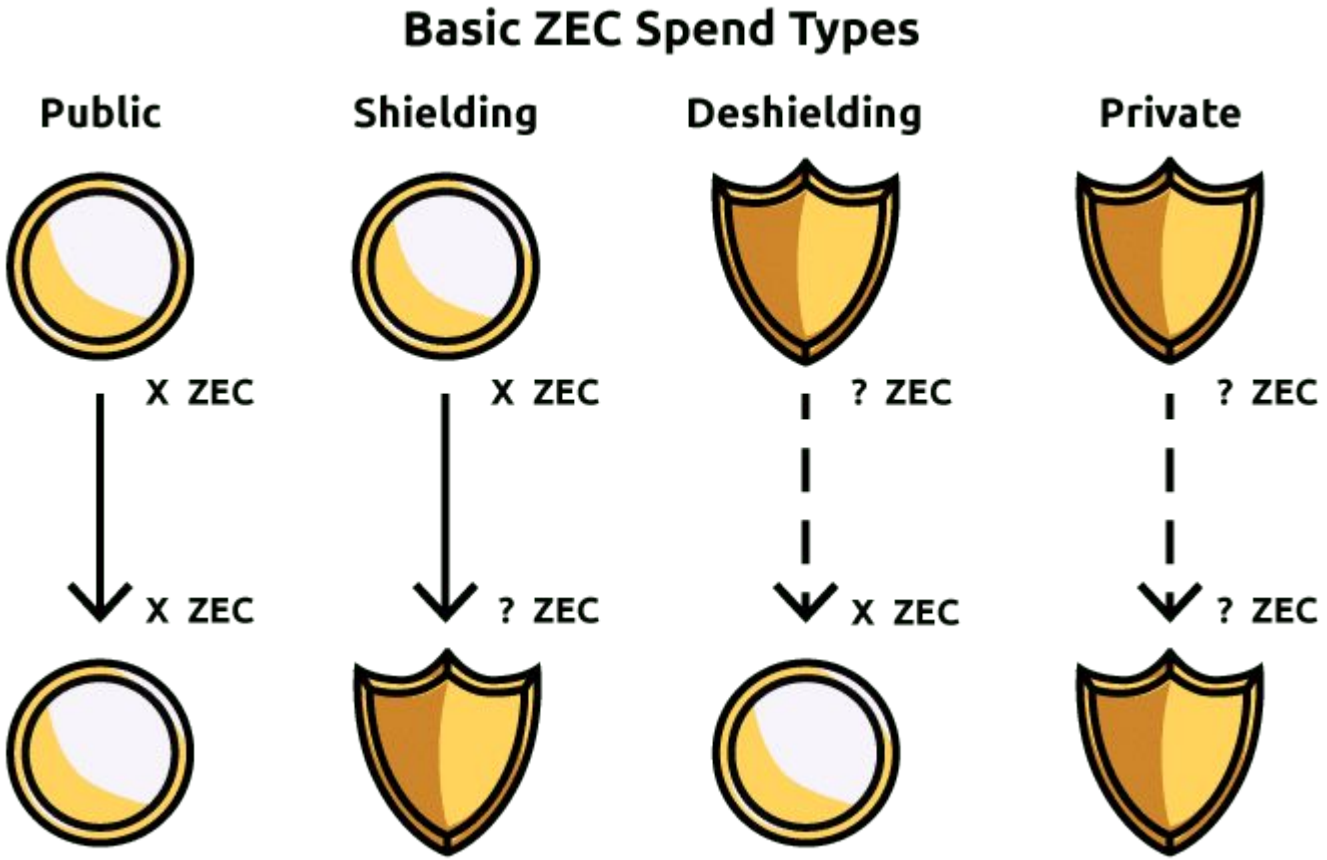
- Sapling 2018
- Blossom 2019
- Heartwood 2020
- Canopy 2020
- NU5

# Account types

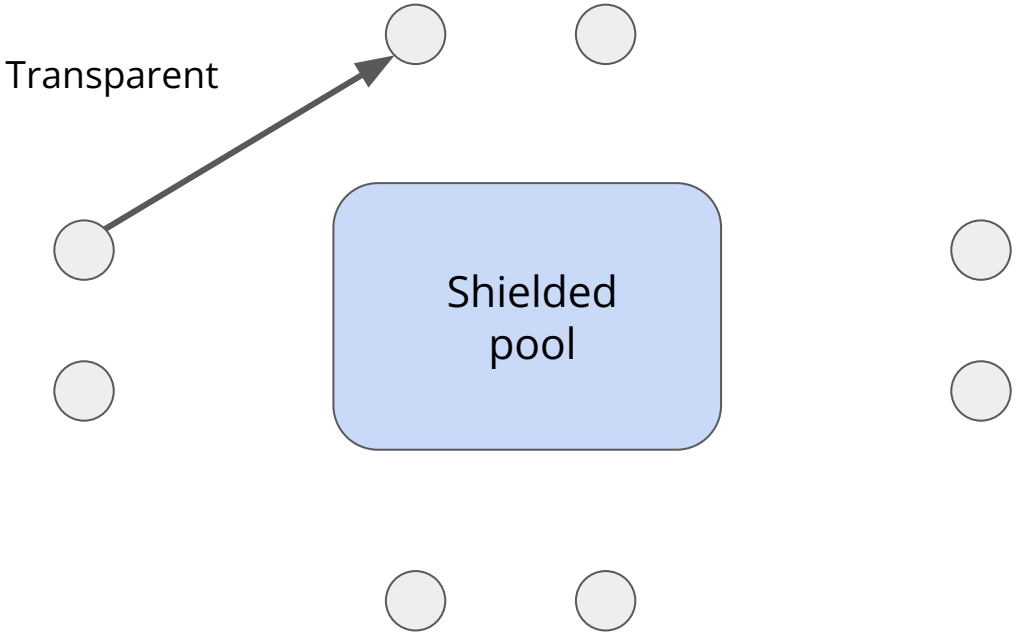
- Transparent
  - t-addresses have the same privacy as Bitcoin (none)

- Shielded
  - z-addresses are private

# Zcash

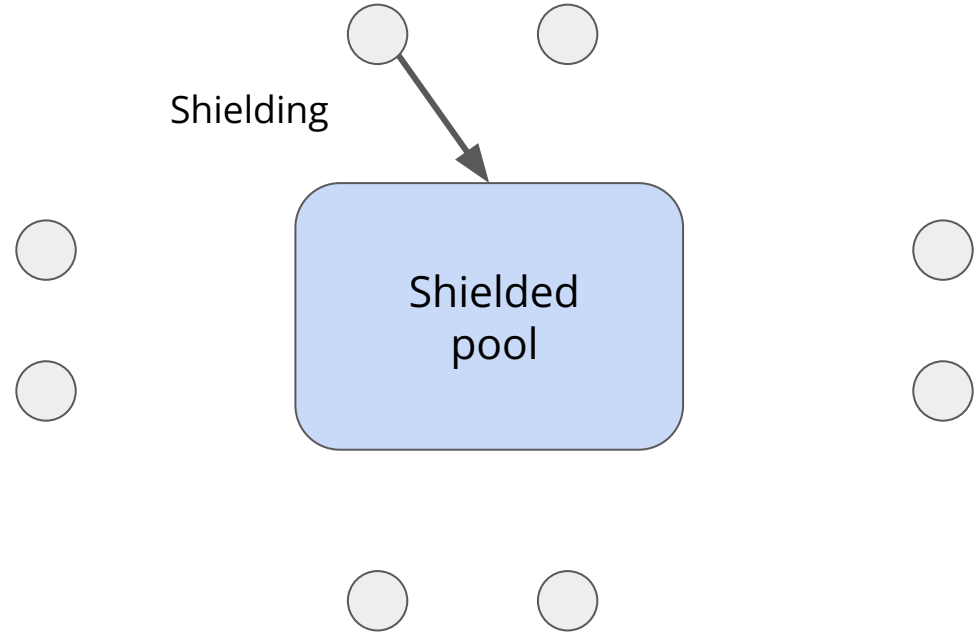

Basic ZEC Spend Types
Public
Shielding
Deshielding
Private
X ZEC
X ZEC
? ZEC
? ZEC
X ZEC
? ZEC
X ZEC
? ZEC

# Zcash transactions



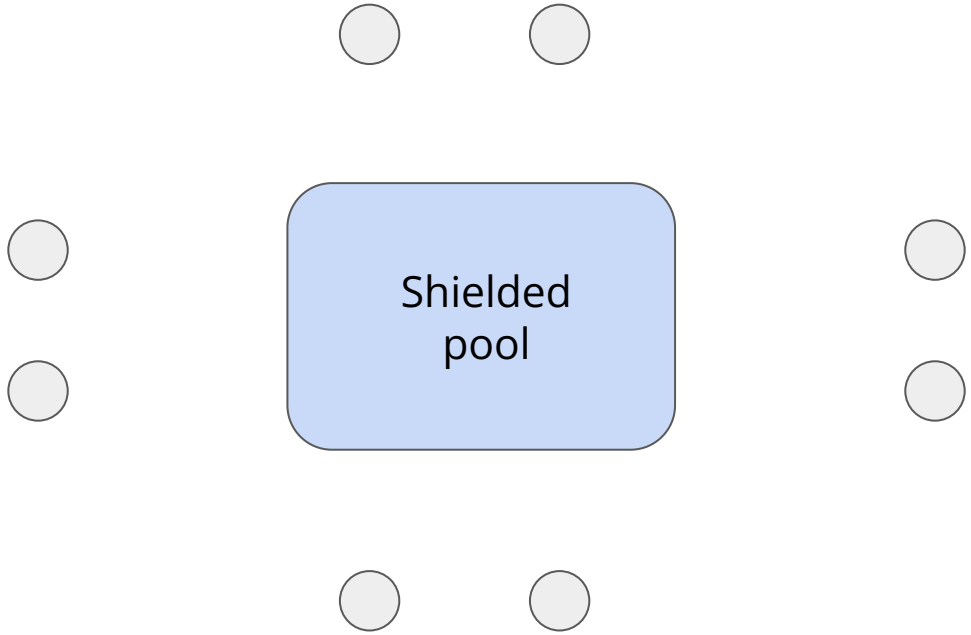

| | Public | Private |
|---|---|---|
| Sender | X | |
| Receiver | X | |
| Amount | X | |

# Zcash transactions




| | Public | Private |
|---|---|---|
| Sender | X | |
| Receiver | | X |
| Amount | X | |

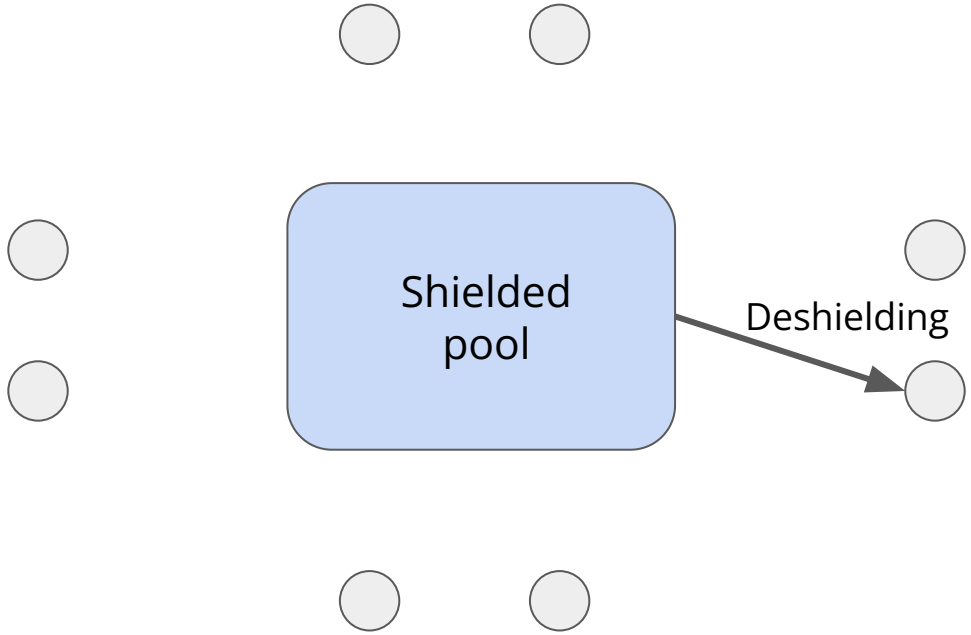
# Zcash transactions




Private

| | Public | Private |
|---|---|---|
| Sender | | X |
| Receiver | | X |
| Amount | | X |

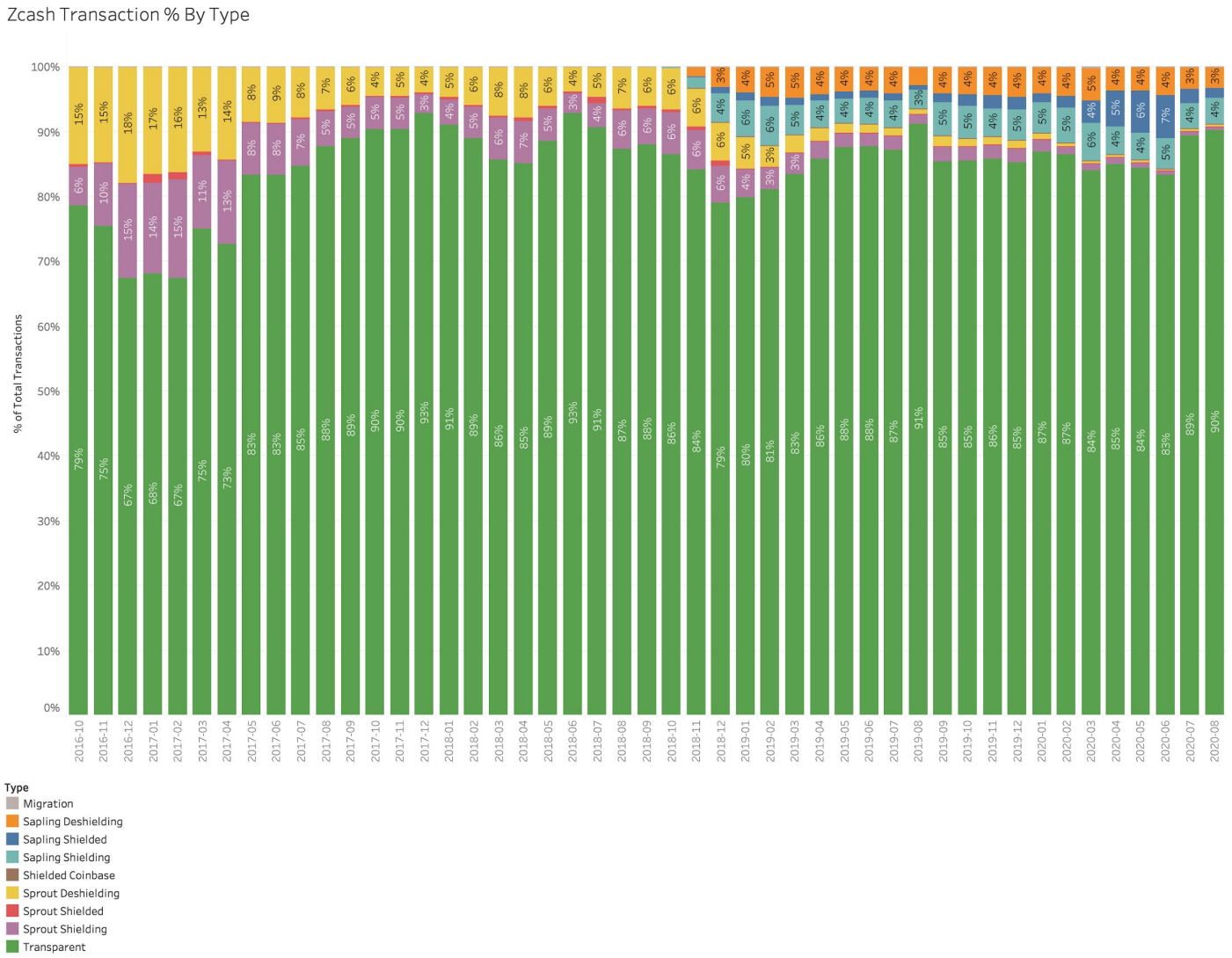
# Zcash transactions




| | Public | Private |
|---|---|---|
| Sender | | X |
| Receiver | X | |
| Amount | X | |

# Most transactions are public

- Only 6% Z-to-Z
- 85% T-to-T

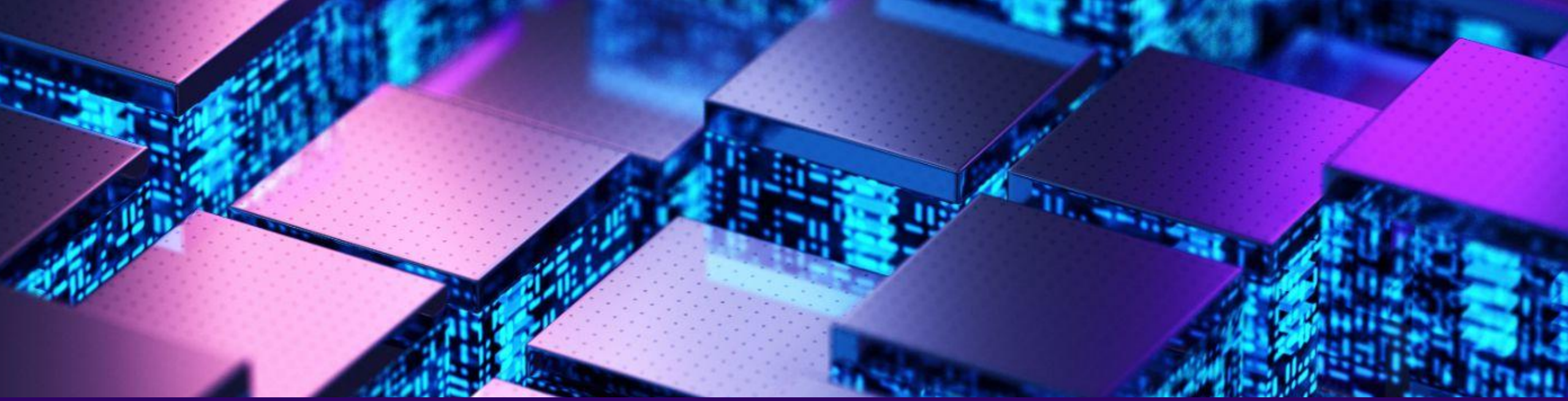


# Founders reward

- Miners receive 80% of ZEC mined
  - Initially 20% split between "founders fund" and core development
  - Now, 20% split between ECC, ZCash Foundation and Grants
  - Cumulative ZEC worth about ~$250M distributed via this mechanism


# Proof of Work

- Zcash is built on Bitcoin, and uses Proof-of-Work consensus
- Zcash uses equihash PoW (unlike Bitcoin)
- Equihash requires more memory than Bitcoin PoW
  - Designed to be “ASIC-resistant”
  - Bitmain released Equihash ASICs in 2021
- Several other coins use equihash (including Bitcoin Gold)



# How does ZCash work?

# Private transactions - Notes and Nullifiers

- Private transactions in Zcash are based on “notes” and “nullifiers”
  - Note is like an encrypted UTXO
  - Nullifier is a receipt that note has been spent
- Note: $(PK, v, r)$
  - $PK$ = receiver’s public key
  - $v$ = value
  - $r$ = randomly generated serial number (required to spend the note)
- Nullifier: Hash($r$)
  - “Nullifies” (spends) the note with serial number $r$

# Shielded transactions

- Miners maintain an append-only list of
  - Committed notes
  - Nullifier hashes

| Notes | Nullifiers |
|---|---|
| $\mathrm{Com}(\ pk_1, v_1, r_1\ )$ | $\mathrm{H}(r_3)$ |
| $\mathrm{Com}(\ pk_2, v_2, r_2\ )$ | |
| $\mathrm{Com}(\ pk_3, v_3, r_3\ )$ | |
| $\mathrm{Com}(\ pk_4, v_4, r_4\ )$ | |
| | |

# Shielded transactions

- To send $v_2$ from $pk_2$ to $pk_5$
  - Create a new note owned by $pk_5$

| **Notes** | **Nullifiers** |
|---|---|
| $\mathrm{Com}(\ pk_1, v_1, r_1\ )$ | $\mathrm{H}(r_3)$ |
| $\mathrm{Com}(\ pk_2, v_2, r_2\ )$ | |
| $\mathrm{Com}(\ pk_3, v_3, r_3\ )$ | |
| $\mathrm{Com}(\ pk_4, v_4, r_4\ )$ | |
| $\mathrm{Com}(\ pk_5, v_5, r_5\ )$ | |

# Shielded transactions

- To send $v_2$ from $pk_2$ to $pk_5$
  - Create a new note owned by $pk_5$
  - Nullify existing note

| Notes | Nullifiers |
|---|---|
| Com( $pk_1$, $v_1$, $r_1$ ) | H($r_3$) |
| Com( $pk_2$, $v_2$, $r_2$ ) | H($r_2$) |
| Com( $pk_3$, $v_3$, $r_3$ ) | |
| Com( $pk_4$, $v_4$, $r_4$ ) | |
| Com( $pk_5$, $v_5$, $r_5$ ) | |

# Shielded transactions

- To send $v_2$ from $pk_2$ to $pk_5$
  - Create a new note owned by $pk_5$
  - Nullify existing note

| Notes | Nullifiers |
|---|---|
| Com( $pk_1$, $v_1$, $r_1$ ) | H($r_3$) |
| Com( $pk_2$, $v_2$, $r_2$ ) | H($r_2$) |
| Com( $pk_3$, $v_3$, $r_3$ ) | |
| Com( $pk_4$, $v_4$, $r_4$ ) | |
| Com( $pk_5$, $v_5$, $r_5$ ) | |

# Shielded transactions

- To send $v_2$ from $pk_2$ to $pk_5$
  - Create a new note owned by $pk_5$
  - Nullify existing note
  - Create ZK proof
    - Prove (in Zero-Knowledge) that $r_2$ appears in list of committed notes
    - Prove (in Zero-Knowledge) that input value ($v_2$) equals output value ($v_5$)

| **Notes** | **Nullifiers** |
|---|---|
| Com( $pk_1$, $v_1$, $r_1$ ) | H($r_3$) |
| Com( $pk_2$, $v_2$, $r_2$ ) | H($r_2$) |
| Com( $pk_3$, $v_3$, $r_3$ ) | |
| Com( $pk_4$, $v_4$, $r_4$ ) | |
| Com( $pk_5$, $v_5$, $r_5$ ) | |

# Shielded transactions

- To send $v_2$ from $pk_2$ to $pk_5$
  - Create a new note owned by $pk_5$
  - Nullify existing note
  - Create ZK proof
    - Prove (in Zero-Knowledge) that $r_2$ appears in list of committed notes
    - Prove (in Zero-Knowledge) that input value ($v_2$) equals output value ($v_5$)
- Validating a transaction
  - Check that $\mathrm{H}(r_2)$ is not in the nullifier list
  - Check ZK Proof

| **Notes** | **Nullifiers** |
|---|---|
| $\mathrm{Com}(\ pk_1, v_1, r_1\ )$ | $\mathrm{H}(r_3)$ |
| $\mathrm{Com}(\ pk_2, v_2, r_2\ )$ | $\mathrm{H}(r_2)$ |
| $\mathrm{Com}(\ pk_3, v_3, r_3\ )$ | |
| $\mathrm{Com}(\ pk_4, v_4, r_4\ )$ | |
| $\mathrm{Com}(\ pk_5, v_5, r_5\ )$ | |

# Shielded transactions

- To send $v_2$ from $pk_2$ to $pk_5$
  - Create a new note owned by $pk_5$
  - Nullify existing note
  - Create ZK proof
    - Prove (in Zero-Knowledge) that $r_2$ appears in list of committed notes
    - Prove (in Zero-Knowledge) that input value ($v_2$) equals output value ($v_5$)
- Validating a transaction
  - Check that $\mathrm{H}(r_2)$ is not in the nullifier list
  - Check ZK Proof

| Notes | Nullifiers |
|---|---|
| $\mathrm{Com}(\ pk_1, v_1, r_1\ )$ | $\mathrm{H}(r_3)$ |
| $\mathrm{Com}(\ pk_2, v_2, r_2\ )$ | $\mathrm{H}(r_2)$ |
| $\mathrm{Com}(\ pk_3, v_3, r_3\ )$ | |
| $\mathrm{Com}(\ pk_4, v_4, r_4\ )$ | |
| $\mathrm{Com}(\ pk_5, v_5, r_5\ )$ | |

Validators can see you “spent” a note in the list of notes, but not which one

# Shielded transactions

- To send $v_2$ from $pk_2$ to $pk_5$
  - Create a new note owned by $pk_5$
  - Nullify existing note
  - Create ZK proof
    - Prove (in Zero-Knowledge) that $r_2$ appears in list of committed notes
    - Prove (in Zero-Knowledge) that input value ($v_2$) equals output value ($v_5$)
- Validating a transaction
  - Check that $\mathrm{H}(r_2)$ is not in the nullifier list
  - Check ZK Proof

| **Notes** | **Nullifiers** |
|---|---|
| $\mathrm{Com}(pk_1, v_1, r_1)$ | $\mathrm{H}(r_3)$ |
| $\mathrm{Com}(pk_2, v_2, r_2)$ | $\mathrm{H}(r_2)$ |
| $\mathrm{Com}(pk_3, v_3, r_3)$ | |
| $\mathrm{Com}(pk_4, v_4, r_4)$ | |
| $\mathrm{Com}(pk_5, v_5, r_5)$ | |

How do you give $r_5$ to owner of $pk_5$?

# Transmitting $r$

- $r_5$ is needed to spend the new note
- Getting $r_5$ to owner of $pk_5$ is complicated
  - Encrypt $r_5$ under $pk_5$ and broadcast
  - Sender still knows $r_5$ and could spend note first
- Exact method changed during different iterations of Zcash