CSE 5095-007: Blockchain Technology

Lecture 16

Anonymity and Privacy in Cryptocurrencies

Ghada Almashaqbeh
UConn - Fall 2022

Outline

- Anonymity and privacy in cryptocurrencies.
  - Background.
  - Centralized mixers.
  - Cryptography Review.
    - Commitments.
    - Zero knowledge proofs.
  - Decentralized mixers - Zerocoin.
  - Anonymous and private payments.
    - UTXO model - Zerocash.
    - Account model - Zether.

Anonymity and Privacy I

- Sensitive information in a cryptocurrency system:
  - Addresses of senders and recipients.
  - Transaction (currency) value.
  - Account balance (for these that use the account model).
  - Executed code (scripts or smart contracts).
  - Inputs and outputs of this executed code.
- Anonymity.
  - Hiding the addresses of senders and recipients.
- Privacy preserving:
  - Generally, it applies to the last four items in the list above.

Anonymity and Privacy II

- In some sources,
  - Hiding identities is also considered a privacy-preserving issue.
  - Hiding balances and transaction values are referred to as confidentiality.
    - E.g., confidential transactions; those with encrypted currency values.
- We will refer to these as:
  - ***Private payments.*** Currency transfer transactions that hide values and balances.
  - ***Secure (or privacy-preserving) function evaluation.*** Computing over private inputs, and possibly, producing private output.
  - ***Function privacy.*** Hiding the function (scripts or code) itself.
  - ***Anonymity.*** Hiding user identities.

Is Bitcoin Anonymous?

- Believed to be.
  - No real identities are required.
  - Users use random-looking keys as pseudonyms.
  - It is advised to generate a new key pair for each new transaction.

WikiLeaks

Bitcoin

Bitcoin is a secure and anonymous digital currency. Bitcoins cannot be easily tracked back to you, and are safer and faster alternative to other donation methods. You can send BTC to the following address:

1HB5XMLmzFVj8ALj6mfBsbifRoD4miY36v

Various sites offer a service to exchange other currency to/from Bitcoins. There are also services allowing trades of goods for Bitcoins. Bitcoins are not subject to central regulations and are still gaining value. To learn more about Bitcoins, visit the website (https://bitcoin.org) or read more on Wikipedia.

For a more private transaction, you can click on the refresh button above to generate a new address

Source: https://shop.wikileaks.org/donate

No it is not …

- The blockchain is public.
  - Transactions do not hide addresses of senders and recipients.
- Transactions linkability.
  - Track transaction flow to infer the real identities of the involved parties.
    - Cluster Bitcoin addresses into entities, link them to identities and/or Bitcoin addresses posted by their owners on forums, blogs, etc., [Reid et al. 2014]
    - Link this flow to users' IPs [Koshy et al. 2014].
      - Here, the use of anonymous communication protocols (e.g., Tor) could be useful. But anonymity is based on the security guarantees of such protocols (recall exit and entry points in Tor see the flow in the clear).

Is Bitcoin Private?

- Also NO.
  - Again, its blockchain is public.
  - Values of transactions are recorded in the clear.
  - Transaction scripts (locking and unlocking) are publicly known and logged in the clear as well.
  - Scripts operate on public inputs and produce public outputs.

How about Ethereum?

- For Ethereum, same as Bitcoin, it is more about functionality extension rather than privacy/anonymity.
- The account model requires different privacy/anonymity techniques than those used in the UTXO model.
- Having arbitrary smart contracts deployed by users raises the expectations.
  - Can these contracts operate on private inputs and produce private outputs?
    - Can existing techniques (e.g., MPC) be used here?
  - Can we preserve the privacy of the code itself? (i.e., hide the performed computation as well.)

Does Anonymity/Privacy Matter?

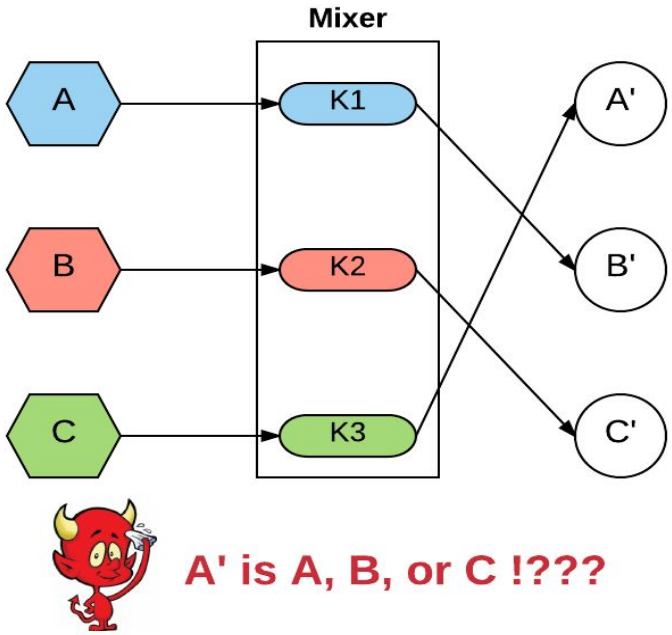
- Just like traditional banking systems, we desire to hide our financial activities when needed/possible.
  - Blockchain records are public, anyone can access them at any time.
- Storing and processing sensitive data.
  - Blockchain-based applications for medical records, trading, auction, voting, etc.
- Without anonymity/privacy, one my forgo the advantages of employing a blockchain in such highly sensitive applications.
  - Front running in auctions, censorship in voting, etc.
- Sometimes in cryptocurrencies coins get tainted.
  - People reject coins that have some undesirable history.
  - But currency is supposed to be fungible in order to serve its basic purposes!

Potential Solutions

- Mixing services (mainly in the context of Bitcoin).
  - Centralized.
  - Decentralized - Zerocoin.
- Anonymous/private cryptocurrencies..
  - UTXO model.
    - ZeroCash - an extension of Bitcoin.
  - Account model.
    - Zether - a token on top of Ethereum.

Mixers

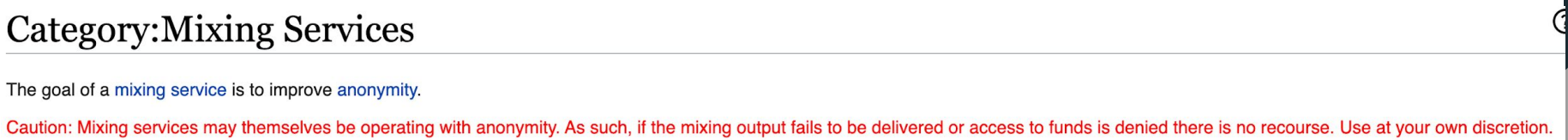
Create Anonymity Sets!



- Break transaction linkability.
  - Participants send their coins to some entity, the mixer (or tumbler).
  - The mixer shuffles these coins and return them to back to the participants.
    - Each party gets same value back but from a different owner (users use fresh addresses to receive these).

Centralized Mixers

From Bitcoin wiki (https://en.bitcoin.it/wiki/Category:Mixing_Services)


Category:Mixing Services

The goal of a mixing service is to improve anonymity.

Caution: Mixing services may themselves be operating with anonymity. As such, if the mixing output fails to be delivered or access to funds is denied there is no recourse. Use at your own discretion.

- Everything is controlled by a trusted party.
  - Parties send their coins with a promise to get them back.
  - Huge trust risk, will the mixer ***return the coins back***?
    - Several theft incidents over the past years.
- The mixer has a full record of which coins were sent to who.
  - It has all transaction linkability information.
  - Will it comply and fully ***delete this record***?
- Do we trust the mixer to randomly shuffle coins?
  - May send coins in a non-random manner allowing deanonymization.

Mixcoin[Bonneau et al., 2014]

- Although anonymous cryptocurrencies were already out there, the goal is to have something efficient and fully compatible with Bitcoin.
- Add accountability to expose theft.
  - A mixer issues a warranty to return the coins.
    - If it does not, the user discloses this warranty, and hence destroying the reputation of the mixer.
  - The mixer creates an escrow address for each party to deposit her coins.
  - Later, the mixer shuffles the escrows and sends each user an equal amount of her coins back (to new fresh addresses).
- Calibrate incentives so that rational mixers will act honestly.
- Propose the use of a series of mixers to reduce the probability of local records risk.

Mixcoin[Bonneau et al., 2014]

- Still same security risks of a centralized mixer.
  - Theft.
    - Maybe it is worth it; destroy reputation but run away with a huge wealth.
  - Delays.
    - Users have to wait for long time to get coins back (to have a large anonymity set).
  - Local records exposure.
    - Mix networks (series of mixers) may not be always available.

- Detour -
Commitments
Zero Knowledge Proofs

Commitments

- A commitment scheme consists of three algorithms:
  - Setup: takes a security parameter $\boldsymbol{\lambda}$ as input and outputs public parameters **pp**.
  - Commit: takes inputs the public parameters **pp**, a message ***m***, and randomness ***r***, and outputs commitment ***c***.
- Opening a commitment ***c*** is simply revealing ***m*** and ***r***, and then verifying that **commit(*m; r*) = *c***.
- Acts like a digital envelope; commit to ***m*** (like a guarantee that ***m*** exists) but without revealing anything about ***m*** before the opening phase.

Commitments - Properties

- Security properties:
  - Hiding: A commitment ***c*** does not reveal anything about ***m***.
  - Binding: A commitment ***c*** to message ***m*** cannot be opened to a different $m' \neq m$
- Homomorphic Commitments:
  - An additively homomorphic commitment scheme is a commitment scheme such that given m_1, m_2, r_1, r_2 we have:

    $$\textbf{commit}(m_1, r_1) + \textbf{commit}(m_2, r_2) = \textbf{commit}(m_1+m_2; r_1+r_2)$$

    (Note: it could be the case that multiplying two commitments produces a commitment to the addition of the two messages. The above is just a symbolic way to represent the property.)

Hash Commitments

- Relies on the security of collision-resistant hash functions.
- Pick a salt ***s***, then compute a commitment to ***m*** as (***H*** is a collision resistant hash function):
  - $c = H(m||s)$
- Hiding: inverting a hash is hard.
- Binding: opening a hash to another $m' \neq m$ requires finding a collision.

Pedersen Commitments

- Work in a cyclic group $G = \langle g \rangle$ (**g** is the generator) of order ***p*** in which the Discrete Log Problem is believed to be hard.
  - Given a g^x it is hard to find ***x***.
- Choose two generators for the group; **g, h**
  - No one knows the discrete log of **g** with respect to **h** and vice versa.
- Commit to a message ***m***:
  - Select a random ***r*** from ***{0, …, p-1}***.
  - Compute $c = g^m h^r$
- Open a commitment:
  - Reveal ***m*** and ***r***

Pedersen Commitments - Security

- Hiding: $\mathbf{h}^r$ is a random element in the group $\boldsymbol{G}$ and so is the commitment $\mathbf{g}^{\boldsymbol{m}}\mathbf{h}^r$
  - A random commitment value does not reveal anything about $\boldsymbol{m}$
- Binding: reduced to DLP,
  - An attacker who can open a commitment to $\boldsymbol{m' \neq m}$ can be used to construct an attacker to break DLP.

Pedersen Commitments - Additively Homomorphic

- Given $c_1 = g^{m1}h^{r1}$ and $c_2 = g^{m2}h^{r2}$
  - $\textbf{commit}(m_1+m_2, r_1+r_2) = c_1c_2 = g^{(m1+m2)}h^{(r1+r2)}$
- Very useful for tracking balances of accounts in a private way, or total of inputs for a transaction.
  - I own x coins, stored in a hidden way on the blockchain
    - as a commitment: $g^x g^{r1}$
  - Bob sent me y coins, also in a hidden way as a commitment: $g^y h^{r2}$
  - The miners can update my account balance without revealing anything.
    - Simply multiply the two commitments together.
    - More like performing arithmetic operations on hidden (secret) data.

Zero Knowledge Proofs I

- Allow a prover to convince a verifier that it knows a witness ω for some statement x without revealing anything about the witness.
  - E.g. proving that a given commitment is for an integer y that is larger than some value z, without revealing the exact value of y.
- A zero knowledge proof system is composed of three algorithms:
  - **Setup:** takes the security parameter κ and specifications of the NP relation (which determines the set of all valid statements x) for which proofs are to be generated, and outputs a set of public parameters pp.
  - **Prove:** takes as inputs pp, a statement x, and a witness ω for x and outputs a proof π proving correctness of x (that it satisfies the NP relation).
  - **Verify:** takes pp, statement x, and π and outputs 1 if the proof is valid, and 0 otherwise.

Zero Knowledge Proofs II

- At a high level:
  - All conditions needed to satisfy the NP relation describing the teh set of all valid x statements are represented as a circuit.
  - A valid proof will be generated upon providing valid inputs that satisfy this circuit.
    - Some of these inputs could be public, which the verifier will use in the verification process,
    - while others are private, which constitute the witness ω that only the prover knows.

Zero Knowledge Proofs–Security

- A secure zero knowledge proof system must satisfy three properties:
  - **Completeness:** ensures that any proof that is generated in an honest way will be accepted by the verifier.
  - **Soundness (or proof-of-knowledge):** states that if a verifier accepts a proof for a statement x then the prover knows a witness ω for x. Put differently, this means that a prover cannot convince a verifier of false statements.
  - **Zero knowledge:** ensures that a proof π for a statement x does not reveal anything about the witness ω.

Additional Properties

- **Succinctness:**
  - The proof size is constant and verification time is linear in the size of the input, regardless of the circuit size representing the underlying NP relation.
  - A proof systems that satisfies completeness, soundness, zero knowledge, and succinctness is denoted as zk-SNARK (zero knowledge succinct non-interactive argument of knowledge).
- **Non-interactivity:**
  - a prover can generate a proof on its own and send it in one message to the verifier.
  - A verifier can verify the proof on its own without any further interaction with the prover.
  - Very desirable property in the blockchain setup.

Schnorr's Protocol

- A proof of knowledge of a discrete logarithm
  - Such as the secret key corresponding to some public key in EC.
- Work in a cyclic group $G = \langle g \rangle$ (g is the generator) of order p in which the Discrete Log Problem is hard.
- For $y = g^x$, one can prove knowing x as follows:
  - First round: prover chooses r in $\{1, \ldots, p-1\}$, prover commits to r by sending $c = g^r$ to the verifier.
  - Second round: verifier replies with a challenge t (selected at random from $\{1, \ldots, p-1\}$).
  - Third round: prover sends $s = r + tx \pmod p$
- Verifier accepts if $g^s \equiv cy^t$
- Usually called a Sigma protocol.
  - Since it consists of three rounds or moves: commit/challenge/respond

Fiat-Shamir Heuristic

- Converts an interactive protocol into a non-interactive one.
  - The prover computes a proof, sends it to the verifier, and the verifier checks the proof
    - One round of communication.
- Works in the random oracle model.
  - Instead of waiting for a random challenge from the verifier, the prover computes it using a random oracle (usually using a hash function modeled as a random oracle).
- For example, in Schnorr's protocol:
  - Instead of having the verifier choose a random challenge ***t***, the prover computes ***t = H(g, y, c)***
  - Then, the verifier sends both ***c*** and ***t*** to the prover to check.

- Zerocoin -
A Decentralized Mixer

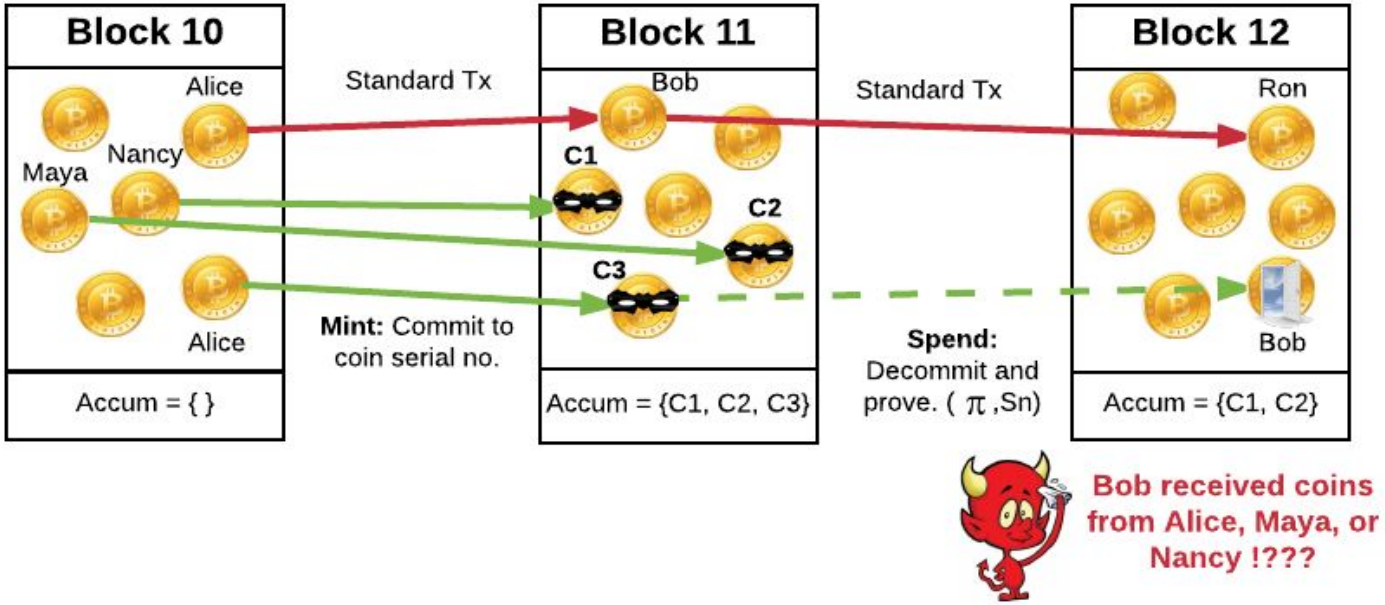
Zerocoin [Miers et al., 2013]

- An extension to Bitcoin to support anonymity, i.e., break the link between the transactions.
- Turn a bitcoin into a zercoin.
  - This will create a pool of zercoins forming an anonymity set.
  - This is called minting.
- Then a zercoin can be sent to a new address.
  - Goes back to Bitcoin.
  - This is called spending.
- Thus, it creates a decentralized mixer without any trusted entity.
  - Users have full control of their coins.
  - The larger the pool, aka anonymity set, the greater the anonymity level.

Zerocoin

- General idea:
  - Creates an anonymity set, a pool of hidden coins in the form of commitments.
    - Clients mint anonymous coins.
    - The coin is simply a random serial number.
  - Spending a coin from that pool does not reveal the owner.
    - It reveals the coin's serial number.
      - Needed to prevent double spending.
    - Provides a NIZK proof that a coin with a specific serial number belongs to the pool.
- Why is it anonymous?
  - NIZK does not provide any info beyond that there exists a zercoin with a given serial number on the blockchain.

Zerocoin Pictorially



Zerocoin Protocol I

- Consists of three algorithms (other than setup):
  - **Mint:** create a zerocoin (i.e., a commitment to some random serial number SN) along with a trapdoor to allow spending the coin later.
  - **Spend:** use the trapdoor to spend C, which reveals the SN of the coin and produces a proof that C is a valid zercoin.
  - **Verify:** verify the proof and the validity of the commitment.
- Minting is sending some bitcoin amount to an escrow (basically an output).
  - The unlocking/locking script relies on providing a valid spend proof.
- Thus, when spending a zercoin, the owner can grab any UTXO to get his/her Bitcoins back.
  - Just like what a mix service does, but in a distributed way.

Zerocoin Protocol II

- Use an efficient RSA one way accumulator to create the anonymity set (or zerocoins pool).
  - Accumulate all zerocoins C1, …, Cn into a small data structure called an accumulator A.
  - During the accumulation process, a membership witness is produced for each zerocoin.
- To spend a coin C, prove the knowledge of a witness such that $C \in A$ and that C opens to serial number SN.

Zerocoin Protocol III

- Accumulator properties:
  - No trusted setup/parties. It can be publicly computed.
  - Bind the computing party to the values in the set.
  - Support efficient non-interactive zero knowledge proof systems.
- Utility of the accumulator.
  - Allows producing a logarithmic size proof instead of a linear one:
    - uses an OR statement represented by the satisfiability of a circuit that outputs 1 if C equals to C1 or C2 or C3 … or Cn

Zerocoin

- Disadvantages:
  - Only hides the originator, but not the destination or the transfer amount.
  - Computationally heavy.
    - A proof has a size of almost 45 KB and requires 450 ms to verify.
  - Not fully compatible with Bitcoin; it requires network protocol changes, which may lead to a hard fork.

- Zerocash -
Private/Anonymous Payments
(UTXO Model)

Zerocash [Ben Sasson et al., 2014]

- An extension to Bitcoin to support anonymity and privacy.
  - It is a new cryptocurrency; builds on Bitcoin but requires major changes to support its functionality.
- Addresses the drawbacks of Zerocoin.
  - Anonymity for destination addresses.
  - Private (or hidden) transfer amount.
  - Direct private transfers.
  - Proof size is less than 1 kB with a verification time of around 6 ms.
- Utilizes recent advances in zk-SNARKS (*zero knowledge - Succinct Non-interactive ARgument of Knowledge*) to optimize proofs needed for spending private coins.

Zerocash Protocol I

- Consists of three main algorithms (apart from setup, create addresses, and receive coins):
  - **Mint.** Mints a new zerocoin. A coin is a commitment to a random serial number (and other related info). An equivalent bitcoin amount is added to a backing pool of coins (like an escrow).
    - All coin commitments are tracked, or accumulated, using a Merkle tree.
    - A mint is from a public coin to a private coin.
  - **Pour.** Spends zerocoins where the destination could be a private output (coin commitments) or a public output (currency value destination address are public).
  - **Verify.** Verify the validity of the transactions (mint/pour), which involves verifying the attached zk-SNARK.

Zerocash Protocol II

- A pour transaction must satisfy a pour statement in order to be valid.
  - The coin (or mint transaction) is well-formed.
  - Public keys and coins serial numbers are derived appropriately (using some PRF constructions).
  - The sender knows the secret key of the public key that owns the coin.
  - The coin commitment appears as a leaf node in the coins Merkle tree.
  - Total input values equals to total output values.
  - A valid signature over these values is provided
- Again, pouring or spending a coin reveals its serial number to prevent double spending.

Zerocash Anonymity/Privacy

- Sender or originator anonymity.
  - Similar to Zerocoin; a pool of coin commitments represents an anonymity set for the sender.
- Destination or recipient anonymity.
  - A recipient address (for a private output) is not stored in the clear on the blockchain. It is part of the coin commitment.
- Private transfer amount.
  - Similar to the recipient address, it is hidden inside the coin commitment.
- To allow a recipient of a private coin to spend it, additional information are sent encrypted to the recipient as part of the pour transaction.
  - Only the recipient can decrypt this ciphertext.

Zerocash Efficiency

- Spending coins, i.e., pour transactions, requires a zk-SNARK proof.
- The previous NP statement (POUR) is represented as an arithmetic circuit.
  - A zk-SNARK is given for the satisfiability of this circuit.
- Performance optimizations can be achieved by carefully designing this circuit to produce a smaller one.
  - All commitment schemes, PRFs, and hashes are instantiated using SHA256.
  - Design an optimized circuit for SHA256 verification, and then use it in building the POUR statement circuit.

- Zether -
Private/Anonymous Payments
(Account Model)

Zether [Bünz et al., 2020]

- An extension to Ethereum to provide anonymous and confidential (or private) payments.
  - Thus it works in the account-based model.
- Private payments rely on two building blocks.
  - ElGamal encryption.
    - Account balances are encrypted.
    - ElGamal is additively homomorphic, hence, adding/deducting currency to/from an account can be done by operating on the ciphertexts (the miners can do that).
  - ∑-Bullets.
    - An optimization of Bulletproofs, an efficient ZKP, to make them more interoperable with Sigma protocols (3 move proof protocols), and thus can be made non-interactive in the random oracle model.
    - Used to prove currency spending transactions.

Zether Protocol I

- Instantiated as a token on top of Ethereum.
  - Its token is denoted as ZTH.
  - It does not introduce any changes to Ethereum's protocol.
- In the Zether smart contract, users can create accounts to hold private balances.
  - Lock Ether in this account to get equivalent ZTH coins.
- All currency transfers must ensure that a MAX value of account balance is not exceeded.
  - To prevent any overflow of private account balances.
- Beside private payments, anonymity can be supported in a similar manner to previous schemes.
  - An anonymity set is chosen by the sender, with a proof attesting that the sender/receiver are members of this set.

Zether Protocol II

- Five main algorithms (apart from setup, create addresses, etc.):
  - **Fund.** Used to fund a private account.
  - **Transfer.** Used to create a private transfer transaction (from a private account to another).
  - **Burn.** Transfer the full amount of a private account balance to a public account.
  - **Lock/unlock.** Locks an account to some address, which transfers the ownership to that address (unlock nullifies that).
    - These are needed to allow interoperability with other smart contracts.
- All these algorithms create transactions corresponding to the required operations that will be processed by the Zether smart contract.

Zether Protocol III

- Burn and transfer requires ZKPs over the private data.
  - **Burn.** Need to prove that
    - the issuer is indeed the owner of the account (knows the secret key associated with the account public key),
    - and that revealed currency amount is equal to the account balance (recall that that balance is encrypted).
  - **Transfer.** It provides two ciphertexts; one to subtract ***b*** coins from the issuer's account, and the other to add ***b*** coins to the recipient's account. Need to prove that
    - Both ciphertexts are well-formed and encrypt the same value.
    - The issuer's balance can cover the transfer amount.
    - ***b*** is a positive value.

Handling Concurrency

- Front running is a serious problem in this setup.
  - After producing some ZKP with respect to an account state, someone may transfer currency to this account.
    - This will change the account state, and hence, invalidate any pending ZKP.
  - This is called front running.
- Addressed by introducing the notion of pending transfers and epochs.
  - An epoch is k consecutive blocks.
  - All transfers to private accounts are put on hold until the end of the epoch.
    - Will be rolled over to the account in the last block of the epoch.
  - Works under the assumption that any private transfer transaction issued in an epoch will be processed in that epoch as well.

Applications

- Usually requires interoperability with other smart contracts.
  - Like a voting contract for example.
- To prevent participants from changing their balances, an account can be locked to the application smart contract.
  - In a sealed bid auction, the bid amount is sent to a private account, then lock this account to the auction contract.
  - This prevents the bidder from spending the bid currency.
  - After opening the bids, the account will unlock the accounts of the lost bids and transfer the winning bid amount to the seller.
- Other applications: stake voting and private payment channels.
- However, being restricted to private/anonymous payments (rather than computations) limits the scope of applications.

References

- [Reid et al. 2014] Reid, Fergal, and Martin Harrigan. "An analysis of anonymity in the bitcoin system." In Security and privacy in social networks, 2013.
- [Koshy et al. 2014] Koshy, Philip, Diana Koshy, and Patrick McDaniel. "An analysis of anonymity in bitcoin using p2p network traffic." In Financial Cryptography, 2014.
- [Bonneau et al., 2014] Bonneau, Joseph, Arvind Narayanan, Andrew Miller, Jeremy Clark, Joshua A. Kroll, and Edward W. Felten. "Mixcoin: Anonymity for Bitcoin with accountable mixes." In Financial Cryptography, 2014.
- [Miers et al., 2013] Miers, Ian, Christina Garman, Matthew Green, and Aviel D. Rubin. "Zerocoin: Anonymous distributed e-cash from bitcoin." In IEEE S&P, 2013.
- [Ben Sasson et al., 2014] Sasson, Eli Ben, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. "Zerocash: Decentralized anonymous payments from bitcoin." In IEEE S&P, 2014.
- [Bünz et al., 2020] Bünz, Benedikt, Shashank Agrawal, Mahdi Zamani, and Dan Boneh. "Zether: Towards privacy in a smart contract world." In Financial Cryptography, 2020.