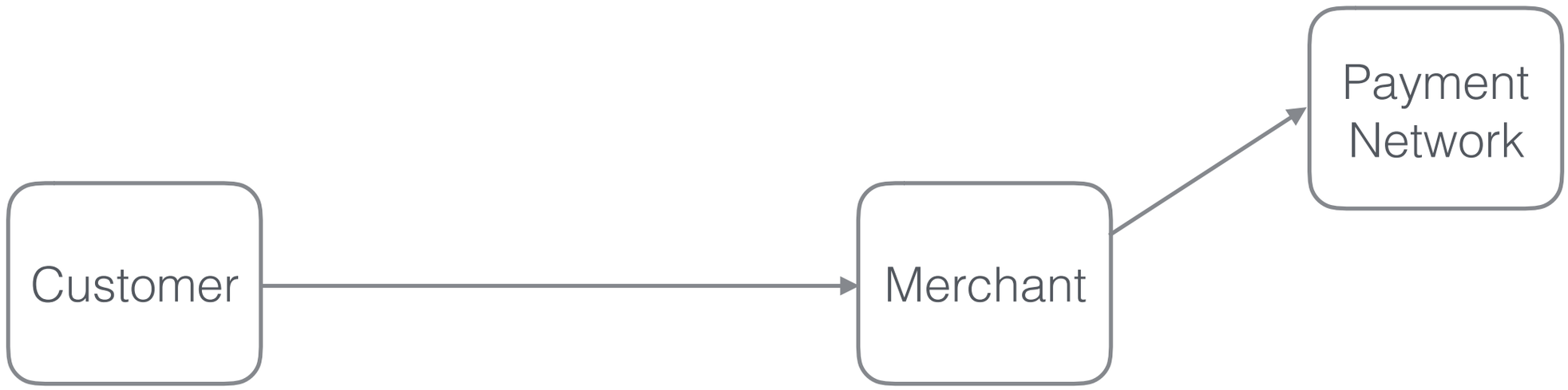
Decentralized
Anonymous
Micropayments

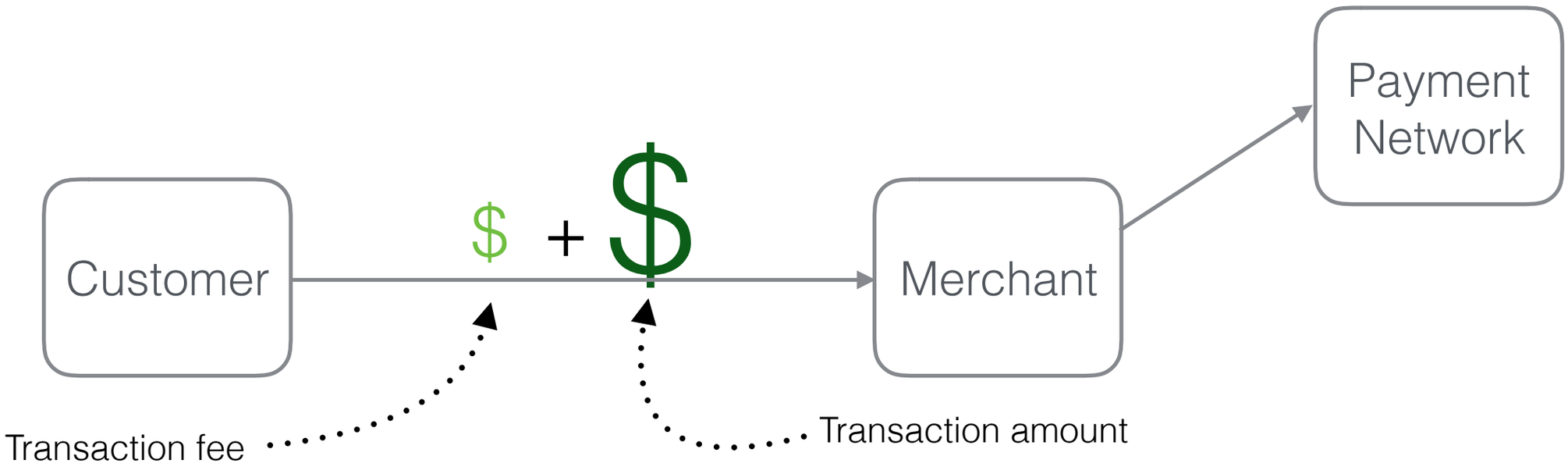
Alessandro Chiesa, Matthew Green, Jingcheng Liu,
Peihan Miao, Ian Miers, **Pratyush Mishra**

http://eprint.iacr.org/2016/1033

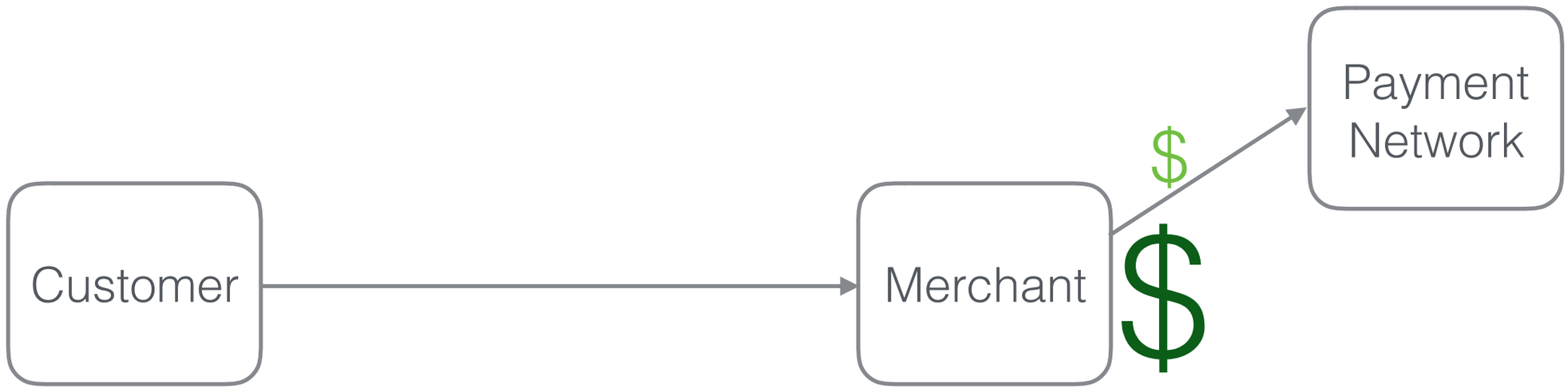
Digital Payments



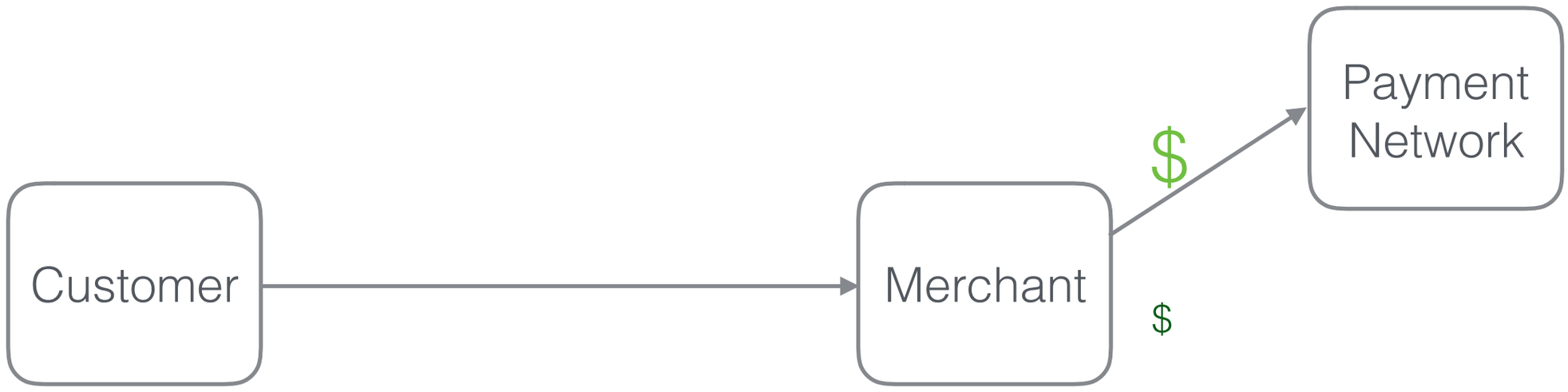
Digital Payments



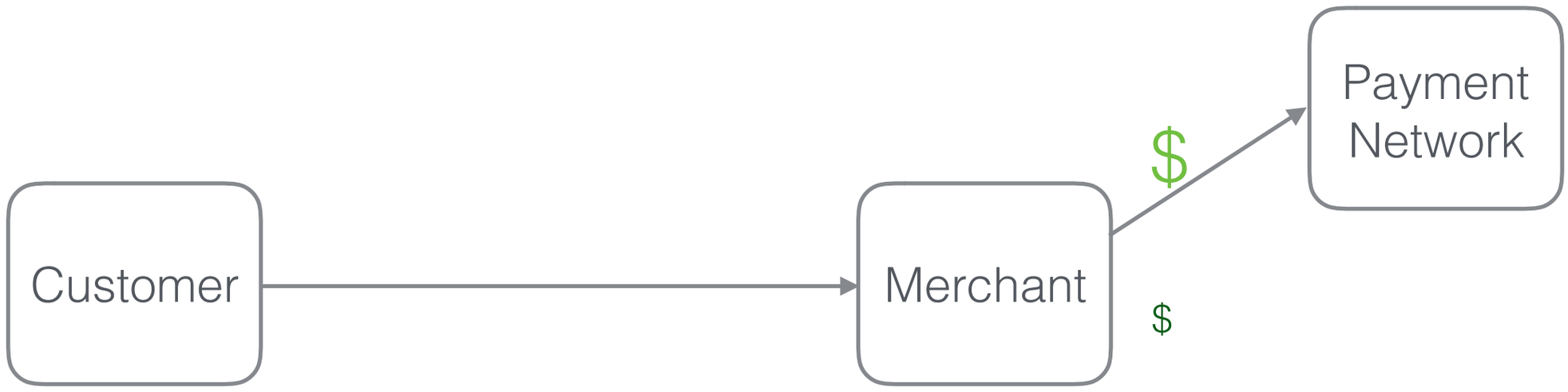
Digital Payments



Digital Payments

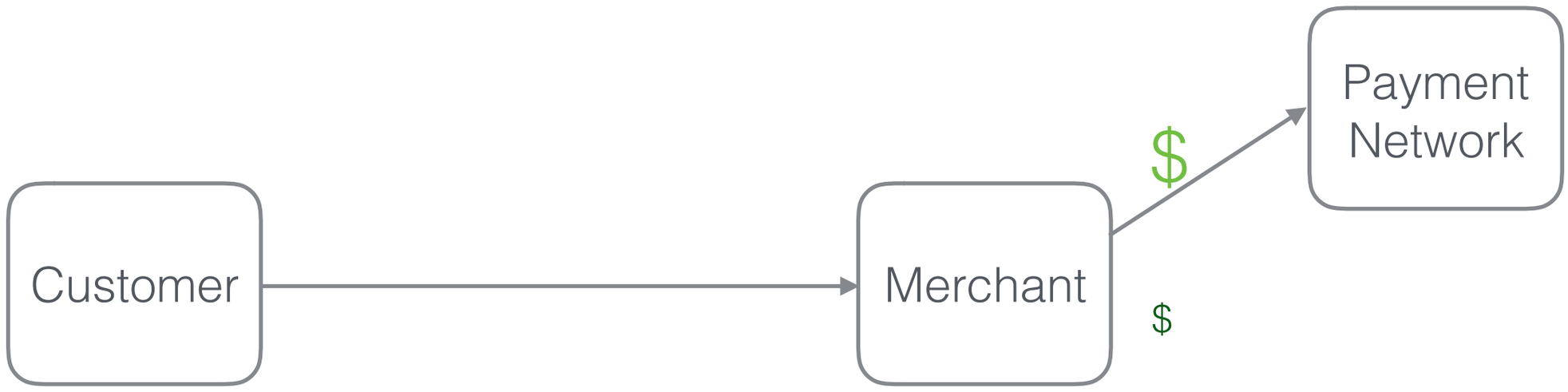


Digital Payments



Supporting small payments is important for applications.

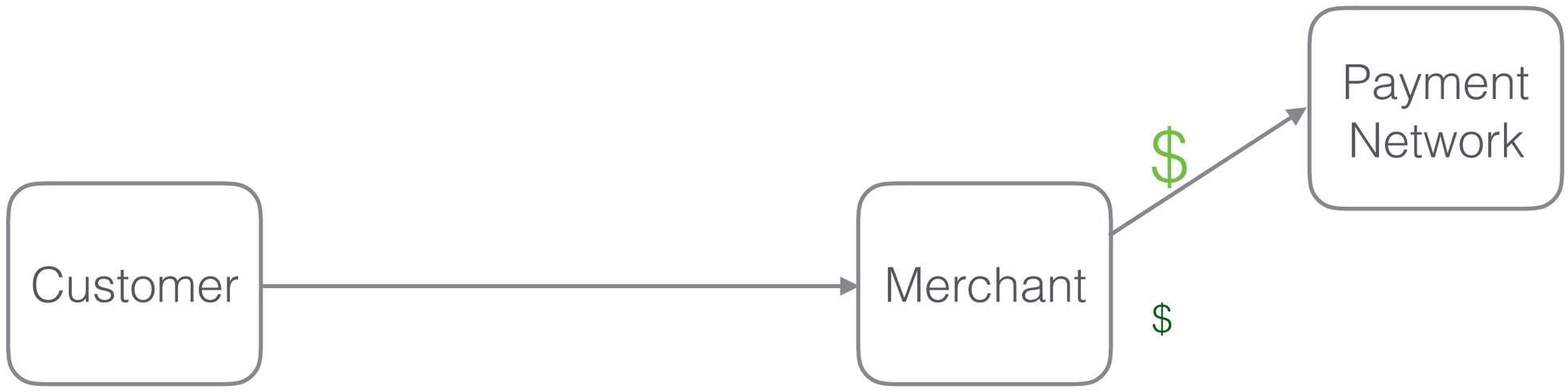
Digital Payments



Supporting small payments is important for applications.

Eg: payments instead of ads while browsing.

Digital Payments

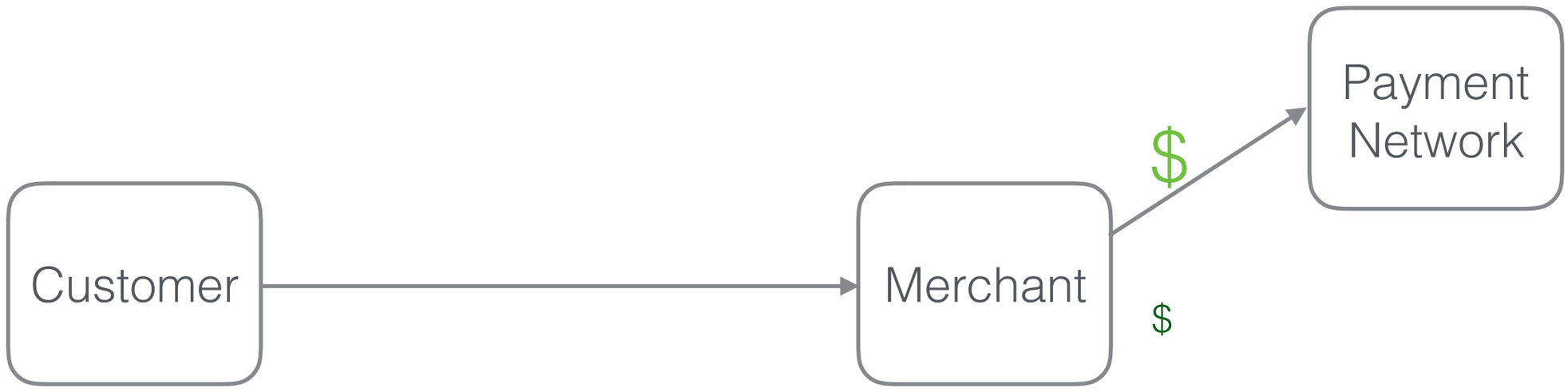


Supporting small payments is important for applications.

Eg: payments instead of ads while browsing.

Rich history of **micropayment schemes** constructions:

Digital Payments



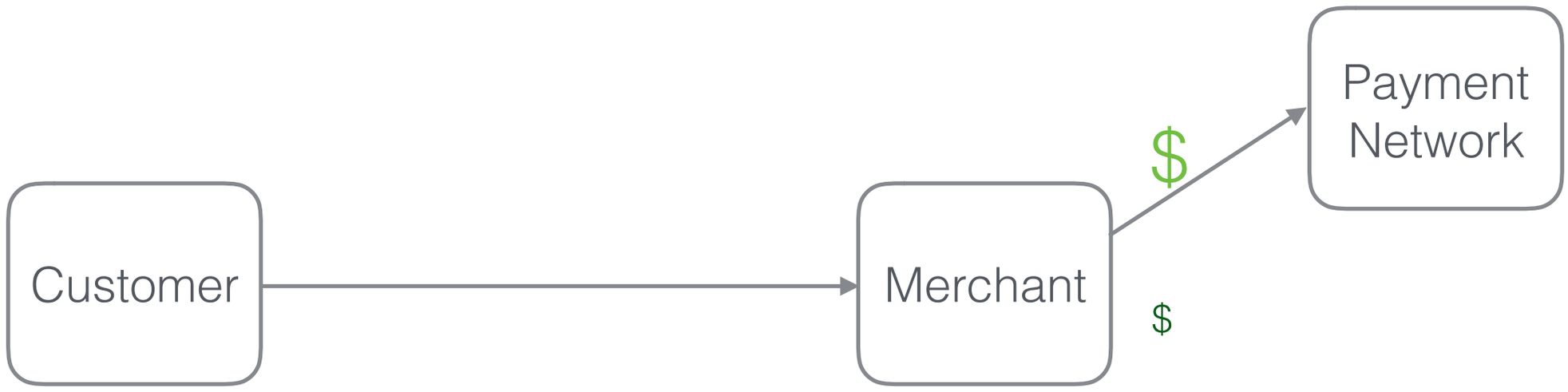
Supporting small payments is important for applications.

Eg: payments instead of ads while browsing.

Rich history of **micropayment schemes** constructions:

[Whe96, Riv97, LO98, JY96, RS01, MR02]…

Digital Payments



Supporting small payments is important for applications.

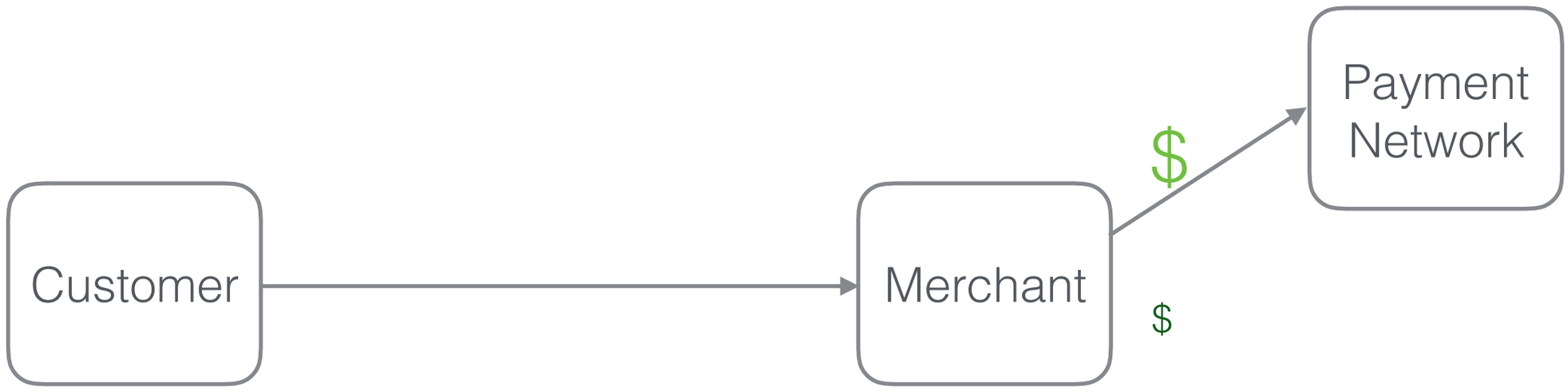
Eg: payments instead of ads while browsing.

Rich history of **micropayment schemes** constructions:

[Whe96, Riv97, LO98, JY96, RS01, MR02]…

… but no widespread deployments across multiple merchants.

Digital Payments



Supporting small payments is important for applications.

Eg: payments instead of ads while browsing.

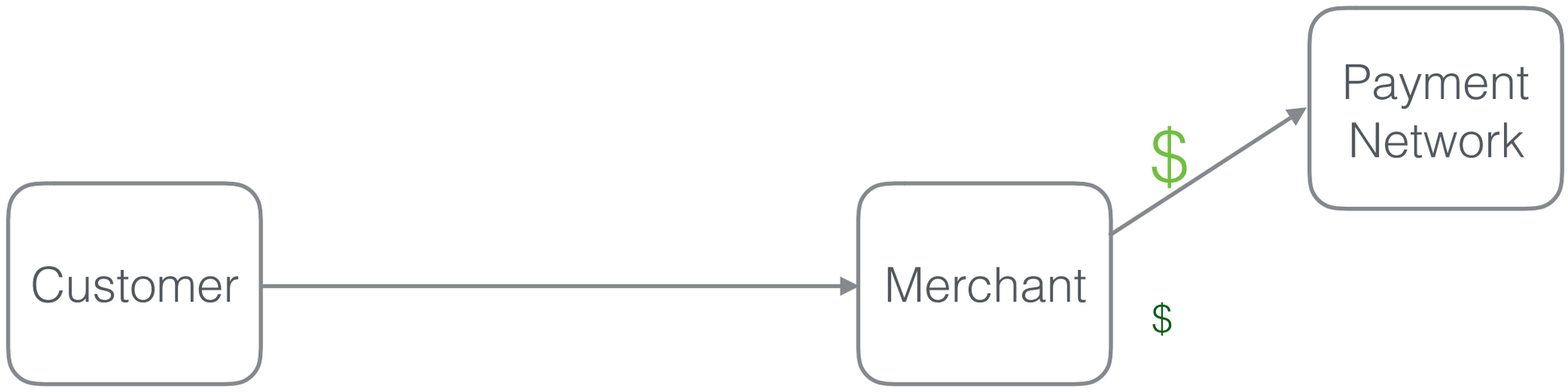
Rich history of **micropayment schemes** constructions:

[Whe96, Riv97, LO98, JY96, RS01, MR02]…

… but no widespread deployments across multiple merchants.

Potential reason: Prior systems required central mediator.

Digital Payments



Supporting small payments is important for applications.

Eg: payments instead of ads while browsing.

Rich history of **micropayment schemes** constructions:

[Whe96, Riv97, LO98, JY96, RS01, MR02]…

… but no widespread deployments across multiple merchants.

Potential reason: Prior systems required central mediator.

Why? Requires creating financial relations, meeting regulations, etc.

Bitcoin

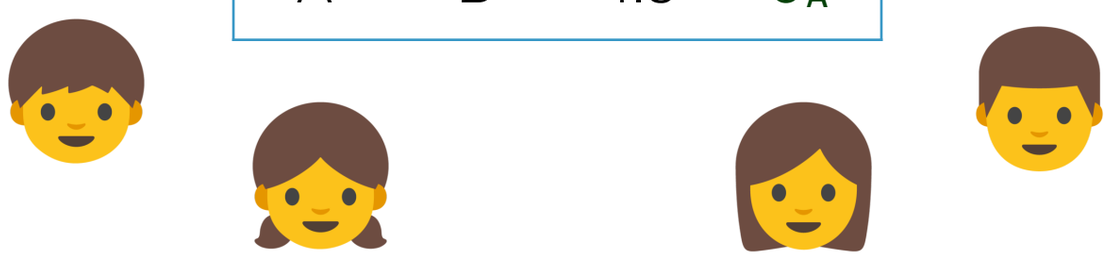
- Decentralized currency w/ quick adoption.
- No need to establish business relations between banks, merchants and regulators.
- To pay, just sign **“from A to B: amt 4.3”**.

LEDGER			
From	To	Amt	Sign
	⋮		
A	M	10	σ_A
M	N	2.3	σ_M

Bitcoin

- Decentralized currency w/ quick adoption.
- No need to establish business relations between banks, merchants and regulators.
- To pay, just sign **“from A to B: amt 4.3”**.

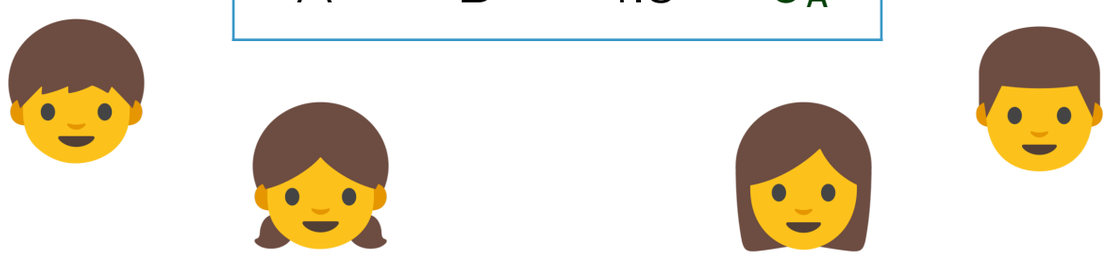
LEDGER			
From	To	Amt	Sign
	⋮		
A	M	10	σ_A
M	N	2.3	σ_M
A	B	4.3	σ_A



Bitcoin

- Decentralized currency w/ quick adoption.
- No need to establish business relations between banks, merchants and regulators.
- To pay, just sign **“from A to B: amt 4.3”**.

LEDGER			
From	To	Amt	Sign
	⋮		
A	M	10	σ_A
M	N	2.3	σ_M
A	B	4.3	σ_A

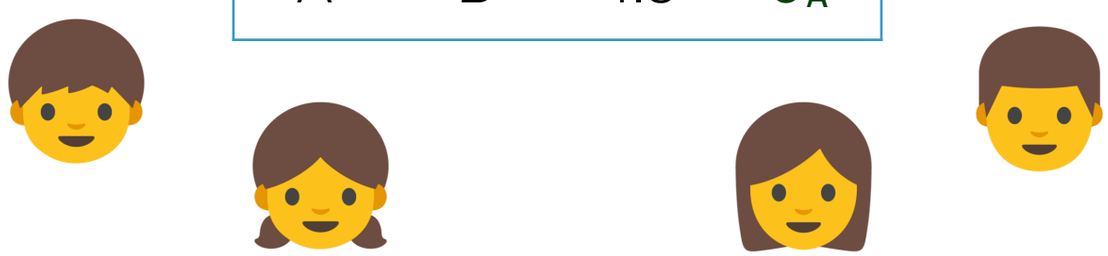


Bitcoin

- Decentralized currency w/ quick adoption.
- No need to establish business relations between banks, merchants and regulators.
- To pay, just sign **“from A to B: amt 4.3”**.

Micropayments on Bitcoin?

LEDGER			
From	To	Amt	Sign
	⋮		
A	M	10	σ_A
M	N	2.3	σ_M
A	B	4.3	σ_A



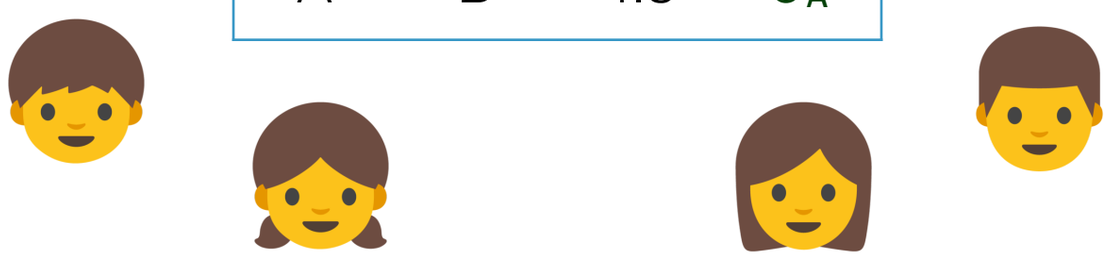
Bitcoin

- Decentralized currency w/ quick adoption.
- No need to establish business relations between banks, merchants and regulators.
- To pay, just sign **“from A to B: amt 4.3”**.

Micropayments on Bitcoin?

Problem 1: High Transaction fees

LEDGER			
From	To	Amt	Sign
	⋮		
A	M	10	σ_A
M	N	2.3	σ_M
A	B	4.3	σ_A



Bitcoin

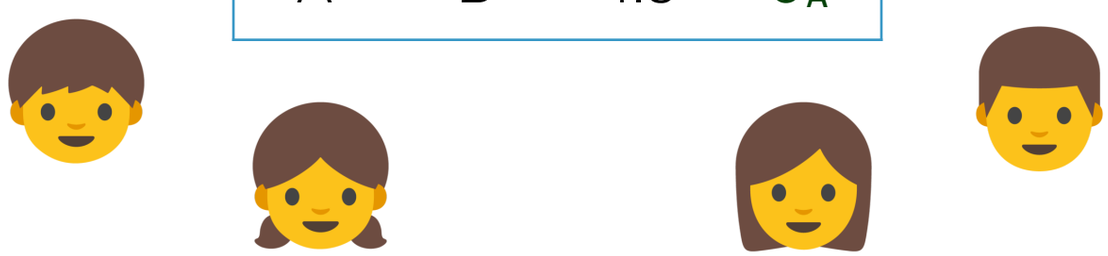
- Decentralized currency w/ quick adoption.
- No need to establish business relations between banks, merchants and regulators.
- To pay, just sign **“from A to B: amt 4.3”**.

Micropayments on Bitcoin?

Problem 1: High Transaction fees

- Projected to get higher.

LEDGER			
From	To	Amt	Sign
	⋮		
A	M	10	σ_A
M	N	2.3	σ_M
A	B	4.3	σ_A



Bitcoin

- Decentralized currency w/ quick adoption.
- No need to establish business relations between banks, merchants and regulators.
- To pay, just sign **“from A to B: amt 4.3”**.

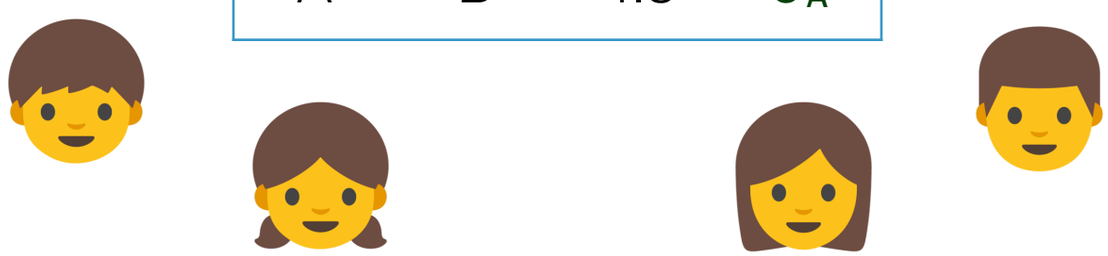
Micropayments on Bitcoin?

Problem 1: High Transaction fees

- Projected to get higher.

Problem 2: Slow Confirmation time

LEDGER			
From	To	Amt	Sign
	⋮		
A	M	10	σ_A
M	N	2.3	σ_M
A	B	4.3	σ_A



Bitcoin

- Decentralized currency w/ quick adoption.
- No need to establish business relations between banks, merchants and regulators.
- To pay, just sign **“from A to B: amt 4.3”**.

Micropayments on Bitcoin?

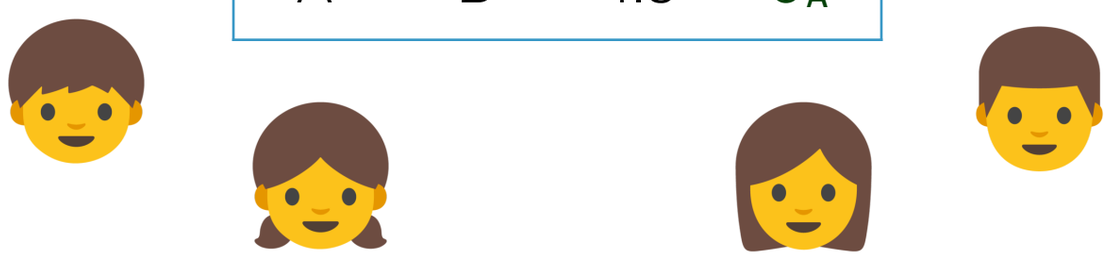
Problem 1: High Transaction fees

- Projected to get higher.

Problem 2: Slow Confirmation time

- Bad for micropayment apps.

LEDGER			
From	To	Amt	Sign
	⋮		
A	M	10	σ_A
M	N	2.3	σ_M
A	B	4.3	σ_A



Bitcoin

- Decentralized currency w/ quick adoption.
- No need to establish business relations between banks, merchants and regulators.
- To pay, just sign **“from A to B: amt 4.3”**.

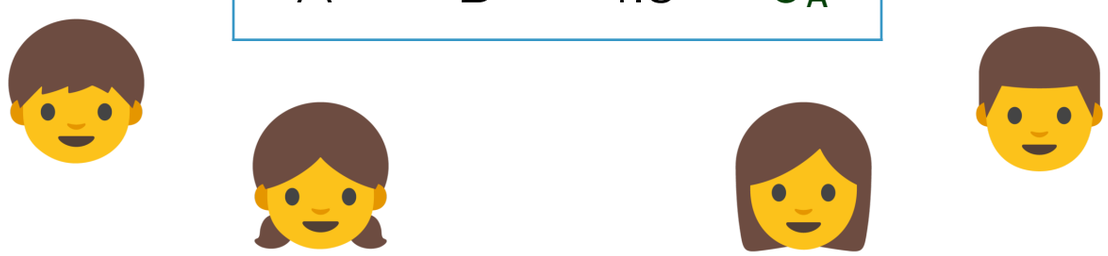
Micropayments on Bitcoin?

Problem 1: High Transaction fees
- Projected to get higher.

Problem 2: Slow Confirmation time
- Bad for micropayment apps.

Problem 3: Lack of Anonymity

LEDGER			
From	To	Amt	Sign
	⋮		
A	M	10	σ_A
M	N	2.3	σ_M
A	B	4.3	σ_A



Bitcoin

- Decentralized currency w/ quick adoption.
- No need to establish business relations between banks, merchants and regulators.
- To pay, just sign **“from A to B: amt 4.3”**.

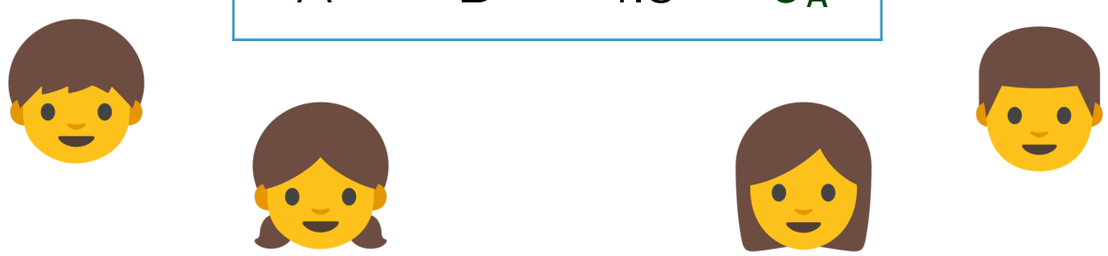
Micropayments on Bitcoin?

Problem 1: High Transaction fees
- Projected to get higher.

Problem 2: Slow Confirmation time
- Bad for micropayment apps.

Problem 3: Lack of Anonymity
- Sender, receiver, amount are all public.

LEDGER			
From	To	Amt	Sign
	⋮		
A	M	10	σ_A
M	N	2.3	σ_M
A	B	4.3	σ_A



Bitcoin

- Decentralized currency w/ quick adoption.
- No need to establish business relations between banks, merchants and regulators.
- To pay, just sign **“from A to B: amt 4.3”**.

Micropayments on Bitcoin?

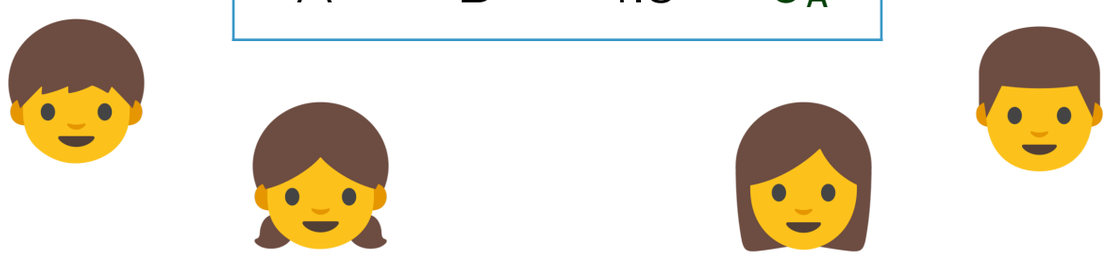
Problem 1: High Transaction fees
- Projected to get higher.

Problem 2: Slow Confirmation time
- Bad for micropayment apps.

Problem 3: Lack of Anonymity
- Sender, receiver, amount are all public.

Consequences:

LEDGER			
From	To	Amt	Sign
	⋮		
A	M	10	σ_A
M	N	2.3	σ_M
A	B	4.3	σ_A



Bitcoin

- Decentralized currency w/ quick adoption.
- No need to establish business relations between banks, merchants and regulators.
- To pay, just sign **“from A to B: amt 4.3”**.

Micropayments on Bitcoin?

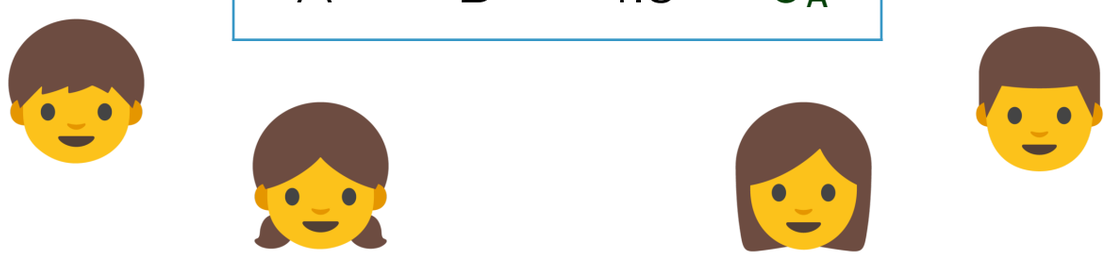
Problem 1: High Transaction fees
- Projected to get higher.

Problem 2: Slow Confirmation time
- Bad for micropayment apps.

Problem 3: Lack of Anonymity
- Sender, receiver, amount are all public.

Consequences:
- No fungibility.

LEDGER			
From	To	Amt	Sign
	⋮		
A	M	10	σ_A
M	N	2.3	σ_M
A	B	4.3	σ_A



Bitcoin

- Decentralized currency w/ quick adoption.
- No need to establish business relations between banks, merchants and regulators.
- To pay, just sign **“from A to B: amt 4.3”**.

Micropayments on Bitcoin?

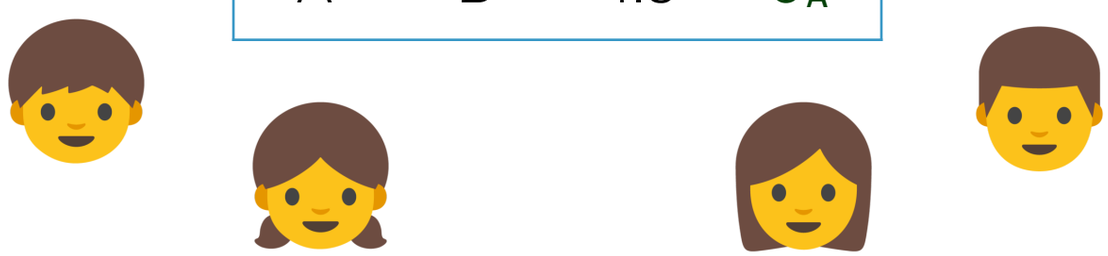
Problem 1: High Transaction fees
- Projected to get higher.

Problem 2: Slow Confirmation time
- Bad for micropayment apps.

Problem 3: Lack of Anonymity
- Sender, receiver, amount are all public.

Consequences:
- No fungibility.
- No privacy. (especially bad for micropayment apps)

LEDGER			
From	To	Amt	Sign
	⋮		
A	M	10	σ_A
M	N	2.3	σ_M
A	B	4.3	σ_A



Bitcoin

- Decentralized currency w/ quick adoption.
- No need to establish business relations between banks, merchants and regulators.
- To pay, just sign **“from A to B: amt 4.3”**.

Micropayments on Bitcoin?

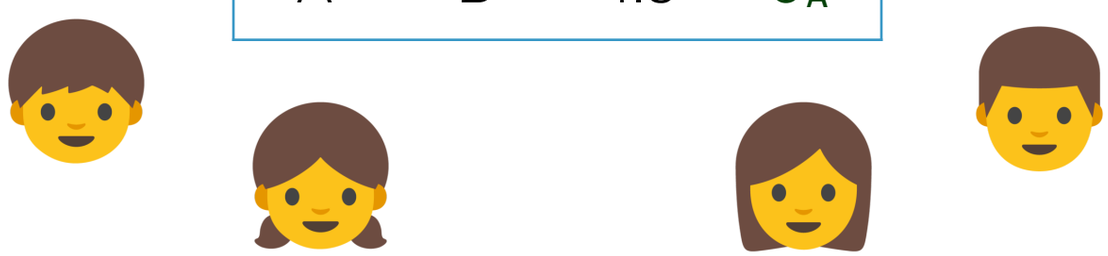
Problem 3: Lack of Anonymity

- Sender, receiver, amount are all public.

Consequences:

- No fungibility.
- No privacy. (especially bad for micropayment apps)

LEDGER			
From	To	Amt	Sign
	⋮		
A	M	10	σ_A
M	N	2.3	σ_M
A	B	4.3	σ_A



Bitcoin

- Decentralized currency w/ quick adoption.
- No need to establish business relations between banks, merchants and regulators.
- To pay, just sign **“from A to B: amt 4.3”**.

Micropayments on Bitcoin?

Pass-Shelat (CCS 2015)

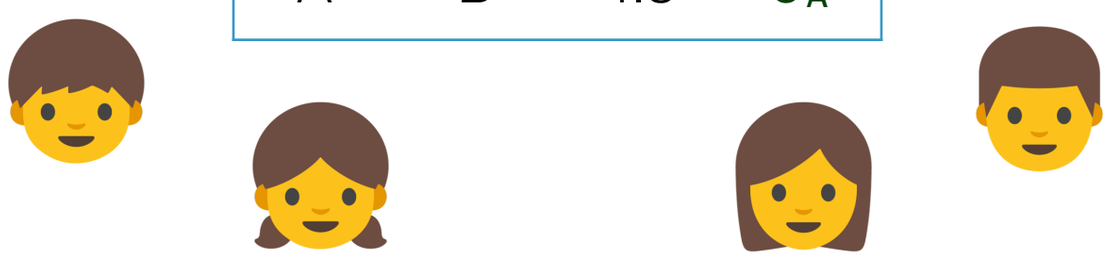
Problem 3: Lack of Anonymity

- Sender, receiver, amount are all public.

Consequences:

- No fungibility.
- No privacy. (especially bad for micropayment apps)

LEDGER			
From	To	Amt	Sign
	⋮		
A	M	10	σ_A
M	N	2.3	σ_M
A	B	4.3	σ_A



Bitcoin

- Decentralized currency w/ quick adoption.
- No need to establish business relations between banks, merchants and regulators.
- To pay, just sign **“from A to B: amt 4.3”**.

Micropayments on Bitcoin?

Pass-Shelat (CCS 2015)

- Probabilistic payments for Bitcoin.

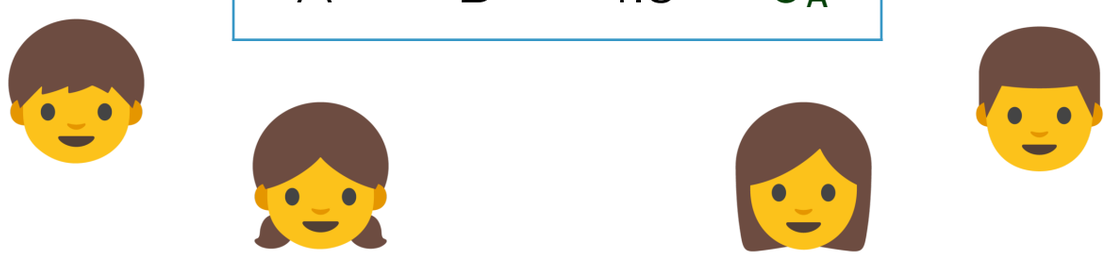
Problem 3: Lack of Anonymity

- Sender, receiver, amount are all public.

Consequences:

- No fungibility.
- No privacy. (especially bad for micropayment apps)

LEDGER			
From	To	Amt	Sign
	⋮		
A	M	10	σ_A
M	N	2.3	σ_M
A	B	4.3	σ_A



Bitcoin

- Decentralized currency w/ quick adoption.
- No need to establish business relations between banks, merchants and regulators.
- To pay, just sign **“from A to B: amt 4.3”**.

Micropayments on Bitcoin?

Pass-Shelat (CCS 2015)

- Probabilistic payments for Bitcoin.
- **Solves problem 1:** Amortized tx fee.

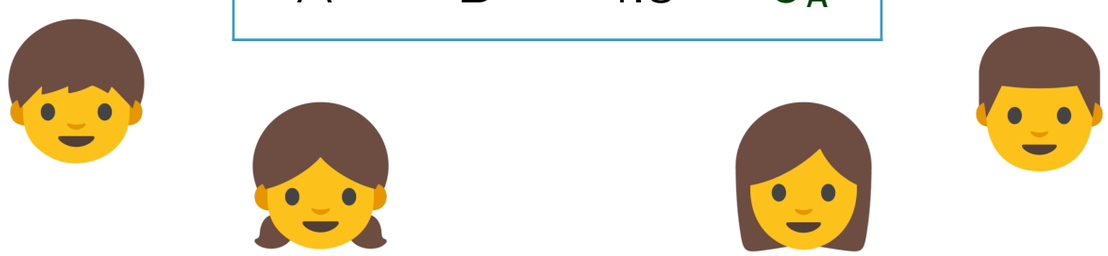
Problem 3: Lack of Anonymity

- Sender, receiver, amount are all public.

Consequences:

- No fungibility.
- No privacy. (especially bad for micropayment apps)

LEDGER			
From	To	Amt	Sign
	⋮		
A	M	10	σ_A
M	N	2.3	σ_M
A	B	4.3	σ_A



Bitcoin

- Decentralized currency w/ quick adoption.
- No need to establish business relations between banks, merchants and regulators.
- To pay, just sign **“from A to B: amt 4.3”**.

Micropayments on Bitcoin?

Pass-Shelat (CCS 2015)

- Probabilistic payments for Bitcoin.
- **Solves problem 1:** Amortized tx fee.
- **Solves problem 2:** Quick confirmation.

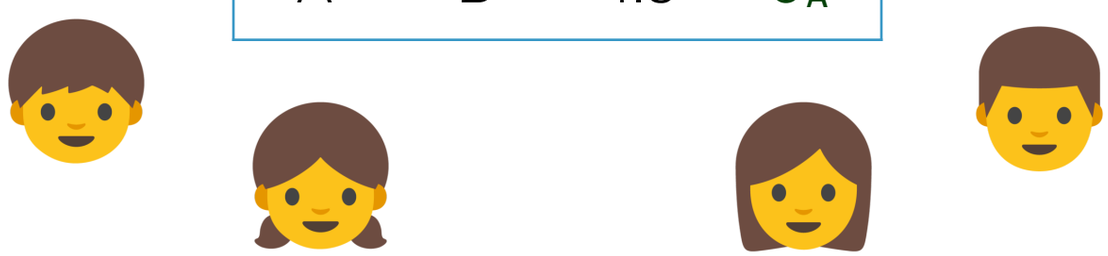
Problem 3: Lack of Anonymity

- Sender, receiver, amount are all public.

Consequences:

- No fungibility.
- No privacy. (especially bad for micropayment apps)

LEDGER			
From	To	Amt	Sign
	⋮		
A	M	10	σ_A
M	N	2.3	σ_M
A	B	4.3	σ_A



Bitcoin

- Decentralized currency w/ quick adoption.
- No need to establish business relations between banks, merchants and regulators.
- To pay, just sign **“from A to B: amt 4.3”**.

Micropayments on Bitcoin?

Pass-Shelat (CCS 2015)

- Probabilistic payments for Bitcoin.
- **Solves problem 1:** Amortized tx fee.
- **Solves problem 2:** Quick confirmation.

Zerocash (Oakland 2014)

- Anonymous Bitcoin-like currency.
- **Solves problem 3:** Hides sender, receiver and amount.

Goal

Goal

micropayments that are:

Goal

micropayments that are:

decentralized (for ease of deployment),

Goal

micropayments that are:

decentralized (for ease of deployment),

anonymous (for fungibility, etc.), and

Goal

micropayments that are:

decentralized (for ease of deployment),

anonymous (for fungibility, etc.), and

offline (for fast response).

Goal

micropayments that are:

decentralized (for ease of deployment),

anonymous (for fungibility, etc.), and

offline (for fast response).

Contributions

Goal

micropayments that are:

decentralized (for ease of deployment),

anonymous (for fungibility, etc.), and

offline (for fast response).

Contributions

1. Definition of **cryptographic primitive** via **ideal functionality**.

Goal

micropayments that are:

decentralized (for ease of deployment),

anonymous (for fungibility, etc.), and

offline (for fast response).

Contributions

1. Definition of **cryptographic primitive** via **ideal functionality**.
2. **Construction** under **standard crypto assumptions**.

Goal

micropayments that are:

decentralized (for ease of deployment),

anonymous (for fungibility, etc.), and

offline (for fast response).

Contributions

1. Definition of **cryptographic primitive** via **ideal functionality**.
2. **Construction** under **standard crypto assumptions**.
3. Techniques: we use two tools:

Goal

micropayments that are:

decentralized (for ease of deployment),

anonymous (for fungibility, etc.), and

offline (for fast response).

Contributions

1. Definition of **cryptographic primitive** via **ideal functionality**.
2. **Construction** under **standard crypto assumptions**.
3. Techniques: we use two tools:
   - **translucent crypto**: new **fractional message transfer** protocol.
     (probabilistic)

Goal

micropayments that are:

decentralized (for ease of deployment),

anonymous (for fungibility, etc.), and

offline (for fast response).

Contributions

1. Definition of **cryptographic primitive** via **ideal functionality**.
2. **Construction** under **standard crypto assumptions**.
3. Techniques: we use two tools:
   - **translucent crypto**: new **fractional message transfer** protocol. (probabilistic)
   - **game theory**: characterization of double-spending.

Probabilistic Payments

Probabilistic Payments

Alice "pays" Bob $0.01

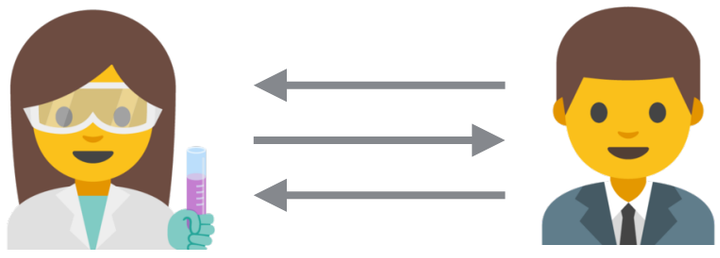
Probabilistic Payments

Alice "pays" Bob $0.01

$1

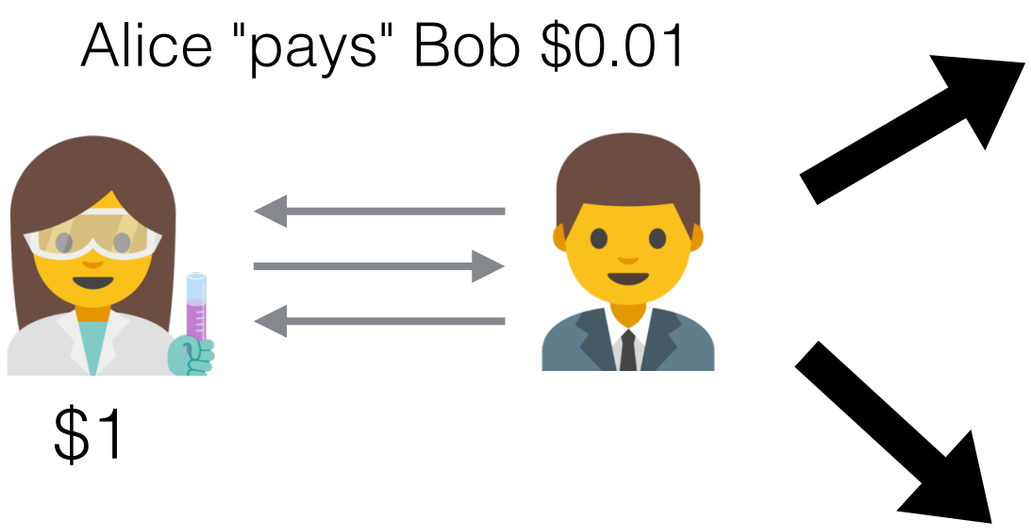
Probabilistic Payments

Alice "pays" Bob $0.01

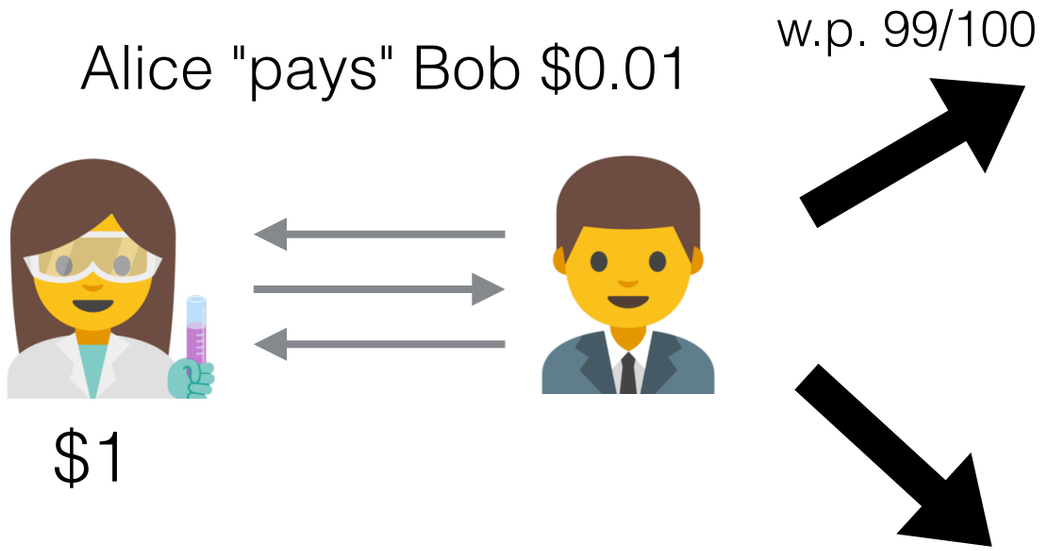


$1

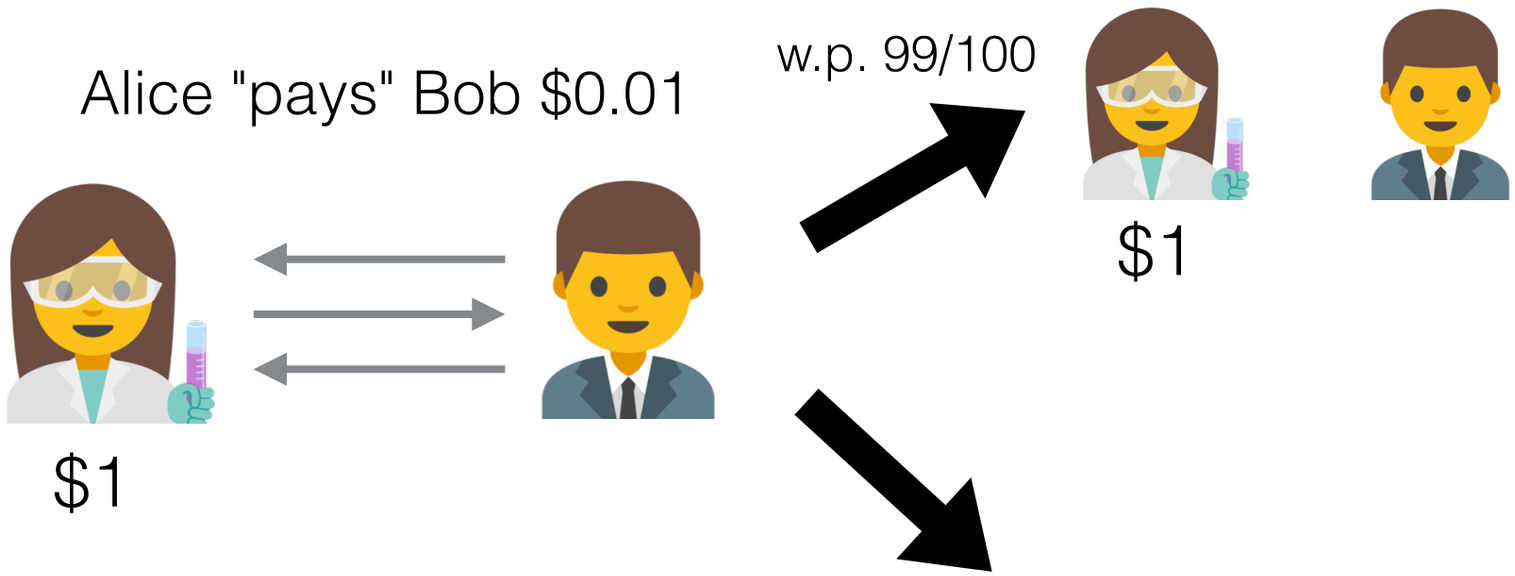
Probabilistic Payments



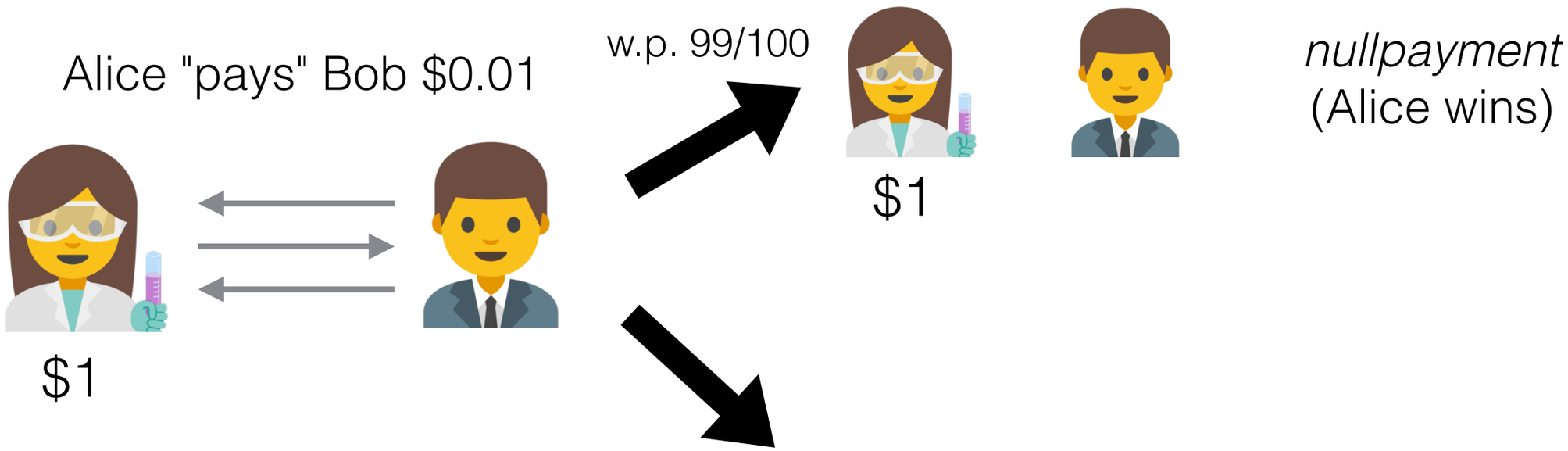
Probabilistic Payments



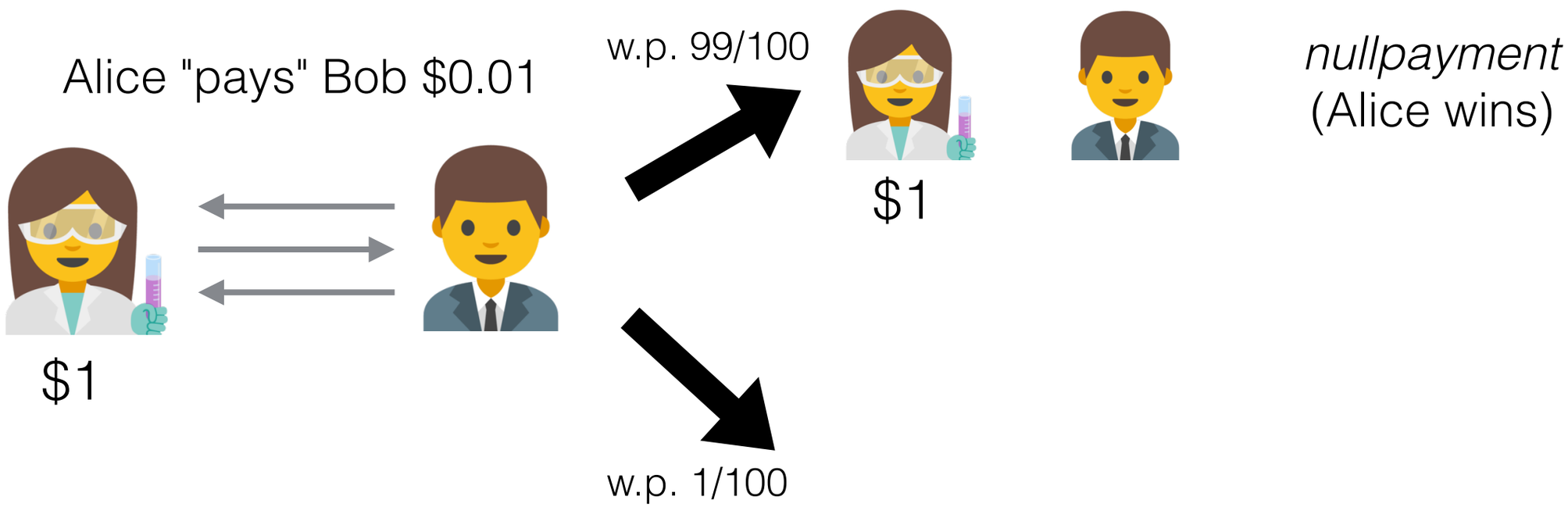
Probabilistic Payments



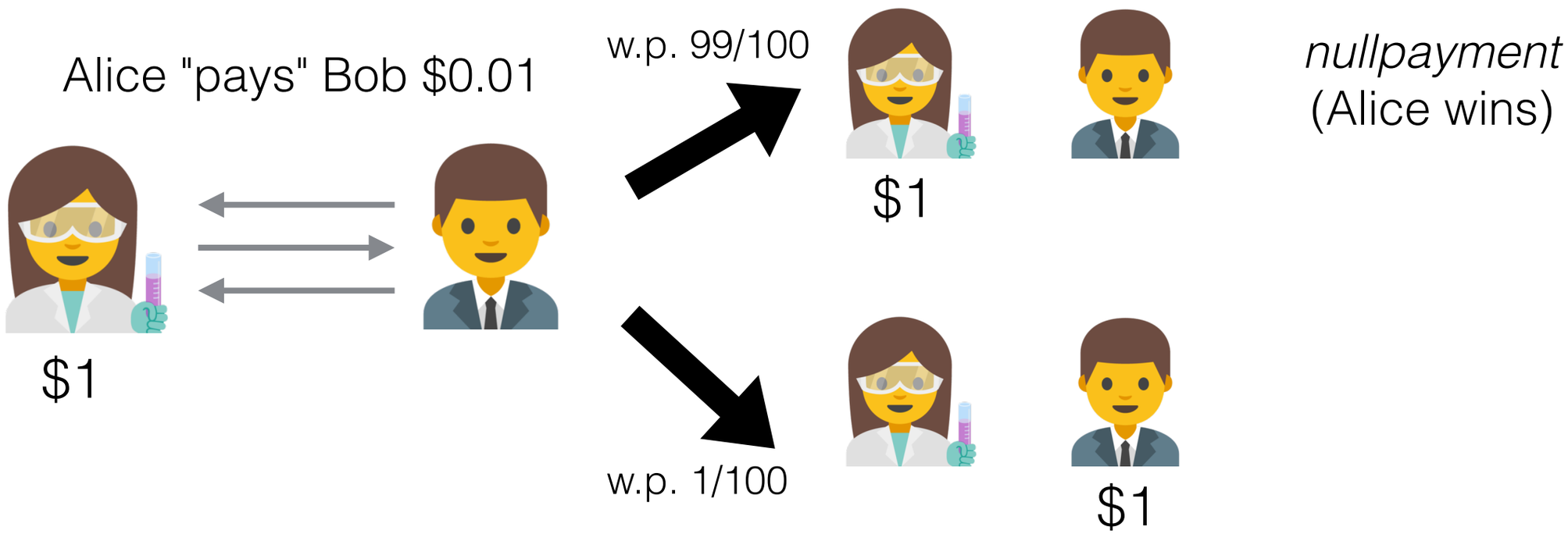
Probabilistic Payments



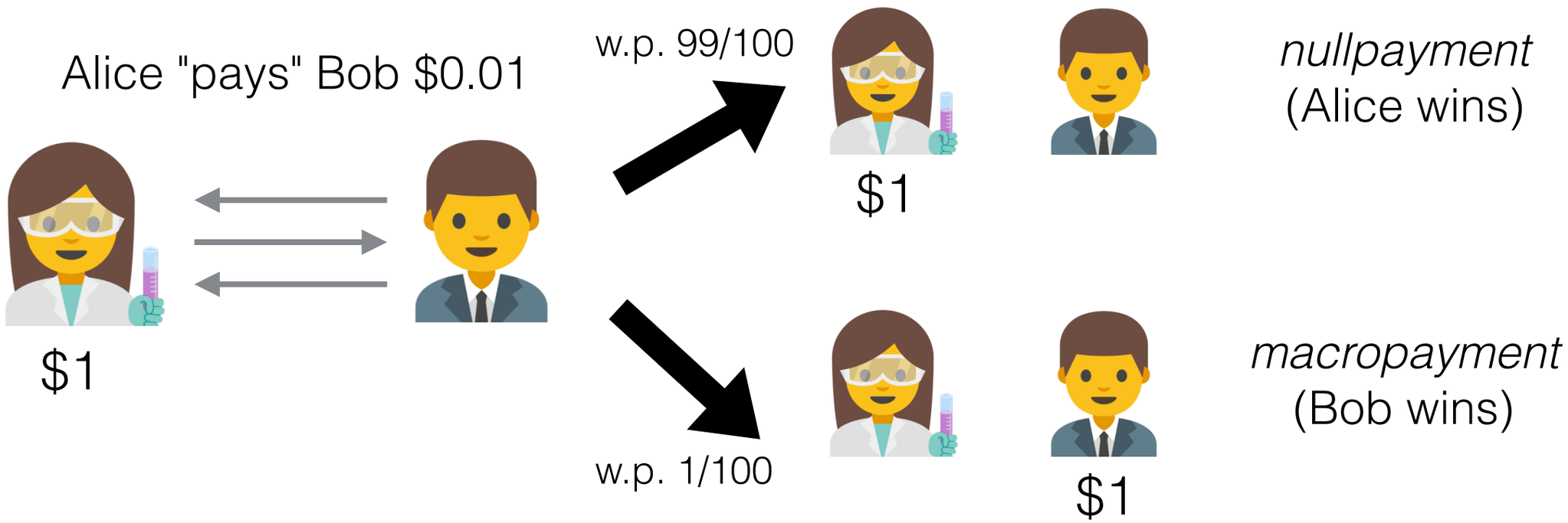
Probabilistic Payments



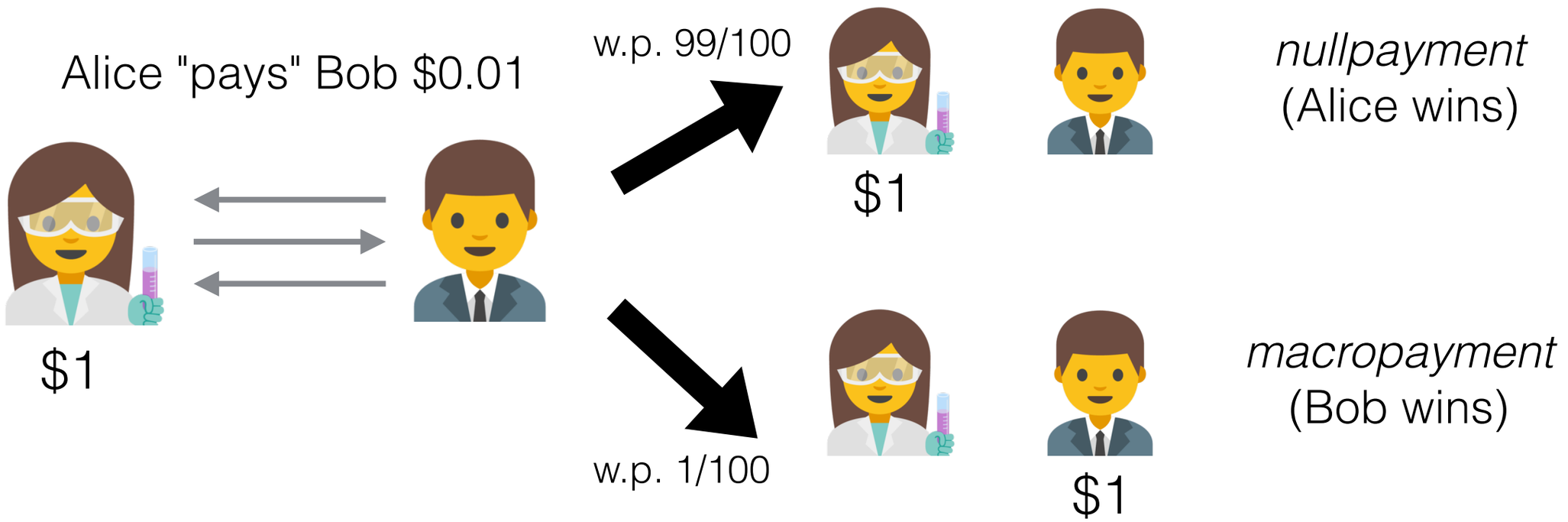
Probabilistic Payments



Probabilistic Payments

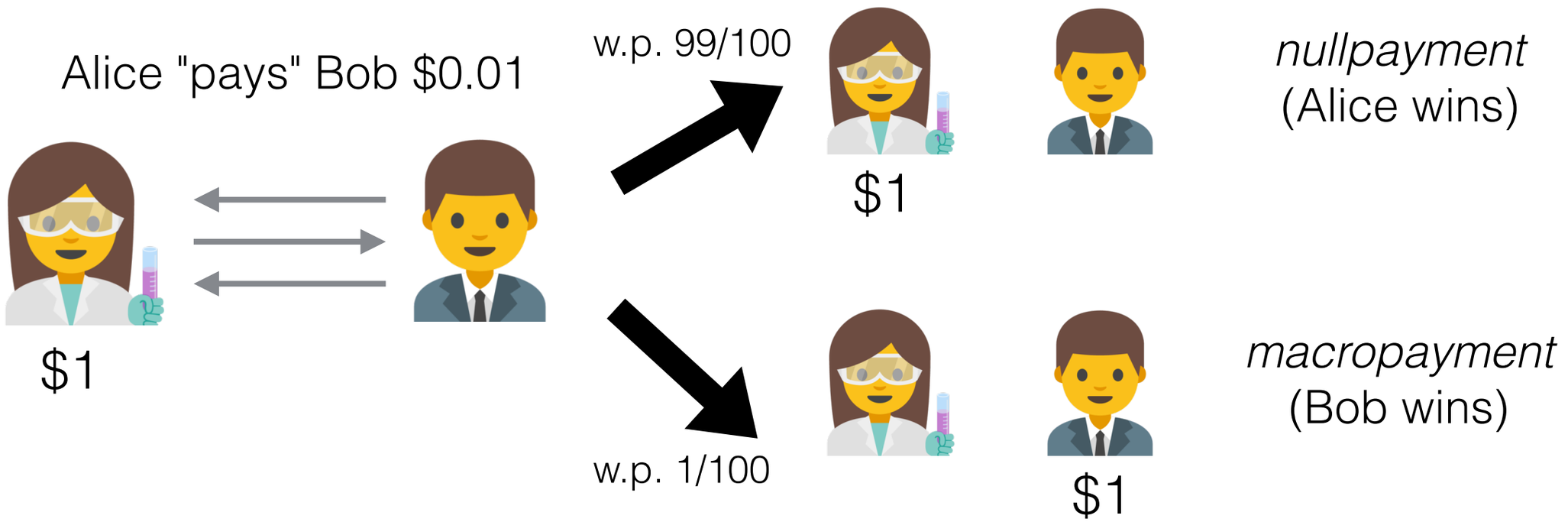


Probabilistic Payments



Probabilistic payments imply micropayments:

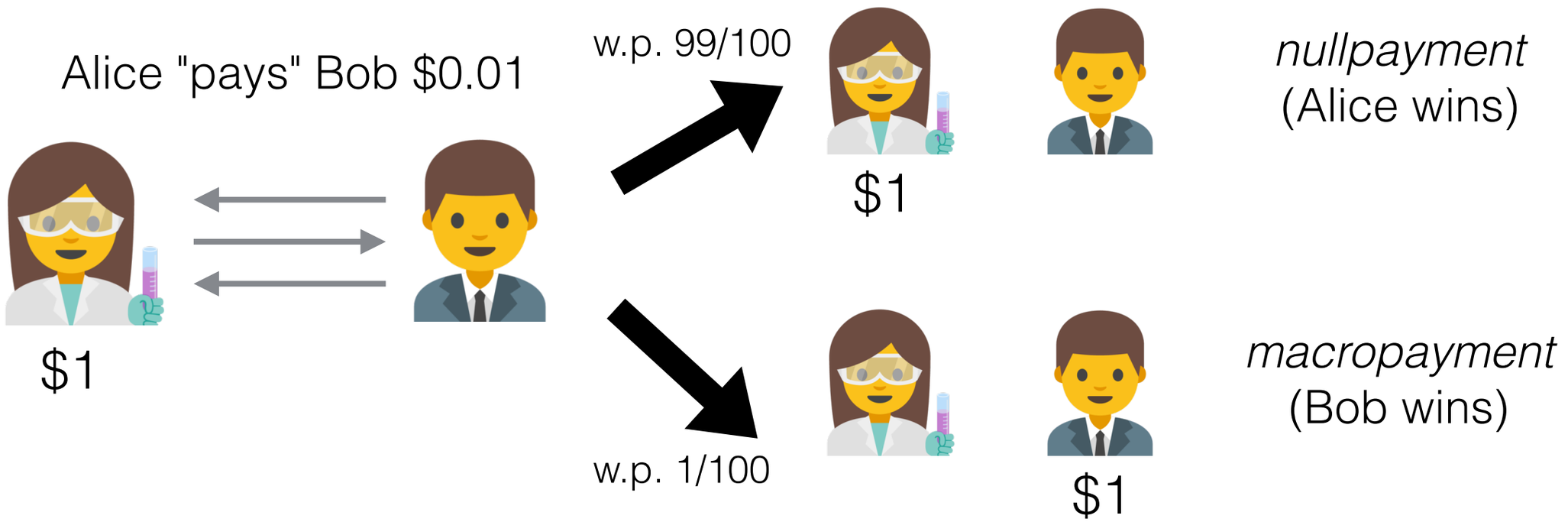
Probabilistic Payments



Probabilistic payments imply micropayments:

Transaction fee is amortized over many payments.

Probabilistic Payments



Probabilistic payments imply micropayments:

Transaction fee is amortized over many payments.

Nullpayments are offline and do not require interaction with payment network.

Building Blocks

Pass-Shelat

Zerocash

Building Blocks

Pass-Shelat

coin-flipping + Bitcoin

Zerocash

Building Blocks

Pass-Shelat

coin-flipping + Bitcoin

Ledger			
From	To	Amt	Sign
	⋮		
M	N	2.3	σ_M
A	M	10	σ_A

Zerocash

Building Blocks

Pass-Shelat

coin-flipping + Bitcoin

1. Alice escrows v.

Ledger			
From	To	Amt	Sign
⋮			
M	N	2.3	σ_M
A	M	10	σ_A
A	E	4.3	σ_A

Zerocash

Building Blocks

Pass-Shelat

coin-flipping + Bitcoin

1. Alice escrows v.
2. Alice and Bob engage in coin-flip.

Ledger			
From	To	Amt	Sign
⋮			
M	N	2.3	σ_M
A	M	10	σ_A
A	E	4.3	σ_A

coin-flip

Zerocash

Building Blocks

Pass-Shelat

coin-flipping + Bitcoin

1. Alice escrows v.
2. Alice and Bob engage in coin-flip.
3. If Alice wins: she can reuse escrow.

Ledger			
From	To	Amt	Sign
⋮			
M	N	2.3	σ_M
A	M	10	σ_A
A	E	4.3	σ_A

coin-flip

Zerocash

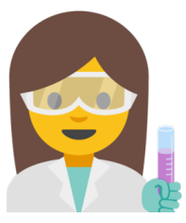
Building Blocks

Pass-Shelat

coin-flipping + Bitcoin

1. Alice escrows v.
2. Alice and Bob engage in coin-flip.
3. If Alice wins: she can reuse escrow.
4. If Bob wins: he gets v.

Ledger			
From	To	Amt	Sign
⋮			
M	N	2.3	σ_M
A	M	10	σ_A
A	E	4.3	σ_A
E	B	4.3	σ_E



coin-flip

Zerocash

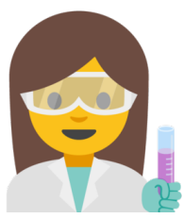
Building Blocks

Pass-Shelat

coin-flipping + Bitcoin

1. Alice escrows v.
2. Alice and Bob engage in coin-flip.
3. If Alice wins: she can reuse escrow.
4. If Bob wins: he gets v.

Ledger			
From	To	Amt	Sign
⋮			
M	N	2.3	σ_M
A	M	10	σ_A
A	E	4.3	σ_A
E	B	4.3	σ_E



coin-flip

Zerocash

zero knowledge proofs + Bitcoin

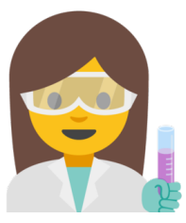
Building Blocks

Pass-Shelat

coin-flipping + Bitcoin

1. Alice escrows v.
2. Alice and Bob engage in coin-flip.
3. If Alice wins: she can reuse escrow.
4. If Bob wins: he gets v.

Ledger			
From	To	Amt	Sign
⋮			
M	N	2.3	σ_M
A	M	10	σ_A
A	E	4.3	σ_A
E	B	4.3	σ_E



coin-flip

Zerocash

zero knowledge proofs + Bitcoin

pk_A, sk_A

Ledger		
Old	New	Proof
⋮		
8436378	cm_1	π_1
6327690	cm_2	π_2

pk_B, sk_B

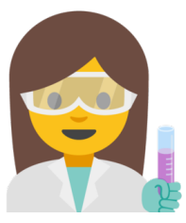
Building Blocks

Pass-Shelat

coin-flipping + Bitcoin

1. Alice escrows v.
2. Alice and Bob engage in coin-flip.
3. If Alice wins: she can reuse escrow.
4. If Bob wins: he gets v.

Ledger			
From	To	Amt	Sign
⋮			
M	N	2.3	σ_M
A	M	10	σ_A
A	E	4.3	σ_A
E	B	4.3	σ_E



coin-flip

Zerocash

zero knowledge proofs + Bitcoin

1. Alice owns coin c_1 with comm cm_1.

pk_A, sk_A

Ledger		
Old	New	Proof
⋮		
8436378	cm_1	π_1
6327690	cm_2	π_2

pk_B, sk_B

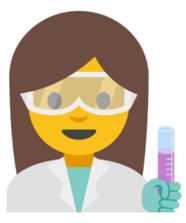
Building Blocks

Pass-Shelat

coin-flipping + Bitcoin

1. Alice escrows v.
2. Alice and Bob engage in coin-flip.
3. If Alice wins: she can reuse escrow.
4. If Bob wins: he gets v.

Ledger			
From	To	Amt	Sign
	⋮		
M	N	2.3	σ_M
A	M	10	σ_A
A	E	4.3	σ_A
E	B	4.3	σ_E



coin-flip

Zerocash

zero knowledge proofs + Bitcoin

1. Alice owns coin c_1 with comm cm_1.
2. To pay Bob, Alice:

pk_A, sk_A

Ledger		
Old	New	Proof
⋮		
8436378	cm_1	π_1
6327690	cm_2	π_2

pk_B, sk_B

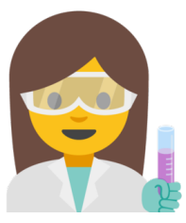
Building Blocks

Pass-Shelat

coin-flipping + Bitcoin

1. Alice escrows v.
2. Alice and Bob engage in coin-flip.
3. If Alice wins: she can reuse escrow.
4. If Bob wins: he gets v.

Ledger			
From	To	Amt	Sign
⋮			
M	N	2.3	σ_M
A	M	10	σ_A
A	E	4.3	σ_A
E	B	4.3	σ_E



coin-flip

Zerocash

zero knowledge proofs + Bitcoin

1. Alice owns coin c_1 with comm cm_1.
2. To pay Bob, Alice:
   a) derives sn_1 from c_1 and sk_A.

sn_1

c_1

pk_A, sk_A

Ledger		
Old	New	Proof
⋮		
8436378	cm_1	π_1
6327690	cm_2	π_2

pk_B, sk_B

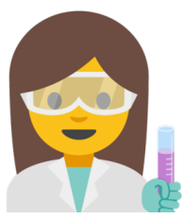
Building Blocks

Pass-Shelat

coin-flipping + Bitcoin

1. Alice escrows v.
2. Alice and Bob engage in coin-flip.
3. If Alice wins: she can reuse escrow.
4. If Bob wins: he gets v.

Ledger			
From	To	Amt	Sign
⋮			
M	N	2.3	σ_M
A	M	10	σ_A
A	E	4.3	σ_A
E	B	4.3	σ_E



coin-flip

Zerocash

zero knowledge proofs + Bitcoin

1. Alice owns coin c_1 with comm cm_1.
2. To pay Bob, Alice:
   a) derives sn_1 from c_1 and sk_A.
   b) creates new coin c_3 with comm cm_3.

Ledger		
Old	New	Proof
⋮		
8436378	cm_1	π_1
6327690	cm_2	π_2

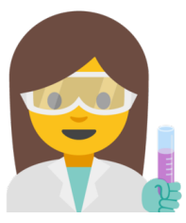
Building Blocks

Pass-Shelat

coin-flipping + Bitcoin

1. Alice escrows v.
2. Alice and Bob engage in coin-flip.
3. If Alice wins: she can reuse escrow.
4. If Bob wins: he gets v.

Ledger			
From	To	Amt	Sign
⋮			
M	N	2.3	σ_M
A	M	10	σ_A
A	E	4.3	σ_A
E	B	4.3	σ_E



coin-flip

Zerocash

zero knowledge proofs + Bitcoin

1. Alice owns coin c_1 with comm cm_1.
2. To pay Bob, Alice:
   a) derives sn_1 from c_1 and sk_A.
   b) creates new coin c_3 with comm cm_3.
   c) creates ZK proof π_3 for above.

sn_1

c_1

pk_A, sk_A

Ledger		
Old	New	Proof
⋮		
8436378	cm_1	π_1
6327690	cm_2	π_2

pk_B, sk_B

Building Blocks

Pass-Shelat

coin-flipping + Bitcoin

1. Alice escrows v.
2. Alice and Bob engage in coin-flip.
3. If Alice wins: she can reuse escrow.
4. If Bob wins: he gets v.

Ledger			
From	To	Amt	Sign
	⋮		
M	N	2.3	σ_M
A	M	10	σ_A
A	E	4.3	σ_A
E	B	4.3	σ_E

coin-flip

Zerocash

zero knowledge proofs + Bitcoin

1. Alice owns coin c_1 with comm cm_1.
2. To pay Bob, Alice:
   a) derives sn_1 from c_1 and sk_A.
   b) creates new coin c_3 with comm cm_3.
   c) creates ZK proof π_3 for above.
   d) appends tx = (sn_1, cm_3, π_3).

Ledger		
Old	New	Proof
⋮		
8436378	cm_1	π_1
6327690	cm_2	π_2
sn_1	cm_3	π_3

Building Blocks

Pass-Shelat

coin-flipping + Bitcoin

1. Alice escrows v.
2. Alice and Bob engage in coin-flip.
3. If Alice wins: she can reuse escrow.
4. If Bob wins: he gets v.

Ledger			
From	To	Amt	Sign
⋮			
M	N	2.3	σ_M
A	M	10	σ_A
A	E	4.3	σ_A
E	B	4.3	σ_E

coin-flip

Zerocash

zero knowledge proofs + Bitcoin

1. Alice owns coin c_1 with comm cm_1.
2. To pay Bob, Alice:
   a) derives sn_1 from c_1 and sk_A.
   b) creates new coin c_3 with comm cm_3.
   c) creates ZK proof π_3 for above.
   d) appends tx = (sn_1, cm_3, π_3).

Cannot link sn_1 with cm_1 without sk_A.

Ledger		
Old	New	Proof
⋮		
8436378	cm_1	π_1
6327690	cm_2	π_2
sn_1	cm_3	π_3

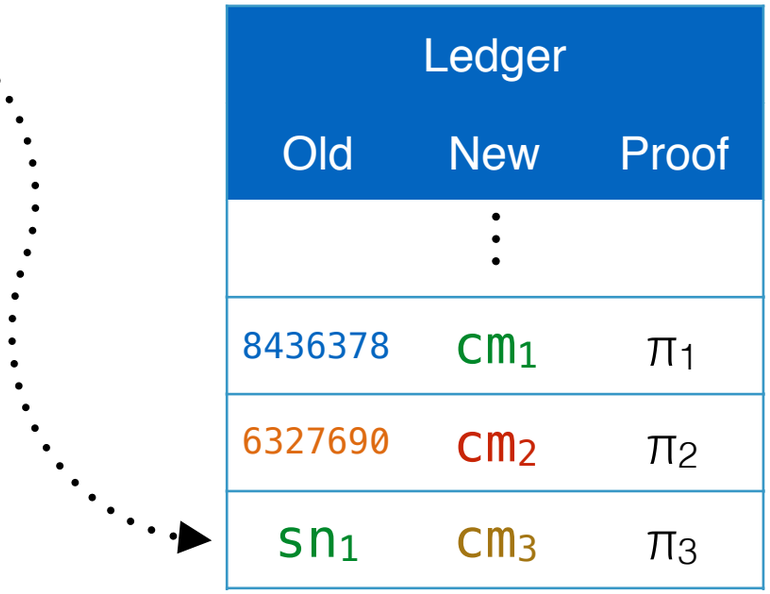
Naive Attempt: PS + Zerocash

Naive Attempt: PS + Zerocash

Ledger		
Old	New	Proof
	⋮	
8436378	cm_1	π_1
6327690	cm_2	π_2

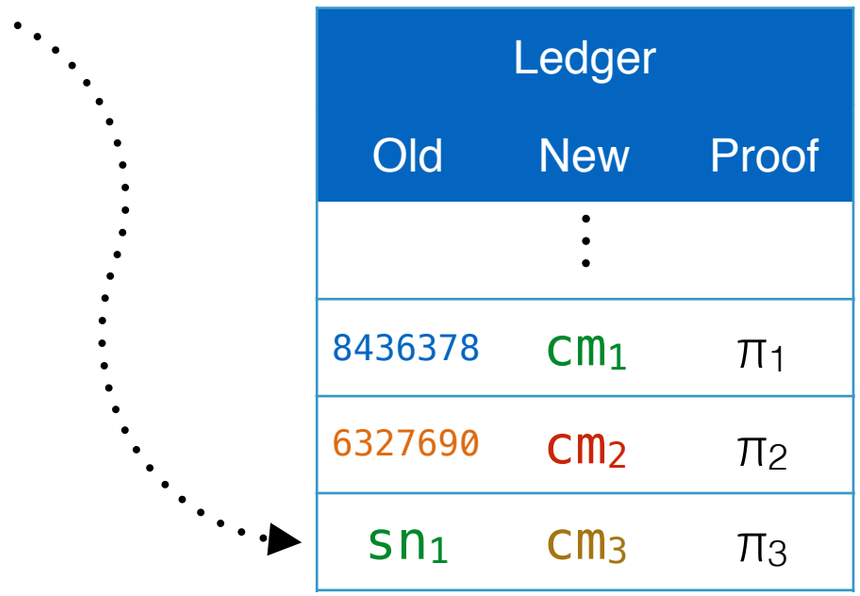
Naive Attempt: PS + Zerocash

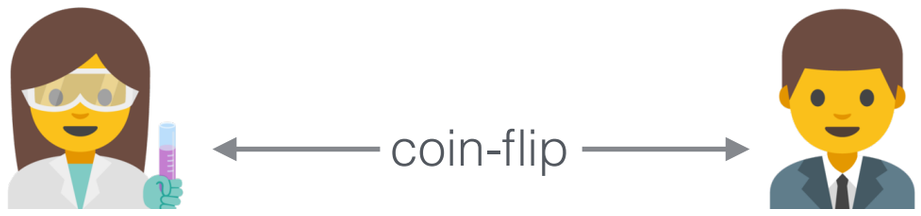
1. Alice escrows v in a Zerocash transaction.



Naive Attempt: PS + Zerocash

1. Alice escrows v in a Zerocash transaction.
2. Alice and Bob engage in coin-flip.

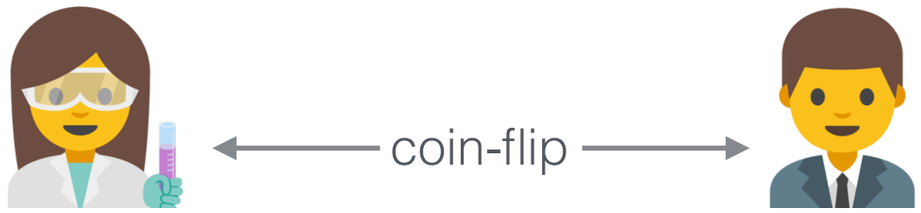




Naive Attempt: PS + Zerocash

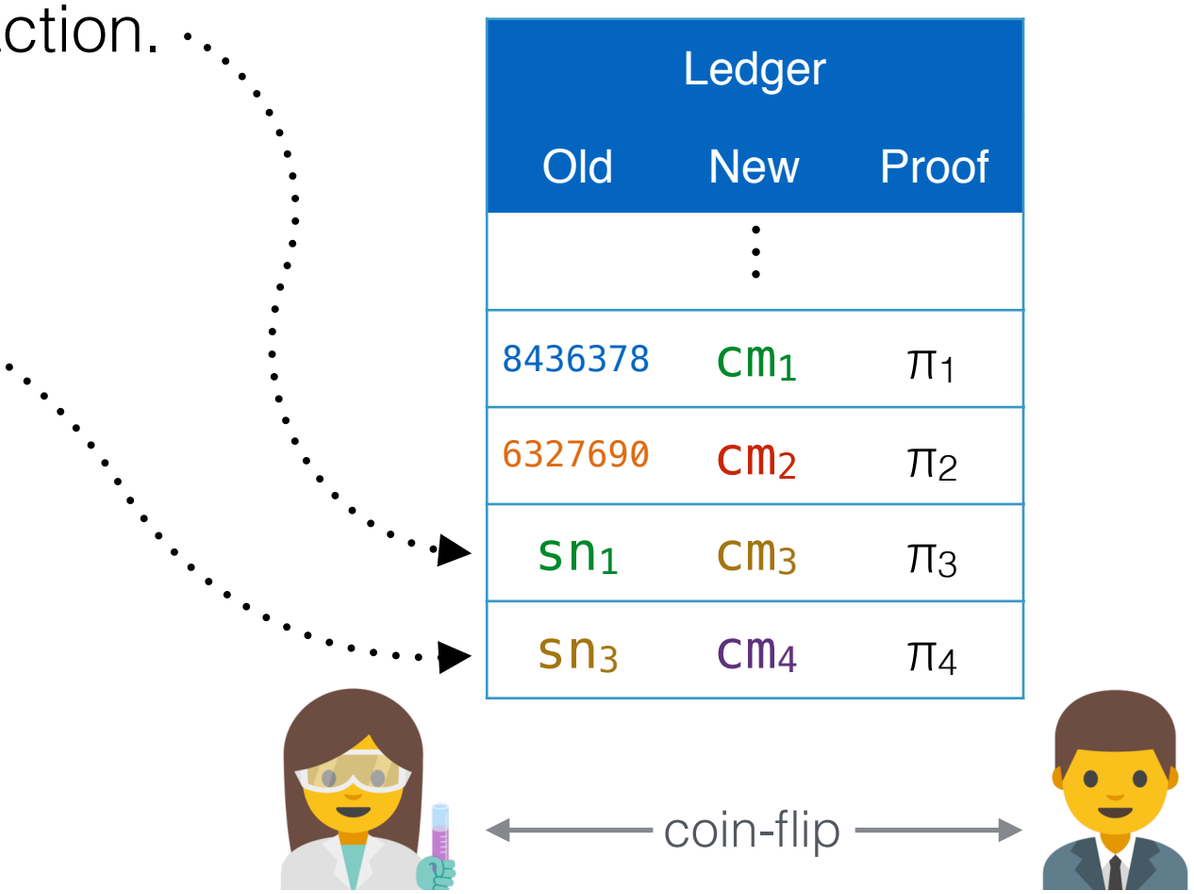
1. Alice escrows v in a Zerocash transaction.
2. Alice and Bob engage in coin-flip.
3. If Alice wins: she can reuse escrow.

Ledger		
Old	New	Proof
	⋮	
8436378	cm_1	π_1
6327690	cm_2	π_2
sn_1	cm_3	π_3



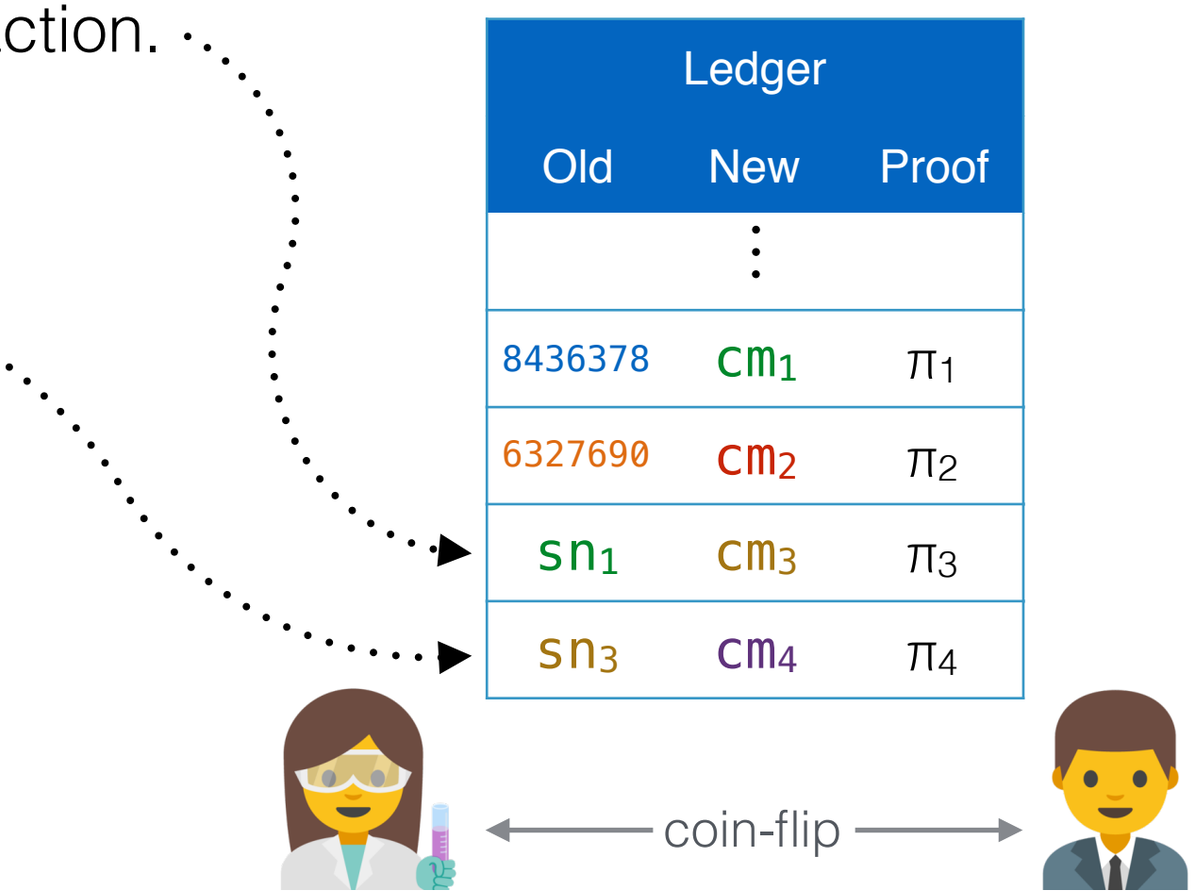
Naive Attempt: PS + Zerocash

1. Alice escrows v in a Zerocash transaction.
2. Alice and Bob engage in coin-flip.
3. If Alice wins: she can reuse escrow.
4. If Bob wins: he gets v.



Naive Attempt: PS + Zerocash

1. Alice escrows v in a Zerocash transaction.
2. Alice and Bob engage in coin-flip.
3. If Alice wins: she can reuse escrow.
4. If Bob wins: he gets v.



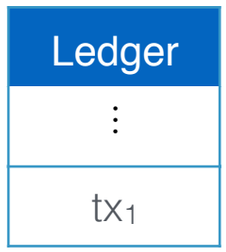
Major Issues:

Linkability

Double Spending

Problem 1: Linkability

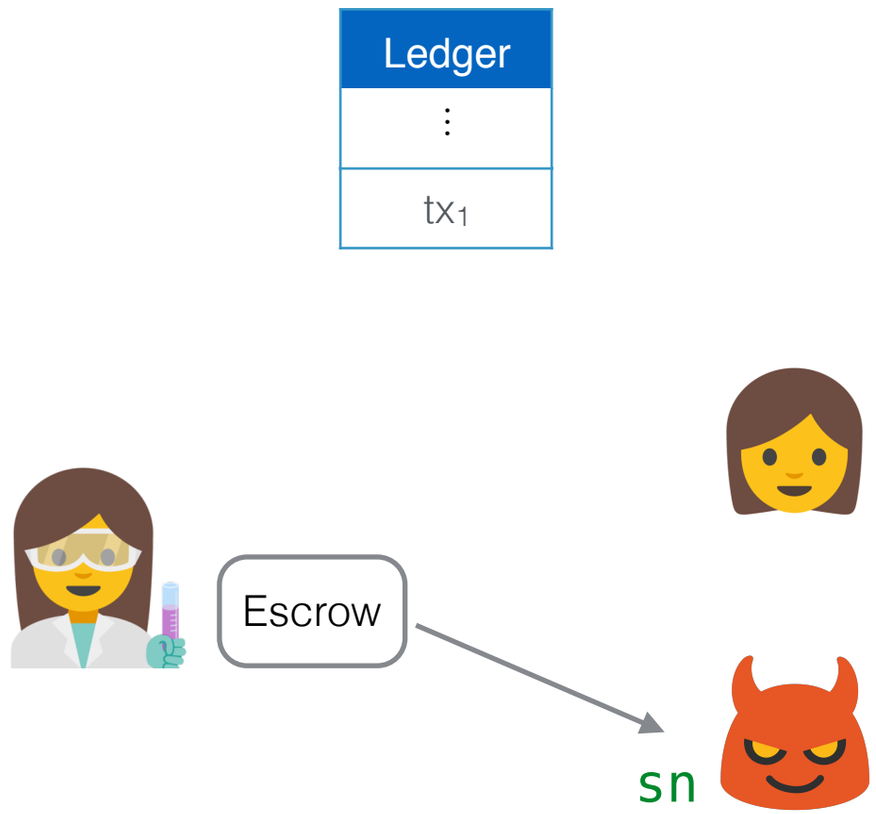
Problem 1: Linkability



Escrow

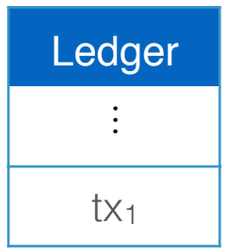
- To amortize transaction fees, Alice has to reuse escrow.
- Bob **always** learns serial number of escrowed coin.

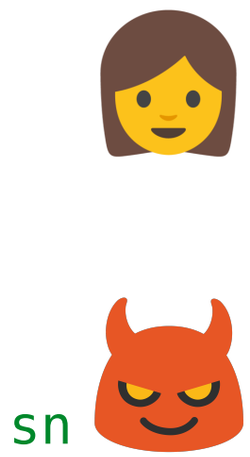
Problem 1: Linkability



- To amortize transaction fees, Alice has to reuse escrow.
- Bob **always** learns serial number of escrowed coin.
  - Can track Alice when she spends coin w/ others.

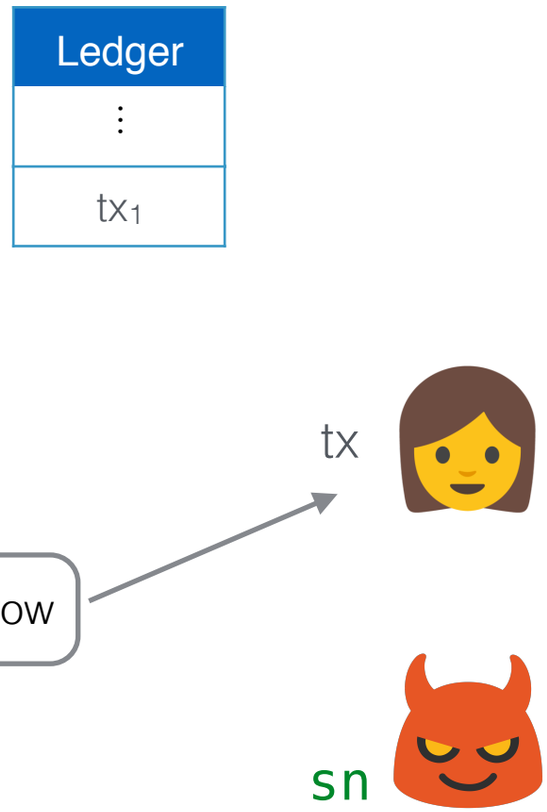
Problem 1: Linkability





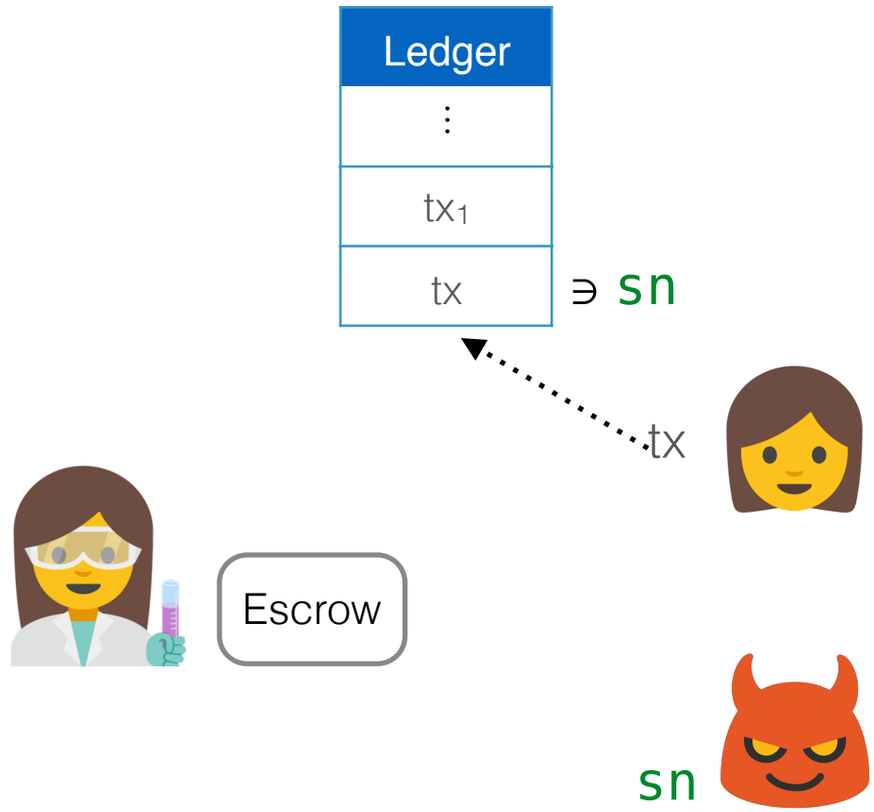
- To amortize transaction fees, Alice has to reuse escrow.
- Bob **always** learns serial number of escrowed coin.
  - Can track Alice when she spends coin w/ others.

Problem 1: Linkability



- To amortize transaction fees, Alice has to reuse escrow.
- Bob **always** learns serial number of escrowed coin.
  - Can track Alice when she spends coin w/ others.

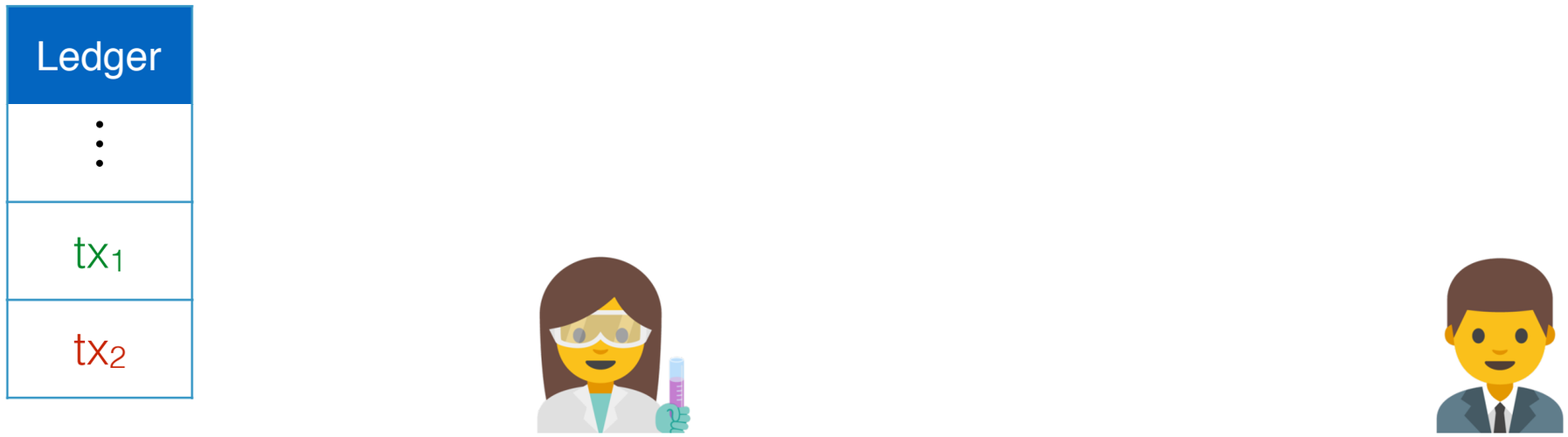
Problem 1: Linkability



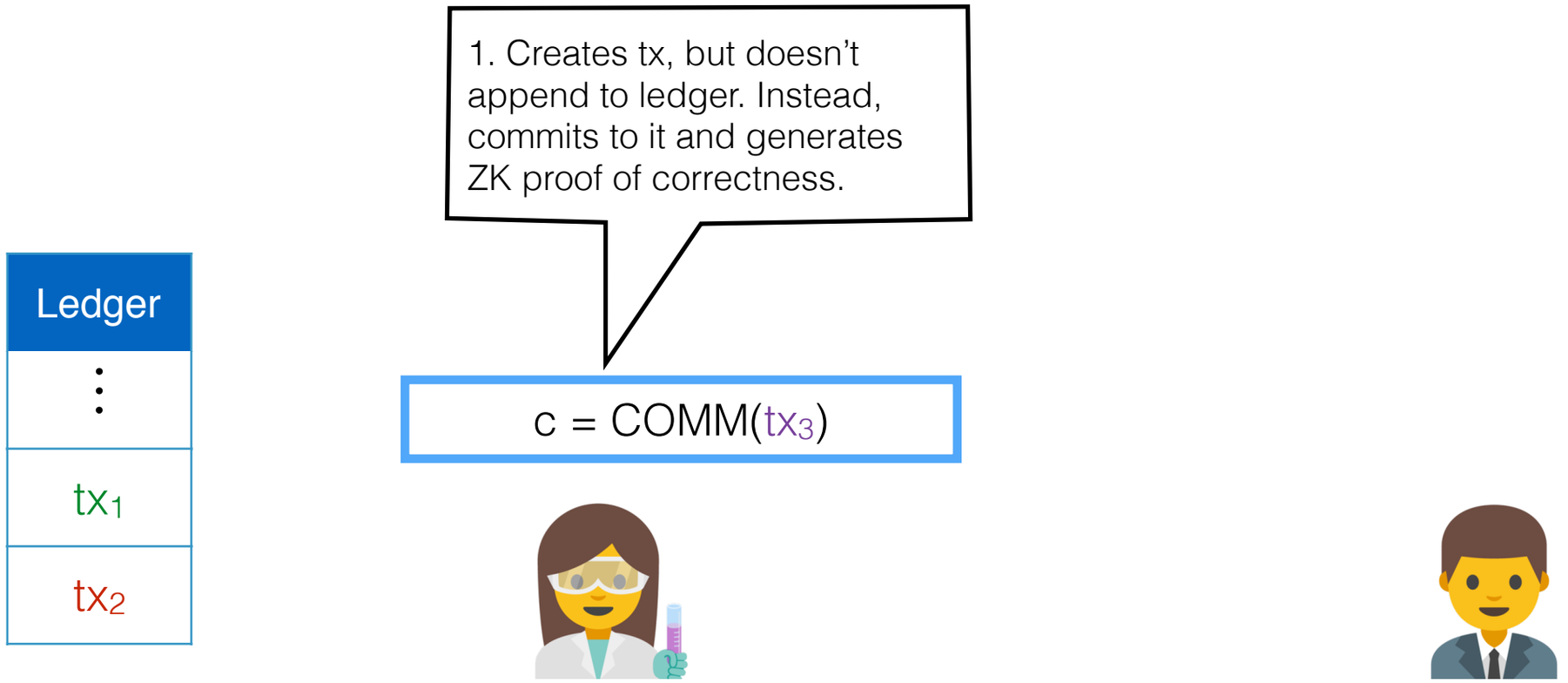
- To amortize transaction fees, Alice has to reuse escrow.
- Bob **always** learns serial number of escrowed coin.
  - Can track Alice when she spends coin w/ others.
- Further attacks lead to loss of most privacy.

Solution: Make sn translucent

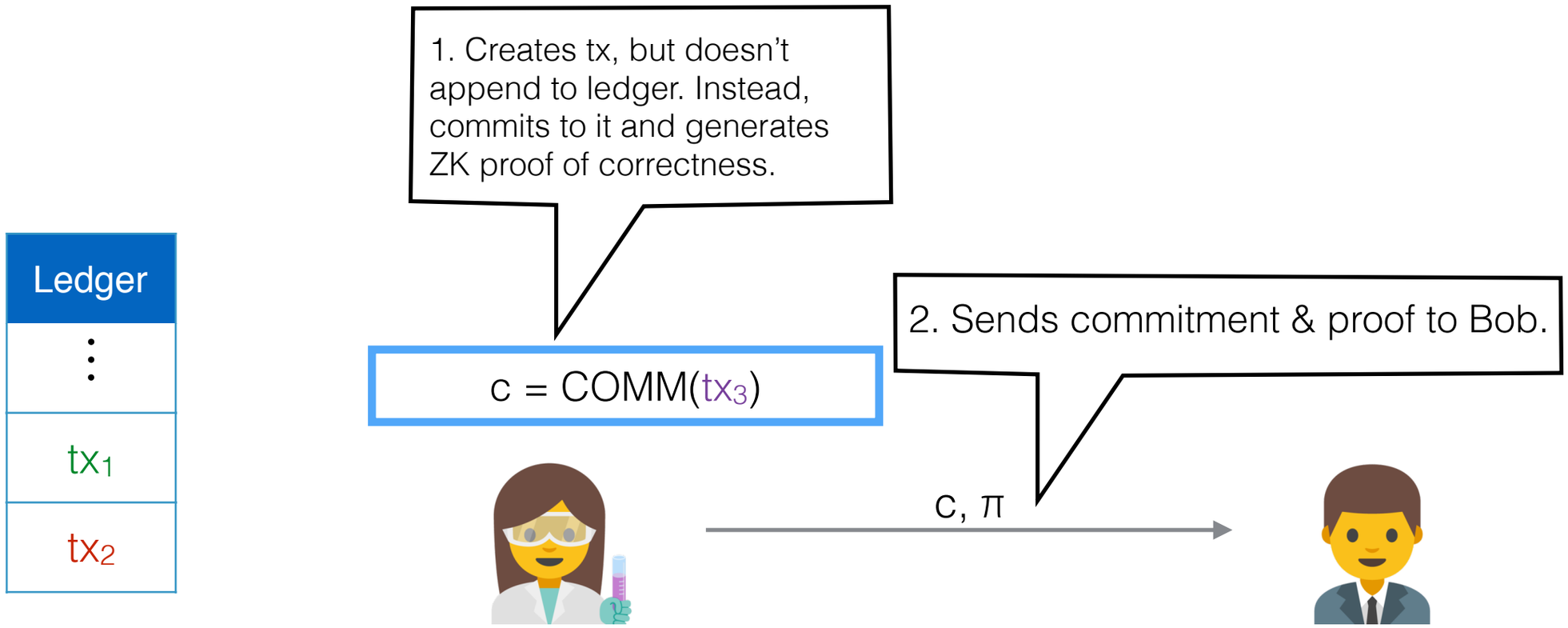
Solution: Make sn translucent



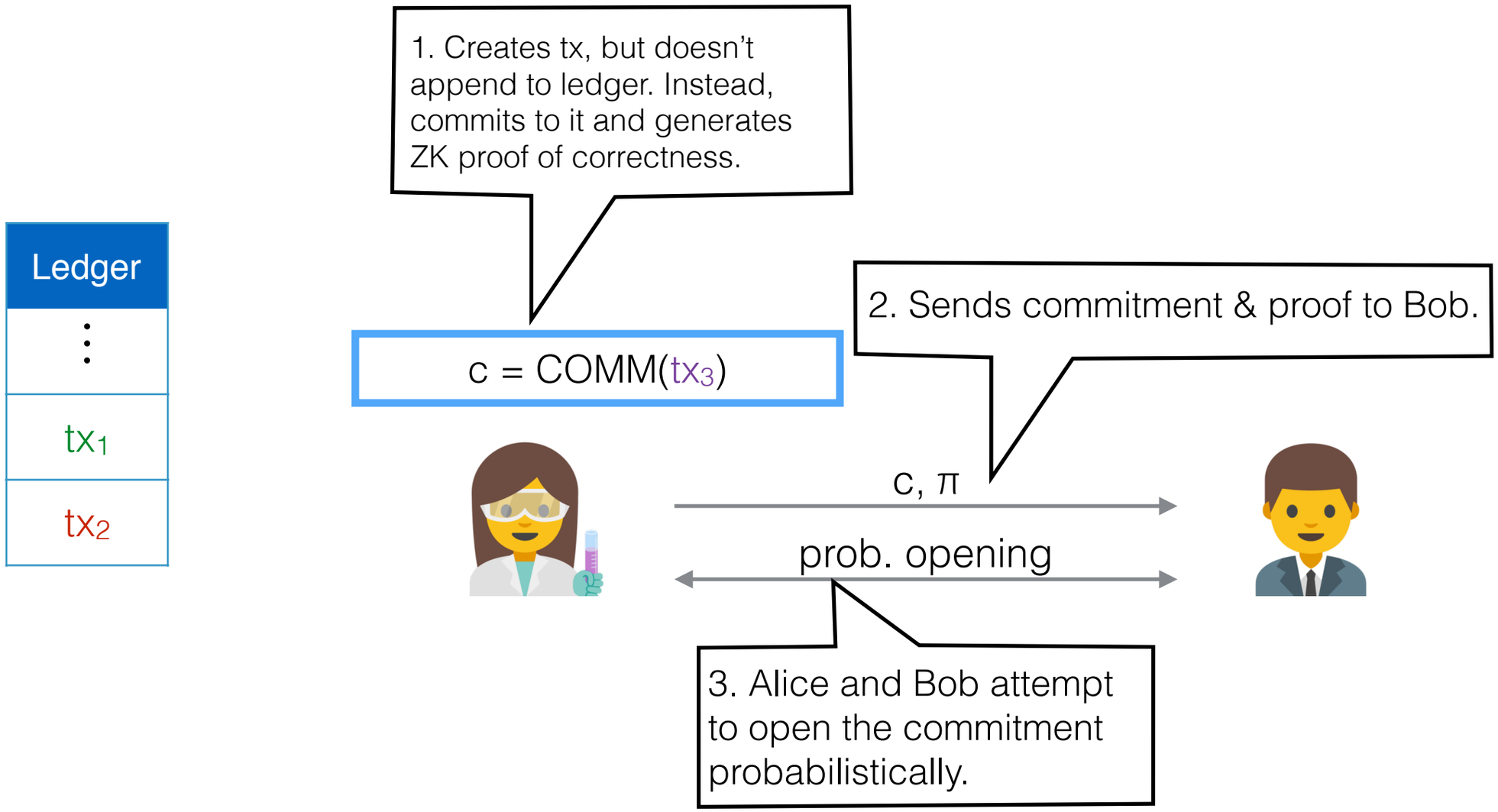
Solution: Make sn translucent



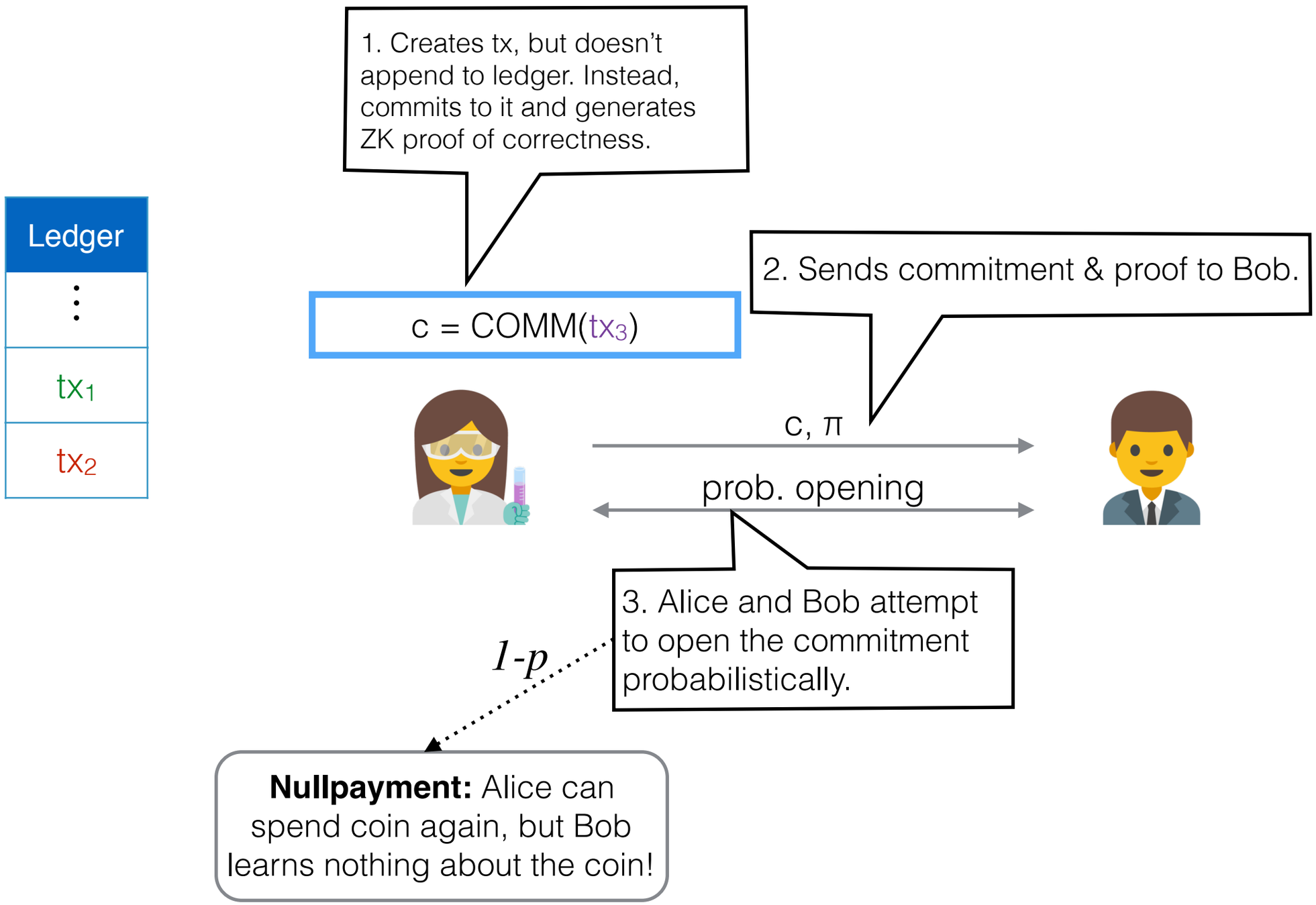
Solution: Make sn translucent



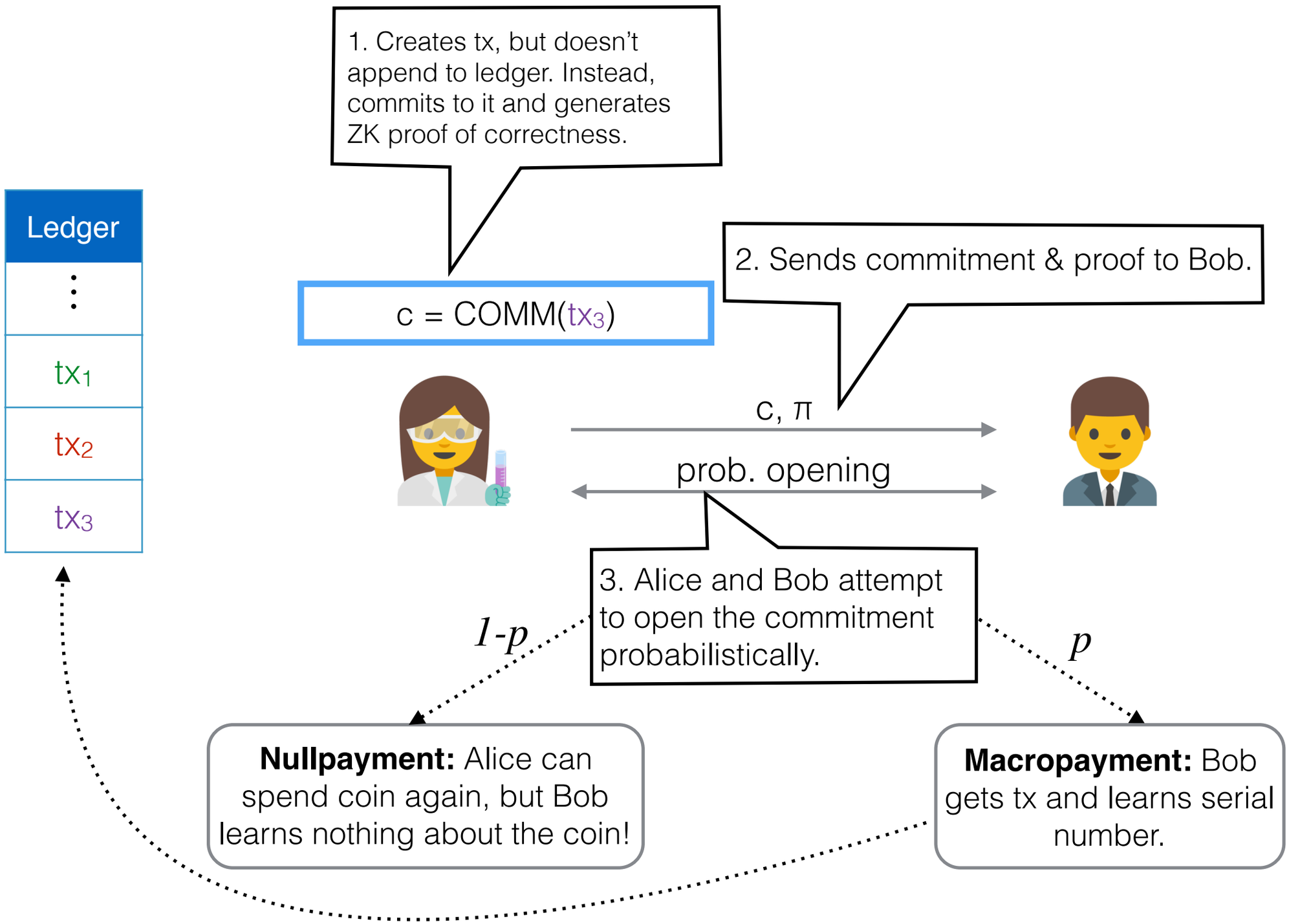
Solution: Make sn translucent



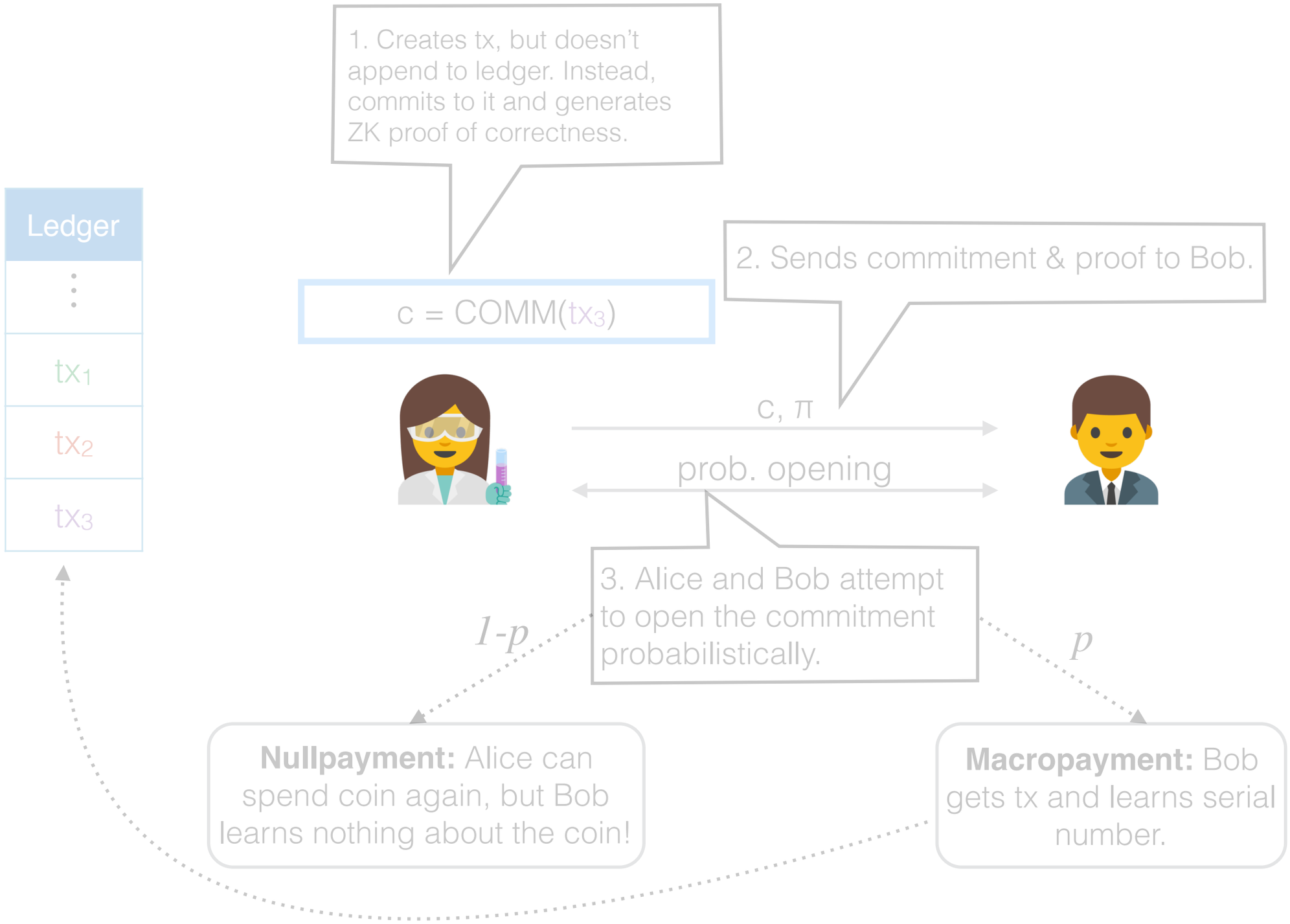
Solution: Make sn translucent



Solution: Make sn translucent



Solution: Make sn translucent



Solution: Make sn translucent

1. Creates tx, but doesn't append to ledger. Instead, commits to it and generates ZK proof of correctness.

Fractional **M**essage **T**ransfer

Fractional hiding: w.p *1-p*, Bob learns nothing about message.
Fractional binding: Bob can always open with probability *p*.

Ledger
tx_1
tx_2
tx_3

c, π

prob. opening

3. Alice and Bob attempt to open the commitment probabilistically.

1-p

p

Nullpayment: Alice can spend coin again, but Bob learns nothing about the coin!

Macropayment: Bob gets tx and learns serial number.

Solution: Make sn translucent

1. Creates tx, but doesn't append to ledger. Instead, commits to it and generates ZK proof of correctness.

Fractional **M**essage **T**ransfer

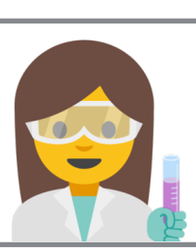
Fractional hiding: w.p *1-p*, Bob learns nothing about message.
Fractional binding: Bob can always open with probability *p*.

Ledger

tx_1

tx_2

tx_3



c, π

prob. opening

Wants fractional hiding

3. Alice and Bob attempt to open the commitment probabilistically.

p

Nullpayment: Alice can spend coin again, but Bob learns nothing about the coin!

Macropayment: Bob gets tx and learns serial number.

Solution: Make sn translucent

1. Creates tx, but doesn't append to ledger. Instead, commits to it and generates ZK proof of correctness.

Fractional **M**essage **T**ransfer

Fractional hiding: w.p *1-p*, Bob learns nothing about message.
Fractional binding: Bob can always open with probability *p*.

Ledger
tx_1
tx_2
tx_3

c, π

prob. opening

Wants fractional hiding

3. Alice and Bob attempt to open the commitment probabilistically.

Wants fractional binding

Nullpayment: Alice can spend coin again, but Bob learns nothing about the coin!

Macropayment: Bob gets tx and learns serial number.

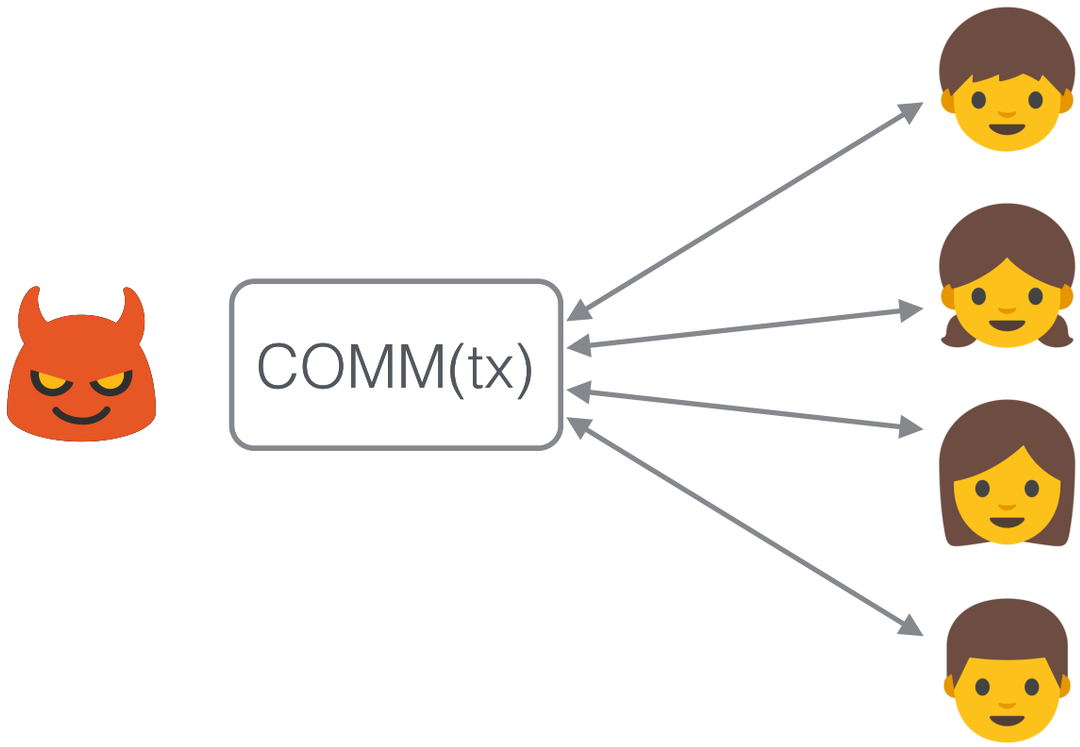
Problem 2: Double-Spending

Problem 2: Double-Spending

Malice can use the same coin in multiple payments **in parallel**.

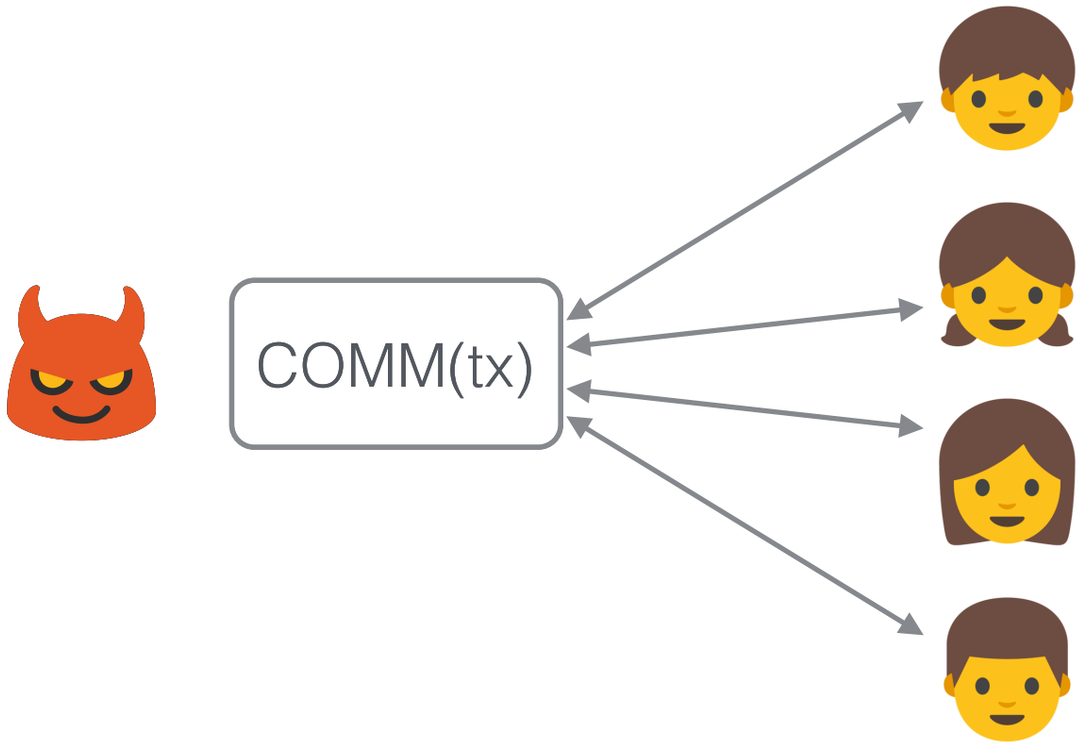
Problem 2: Double-Spending

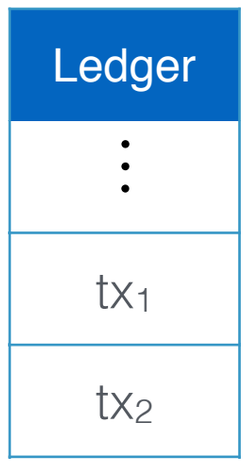
Malice can use the same coin in multiple payments **in parallel**.



Problem 2: Double-Spending

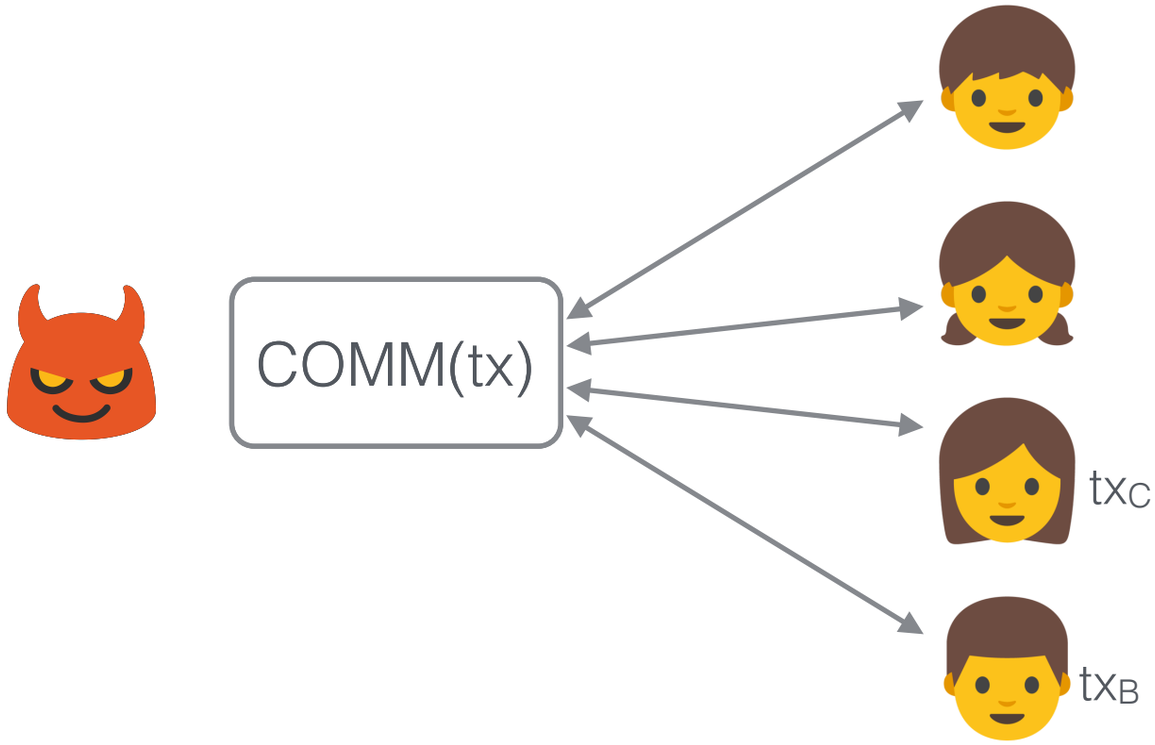
Malice can use the same coin in multiple payments **in parallel**.

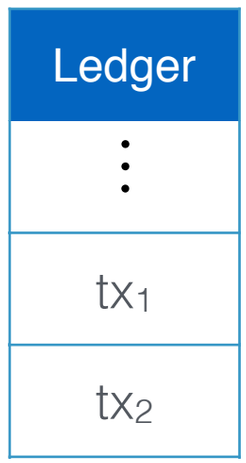




Problem 2: Double-Spending

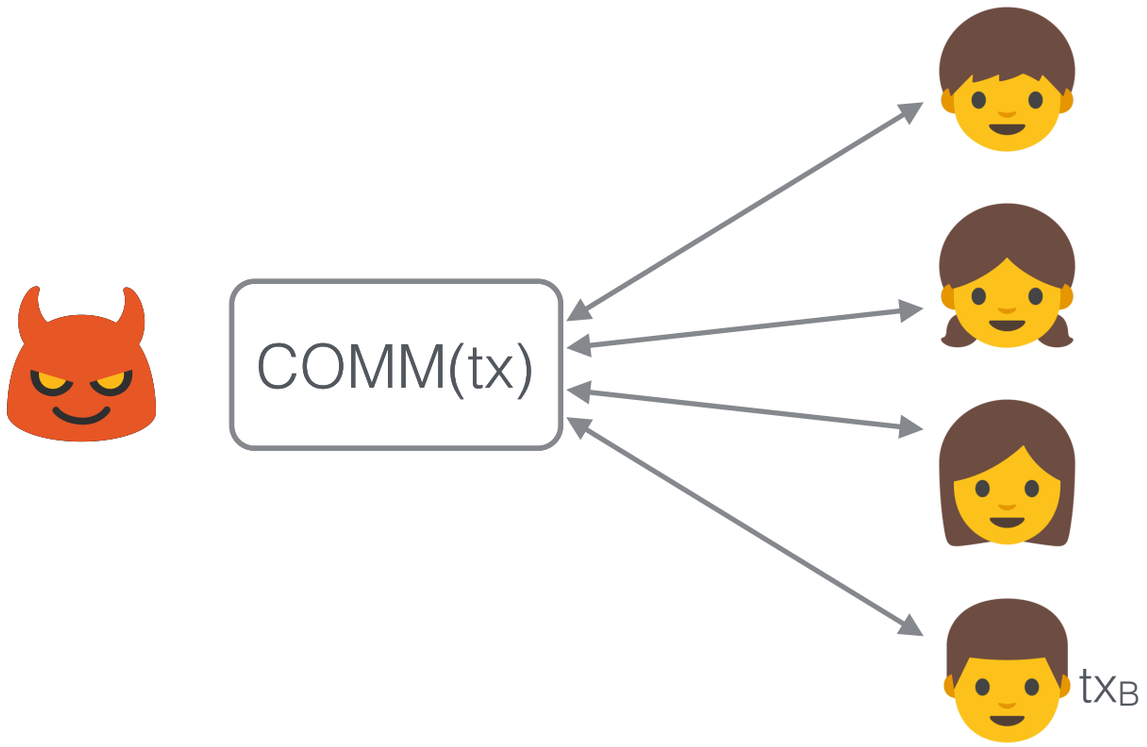
Malice can use the same coin in multiple payments **in parallel**.

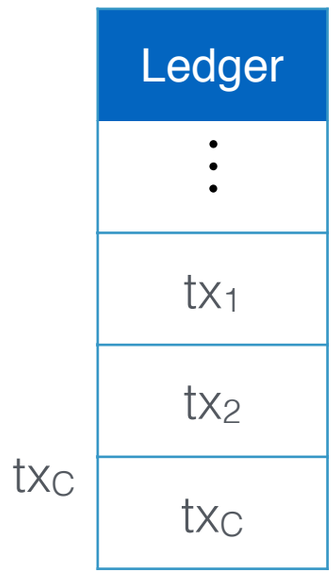




Problem 2: Double-Spending

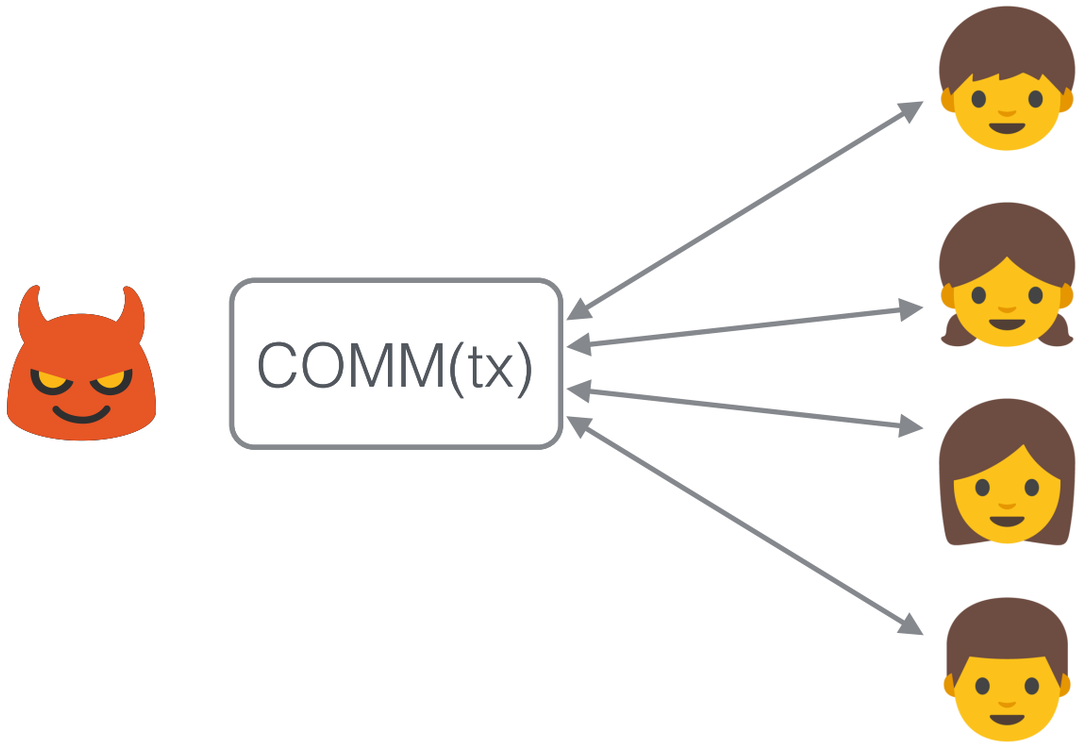
Malice can use the same coin in multiple payments **in parallel**.

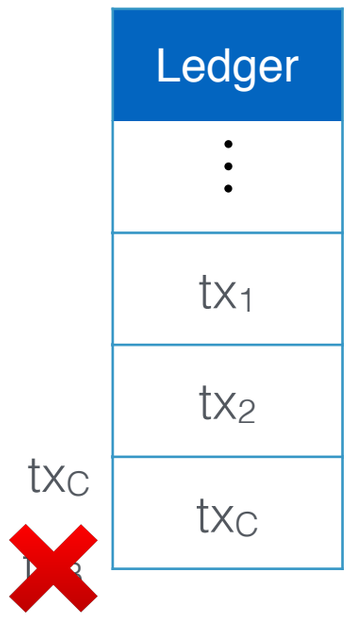




Problem 2: Double-Spending

Malice can use the same coin in multiple payments **in parallel**.

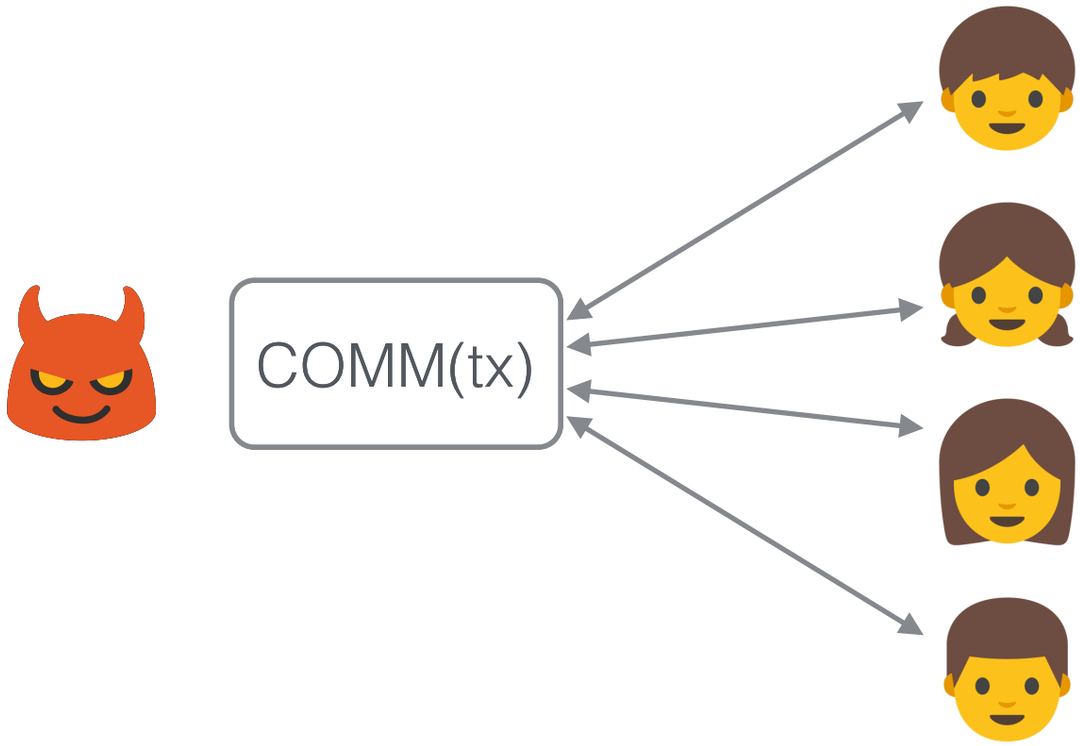


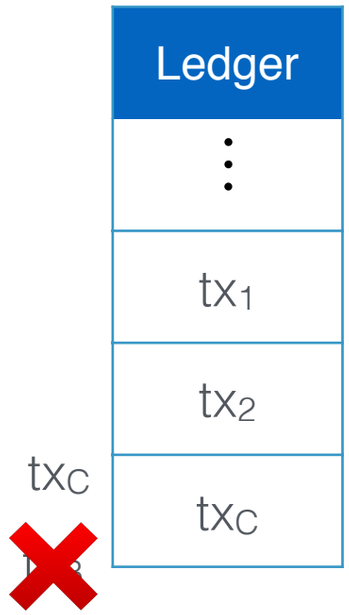


Problem 2: Double-Spending

Malice can use the same coin in multiple payments **in parallel**.

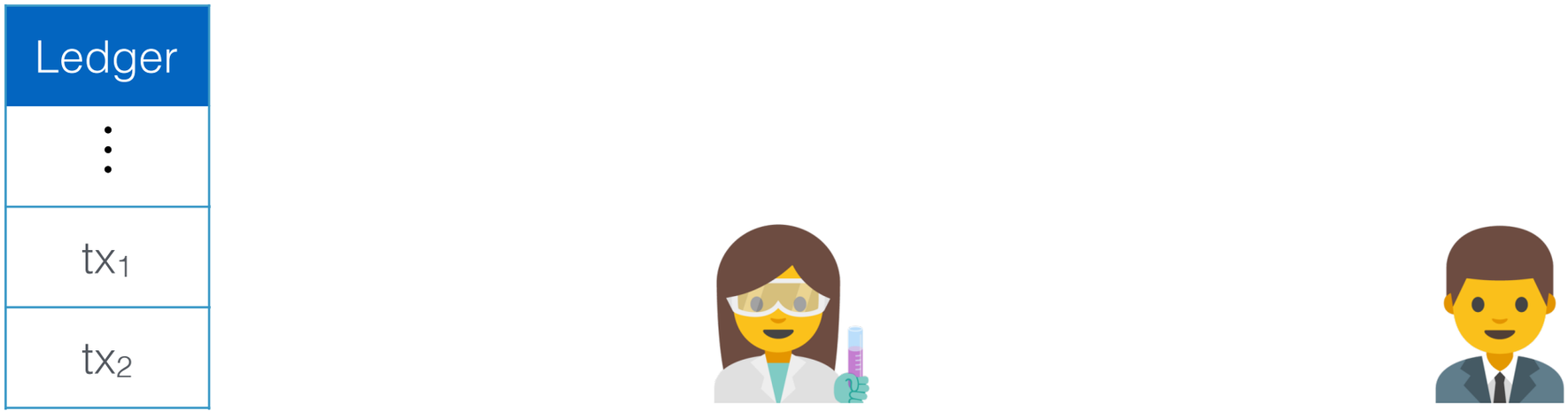
Offline setting ⇒ such attacks **cannot be prevented**.



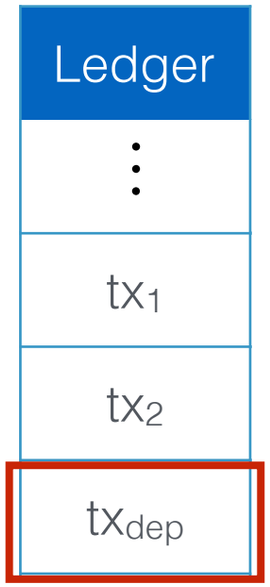


Solution: deposits + rationality

Solution: deposits + rationality

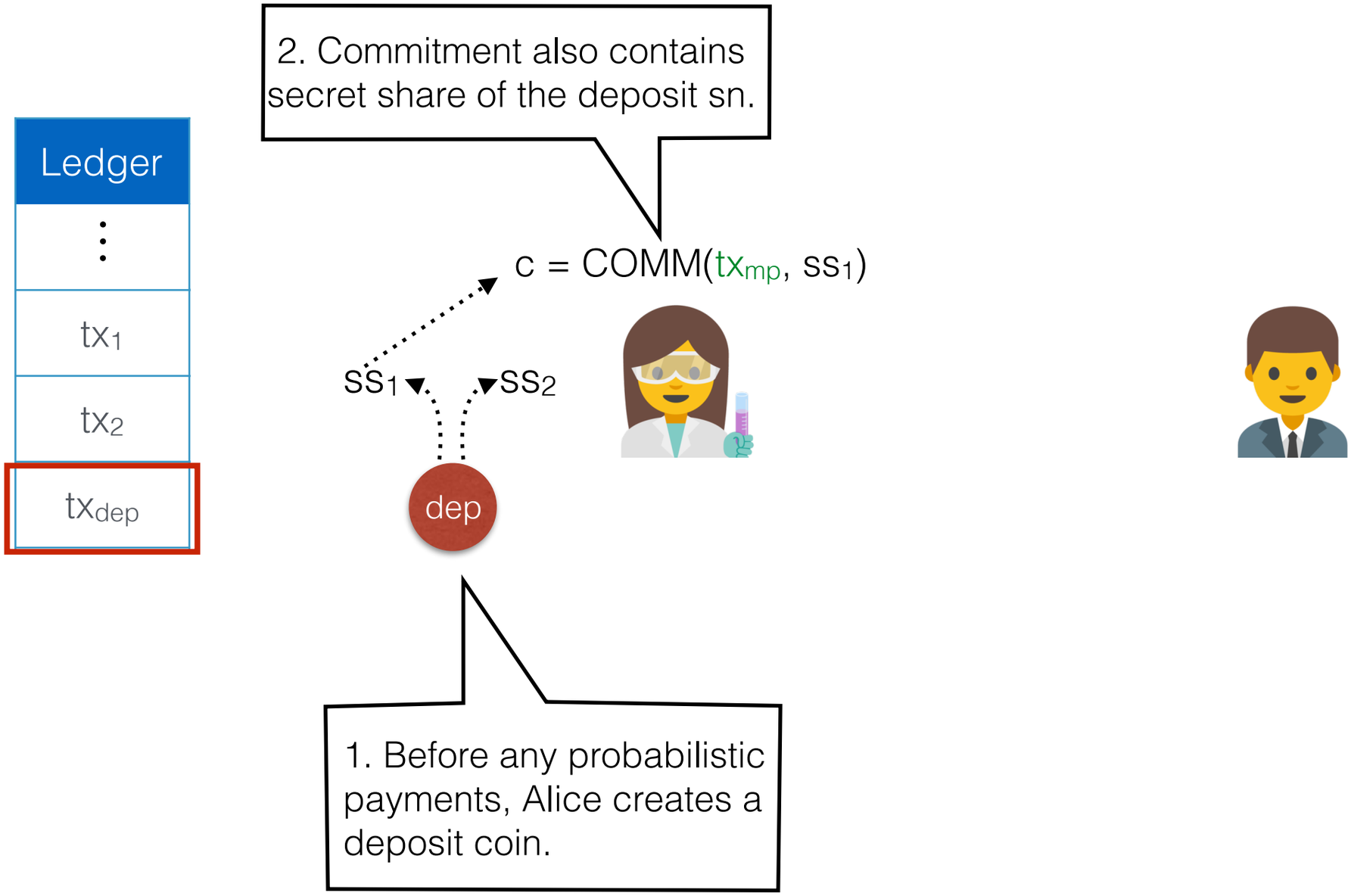


Solution: deposits + rationality

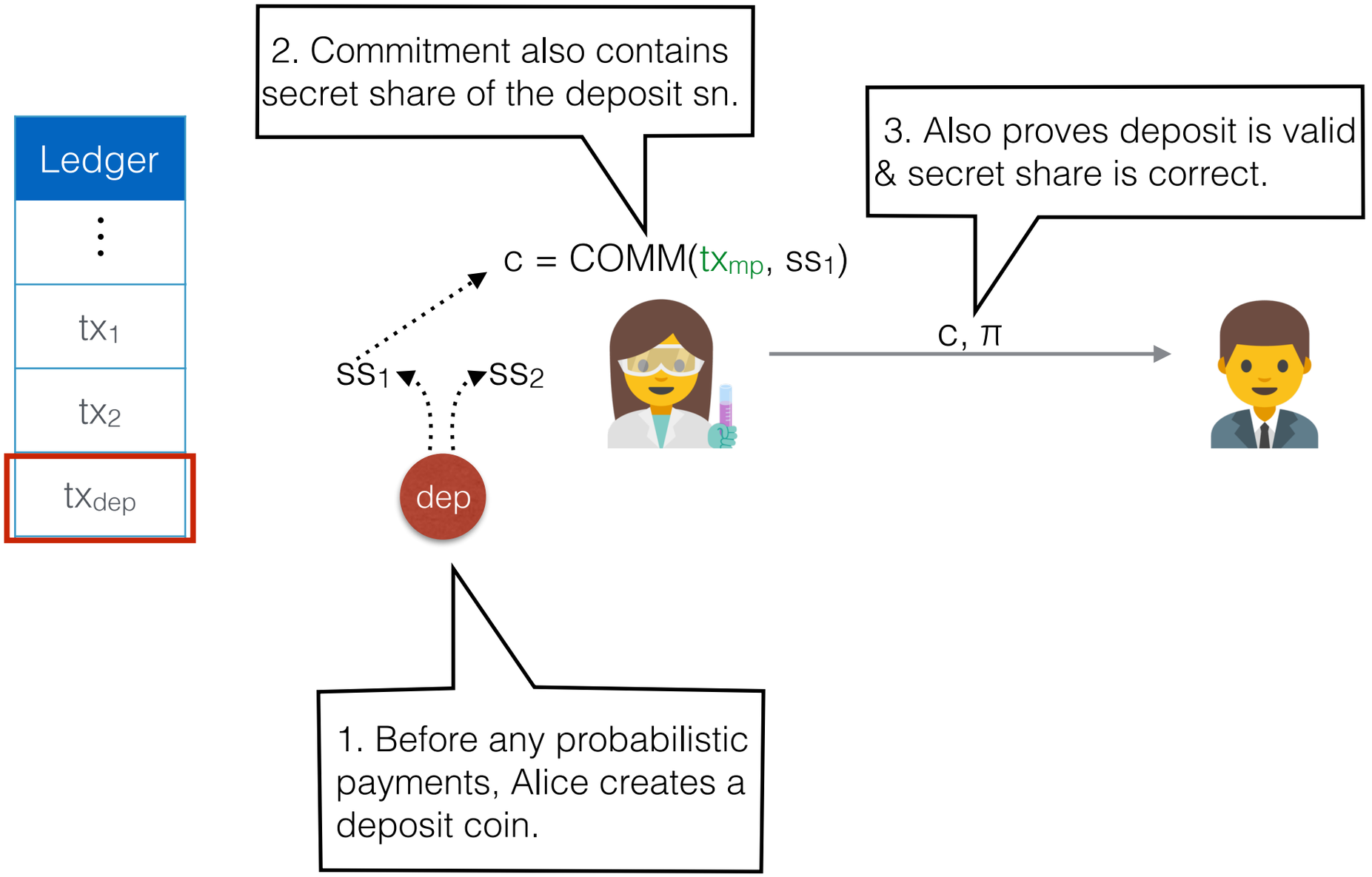


1. Before any probabilistic payments, Alice creates a deposit coin.

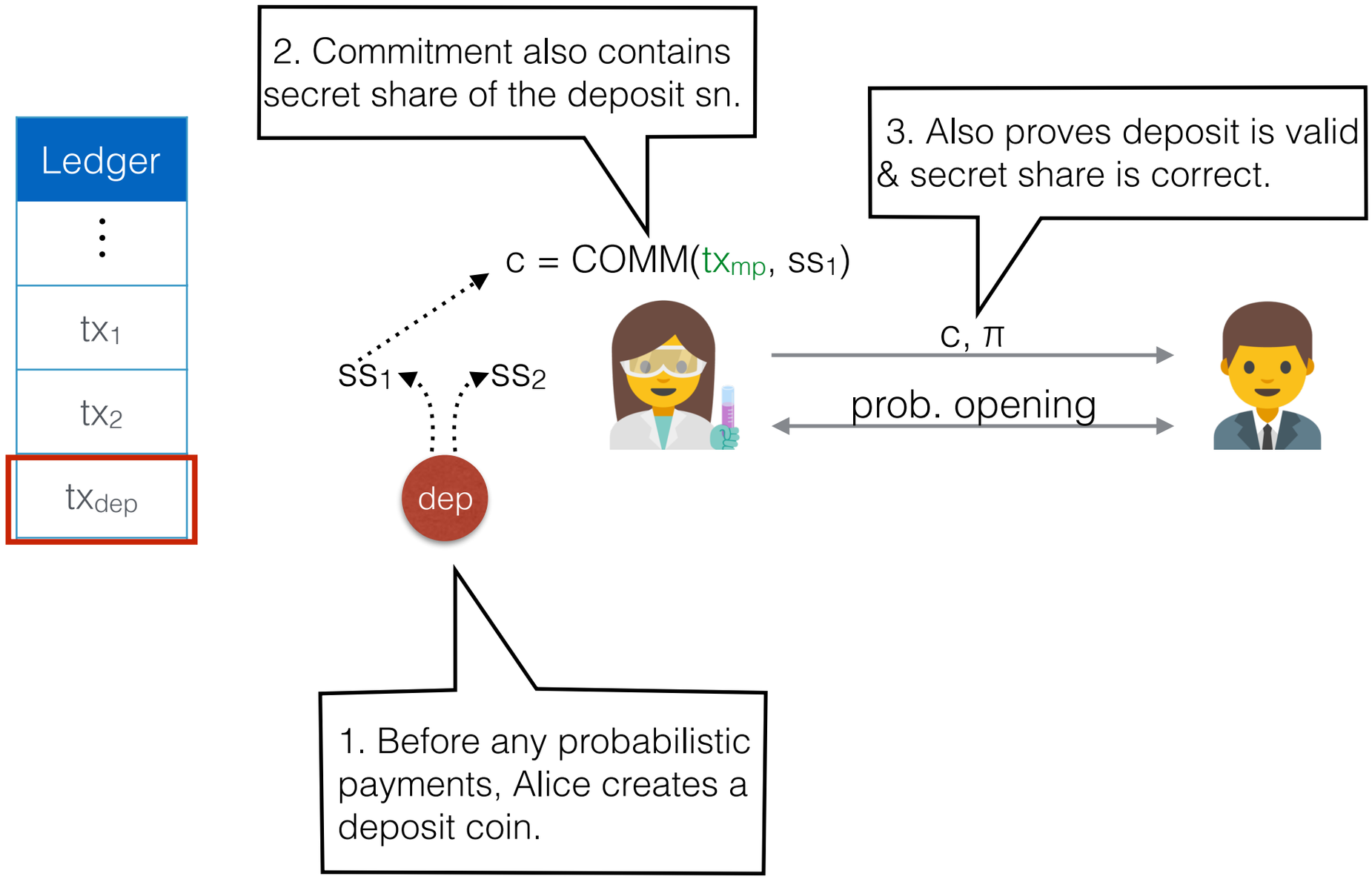
Solution: deposits + rationality



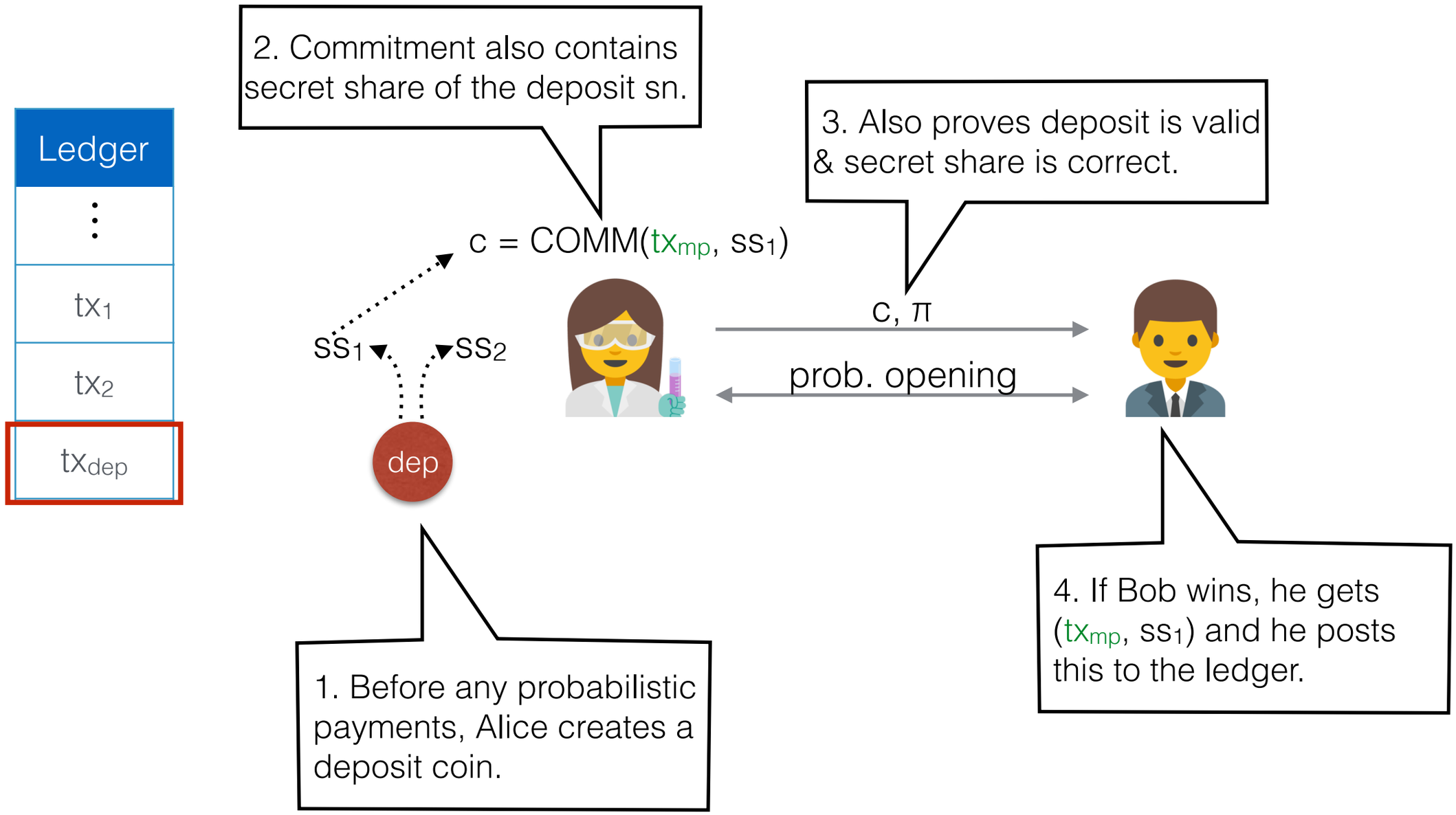
Solution: deposits + rationality



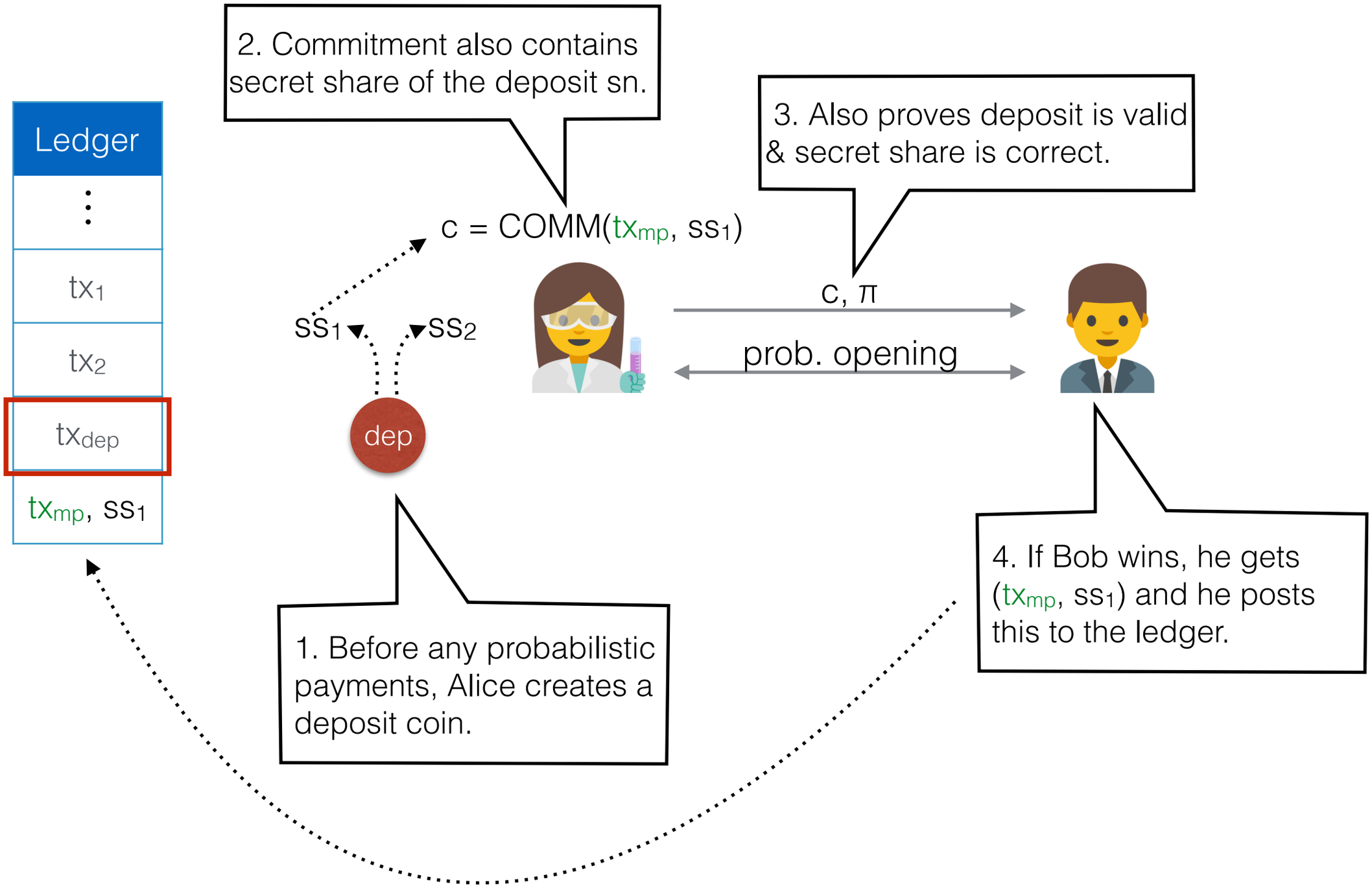
Solution: deposits + rationality



Solution: deposits + rationality

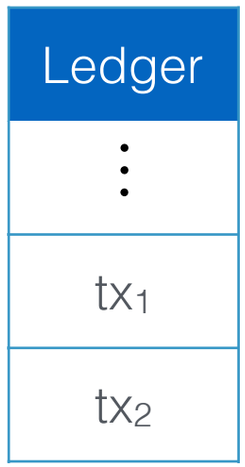


Solution: deposits + rationality

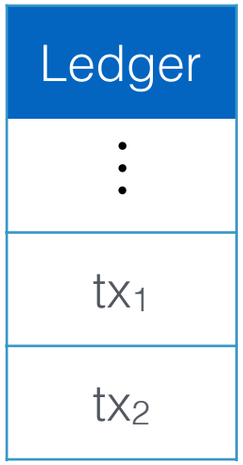


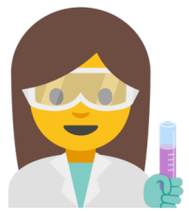
Why does this work?

Why does this work?

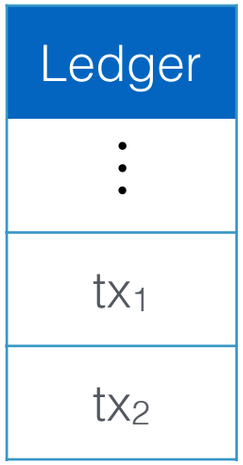


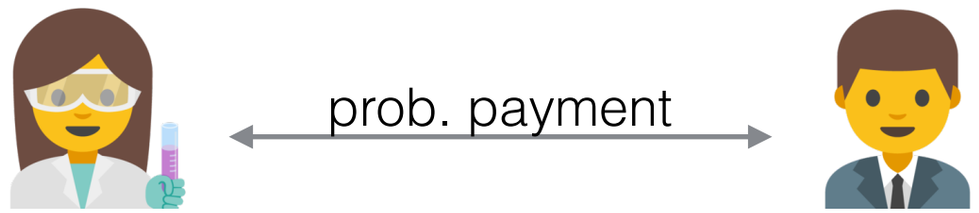
Why does this work?



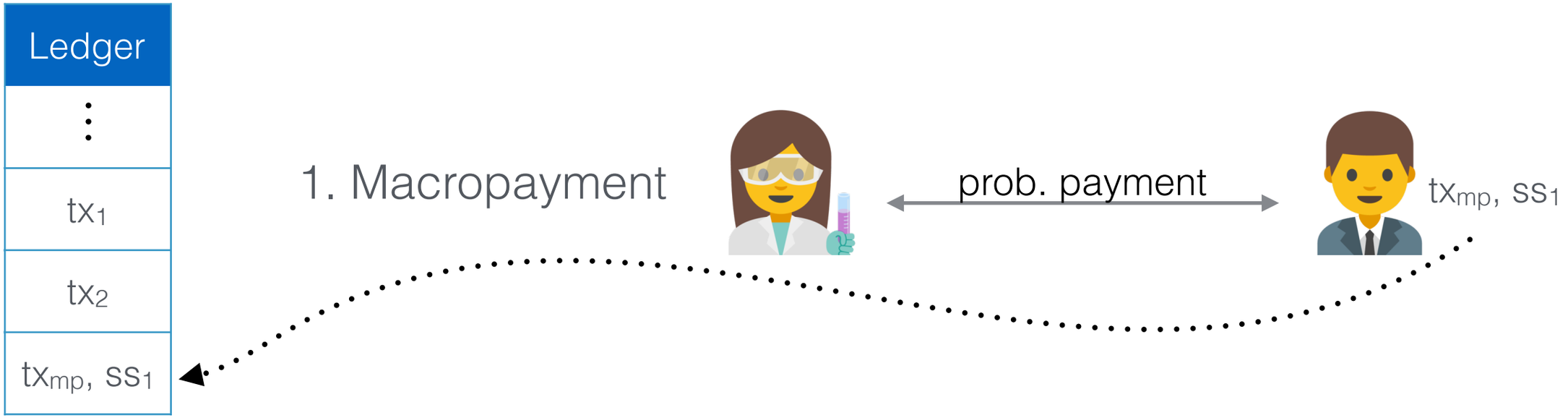


Why does this work?

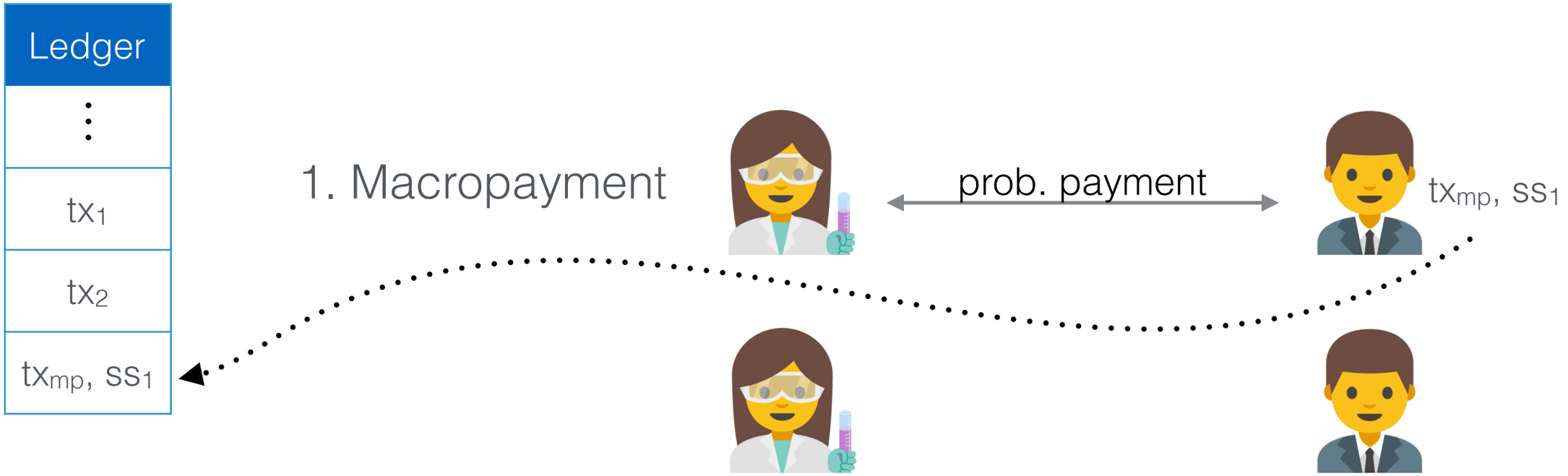




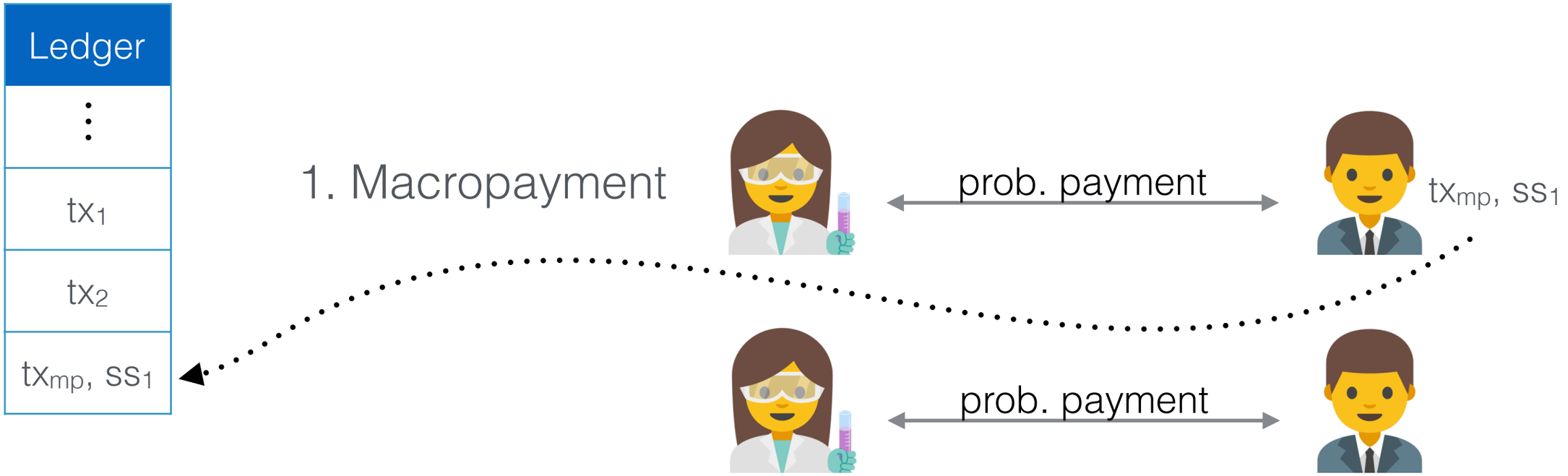
Why does this work?



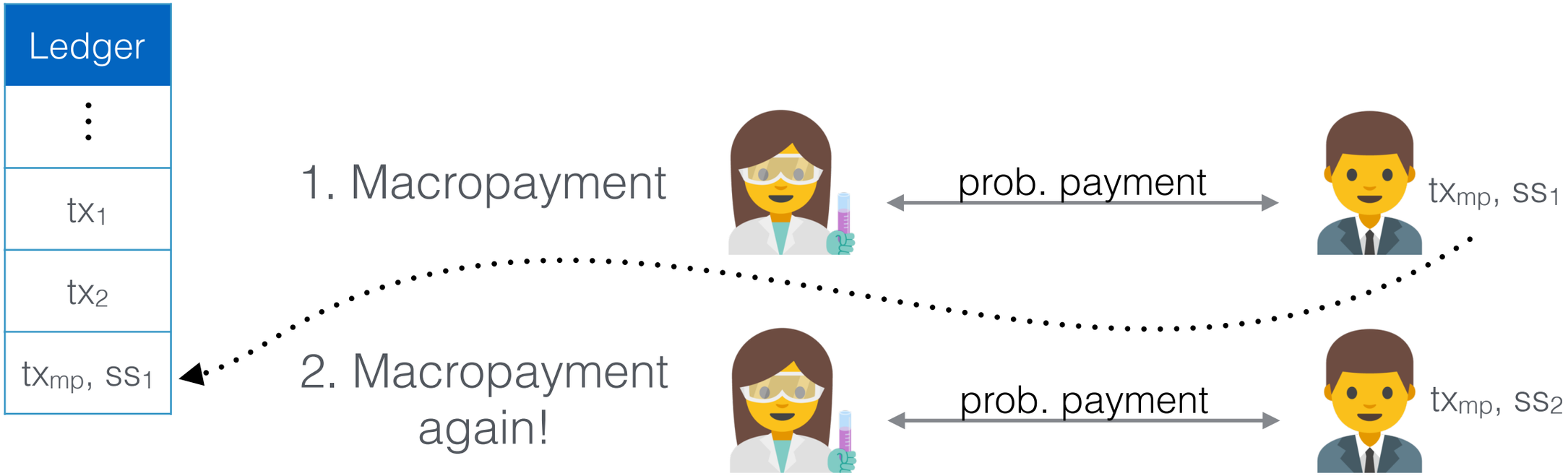
Why does this work?



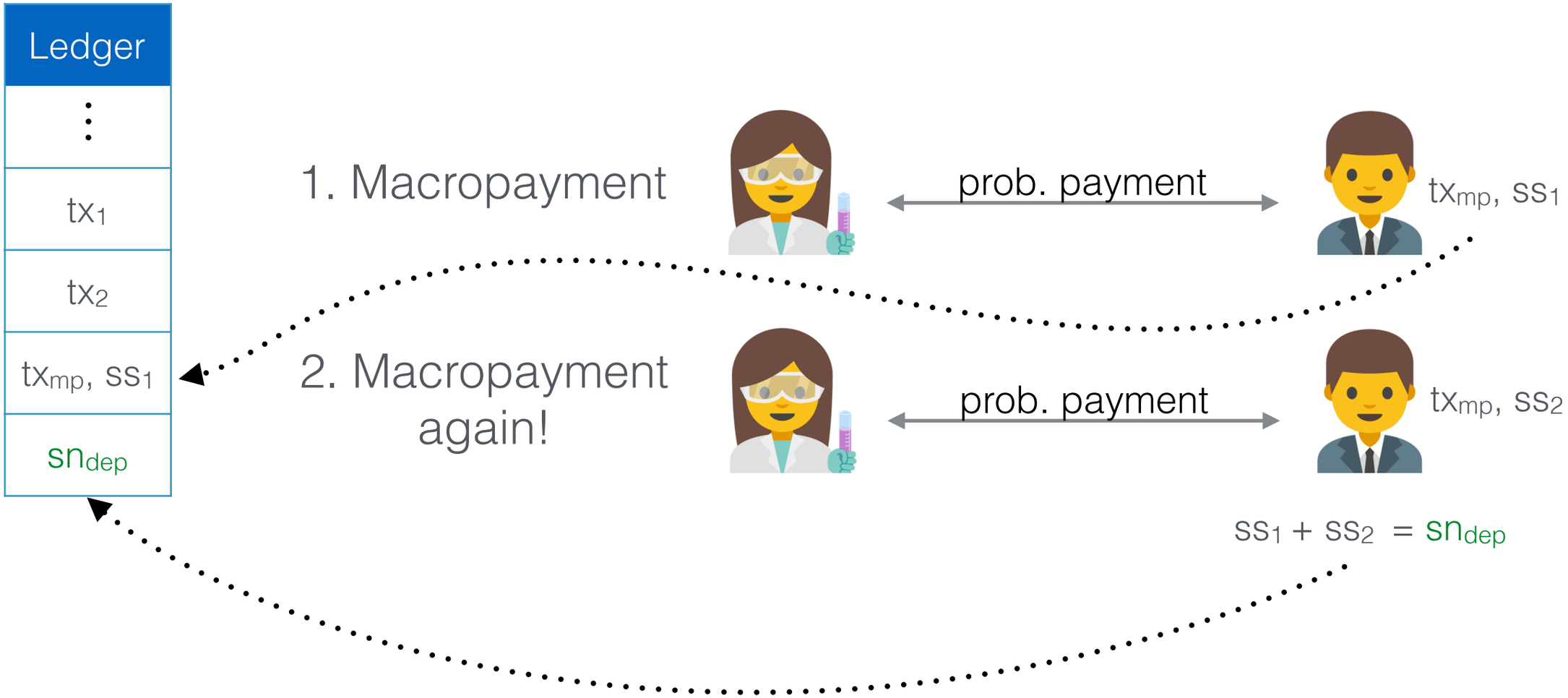
Why does this work?



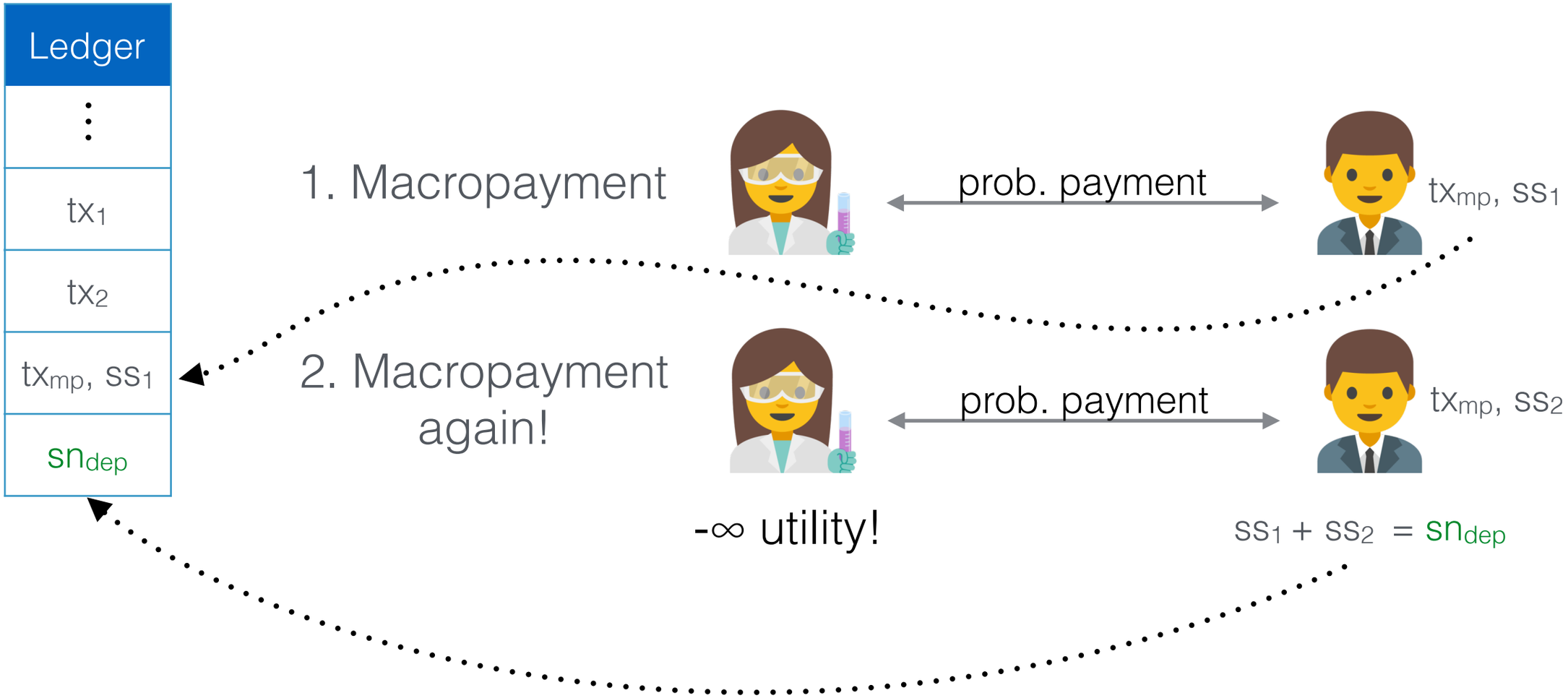
Why does this work?



Why does this work?



Why does this work?



So far

So far

Probabilistic opening:

So far

Probabilistic opening:

Deposits:

So far

Probabilistic opening: prevents linkability.

Deposits:

So far

Probabilistic opening: prevents linkability.

Deposits: prevent double-spending.

So far

Probabilistic opening: prevents linkability.

Deposits: prevent double-spending.

Are we done?

So far

Probabilistic opening: prevents linkability.

Deposits: prevent double-spending.

Are we done?

Functionality:

So far

Probabilistic opening: prevents linkability.

Deposits: prevent double-spending.

Are we done?

Functionality:

Feature: Customers should be able to withdraw deposits.

So far

Probabilistic opening: prevents linkability.

Deposits: prevent double-spending.

Are we done?

Functionality:

Feature: Customers should be able to withdraw deposits.

Problem: Customer can withdraw before revocation.

So far

Probabilistic opening: prevents linkability.

Deposits: prevent double-spending.

Are we done?

Functionality:

Feature: Customers should be able to withdraw deposits.

Problem: Customer can withdraw before revocation.

Problem: What if merchant refuses to reply?

So far

Probabilistic opening: prevents linkability.

Deposits: prevent double-spending.

Are we done?

Functionality:

Feature: Customers should be able to withdraw deposits.
Problem: Customer can withdraw before revocation.
Problem: What if merchant refuses to reply?

Economic analysis: How to set deposit value?

So far

Probabilistic opening: prevents linkability.

Deposits: prevent double-spending.

Are we done?

Functionality:

Feature: Customers should be able to withdraw deposits.
Problem: Customer can withdraw before revocation.
Problem: What if merchant refuses to reply?

Economic analysis: How to set deposit value?

See paper for solutions!

Takeaways

Takeaways

Used translucent crypto + game theory to construct

Takeaways

Used translucent crypto + game theory to construct

Decentralized
Anonymous
Micropayments

Takeaways

Used translucent crypto + game theory to construct

Decentralized
Anonymous
Micropayments

Game-theoretic analysis more broadly applicable:

- Eg: Pass-Shelat do not specify value of deposit.
- Eg: Probabilistic smart contracts.

Takeaways

Used translucent crypto + game theory to construct

Decentralized
Anonymous
Micropayments

Game-theoretic analysis more broadly applicable:

- Eg: Pass-Shelat do not specify value of deposit.
- Eg: Probabilistic smart contracts.

We also discovered pain points in Zerocash interface.

- Resulted in a more “programmable” interface.

Takeaways

Used translucent crypto + game theory to construct

Decentralized
Anonymous
Micropayments

Game-theoretic analysis more broadly applicable:

Eg: Pass-Shelat do not specify value of deposit.
Eg: Probabilistic smart contracts.

We also discovered pain points in Zerocash interface.

Resulted in a more “programmable” interface.

Thanks!

http://eprint.iacr.org/2016/1033