Zerocash: Decentralized Anonymous Payments from Bitcoin

— Rujia Li

UNIVERSITY OF BIRMINGHAM

Outline

1. The anonymous problem of Bitcoin(or similar ledger-based currencies)

2. Zerocash solves the lack of anonymity of bitcoin

3. The security and performance of Zerocash

4. Conclusion and future works

Pseudonymous and anonymous

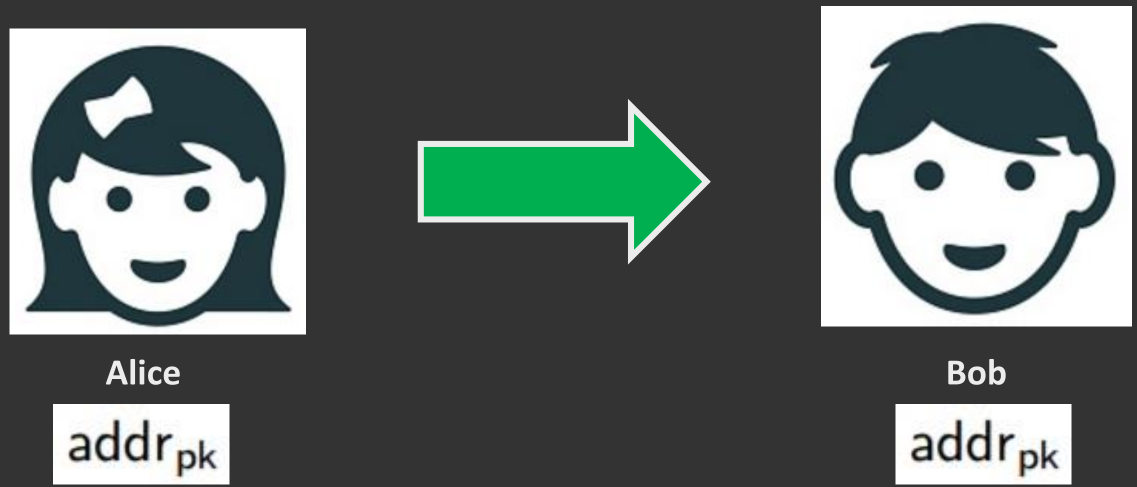
Pseudonymous ≠ Anonymous

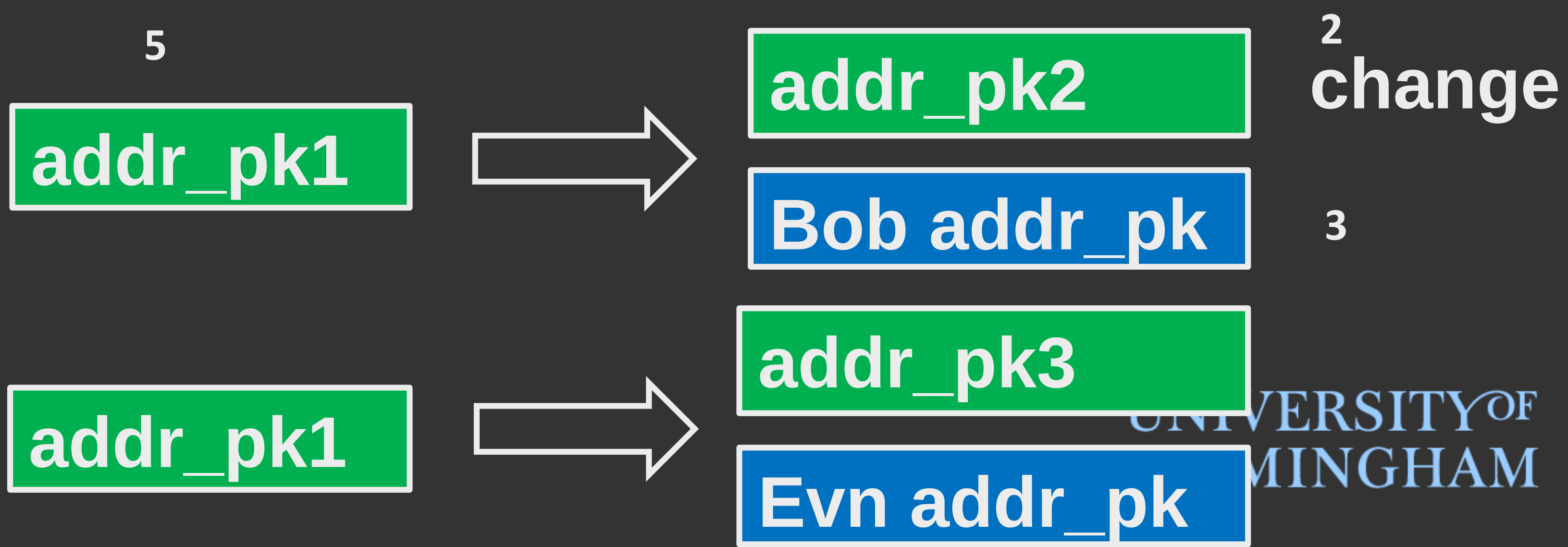
"Pseudonymous", it means you are not using your real, legal name to identify yourself.

"Anonymous" it means that someone's identity is completely unknown, you can't associate the name with any individual.

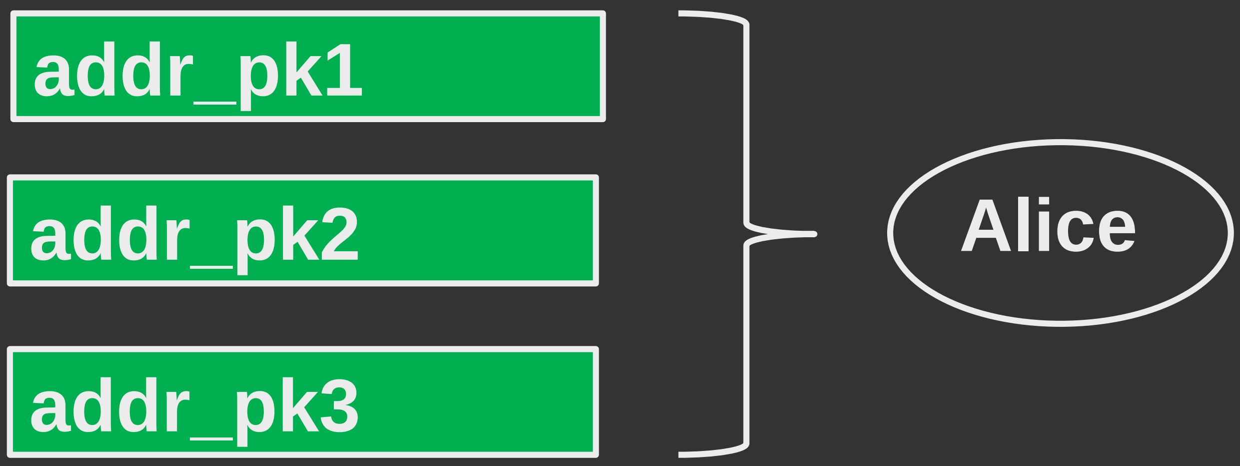
UNIVERSITY OF BIRMINGHAM

Bitcoin transaction





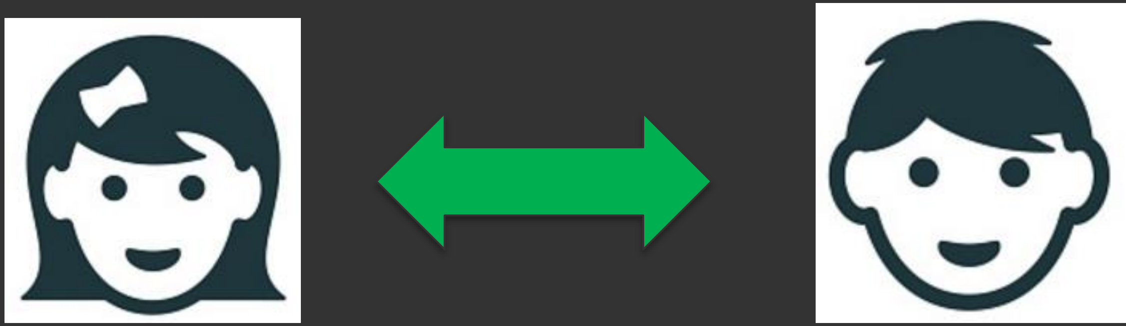
Address linkability



The hacker may deduce the 3 addresses which are belonged to one person

Zerocash transaction

They use the random string to represent user's identity



Alice

random string

Bob

random string

UNIVERSITY OF BIRMINGHAM

Collision Resistant Hash (CRH)

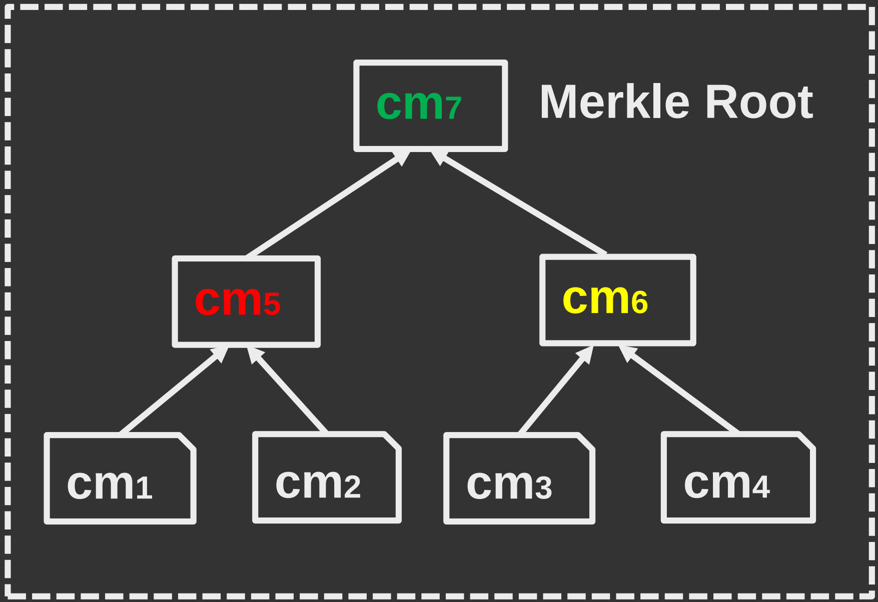
- Function H mapping long string to shorter ones

- Easy to compute(ZeroCash uses SHA256)

- Hard to find 2 long strings mapped to same short one

UNIVERSITY OF BIRMINGHAM

Merkle Tree



cm_5 = CRH(cm_1, cm_2)

cm_6 = CRH(cm_3, cm_4)

cm_7 = CRH(cm_5, cm_6)

Used to prove “I know some data committed in one of cm1,cm2,…cmN”

UNIVERSITY OF BIRMINGHAM

Zero-knowledge proof

“ A zero-knowledge proof is a method by which one party (the prover) can prove to another party (the verifier) that a given statement is true, without conveying any information apart from the fact that the statement is indeed true. ”

—S.Goldwasser

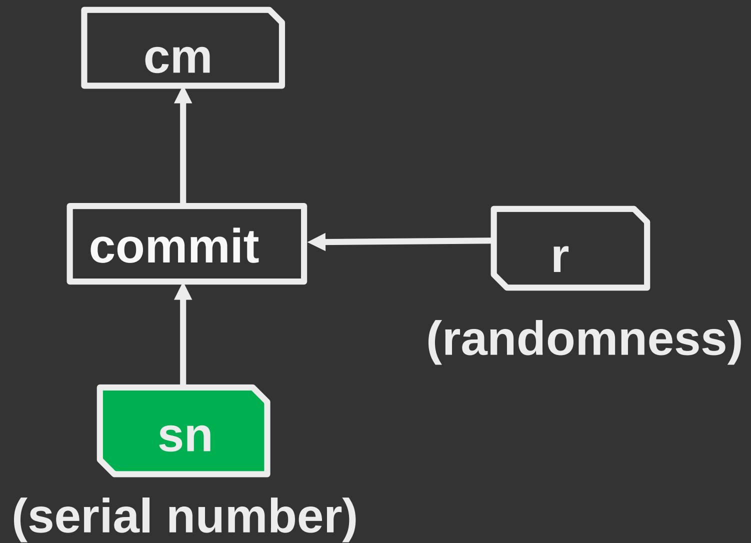
UNIVERSITY OF BIRMINGHAM

https://en.wikipedia.org/wiki/Zero-knowledge_proof

Zero-knowledge proof example

Alice can use H to commit to string sn(256 bits long)

- Pick random r (256 bit long)
- Publish cm = H(sn, r)
- Alice can prove she knows sn by revealing r
- Bob can't learn much about sn from cm



UNIVERSITY OF BIRMINGHAM

Step1: Creating payment addresses

$$\mathsf{addr}_{\mathsf{pk}} = (a_{\mathsf{pk}}, \mathsf{pk}_{\mathsf{enc}})$$

The public address addr_pk is published and enables others to direct payments to the user

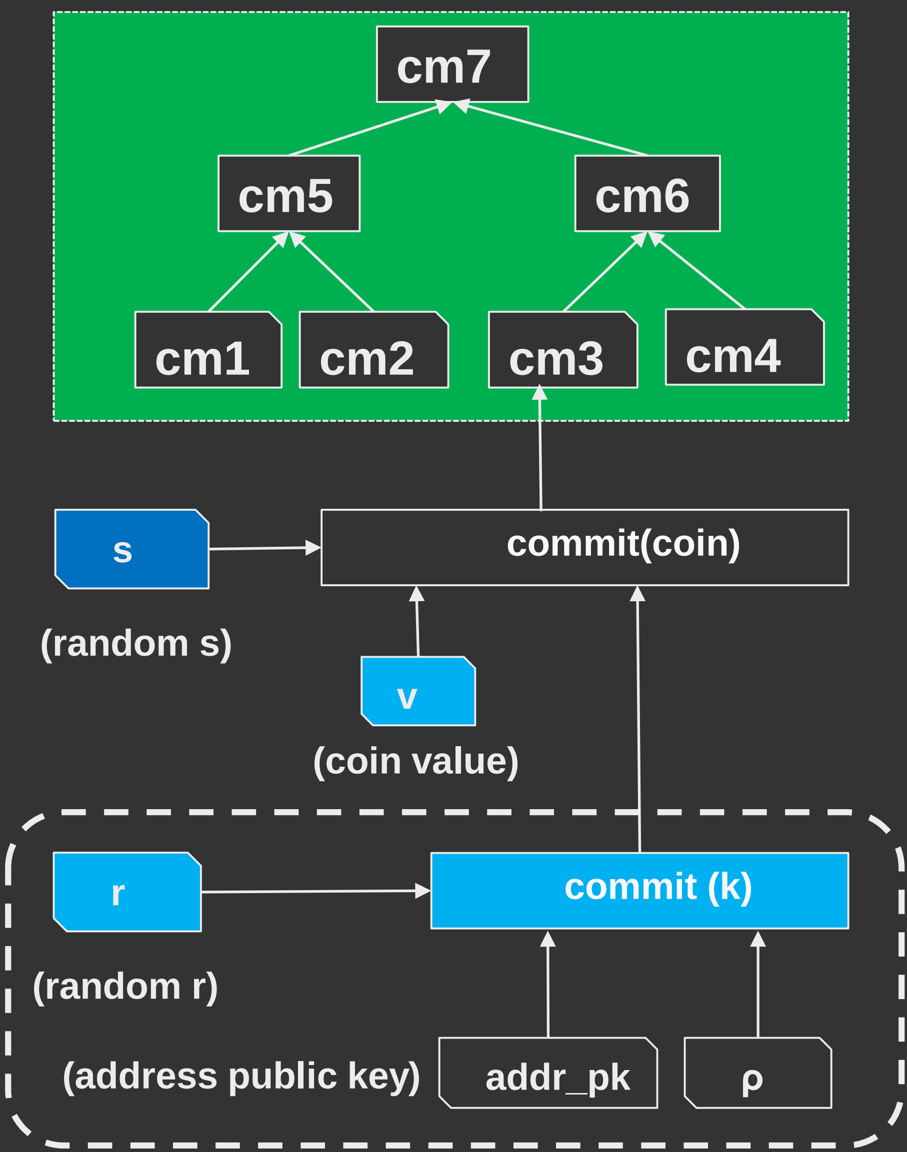
$$\mathsf{addr}_{\mathsf{sk}} = (a_{\mathsf{sk}}, \mathsf{sk}_{\mathsf{enc}})$$

The secret address addr_sk is used to redeem coins sent to addr_pk.

The next step: To find the relationship between random string ρ and the address.

UNIVERSITY OF BIRMINGHAM

Step2: Minting coins



$k := \mathsf{COMM}_r(a_{\mathsf{pk}} \| \rho)$

$\mathsf{cm} := \mathsf{COMM}_s(v \| k)$

$\mathsf{tx}_{\mathsf{Mint}} := (v, k, s, \mathsf{cm})$

$\mathbf{c} := (a_{\mathsf{pk}}, v, \rho, r, s, \mathsf{cm})$

To prove coin c has value v and coin address is addr_pk

UNIVERSITY OF BIRMINGHAM

Step1 and step2 recap

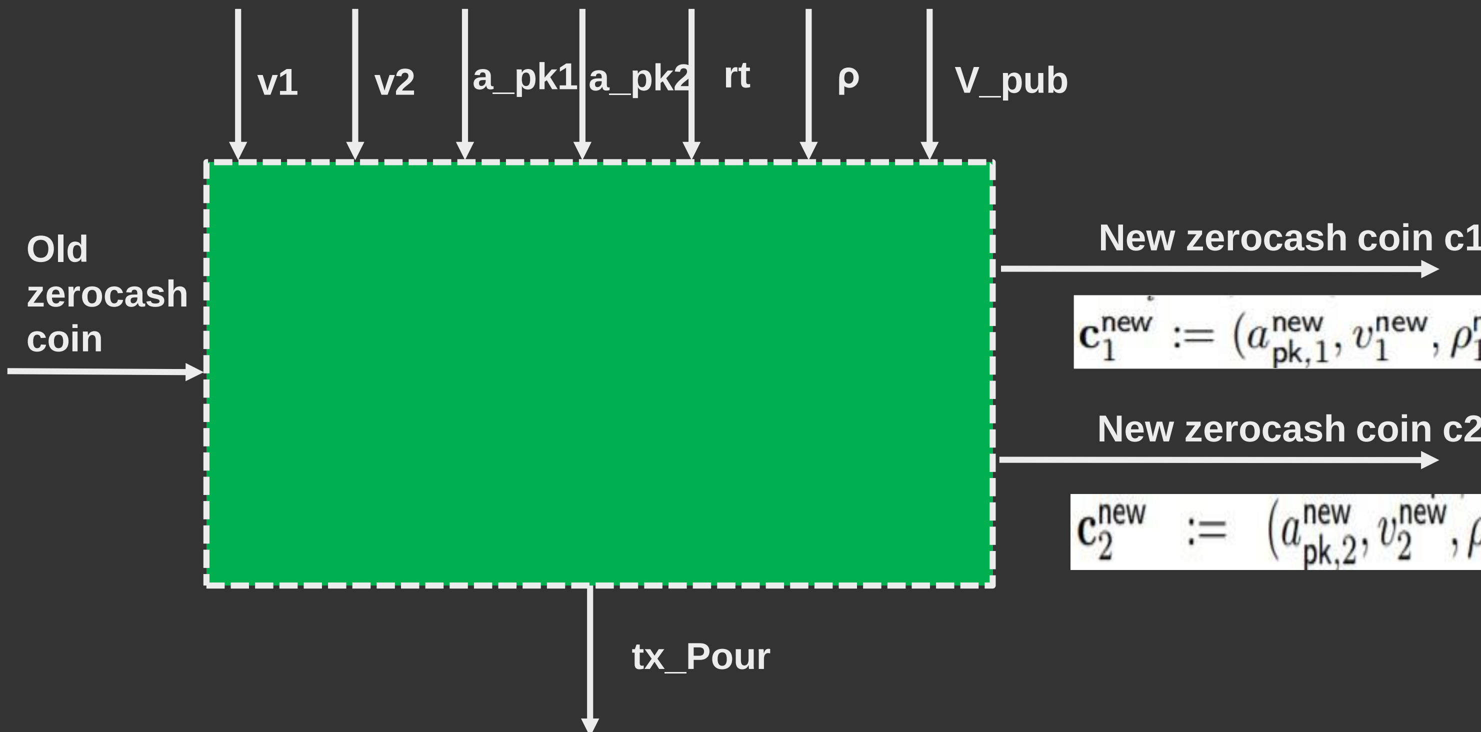
1. The user u has the public address and private address.

2. The coin c is belonged to the user u and its value is v

3. The next step: how does the user u spend the coin c ?

UNIVERSITY OF BIRMINGHAM

Step3: Pouring transaction



$$\mathbf{c}_1^{\text{new}} := (a_{\text{pk},1}^{\text{new}}, v_1^{\text{new}}, \rho_1^{\text{new}}, r_1^{\text{new}}, s_1^{\text{new}}, \text{cm}_1^{\text{new}})$$

$$\mathbf{c}_2^{\text{new}} := (a_{\text{pk},2}^{\text{new}}, v_2^{\text{new}}, \rho_2^{\text{new}}, r_2^{\text{new}}, s_2^{\text{new}}, \text{cm}_2^{\text{new}})$$

$$\text{tx}_{\text{Pour}} = (\text{rt}, \text{sn}_1^{\text{old}}, \text{cm}_1^{\text{new}}, \text{cm}_2^{\text{new}}, v_{\text{pub}}, \text{info}, *)$$

UNIVERSITY OF BIRMINGHAM

Step4: Verifying transactions

1. The coin c1 is belonged to the user u1 and its value is v1

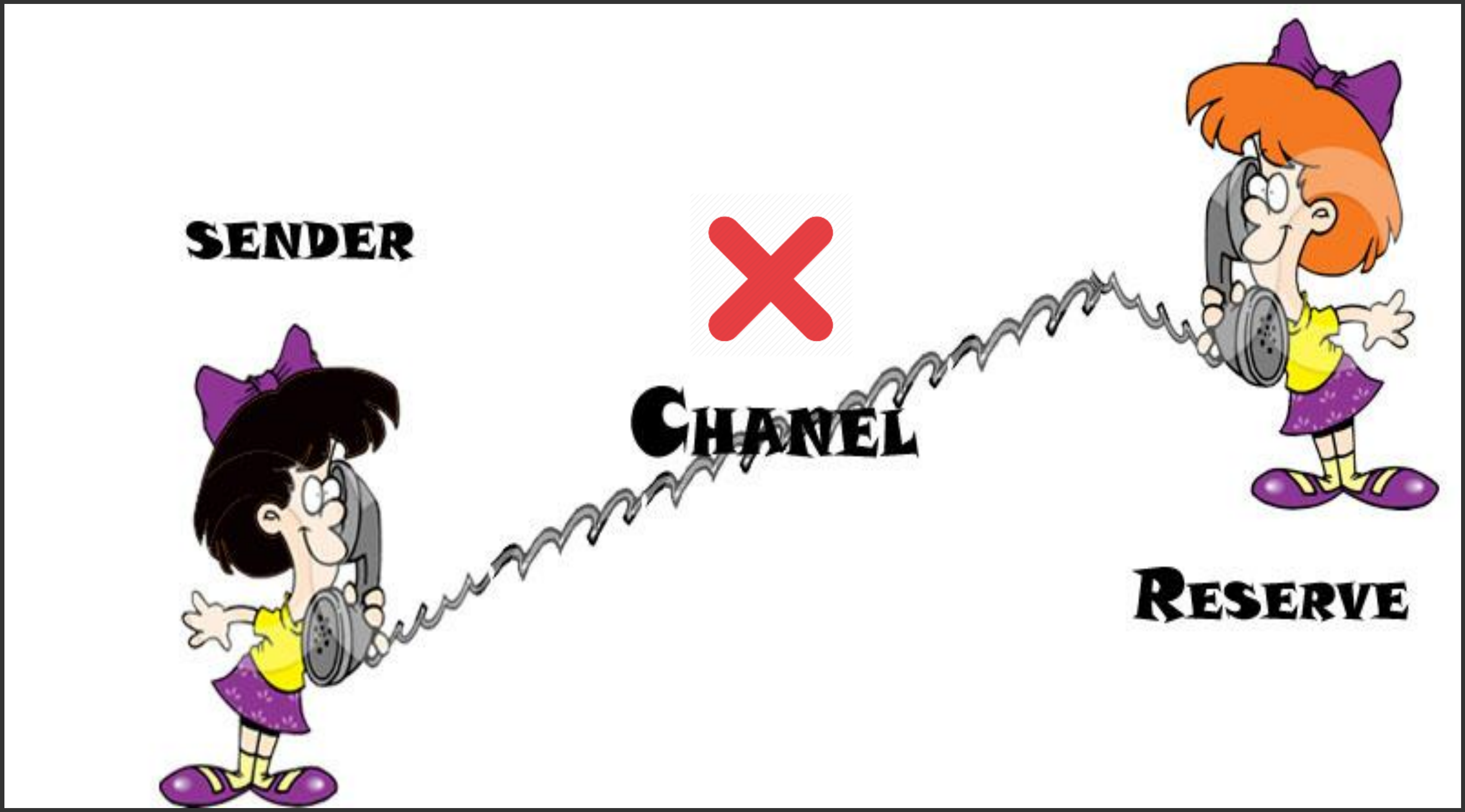
2. The coin c2 is belonged to the user u2 and its value is v2

3. $v\_old = v\_new1 + v\_new2 + v\_pub$

4. The next step: how to distribute ρ_new1 and ρ_new2 to user1 and user2 ?

UNIVERSITY OF BIRMINGHAM

Encrypted channel ?



1. It Needs additional infrastructure
2. Inefficient and not secure

UNIVERSITY OF BIRMINGHAM

Step5: Distribute random string

$$\mathsf{addr}_{\mathsf{pk}} = (a_{\mathsf{pk}}, \mathsf{pk}_{\mathsf{enc}}) \qquad \rho_1^{\mathsf{new}}$$

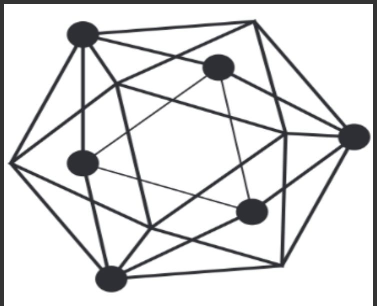
$$\Downarrow \qquad \Downarrow$$

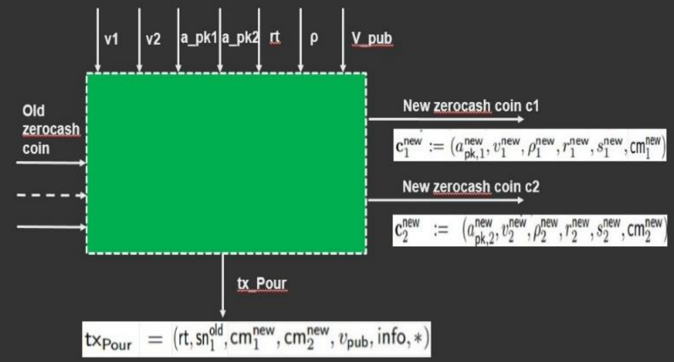
$$\mathbf{C}_1 = \mathsf{pk}^{\mathsf{new}}_{\mathsf{enc},1}\ (v_1^{\mathsf{new}}, \rho_1^{\mathsf{new}}, r_1^{\mathsf{new}}, s_1^{\mathsf{new}})$$

$$\Downarrow$$

$$\mathsf{tx}_{\mathsf{Pour}} = (\mathsf{rt}, \mathsf{sn}_1^{\mathsf{old}}, \mathsf{cm}_1^{\mathsf{new}}, \mathsf{cm}_2^{\mathsf{new}}, v_{\mathsf{pub}}, \mathsf{info}, *)$$

$$\Downarrow$$



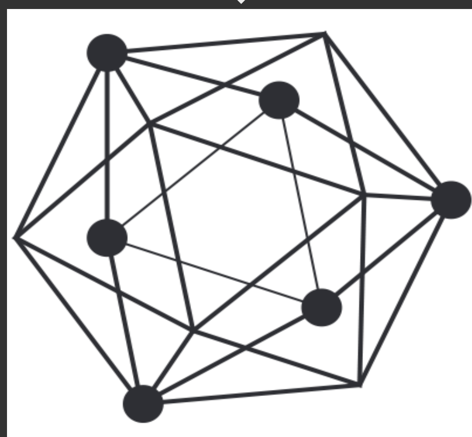


Put the encrypted ρ_new1 on the public ledger

UNIVERSITY OF BIRMINGHAM

Step5: Distribute random string

$$\mathsf{addr_{sk}} = (a_{\mathsf{sk}}, \mathsf{sk_{enc}})$$



$$\mathbf{C}_1 = \mathsf{pk}^{\mathsf{new}}_{\mathsf{enc},1} \; (v^{\mathsf{new}}_1, \rho^{\mathsf{new}}_1, r^{\mathsf{new}}_1, s^{\mathsf{new}}_1)$$

$$\rho^{\mathsf{new}}_1$$

The user u1 can find and decrypt the message (using his $\mathsf{sk}^{\mathsf{new}}_{\mathsf{enc},1}$) by scanning the pour transactions on the public ledger

UNIVERSITY OF BIRMINGHAM

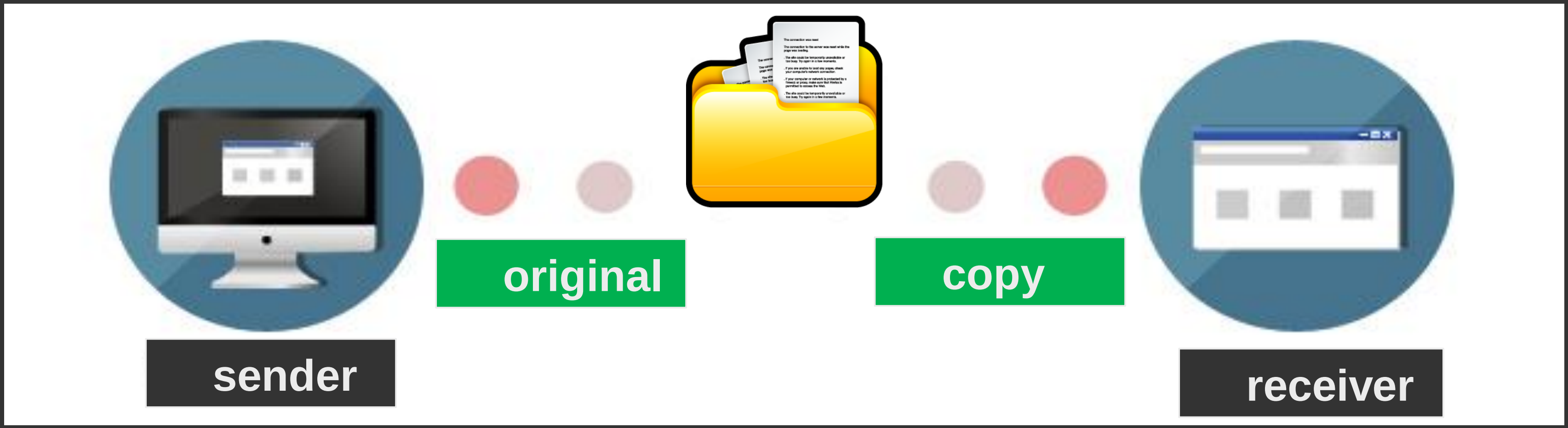
Step4 and step5 recap

1. We have generated the new coin c1 and c2. Also, the users can spend the coins by revealing the new ρ.

2. We have known how to distribute ρ.

3. The last step: how to prevent double spending ?

Double spending problem



sending the files

sending the assets

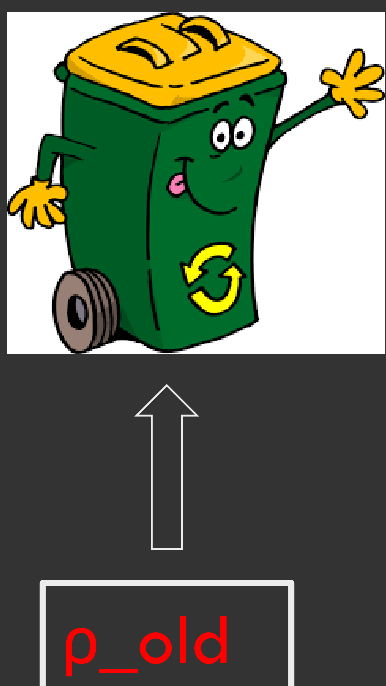
UNIVERSITY OF BIRMINGHAM

Preventing double spending

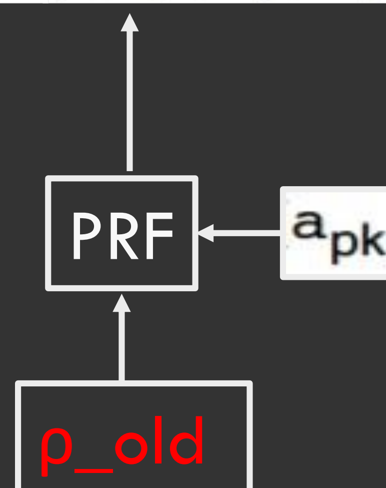
The user 1 can spend coin c1 due to ρ_new1

The user 2 can spend coin c2 due to ρ_new2

The old user also can spend old coin c due to ρ_old



$$\mathsf{tx}_{\mathsf{Pour}} = (\mathsf{rt}, \mathsf{sn}_1^{\mathsf{old}}, \mathsf{cm}_1^{\mathsf{new}}, \mathsf{cm}_2^{\mathsf{new}}, v_{\mathsf{pub}}, \mathsf{info}, *)$$



Performance of Zerocash algorithms

Intel Core i7-4770 @ 3.40GHz with 16GB of RAM (1 thread)		
Setup	Time	5 min 17 s
	pp	896 MiB
CreateAddress	Time	326.0 ms
	$\mathsf{addr}_{\mathsf{pk}}$	343 B
	$\mathsf{addr}_{\mathsf{sk}}$	319 B
Mint	Time	23 µs
	Coin **c**	463 B
	$\mathsf{tx}_{\mathsf{Mint}}$	72 B
Pour	Time	2 min 2.01 s
	$\mathsf{tx}_{\mathsf{Pour}}$	996 B[16]
VerifyTransaction	mint	8.3 µs
	pour (excludes L scan)	5.7 ms
Receive	Time (per pour tx)	1.6 ms

UNIVERSITY OF BIRMINGHAM

The security of ZeroCash

1. Ledger indistinguishability
   - Nothing revealed beside public information, even by chosen-transaction adversary.

2. Balance
   - can't own more money than received or minted

3. Transaction non-malleability
   - can't manipulate transactions en route to ledger

UNIVERSITY OF BIRMINGHAM

Conclusion

1. Zerocash enable one user to pay another user directly via random string without reveal neither the origin,destination or amount

2. The security and performance of Zerocash

UNIVERSITY OF BIRMINGHAM

Questions

Zerocash: Decentralized Anonymous Payments from Bitcoin

UNIVERSITY OF BIRMINGHAM