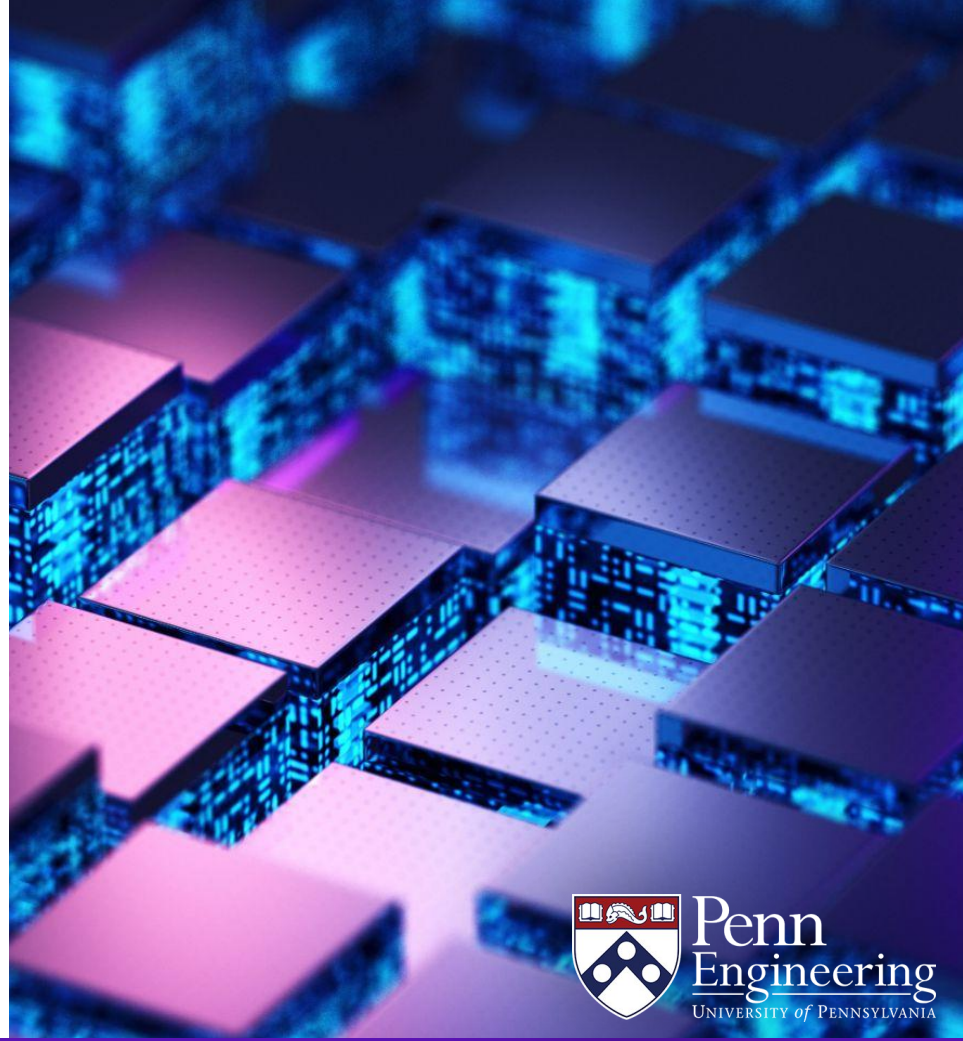
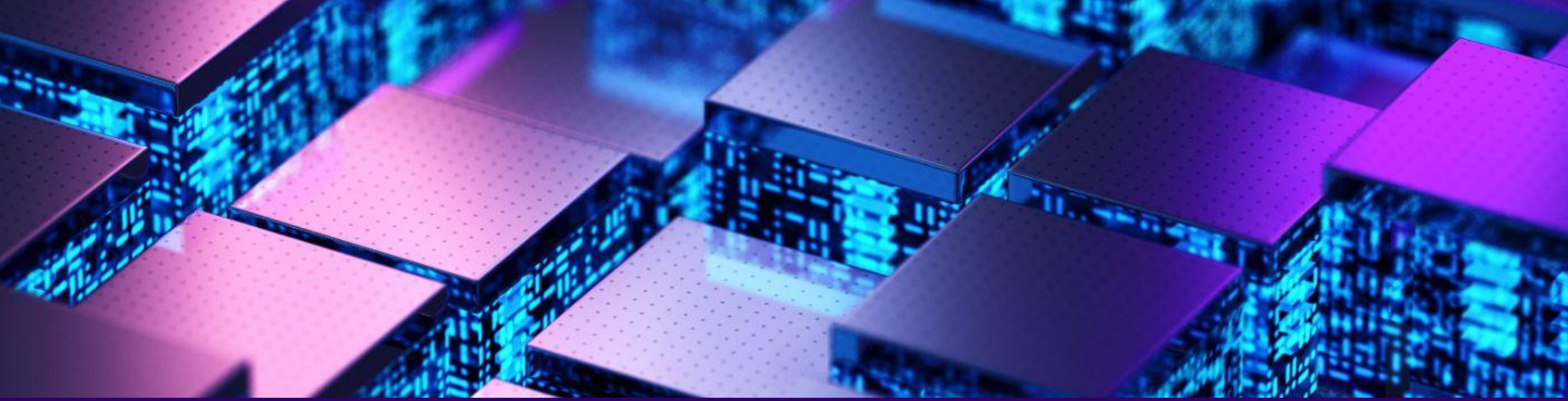
EAS 5830: BLOCKCHAINS

Applications of ZKPs

Professor Brett Hemenway Falk





Compact blockchains

Blockchains are large

- The size of the Bitcoin blockchain is over 400Gb
- Validating the state of the system requires reading the entire blockchain
- When a new TX is proposed, validators need to check whether input is in UTXO set
    - Requires keeping an up-to-date list of UTXOs

Compact blockchains

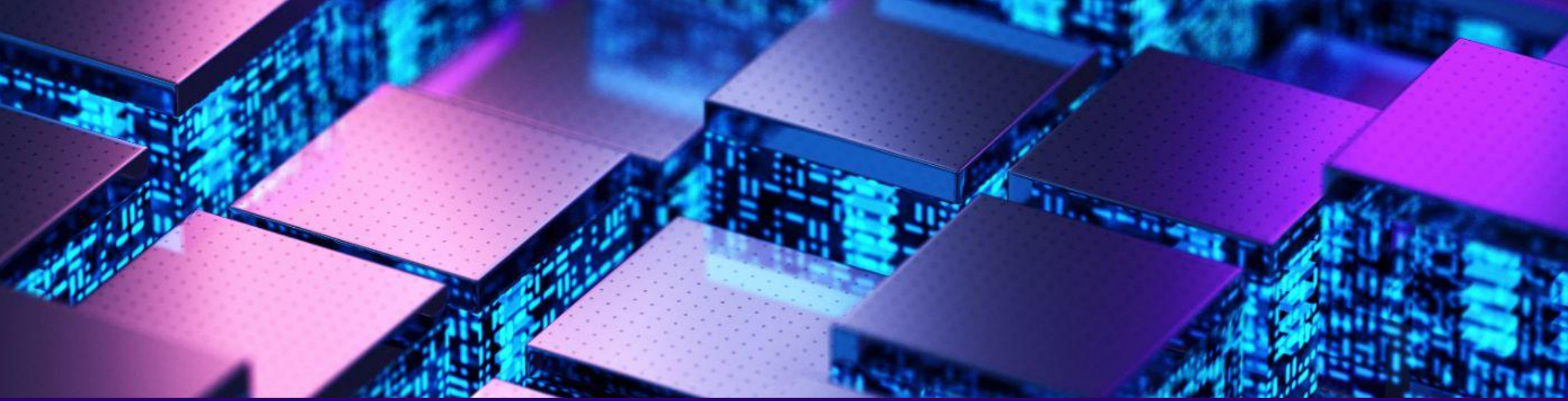
- Mina Protocol
- Uses account balance model (not UTXO model)
- Current state is a list of all account balances
- Each block contains a root to the Merkle Tree of all balances
- Each block contains a ZKP that the current state is a valid
  - Each block contains a proof that the current state results from a sequence of valid updates from the genesis block
  - Requires *incremental* ZKPs: Each block builds on the proof contained in the previous block

Compact blockchains

- Validating state of the system only requires downloading the last block
  - Extremely efficient!
- Actually learning all account balances requires downloading entire Merkle tree
  - Size proportional to number of accounts
- Wallets can keep track of single account balance efficiently
  - Track account balance
  - Merkle proof that current account balance is consistent with Merkle root

Compact blockchains

- ZK Rollups use the same idea
- Rollup state is small enough to be verified by a smart contract



Private transactions

Private transactions

ZK proofs can prove that a transaction is valid without revealing transaction details

Private ledgers

- Zcash (Layer 1)
- Tornado Cash (Smart Contracts)
- Aleo (Layer 1)
- ZKledger (Academic)
- Solidus (Academic)

ZKLedger

- Requires additively homomorphic encryption
- $\text{Enc}_{pk}(X) + \text{Enc}_{pk}(Y) = \text{Enc}_{pk}(X+Y)$
- Many efficient homomorphic cryptosystems
  - El-Gamal
  - Paillier
  - Lattice-based systems
    - PALISADE
    - HElib
    - SEAL
    - TFHE

ZKledger

	Bank 1	Bank 2	Bank 3	...	Bank n
Balance	Enc(b_1)	Enc(b_2)	Enc(b_3)		Enc(b_n)

- Permissioned system
  - Fixed number of users
- State of the system is a list of encrypted account balances

ZKledger

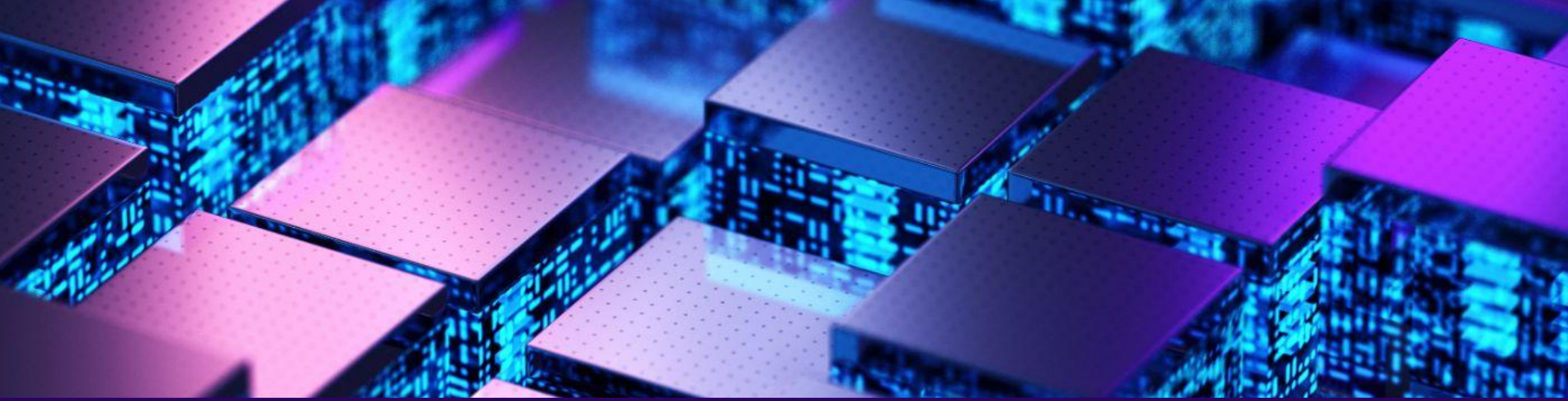
	Bank 1	Bank 2	Bank 3	...	Bank n
Balance	Enc(b_1)	Enc(b_2)	Enc(b_3)	...	Enc(b_n)
TX	Enc(-1)	Enc(0)	Enc(1)	...	Enc(0)

- Update is an encrypted vector
- ZK proof that the update is valid
  - Vector sums to 0
  - You know the private keys for all accounts being decremented
  - No account balance ends up negative

ZKledger

	Bank 1	Bank 2	Bank 3	…	Bank n
Balance	$\text{Enc}(b_1\text{-}1)$	$\text{Enc}(b_2)$	$\text{Enc}(b_3\text{+}1)$	…	$\text{Enc}(b_n)$

- Updating requires homomorphic encryption
- $\text{Enc}(x) + \text{Enc}(y) = \text{Enc}(x+y)$



Privacy-preserving credentials

List of Decentralized Identity Tools

Discover 57 Decentralized Identity Tools across the most popular web3 ecosystems with Alchemy's Dapp Store. Also explore related collections including Web3 Credential Tools, DAO Reputation Tools, Web3 Authentication Tools. Is your project missing from the list? Submit your project to Alchemy's list of Decentralized Identity Tools and we'll review it!

Validating credentials

Validating credentials

Validating credentials

Validating credentials

Owner

Issuer

Verifier

Validating credentials

Owner

Issuer

- Casino wants to verify you are 21 years old
- User may not want to reveal
  - name
  - home address
  - weight
  - etc

Verifier

Validating credentials

Issuer

Owner

- DMV Signs DL
- Owner provides Encrypted DL to Casino, and makes ZKP::
  - The DL was signed by DMV
  - The DOB is more than 21 years ago

Verifier