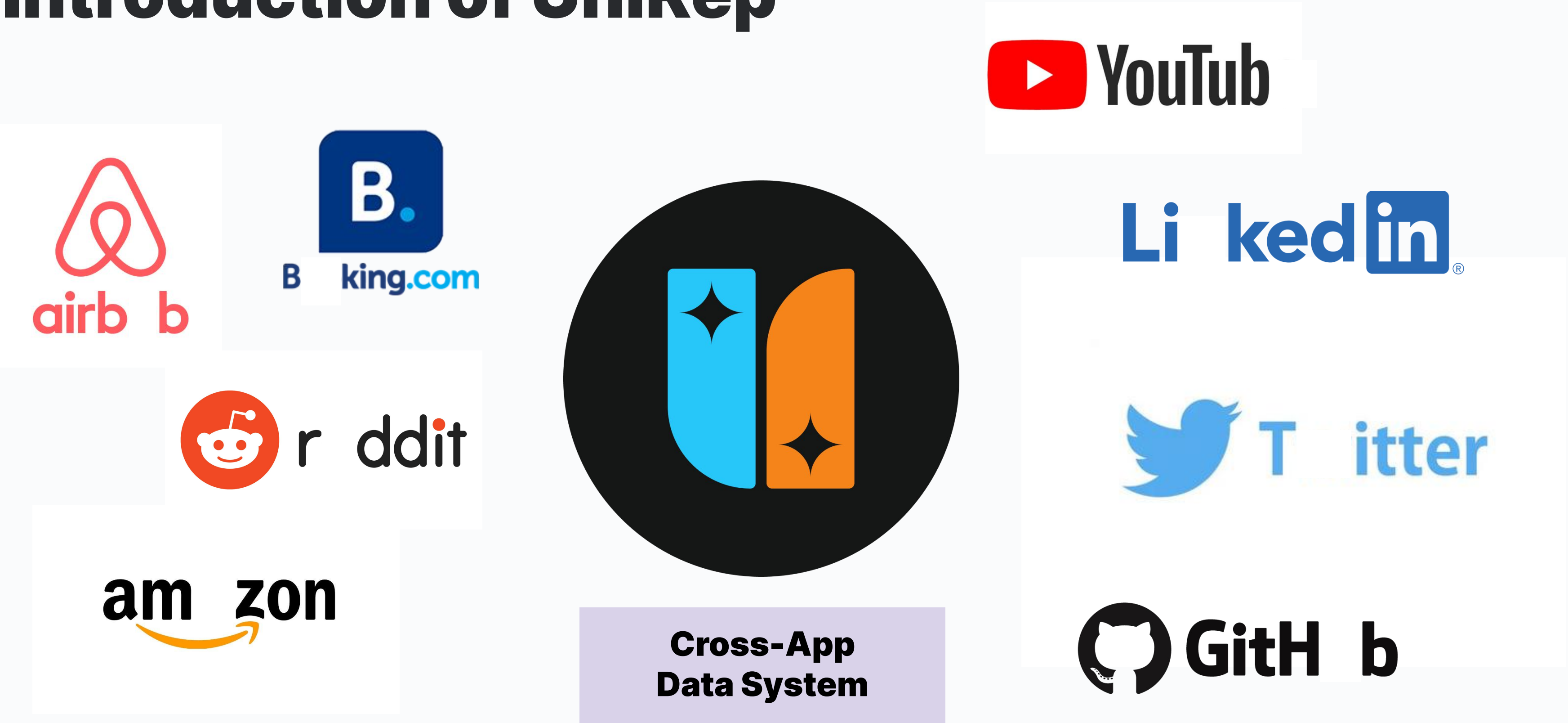
Hackmd notes:
https://hackmd.io/025gduq8Rfmg57Bo1KGTkQ?both

Privacy and Scaling Explorations

UniRep V2 - anonymous data system

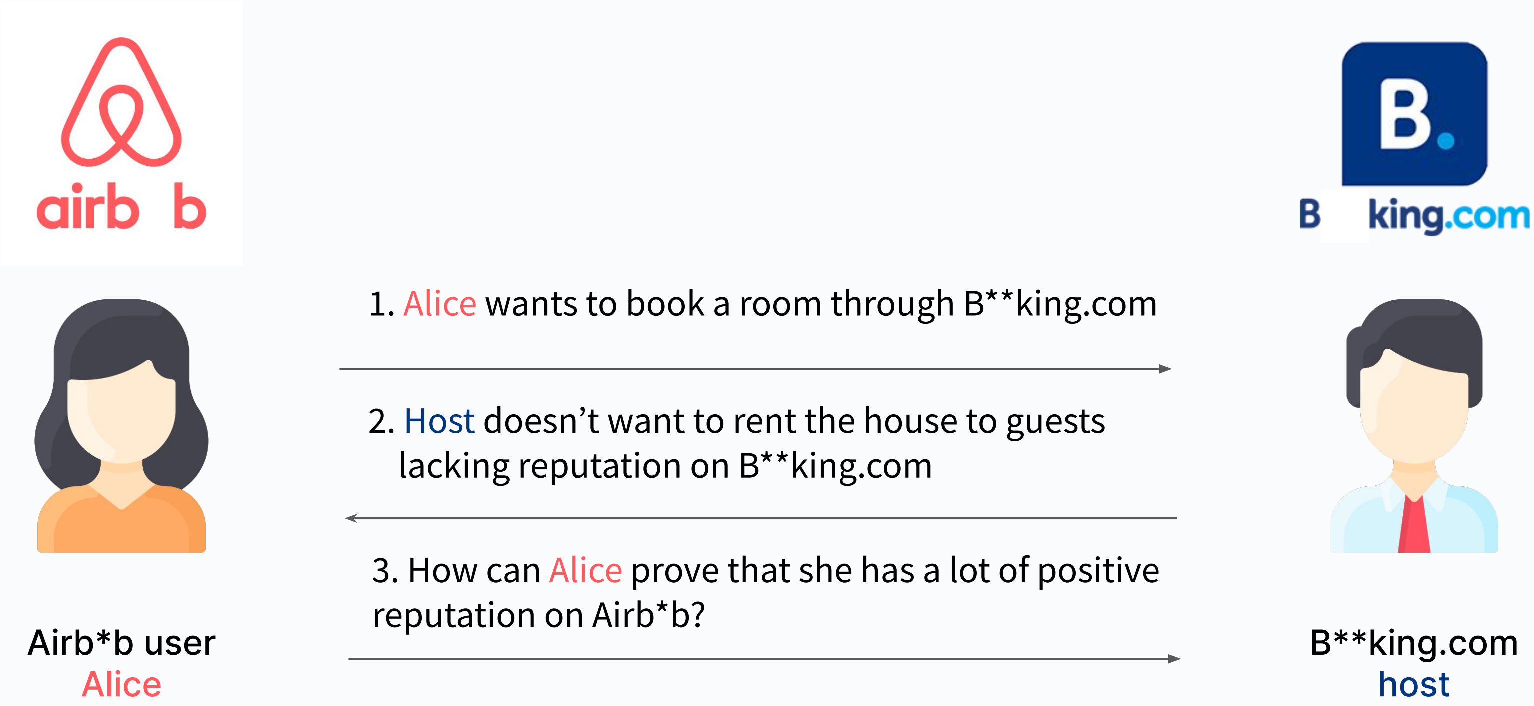
Vivian

Introduction of UniRep



Cross-App
Data System

Introduction of UniRep



1. Alice wants to book a room through B**king.com

2. Host doesn’t want to rent the house to guests lacking reputation on B**king.com

3. How can Alice prove that she has a lot of positive reputation on Airb*b?

Airb*b user
Alice

B**king.com
host

Introduction of UniRep

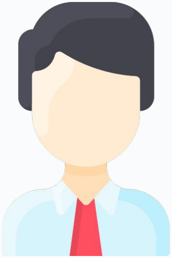
Airb*b user
Alice

How can Alice prove that she has a lot of positive reputation on Airbnb?

e.g. Alice takes a screenshot

- It compromises Alice’s privacy
- Screenshot can easily be forged
- Host cannot be sure that Alice did not forge the screenshot

Unirep can solve the problem



B**king.com
host

UniRep

- Users can **receive** attestations
- Voluntarily **prove** how much data they have
- Users cannot refuse to receive the attestations

Version 1

- Unireversal Reputation
- Data: positive reputation, negative reputation, graffiti

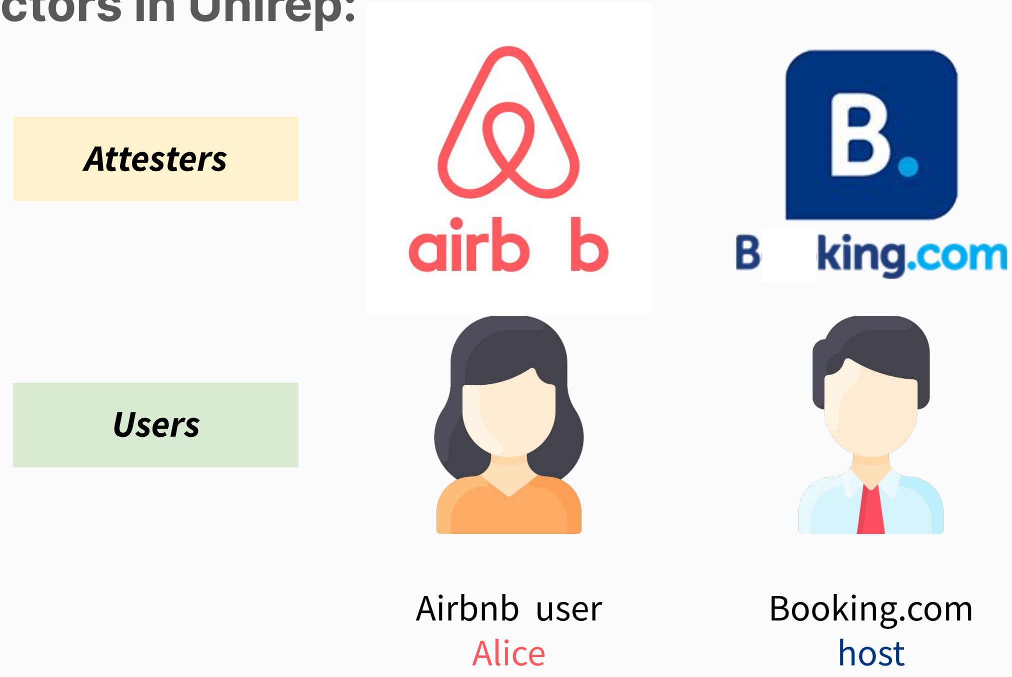
Version 2

- Anonymous data system
- Data: sum data fields, replacement data fields

https://developer.unirep.io/docs/welcome

Protocol

- **Actors in Unirep:**



Attesters

Users

Airbnb user
Alice

Booking.com
host

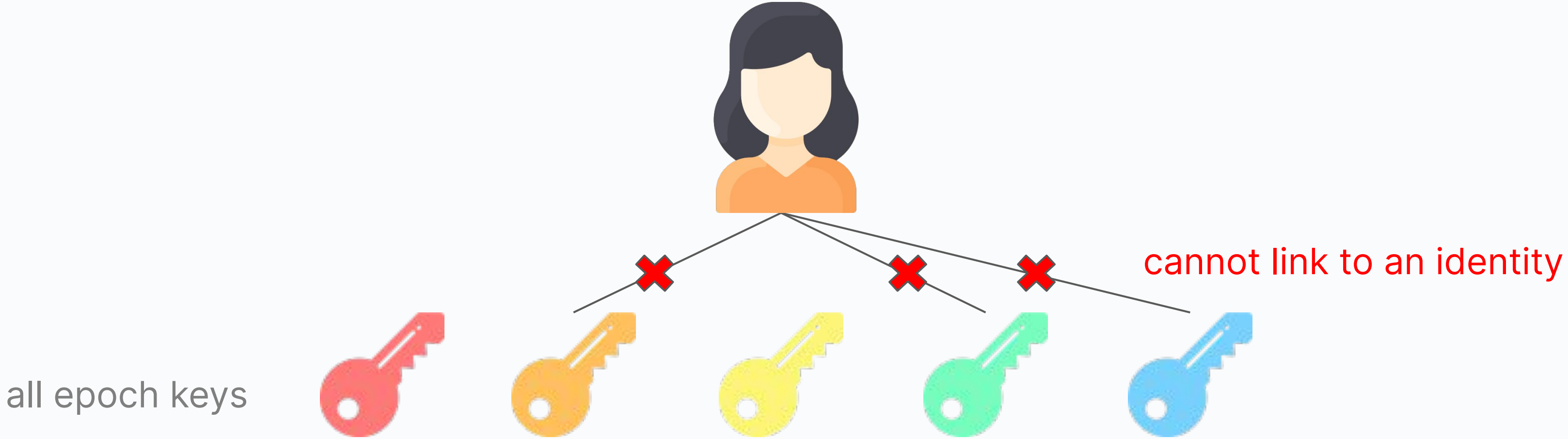
- non-anonymous
- Give attestations

- anonymous
- Receive data
- Prove data
- (Use data)

https://developer.unirep.io/docs/protocol/users-and-attesters

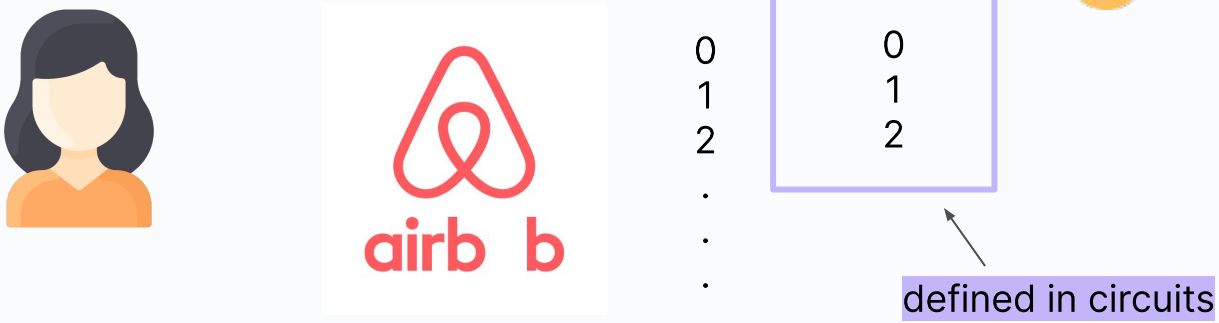
Protocol

- **Anonymity:**
  - User uses a **temporary identity** to receive attestations, called an ***epoch key***.
  - User can generate *k* epoch keys within an ***epoch*** (e.g. 7 days).
  - User can receive all data given to these *k* epoch keys.



Protocol

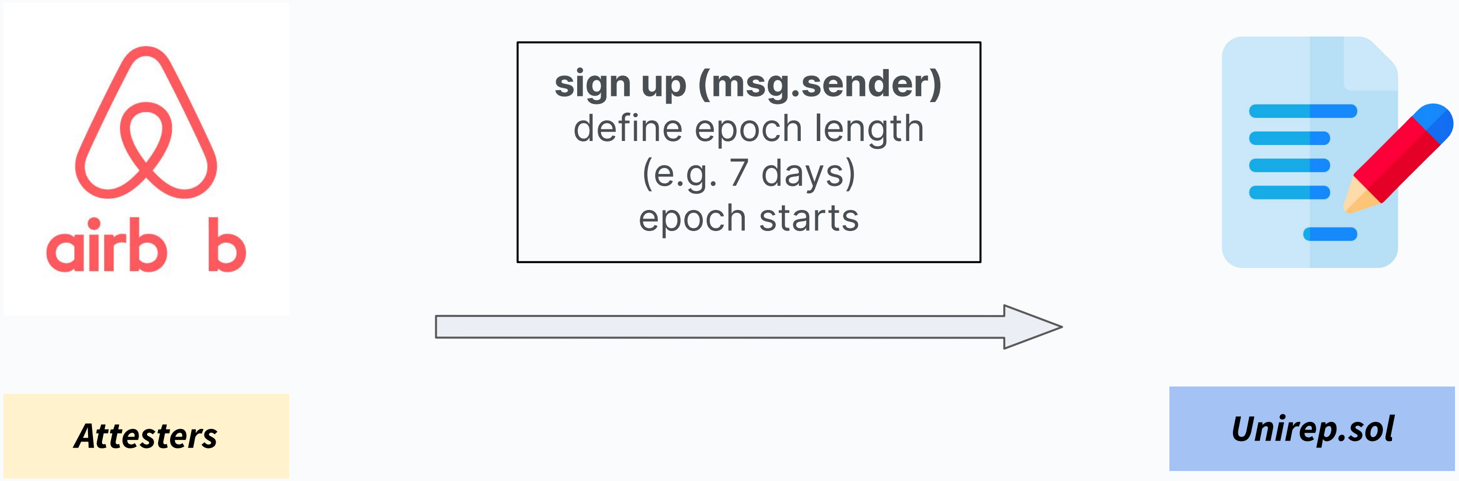
- **Epoch key:**
  - `hash([identitySecret, attesterId, epoch, nonce])`



https://developer.unirep.io/docs/protocol/epoch-key

Protocol

- **Attester sign up:**



sign up (msg.sender)
define epoch length
(e.g. 7 days)
epoch starts

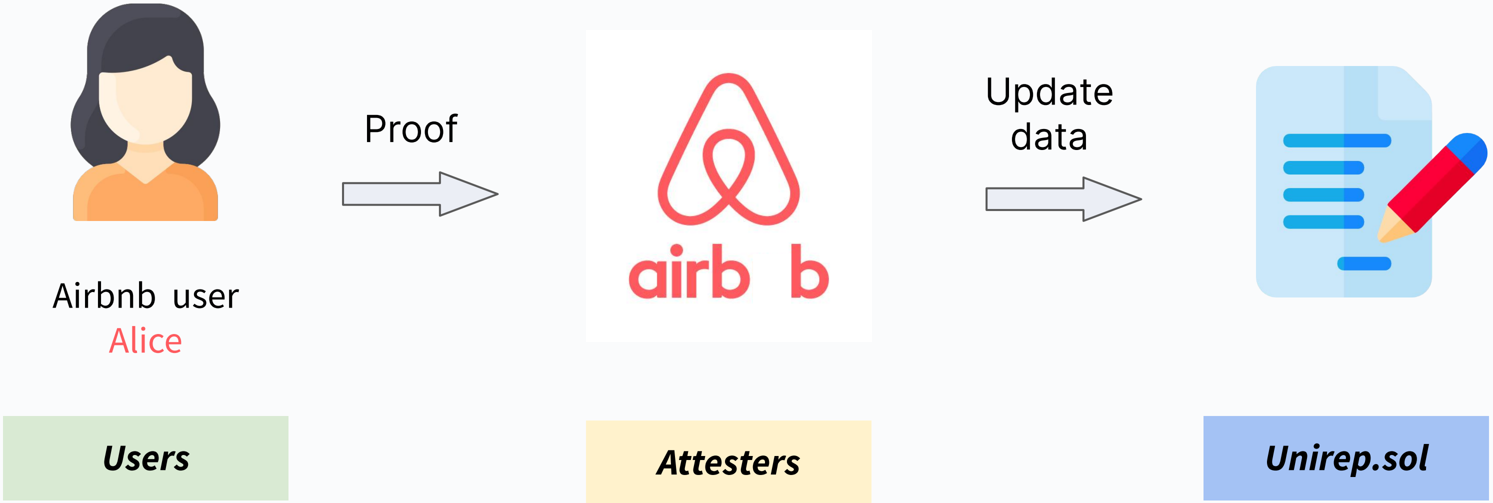
Attesters

Unirep.sol

1. EOA
2. smart contract

https://developer.unirep.io/docs/contracts-api/unirep-sol#attestersignup

Protocol



Airbnb user
Alice

Proof

Update
data

Users

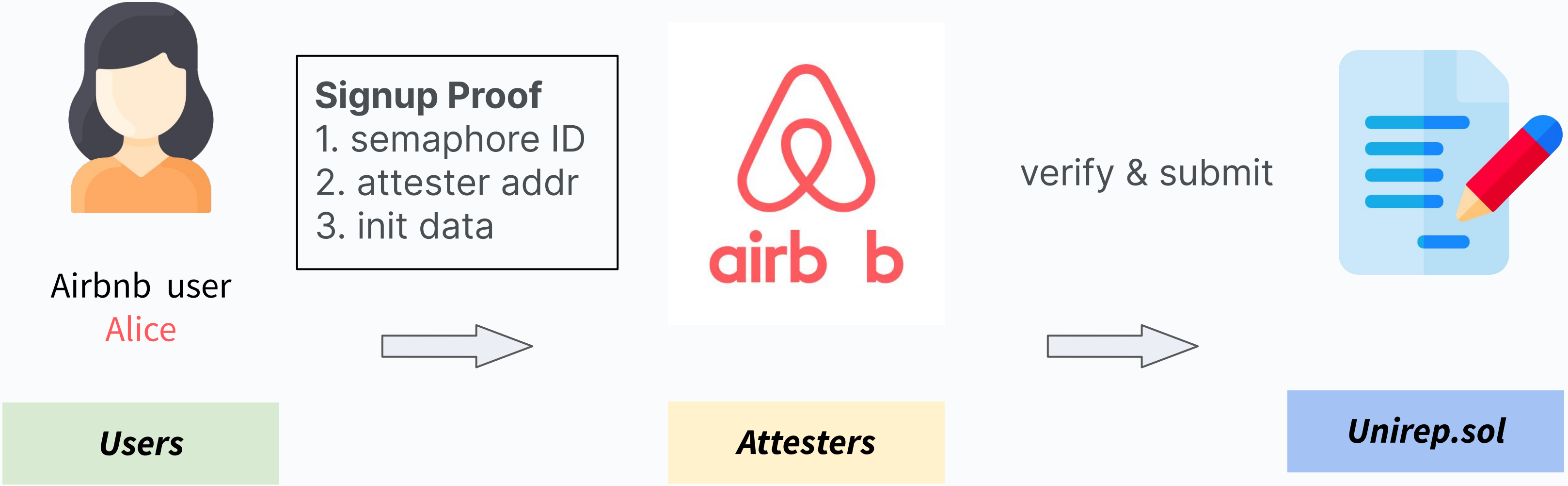
Attesters

Unirep.sol

1. EOA
2. smart contract

Protocol

- **User sign up:**



Signup Proof
1. semaphore ID
2. attester addr
3. init data

Airbnb user
Alice

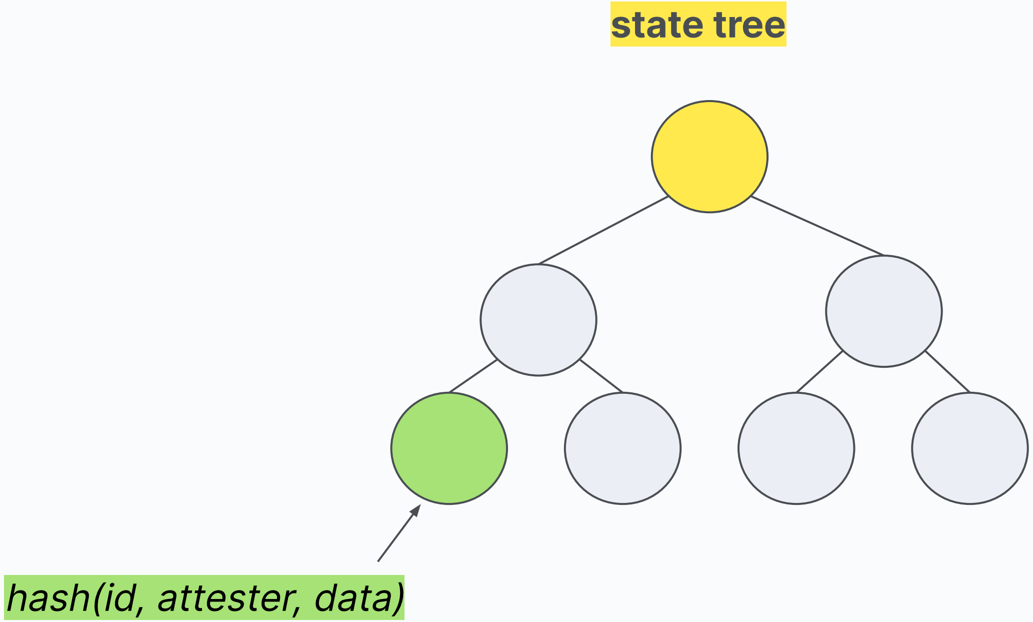
verify & submit

https://developer.unirep.io/docs/contracts-api/unirep-sol#usersignup

PSE

Protocol

- **state tree:**



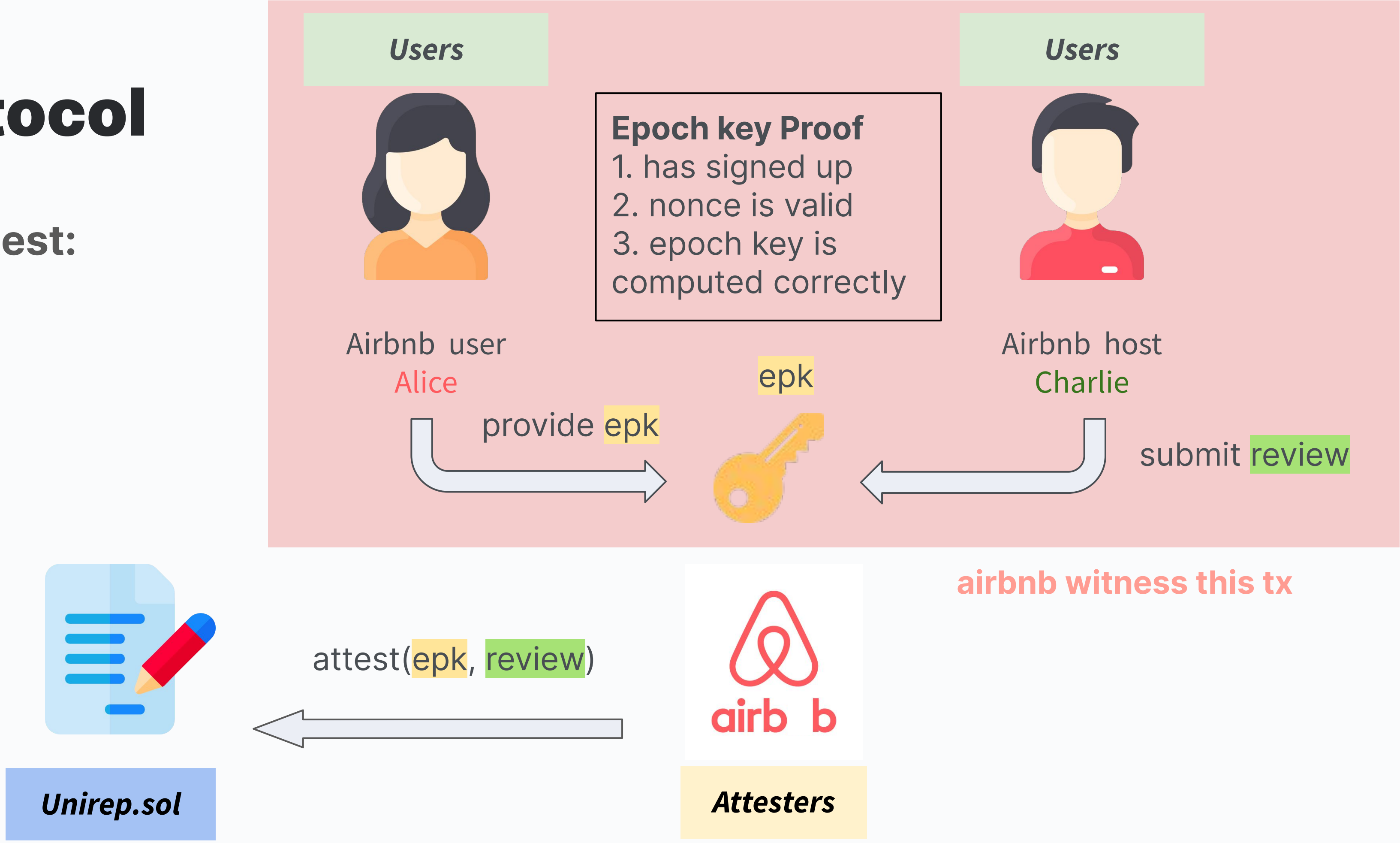
state tree

hash(id, attester, data)

https://developer.unirep.io/docs/protocol/trees#state-tree

Protocol

- **Attest:**



Users

Users

Epoch key Proof
1. has signed up
2. nonce is valid
3. epoch key is computed correctly

Airbnb user
Alice

Airbnb host
Charlie

epk

provide epk

submit review

airbnb witness this tx

attest(epk, review)

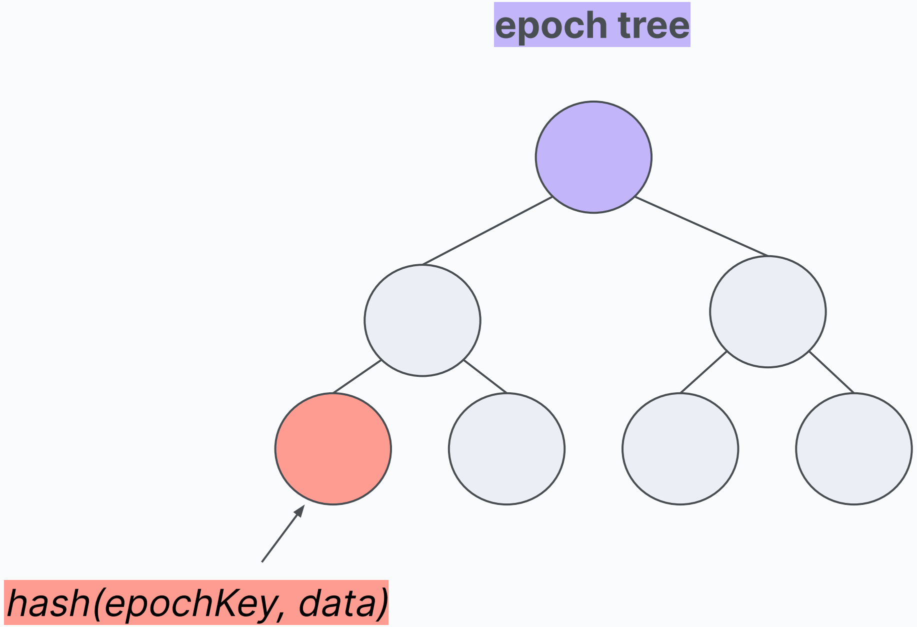
Unirep.sol

Attesters

https://developer.unirep.io/docs/contracts-api/unirep-sol#attest

Protocol

- **epoch tree:**



epoch tree

hash(epochKey, data)

https://developer.unirep.io/docs/protocol/trees#epoch-tree

Protocol

Airbnb user
Alice

		sum data fields		replacement data fields
nonce		communication	cleanliness	reviews
0	epoch key 0×123	3	5	good
1	epoch key 0×456	4	3	bad
2	epoch key 0×789	5	2	okay

https://developer.unirep.io/docs/protocol/data

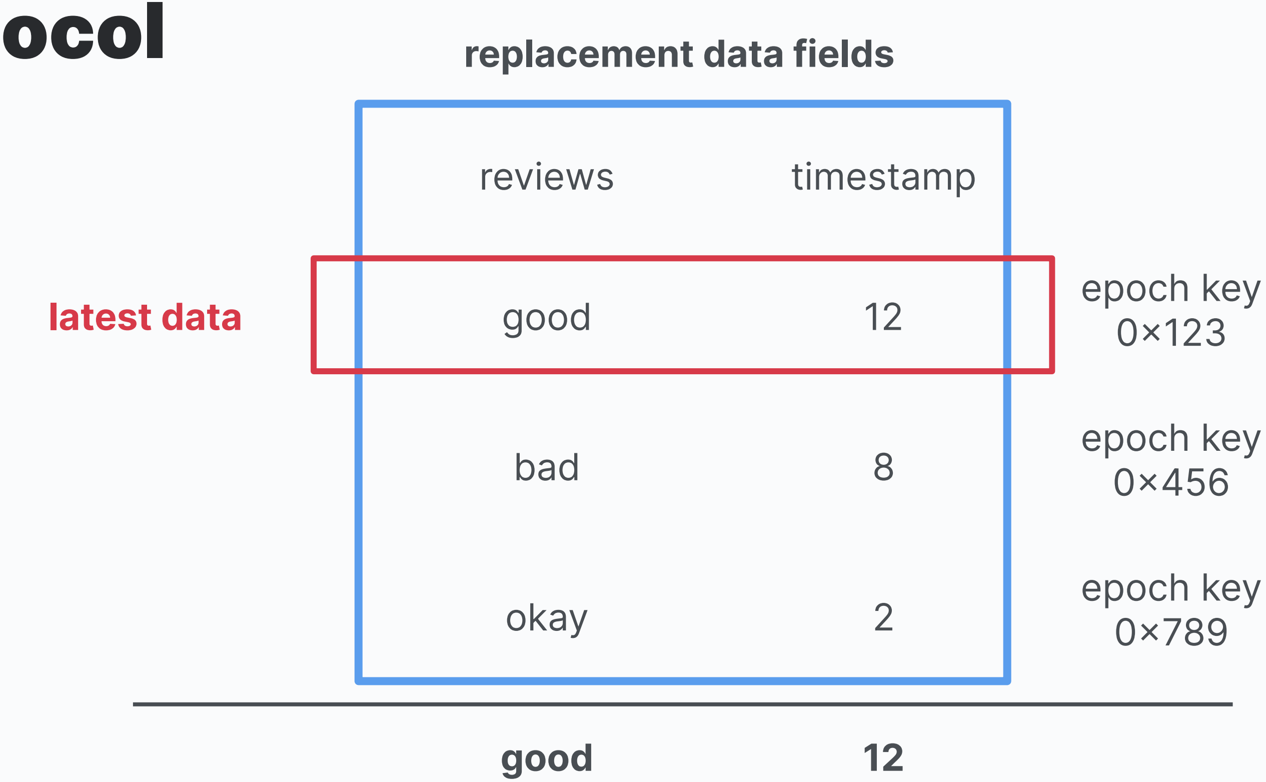
Protocol

sum data fields

	communication	cleanliness	count	
	3	5	1	epoch key 0×123
	4	3	1	epoch key 0×456
+)	5	2	1	epoch key 0×789
	12	**10**	**3**	

https://developer.unirep.io/docs/protocol/data

Protocol



replacement data fields

	reviews	timestamp	
latest data	good	12	epoch key 0×123
	bad	8	epoch key 0×456
	okay	2	epoch key 0×789
	good	**12**	

https://developer.unirep.io/docs/protocol/data

Protocol

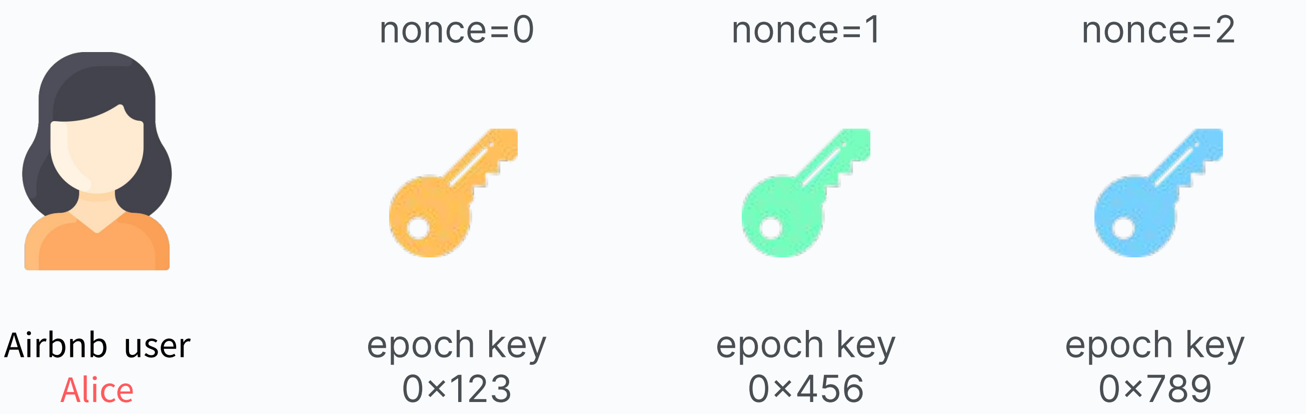
- **Receive data:**

(
    communication,
    cleanliness,
    count,
    review,
    timestamp
)

	🔑	🔑	🔑
	3,5,1,good,12	4,3,1,bad,8	5,2,1,okay,2
	4,2,1,bad,13	3,4,1,good,15	5,5,1,good,19
	3,3,1,bad,25	3,3,1,okay,20	5,2,1,good,21
	3,3,1,okay,29	4,4,1,good,33	3,2,1,bad,35
	…	…	…
final data	13,13,4,okay,29	50,60,20,bad,60	40,38,10,good,88

Protocol

- **Receive data:**
  1. **Prove Alice owns which epoch keys**



nonce=0

nonce=1

nonce=2

Airbnb user
Alice

epoch key
0×123

epoch key
0×456

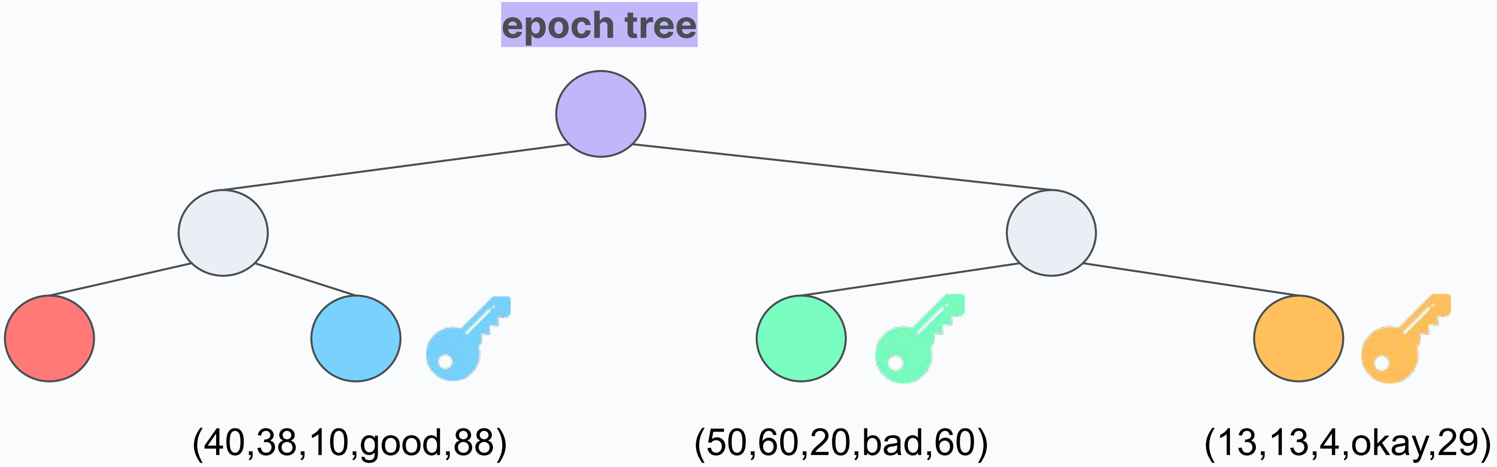
epoch key
0×789

https://developer.unirep.io/docs/protocol/user-state-transition

Protocol

- **Receive data:**

    2. **Prove epoch key status in epoch tree**



https://developer.unirep.io/docs/protocol/user-state-transition

Protocol

- **Receive data:**
  3. **Calculate the final data status**

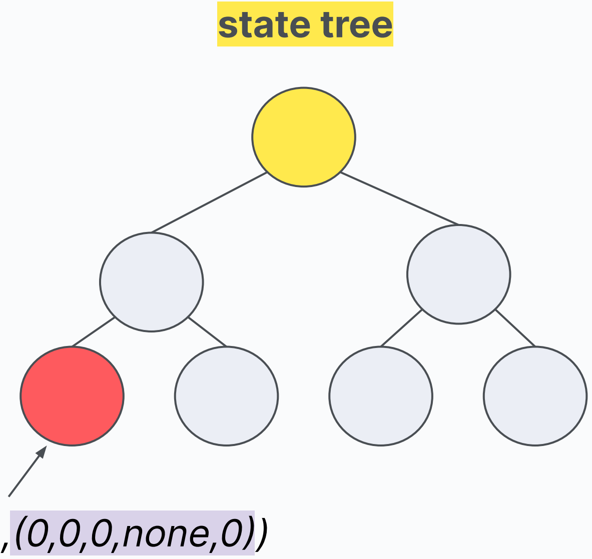
	sum data fields			replacement data fields	
epoch keys	**communication**	**cleanliness**	**count**	**reviews**	**timestamp**
0x789	40	38	10	good	88
0x456	50	60	20	bad	60
0x123	13	13	4	okay	29
final data	**103**	**111**	**34**	**good**	**88**

https://developer.unirep.io/docs/protocol/user-state-transition

Protocol

- **Receive data:**

  4. **Check state tree (sign up) status**



*The status changing in current epoch has not been included yet

https://developer.unirep.io/docs/protocol/user-state-transition

Protocol

- **Receive data:**

  5. **Compute the updated status (state tree leaf)**

	communication	cleanliness	count	reviews	timestamp
old status	0	0	0	none	0
new status	103	111	34	good	88
final status	**103**	**111**	**34**	**good**	**88**

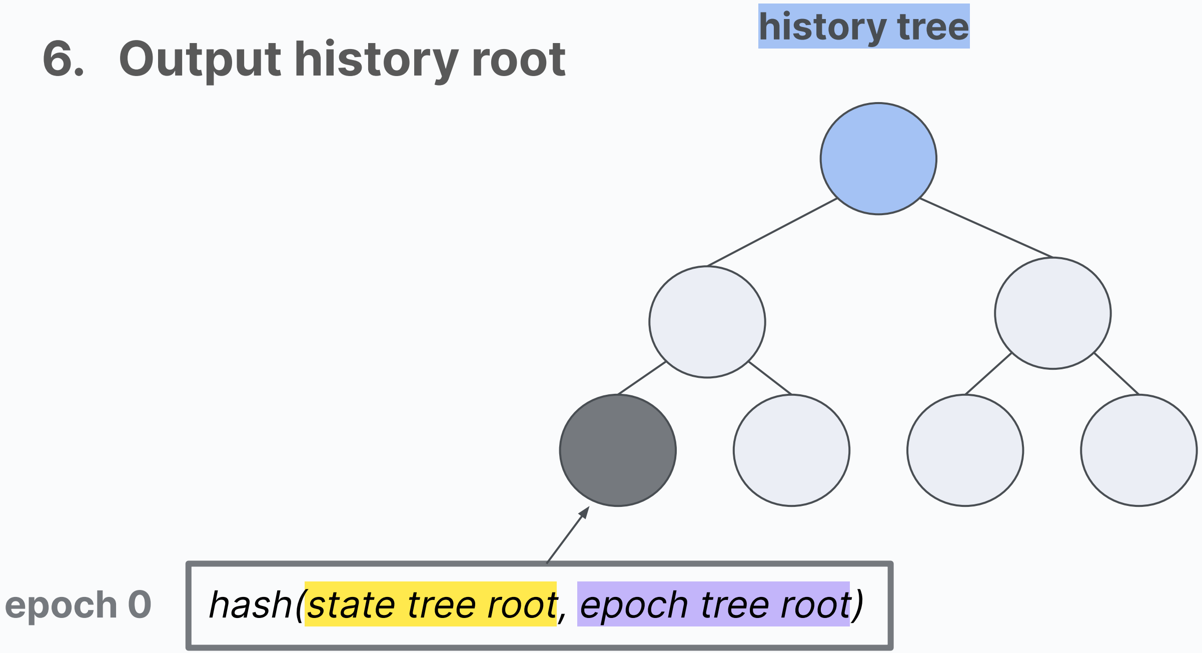
new leaf = hash(Alice, Airbnb, (103,111,34,good,88))

https://developer.unirep.io/docs/protocol/user-state-transition

PSE

Protocol

- Receive data:
  6. Output history root



history tree

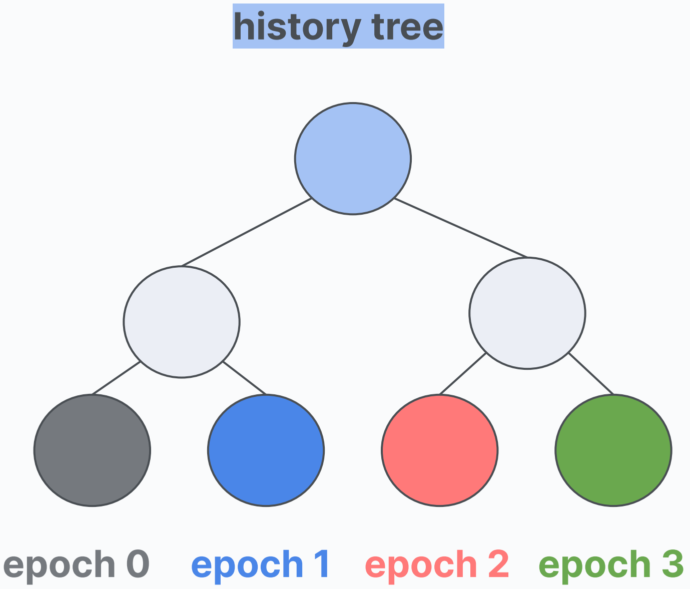
epoch 0

hash(state tree root, epoch tree root)

https://developer.unirep.io/docs/protocol/trees#history-tree

Protocol

- Example of the history tree

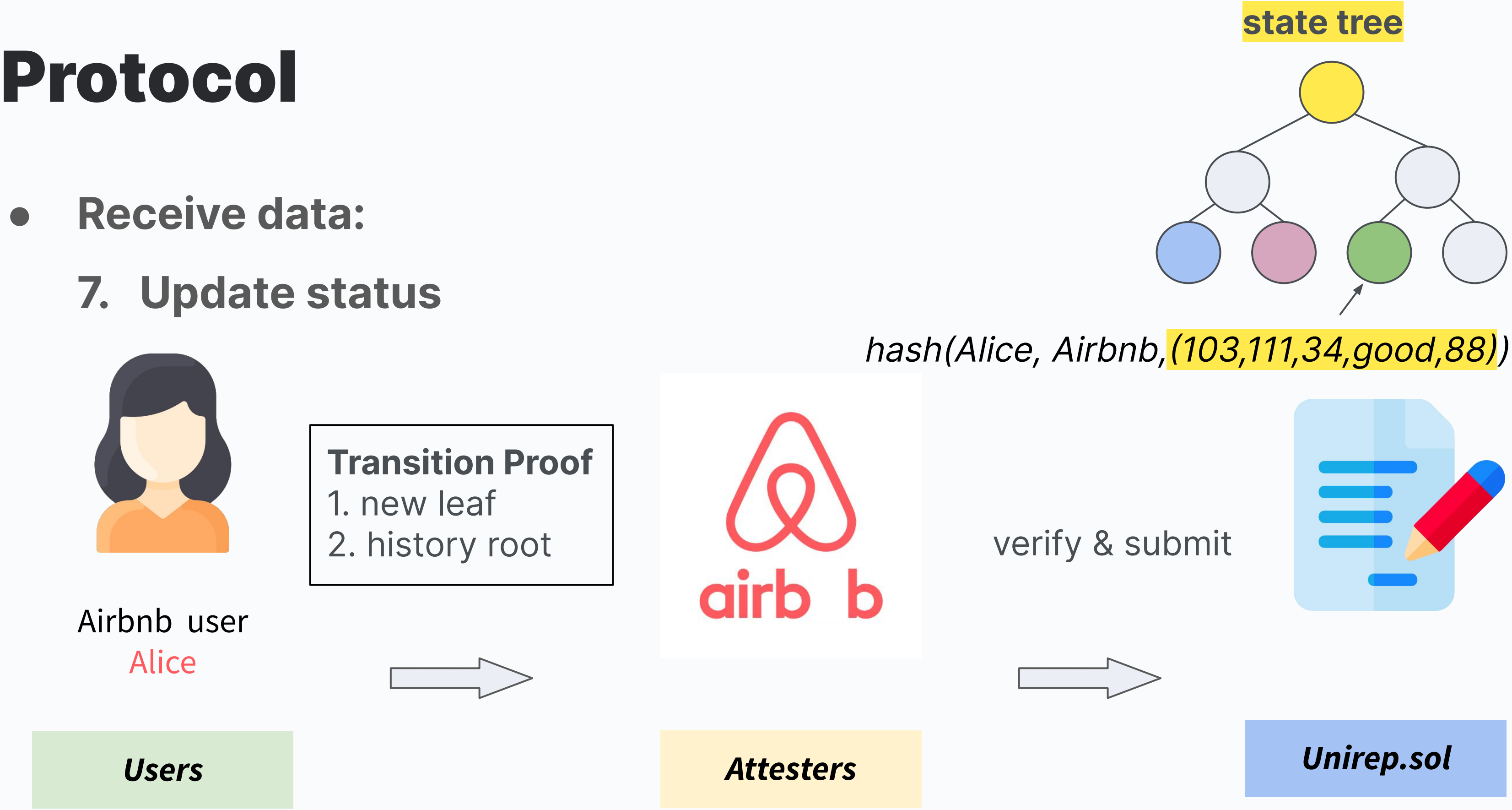


https://developer.unirep.io/docs/protocol/trees#history-tree

Protocol

- **Receive data:**

  7. **Update status**



state tree

hash(Alice, Airbnb,(103,111,34,good,88))

Transition Proof
1. new leaf
2. history root

Airbnb user
Alice

verify & submit

Users

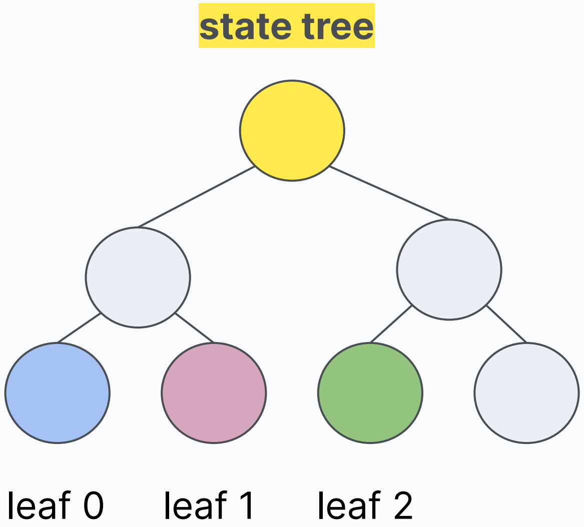
Attesters

Unirep.sol

https://developer.unirep.io/docs/contracts-api/unirep-sol#userstatetransition

Protocol

- **Example of state tree of epoch 1**



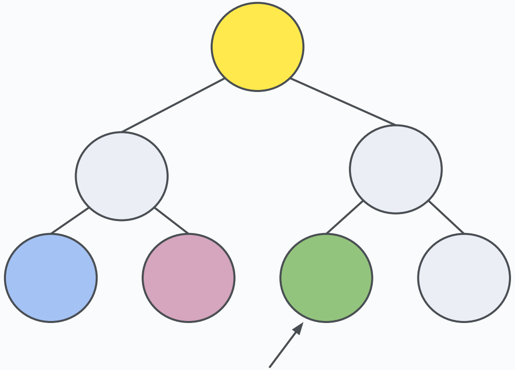
Index	value	type
leaf 0	hash(Cindy, Airbnb, (0,0,0,none,0))	sign up
leaf 1	hash(Tom, Airbnb, (2,3,5, bad, 3))	transition
leaf 2	hash(Alice, Airbnb,(103,111,34,good,88))	transition
leaf i	*hash(user_i, Airbnb, data)*	*sign up/ transition*

https://developer.unirep.io/docs/protocol/trees#state-tree

Protocol

- **Prove data:**
  1. **State tree membership**

state tree



hash(Alice, Airbnb,(103,111,34,good,88))

https://developer.unirep.io/docs/circuits-api/circuits#prove-reputation-proof

Protocol

- **Prove data:**

   2. **Claim data**

	communication	cleanliness	count	reviews	timestamp
final status	**103**	**111**	**34**	**good**	**88**

e.g.

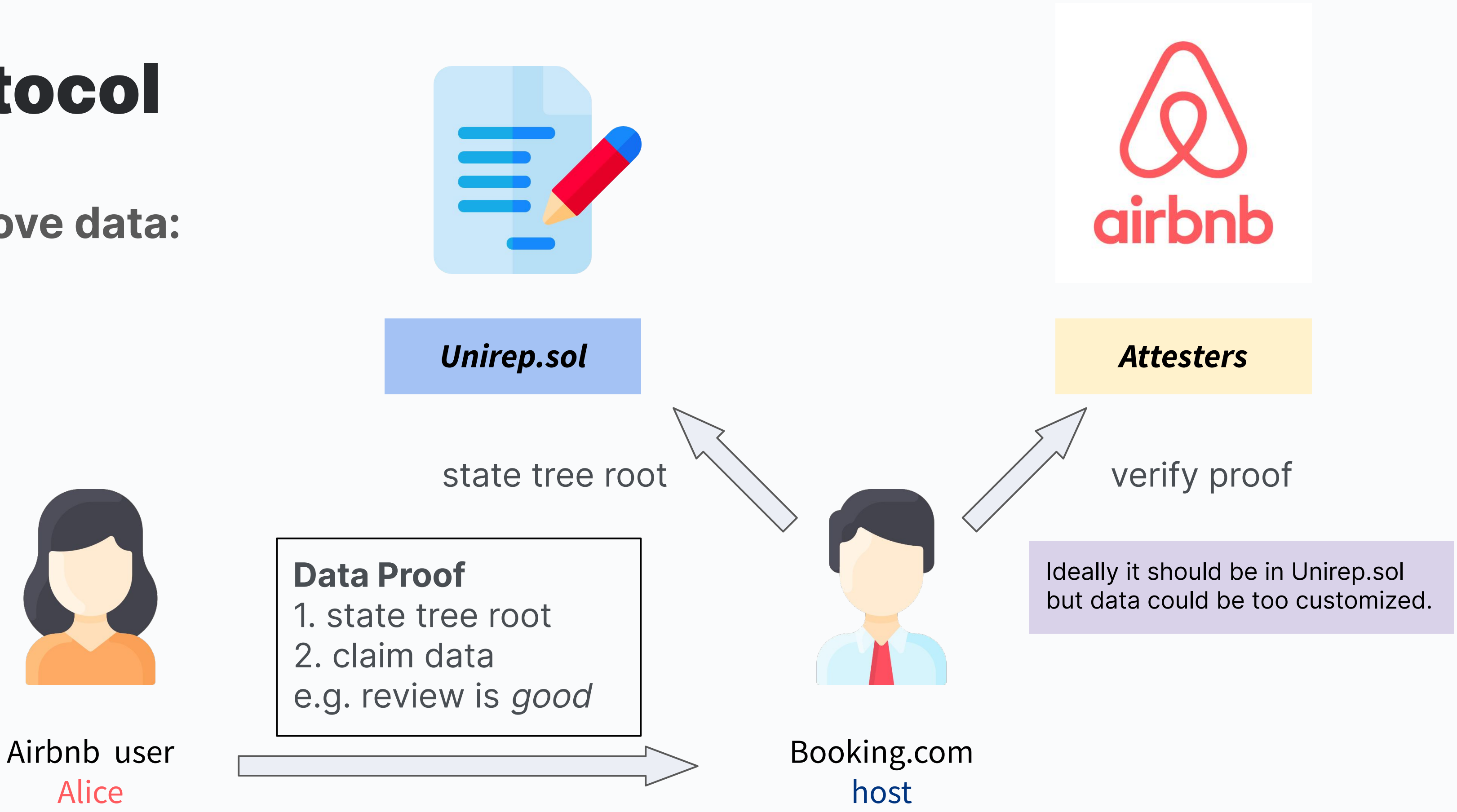
- communication rate (103/34=3.03) **> 3**
- cleanliness rate (111/34=3.26) **> 3**
- count **> 10**
- reviews **== good**
- ...

without revealing the exact data

https://developer.unirep.io/docs/circuits-api/circuits#prove-reputation-proof

Protocol

- **Prove data:**



Unirep.sol

Attesters

state tree root

verify proof

Ideally it should be in Unirep.sol but data could be too customized.

Data Proof
1. state tree root
2. claim data
e.g. review is *good*

Airbnb user
Alice

Booking.com
host

https://developer.unirep.io/docs/circuits-api/circuits#prove-reputation-proof

Example

- **Prove data from web2**
  - **e.g. Github**

	increased followers	decreased followers	increased starts	decreased stars	username
data	**30**	**1**	**40**	**3**	**vivianjeng**

e.g.

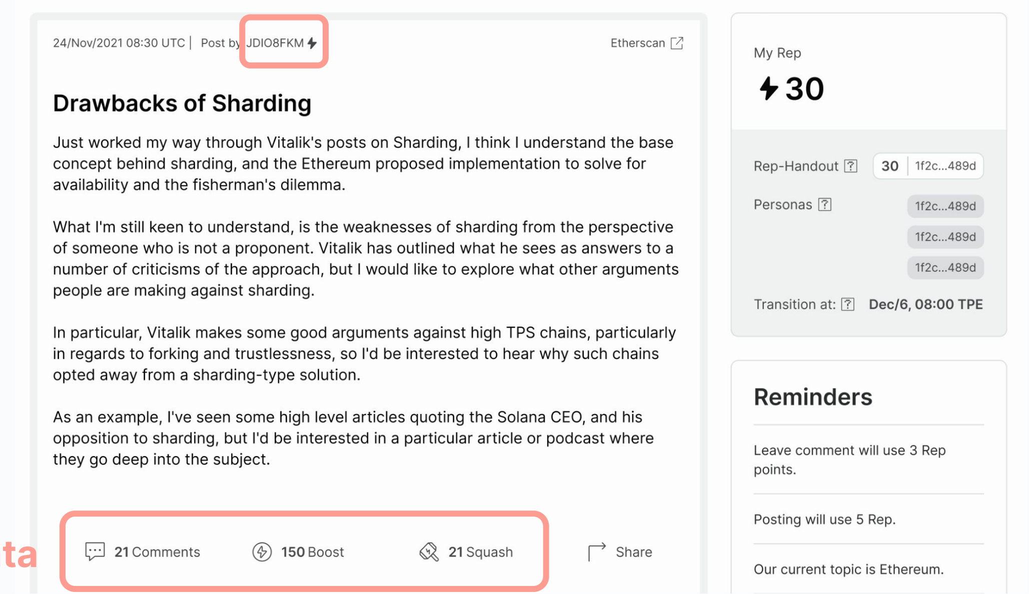
- current followers (30-1) **> 10**
- current stars (40-3) **> 10**
- username **== vivianjeng**

https://github.com/kittybest/my-badge

PSE

Example

- web3 anonymous social media/forum



24/Nov/2021 08:30 UTC | Post by JDIO8FKM ⚡

Etherscan

Drawbacks of Sharding

Just worked my way through Vitalik's posts on Sharding, I think I understand the base concept behind sharding, and the Ethereum proposed implementation to solve for availability and the fisherman's dilemma.

What I'm still keen to understand, is the weaknesses of sharding from the perspective of someone who is not a proponent. Vitalik has outlined what he sees as answers to a number of criticisms of the approach, but I would like to explore what other arguments people are making against sharding.

In particular, Vitalik makes some good arguments against high TPS chains, particularly in regards to forking and trustlessness, so I'd be interested to hear why such chains opted away from a sharding-type solution.

As an example, I've seen some high level articles quoting the Solana CEO, and his opposition to sharding, but I'd be interested in a particular article or podcast where they go deep into the subject.

21 Comments 150 Boost 21 Squash Share

user data

My Rep

⚡30

Rep-Handout [?] 30 1f2c...489d

Personas [?] 1f2c...489d
1f2c...489d
1f2c...489d

Transition at: [?] Dec/6, 08:00 TPE

Reminders

Leave comment will use 3 Rep points.

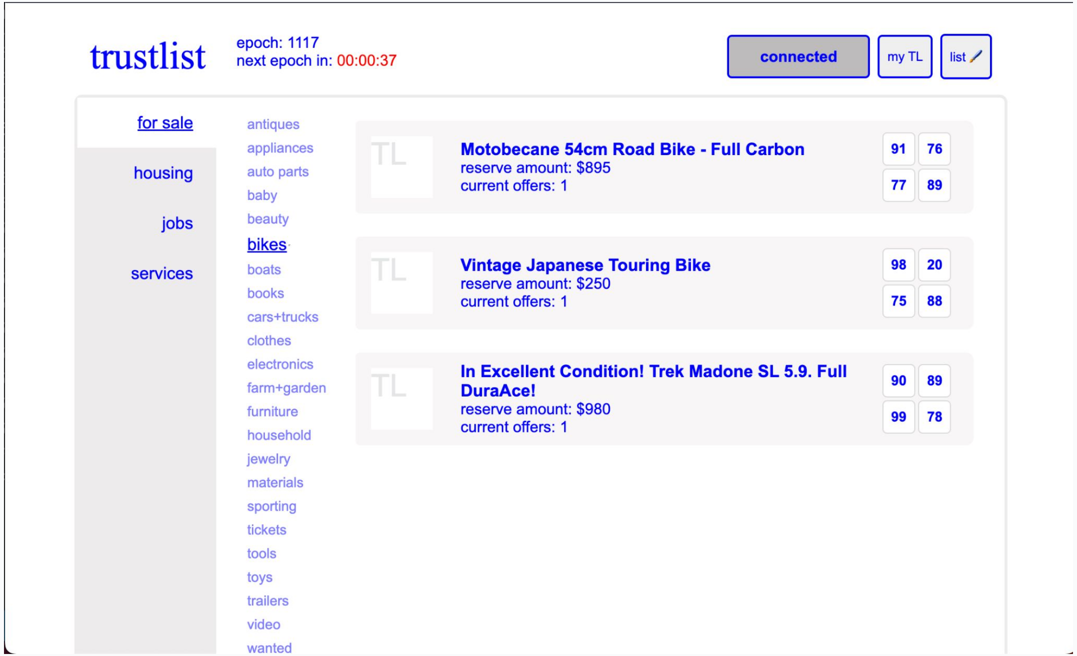
Posting will use 5 Rep.

Our current topic is Ethereum.

https://github.com/unirep/unirep-social

Example

- **web3 C2C service**



epoch: 1117
next epoch in: 00:00:37

Motobecane 54cm Road Bike - Full Carbon
reserve amount: $895
current offers: 1

91	76
77	89

Vintage Japanese Touring Bike
reserve amount: $250
current offers: 1

98	20
75	88

In Excellent Condition! Trek Madone SL 5.9. Full DuraAce!
reserve amount: $980
current offers: 1

90	89
99	78

https://github.com/CJ-Rose/trustlist

Thank you

Github: https://github.com/unirep/unirep

Twitter: @UniRep_Protocol