

CPSC 3640/5400: Decentralized Payments, Contracts, and Finance for Humans and AI

Lecture 10: DeFi advantage: Privacy-preserving asset transfers

Fan Zhang
Yale University
2026 Fall

Recap: Bitcoin has weak privacy

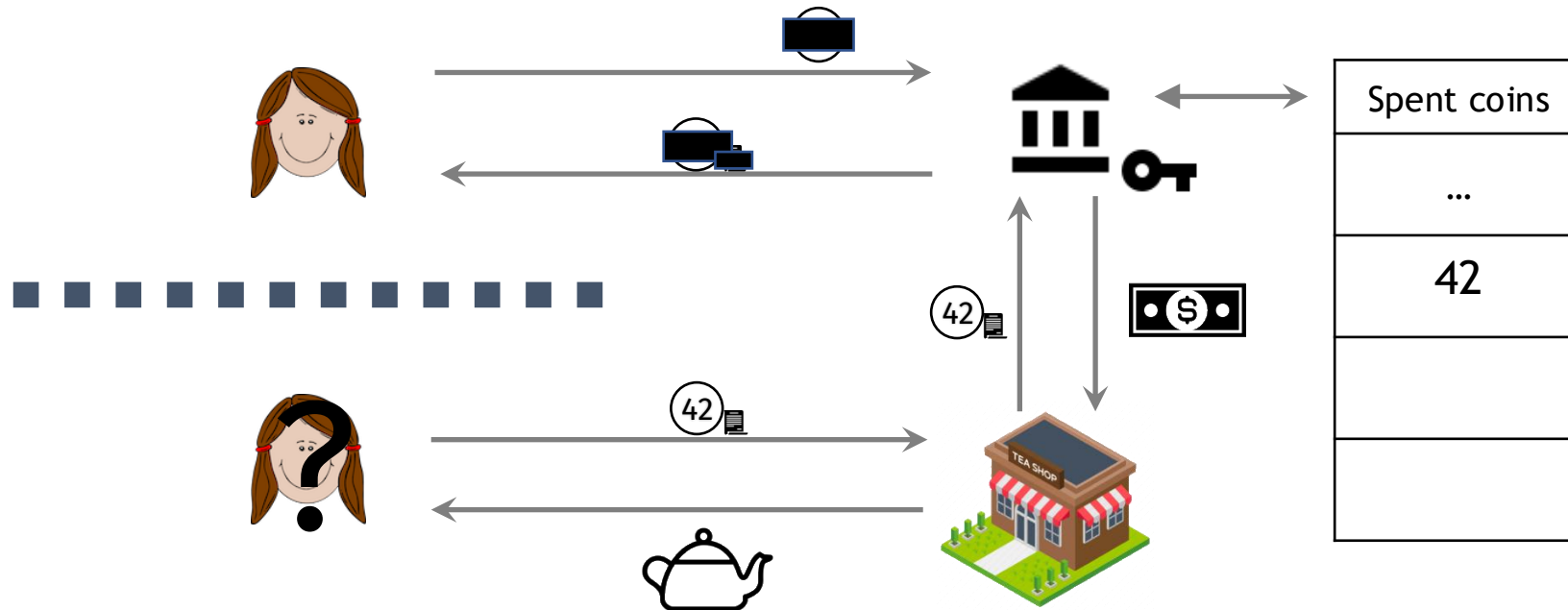


- All Bitcoin transactions are public
- Users use pseudonyms (PKs) but can be potentially deanonymized by transaction history
- Key technique: cluster PKs and UTXOs using various heuristics
 - Can you recall some?
- Companies doing this professionally



Recap: private digital money and e-cash

- Privacy goals of e-cash: cash-like privacy
- E-cash uses **blind signature** to decouple issuance & spending

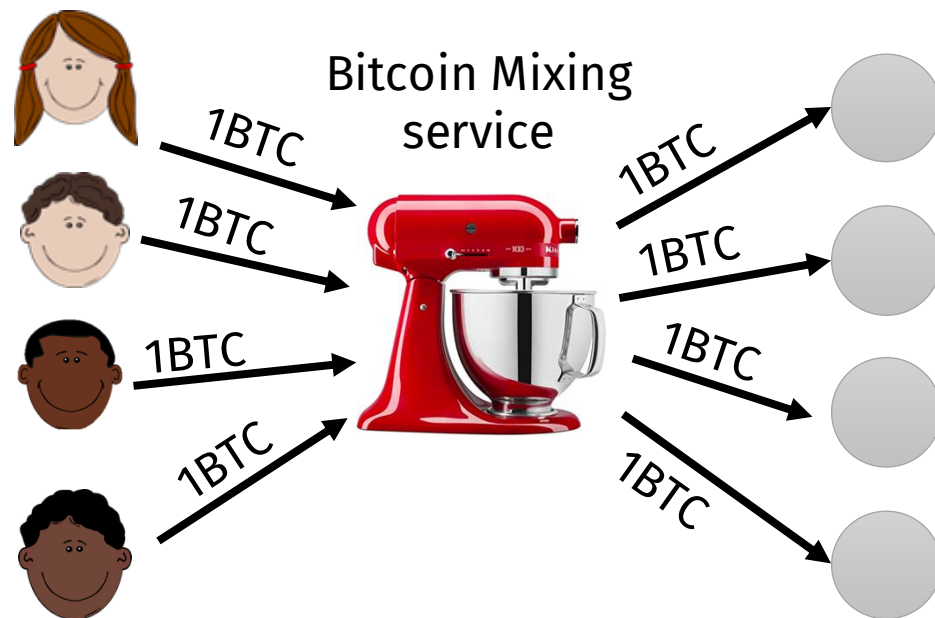


Recap: private digital money and e-cash

- E-cash did not get real-world adoption
 - Perhaps due to the lack of interest in privacy
 - Or the lack of “trusted banks”
- Bitcoin and other cryptocurrencies doesn't require a trusted bank, but they are not anonymous either.
- **How to make Bitcoin more private?**
 - CoinJoin, Monero, Zerocash
- **How to make Ethereum (or ERC-20) more private?**
 - It's in fact harder (we will see Tornado Cash)

Today

Fundamental idea: Mixing



Breaks clustering heuristics: Each output links to each input with equal probability $\frac{1}{4}$

1. Each user sends (input, *fresh address*) to the service
2. The service creates a transaction with *N inputs* and *N outputs to fresh addresses* in a *random* order
3. Each user sends the signature for *her input*

Fundamental idea: Mixing



- Drawbacks?

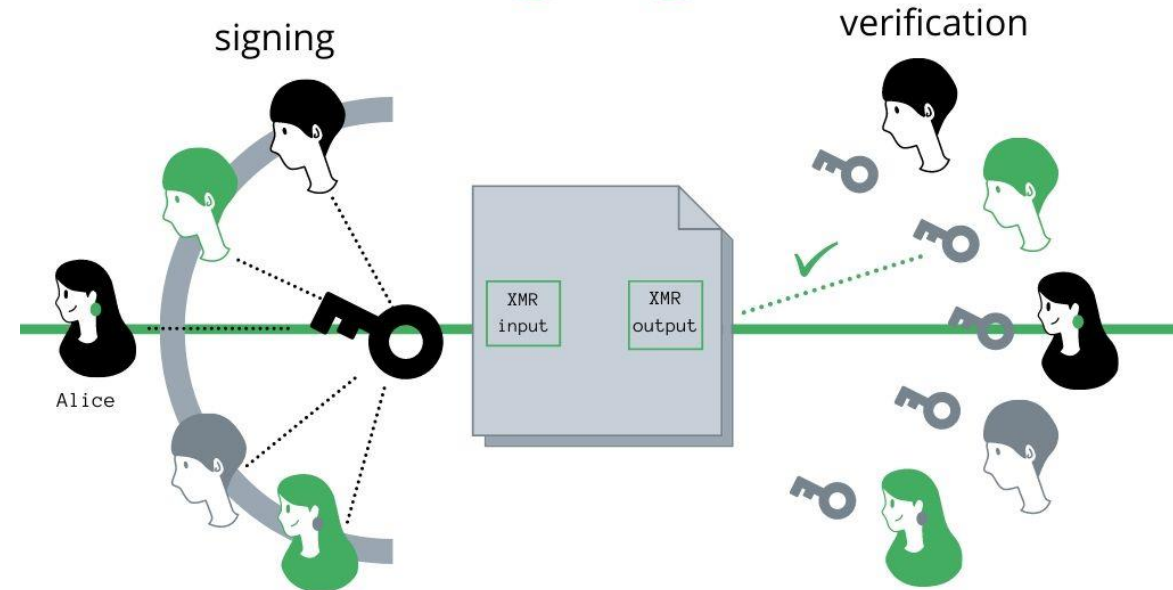
- No privacy against the mixing service
- Anonymity-latency tension: stronger anonymity requires longer wait

Monero: Bitcoin with built-in
mixing

Ring Signature

- Special kind of signature that hides the signer among decoys
- $\sigma = \text{Sign}(m, sk_i, pk_1, \dots, pk_n)$
 - sk_i is the secret key of pk_i
 - pk_j where $j \neq i$ are decoys
- $\text{Verify}(m, \sigma, pk_1, \dots, pk_n)$

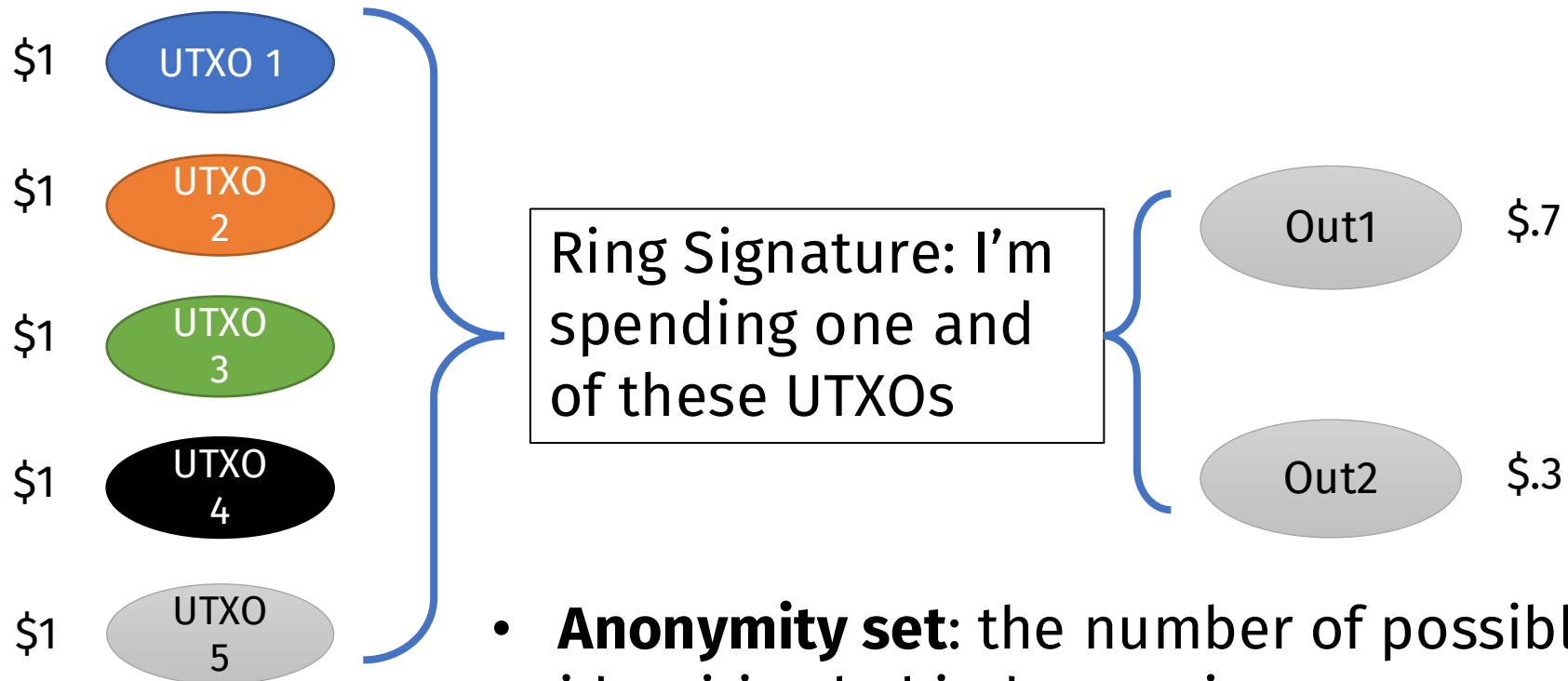
Monero Ring Signatures



Alice wants to send XMR. The protocol automatically combines her address with random decoy addresses to form a single ring signature which signs the transaction. Observers cannot tell who initiated the exchange, but can verify the transaction with cryptographic key images.



CryptoNote, Monero (simplified)



- **Anonymity set:** the number of possible identities behind an action.
 - **In this case 5.**
 - The larger the better (what's the ideal?)
- Ring signatures don't scale to many decoys.

Mixing with a *smart contract*

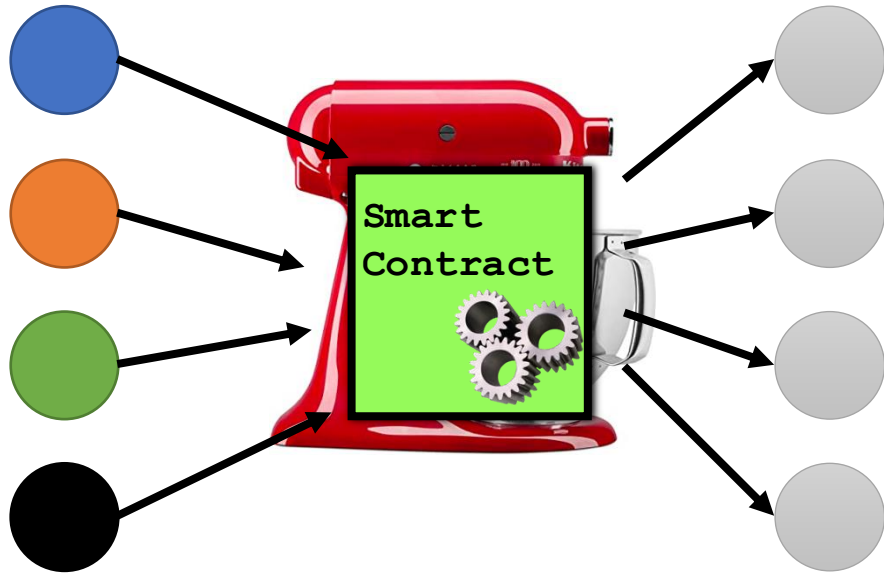
tornado

CASH



Setup (only allow for fixed value)

- Smart contract (the mixer) maintains
 - A list of deposits
 - A list of spent coins



Smart Contract State
(in practice, the Merkle roots)

Deposit	Spent
CM_1	SN_1
CM_2	SN_2
...	...

Commitments



- Like a secret note sealed in an envelope
 - $\text{Commit}(s) \rightarrow (c, \text{opening})$
 - s = the secret note, c = sealed envelop, opening = the opening key
 - $\text{Open}(c, s, \text{opening}) \rightarrow \{0,1\}$
 - Testing if s is the correct note in c

Hiding: c does not reveal anything about s

Binding: a given c cannot be opened to two different s

Example:

- $\text{Commit}(s)$: sample r , output $(c = H(s||r), r)$
- $\text{Open}(c, s, r)$: return $c == H(s||r)$

Second ingredient: zero-knowledge proofs (ZKPs)

A powerful tool for proving **knowledge** of inputs that satisfy a statement **without revealing the inputs**

More formally: Given a function $f(x)$ known to prover & verifier

- Prover can generate a short proof that convinces the verifier that she knows w such that $f(w)=0$.
- Soundness: prover cannot convince verifier unless she indeed knows w
 - defining what it mean for someone to know something is a challenge
- zero-knowledge: proof does not leak any information about w

Example: “I have a solution to Sudoku puzzle P”

- Call this puzzle P

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Hard to solve but easy to verify a solution S

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Call the verification algorithm $\text{SudokuVerify}(P, S) \rightarrow \{0, 1\}$

Example: “I have a solution to Sudoku S”

To do so, we just need to express the statement in the following function:

```
f(x) {  
    Output 0 if SudokuVerify(P, x) == 1  
}
```

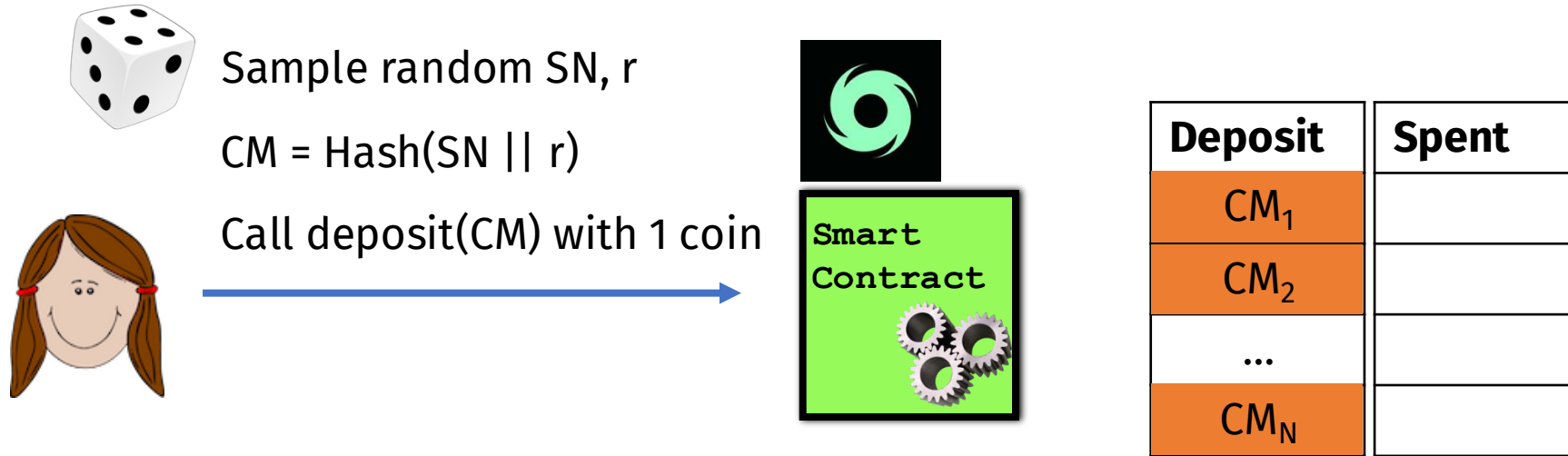
Then ask prover to

In general, **no matter how complex f** is:

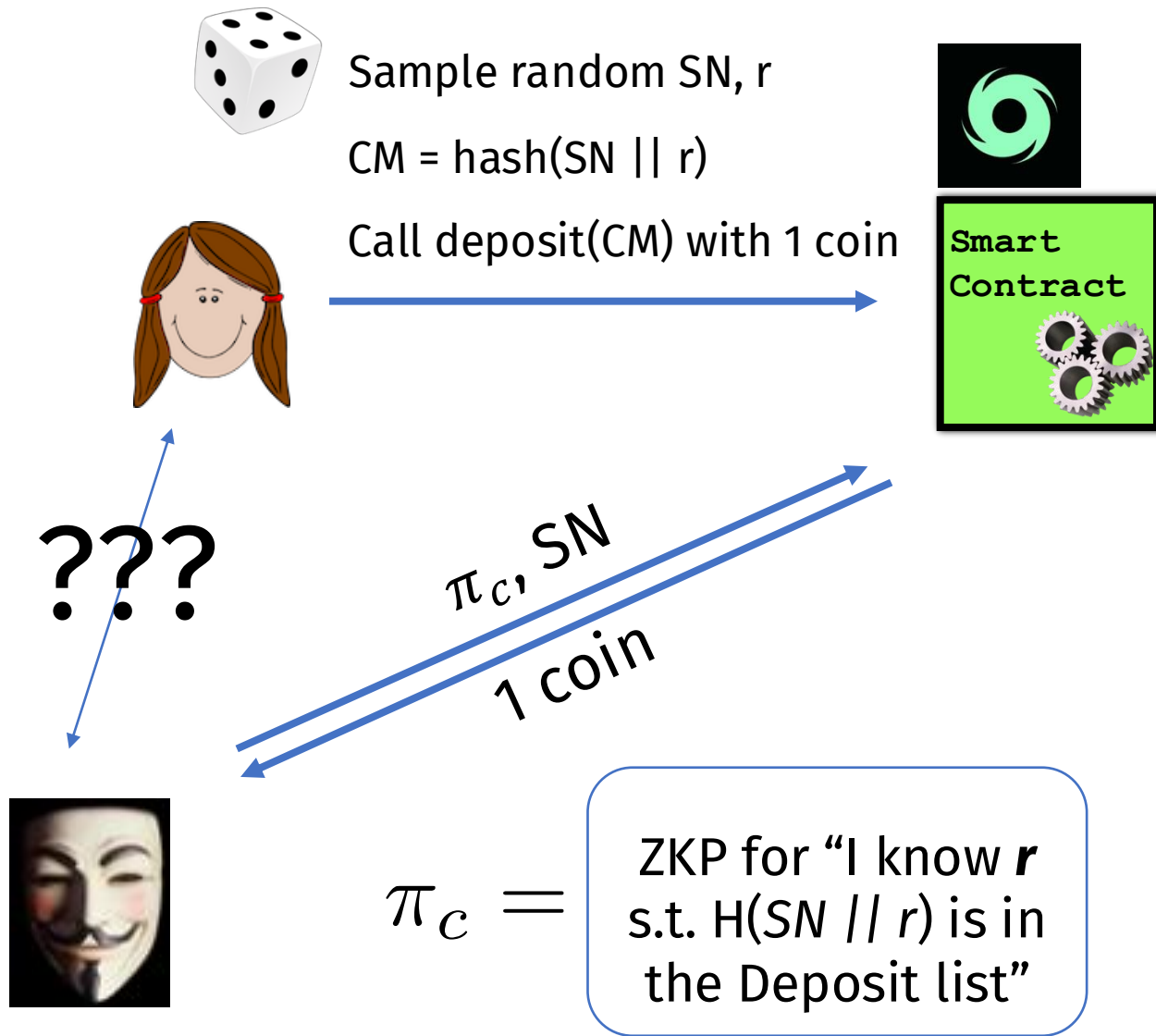
- Proof is small (e.g., a few KB)
- Verification of a proof is extremely fast
- Proof convinces **everyone** (public-verifier)

Catch: generating proof can be slow (e.g., hours)

Deposit coins by creating a commitment



Withdraw Mixed Coins by opening a commitment



Why recording serial numbers?

Deposit	Spent
CM_1	SN
CM_2	
CM_N	

Unlinkability:

To an observer, the spender could be *any* depositor (they form the *anonymity set*).

Important details

How to pay tx fee for withdrawal transaction?

- Using your own ether?
- That links you to the withdrawal, defeating the purpose.
- Solution idea: Tornado Cash sets up **relays** who sends transactions on user's behalf and collect a small fee from withdrawal.
- Risks (for relay, and for users)?

The trouble with unlinkability

- North Korea's Lazarus Group used Tornado Cash to launder their hacked coins.
- Unlinkability makes it hard to sanction the hackers, without also sanction the legit users.



The U.S. sanctions on Tornado Cash

- Aug 8, 2022: The Treasury's Office of Foreign Assets Control (OFAC) put Tornado Cash on its sanctions list, including **the addresses of its smart contracts**.
- **Implications:** Any U.S. person who deposited, withdrew, or even received funds from those addresses could be violating sanctions law.

A bit of painting with a broad brush

- Innocent users are affected
 - E.g., Vitalik had used it to donate to Ukraine.
- “Any U.S. person who ... received funds from those addresses could be violating sanctions law.” Someone sent ETH from Tornado to Jimmy Fallon, Shaq, and others, making them 'sanctions violators’.

 **FatMan** 
@FatManTerra · [Follow](#)

Notable individuals/companies who just received funds from a government-sanctioned entity:

- Jimmy Fallon
- Shaquille O'Neal
- PUMA
- Randi Zuckerberg
- Logan Paul
- Brian Armstrong
- Steve Aoki
- Ukraine Crypto Donation
- Dave Chappelle
- Beeple

Field day for investigators.

 **db**  @tier10k

Jimmy Fallon is a DPRK Operative: Source

Age	From	To
20 mins ago	 Tornado.Cash : 0.1 ETH	 Jimmy Fallon

PRESS RELEASES

Tornado Cash Delisting



March 21, 2025

WASHINGTON — Based on the Administration's review of the novel legal and policy issues raised by use of financial sanctions against financial and commercial activity occurring within evolving technology and legal environments, we have exercised our discretion to remove the economic sanctions against Tornado Cash as reflected in Treasury's Monday filing in *Van Loon v. Department of the Treasury*.

- Sanction lifted by OFACT in 2025
- Developer Storm still being persecuted (retrial in 2027)

Is there a way to reconcile compliance and privacy?

- A better idea: stop only the hackers' fund.
- Technically, this is not hard.
- Option 1: having a user prove that their deposit is **outside** a set of flagged funds.
- Option 2: having a user prove that their deposit is **inside** an allow list.
- Both can be done with an additional ZKP.

Privacy Pool

- An association set contains deposits accepted under a screening policy. An association set provider (ASP) publishes its Merkle root.
- At withdrawal, a ZKP proves the coin is valid and its deposit belongs to this set, without identifying the deposit.

Privacy Pool

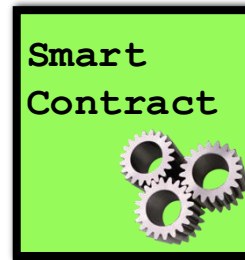


Sample random SN, r

$CM = \text{Hash}(SN || r)$



Call $\text{deposit}(CM)$ with 1 coin



Deposit	Spent
CM_1	
CM_2	
...	
CM_N	

Screened PKs	
PK_1	

Privacy Pool



Sample random SN, r

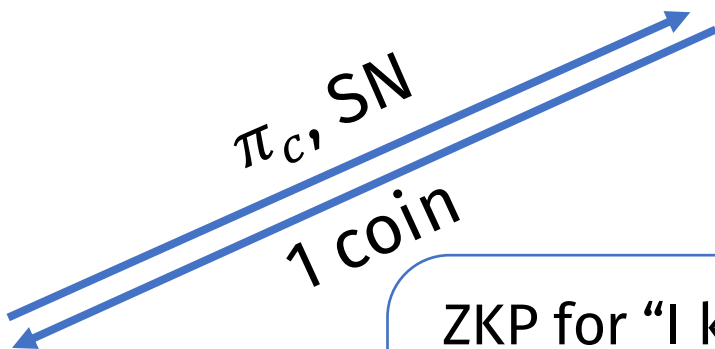
$CM = \text{hash}(SN \parallel r)$



Call $\text{deposit}(CM)$ with 1 coin



π_c, SN
1 coin



$\pi_c =$

ZKP for "I know r s.t.
 $H(SN \parallel r)$ is in the
Deposit list" AND "I
know sk whose pk is in
screened PKs."

Why recording
serial numbers?

Deposit	Spent
CM_1	SN
CM_2	
CM_N	



Screened PKs	
PK_1	

Hands-On Component: Playing with Privacy Pools

- We selfhosted a modified version of Privacy Pool
 - We added support for Sepolia
 - We added support for a custom ERC-20 token
- Two tasks
 - Obtain some BULLDOGS tokens
 - Use Privacy Pool to mix 5000 of them.

Tornado Cash is not a payment system

- A mixer for you to mix your coins
- Not for you to pay some else
- Tornado cash is a simplified version of Zcash
- Zcash supports full fledged private payments

Optional slides on zcash

2014 IEEE Symposium on Security and Privacy

Zerocash: Decentralized Anonymous Payments from Bitcoin

Eli Ben-Sasson*, Alessandro Chiesa[†], Christina Garman[‡], Matthew Green[‡], Ian Miers[‡], Eran Tromer[§], Madars Virza[†]

*Technion, eli@cs.technion.ac.il

[†]MIT, {alexch, madars}@mit.edu

[‡]Johns Hopkins University, {cgarman, imiers, mgreen}@cs.jhu.edu

[§]Tel Aviv University, tromer@cs.tau.ac.il



(Note that Zerocash predates Tornado Cash by 6 years)

Challenges in enabling anonymous payments

- Basic idea: spending a commitment creates *new commitments*

So far: $CM = \text{hash}(SN || r)$

- How to create new commitments?
 - In particular, how to create SNs?
- How to enable payment of arbitrary value?
- How to validate solvency (output sum < input sum) when both are hidden?

New SNs

- Suppose Alice (A) wants to pay Bob (B). Alice needs to create a new coin with a new SN.
- **B** should not be actively involved (payee may not be online)
- **A** can't determine the SN by herself (why?)
 - A can trace B
 - A can create a spent SN so that B can't spend
 - A can spend before B can
- Solution: A contributes randomness ρ
 - $sn = H(\rho|sk_B)$; Only B knows sk_B (and thus SN)

Coins

Each coin is represented by a commitment to

- owner's PK
- value v
- ρ : randomness for serial number

Technically, a coin is $(\overbrace{pk, v, \rho, r, s}^{\text{Known only to the owner}}, cm)$

- r, s are secret blinding factors

- $k = Com_r(pk|\rho)$
- $cm = Com_s(v|k)$

- K commits to only PK and ρ
- Useful when minting (next slide)

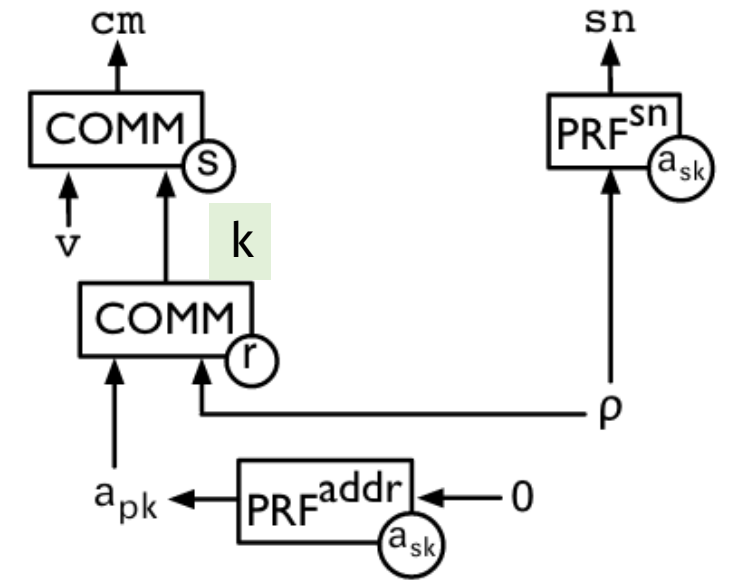


Fig. 1 in Zerocash paper.

Mint zerocash (from Bitcoin)

- To mint an anonymous coin of v BTC, user sends v BTC to an escrow with (v, k, s, cm) in transaction metadata (e.g., OP_RETURN).
 - v : the value
 - k commits to PK, ρ
 - cm commits to v and k
- **cm now represents v “anonymous BTC”**
- This “escrow” can be implemented by modifying Bitcoin protocol or building an overlay on top of Bitcoin
- Zcash took the first approach.

Zerocash: Alice Pays Bob \$3

blockchain

Serial #'s	31	29	34	11	7	8	6	17				
Coins												

1. Alice samples new ρ_1, r_1, s_1
2. creates $k_1 = \text{Com}(pk_B | \rho_1)$ and $cm_1 = \text{Com}(\$3 | k_1)$
3. New coin is $c_{out_1} = (pk_B, \$3, \rho_1, r_1, s_1, cm_1)$



Cout₁

$c_{in} = (pk^{old}, \$10, \rho^{old}, r^{old}, s^{old}, cm^{old})$
 Suppose $sn = H(\rho, sk^{old}) = 42$

C_{in}



1. Alice samples new ρ_2, r_2, s_2
2. creates $k_1 = \text{Com}(pk_A | \rho_2)$ and $cm = \text{Com}(\$7 | k_1)$
3. New coin is $c_{out_2} = (pk_A, \$7, \rho_2, r_2, s_2, cm_1)$



Cout₂

ZeroCash: Alice Pays Bob \$3

blockchain

Serial #'s

31	29	34	11	7	8	6	17				
----	----	----	----	---	---	---	----	--	--	--	--

Coins

											
---	---	--	---	---	---	---	---	--	--	--	--

$$c_{in} = (pk^{old}, \$10, \rho^{old}, r^{old}, s^{old}, cm^{old})$$

Suppose $sn = H(\rho, sk^{old}) = 42$

$$c_{out_1} = (pk_B, \$3, \rho_1, r_1, s_1, cm_1)$$

c_{in}



$\pi =$



c_{out_1}

witness:

$(sk^{old}, c_{in}, c_{out_1}, c_{out_2})$

Public input:

$(cm_1, cm_2, 42)$

- Three coins are well formed
- sk^{old} matches pk^{old}
- $42 = H(\rho^{old}, sk^{old})$
- cm^{old} is in the list of coins
- $v_1 + v_2 \leq v^{old}$ (in c_{in})

$$c_{out_2} = (pk_A, \$7, \rho_2, r_2, s_2, cm_1)$$



c_{out_2}

ZeroCash: Alice Pays Bob \$3

blockchain

Serial #'s

31	29	34	11	7	8	6	17				
----	----	----	----	---	---	---	----	--	--	--	--

Coins

											
---	---	--	---	---	---	---	---	--	--	--	--

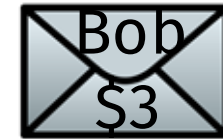
$$c_{in} = (pk^{old}, \$10, \rho^{old}, r^{old}, s^{old}, cm^{old})$$

$$\text{Suppose } sn = H(\rho, sk^{old}) = 42$$

c_{in}



$$c_{out_1} = (pk_B, \$3, \rho_1, r_1, s_1, cm_1)$$



c_{out_1}

- Finally, Alice sends $tx_{pour} = (\pi, sn^{old}, cm_1, cm_2)$ to blockchain
- Send c_{out_1} to Bob in private because Bob needs ρ_1, r_1, s_1 to spend cm_1

$$c_{out_2} = (pk_A, \$7, \rho_2, r_2, s_2, cm_1)$$



c_{out_2}

That's pretty much how Zerocash works...



- Some technical details omitted (e.g., how to send coins on-chain)
- Zcash is the commercial implementation of zerocash

Summary:

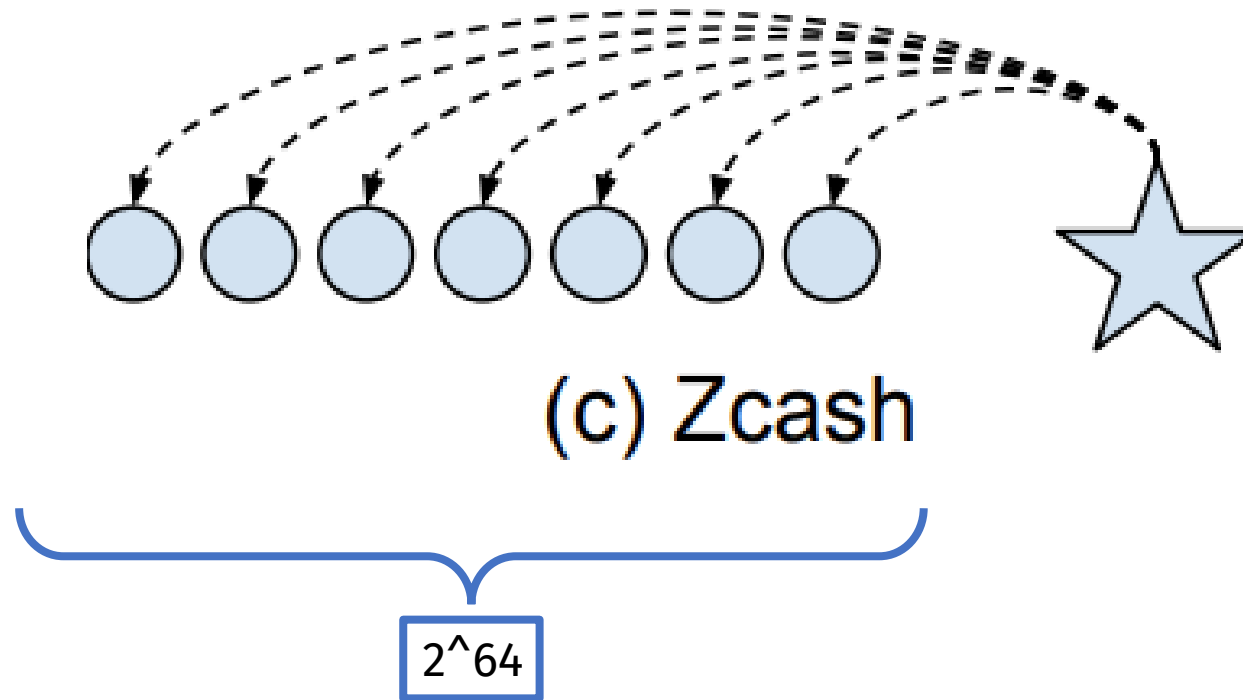
- coins are commitment to (PK, v, p)
 - PK: owner
 - v: value
 - p: randomness for SN
- spending = reveal SN & proving SK
- transfer involves spending and creating new commitments

Technically

- a coin is (pk, v, ρ, r, s, cm)
 - where r, s are random blinding factors
 - $k = Com(pk|\rho)$ and $cm = Com(v|k)$
- To mint an anonymous coin of v BTC, user sends v BTC to an escrow with (v, k, s, cm) in transaction metadata.
- To spend, create new coins and send $tx_{pour} = (\pi, sn^{old}, cm_1, cm_2)$

What is the anonymity set of Zerocash?

Anonymity set is the entire user universe



Any questions?