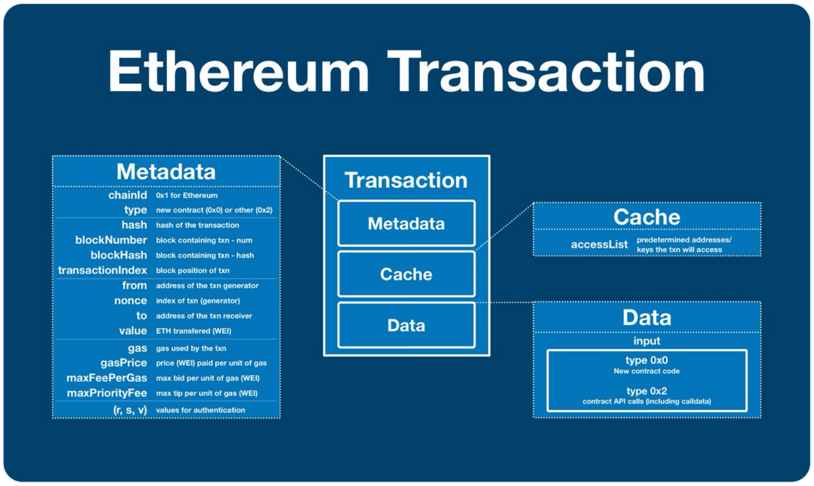
Haym Salomon @SalomonCrypto
Sep 9 • 19 tweets • SalomonCrypto/status/1568092433803808770

(1/18) @ethereum Fundamentals: Transactions

Sent $ETH? LP'ed into an AMM? Deployed a new contract? Everything you do on the World Computer leaves an on-chain record. Ever wonder what's inside your transactions?

A field-by-field guide to the atomic unit of Ethereum computing



Ethereum Transaction

Metadata

chainId	0x1 for Ethereum
type	new contract (0x0) or other (0x2)
hash	hash of the transaction
blockNumber	block containing txn - num
blockHash	block containing txn - hash
transactionIndex	block position of txn
from	address of the txn generator
nonce	index of txn (generator)
to	address of the txn receiver
value	ETH transfered (WEI)
gas	gas used by the txn
gasPrice	price (WEI) paid per unit of gas
maxFeePerGas	max bid per unit of gas (WEI)
maxPriorityFee	max tip per unit of gas (WEI)
(r, s, v)	values for authentication

Transaction
- Metadata
- Cache
- Data

Cache

accessList	predetermined addresses/ keys the txn will access

Data

input

type 0x0
New contract code

type 0x2
contract API calls (including calldata)

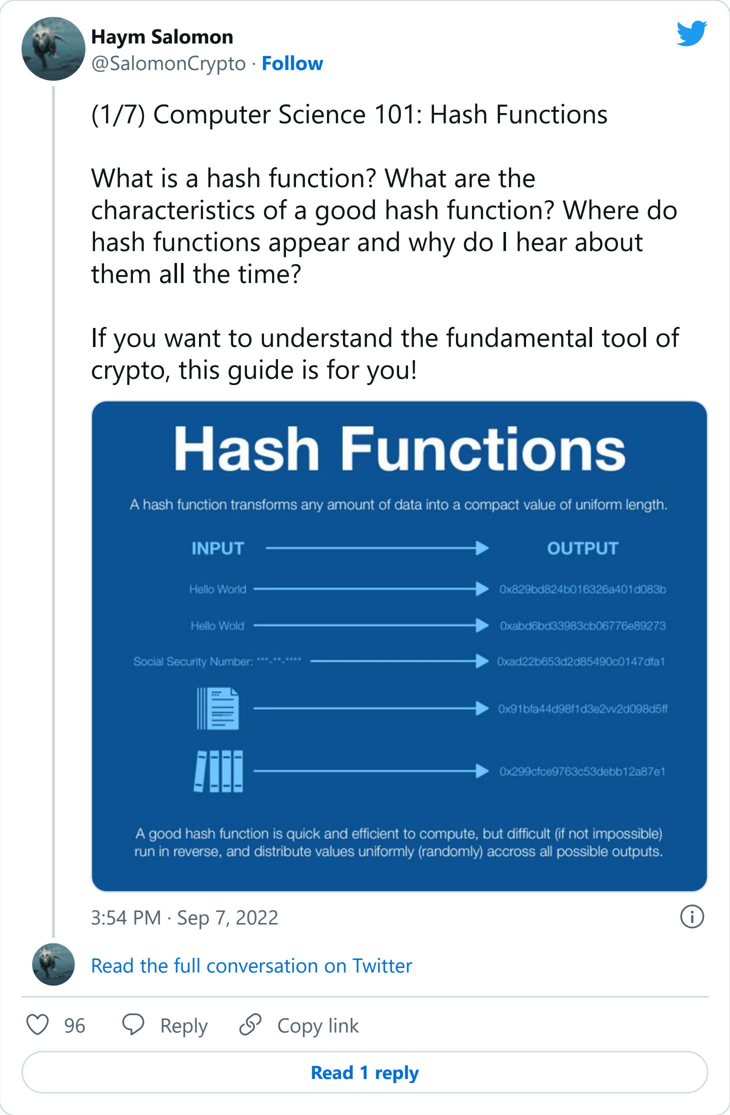
(2/18) @ethereum is the World Computer: a globally shared utility that exists between a network of 1000s of computers

Users interact with Ethereum through a wallet (like @MetaMask), which creates and sends txns to the network. Once accepted, the txns are written into a block.

(3/18) Perquisite - hashing, (applying a hash function)

Hash function: a piece of code used to transform any amount of data into a compact, uniform value. The input can be of arbitrary length but the output is always the same length.

(Good) hash functions are non-reversible.



Haym Salomon
@SalomonCrypto · Follow

(1/7) Computer Science 101: Hash Functions

What is a hash function? What are the characteristics of a good hash function? Where do hash functions appear and why do I hear about them all the time?

If you want to understand the fundamental tool of crypto, this guide is for you!

Hash Functions

A hash function transforms any amount of data into a compact value of uniform length.

INPUT → OUTPUT
Hello World → 0x829bd824b016326a401d083b
Hello Wold → 0xabd6bd33983cb06776e89273
Social Security Number: ***-**-**** → 0xad22b653d2d85490c0147dfa1
→ 0x91bfa44d98f1d3e2vv2d098d5ff
→ 0x299cfce9763c53debb12a87e1

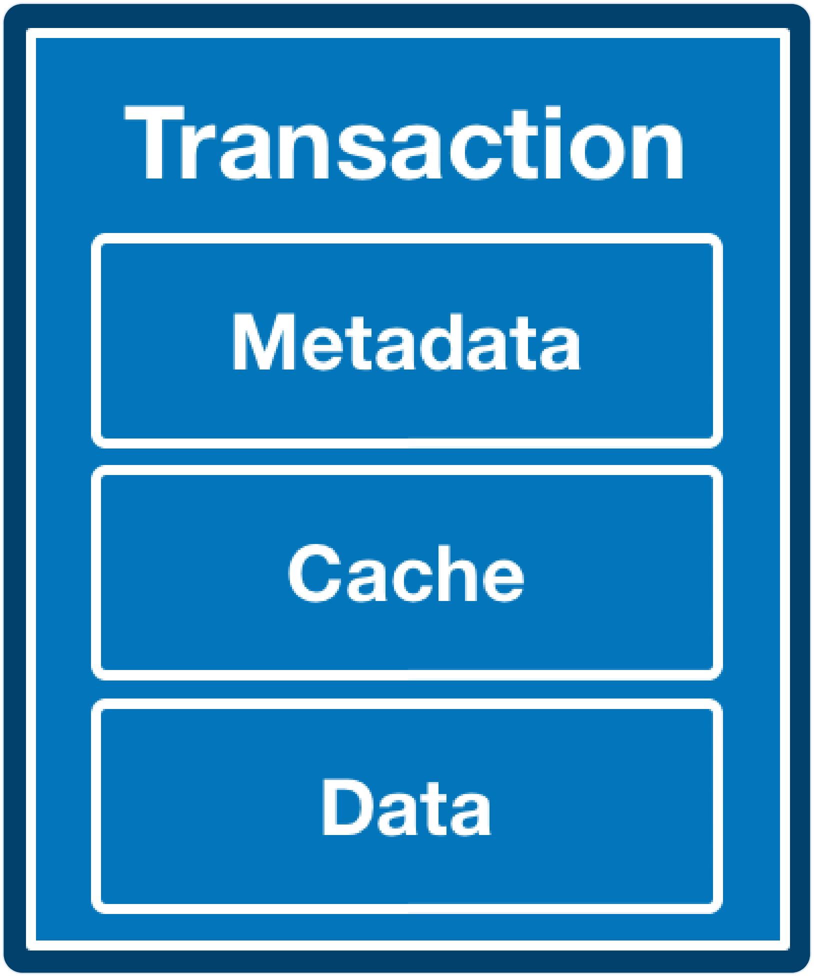
A good hash function is quick and efficient to compute, but difficult (if not impossible) run in reverse, and distribute values uniformly (randomly) accross all possible outputs.

3:54 PM · Sep 7, 2022

Read the full conversation on Twitter

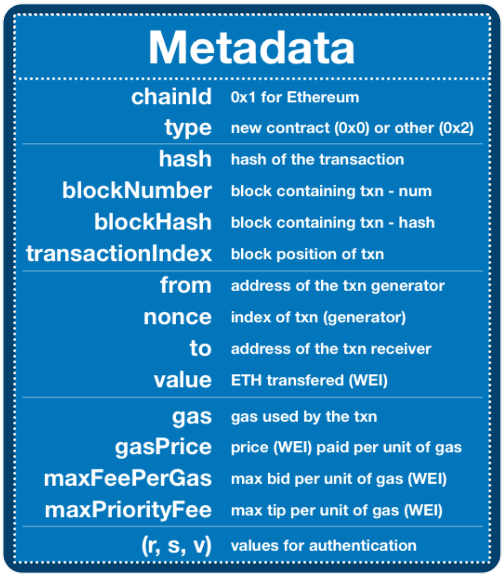
(4/18) An @ethereum transaction is made of up 3 parts:

- metadata, including to/from, $ETH amount, gas details and signature data
- cache, a list of accounts and keys the transaction expects to use
- data, the payload of the transaction (smart contract code or API call)



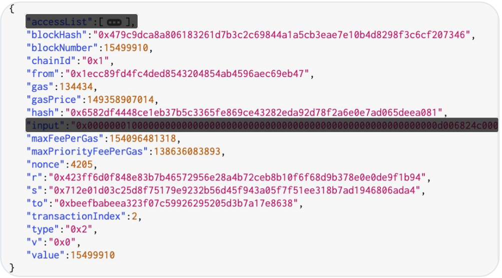
(5/18) Metadata - information about the transaction

The attached images show all the metadata fields. We will discuss the non-obvious ones in the tweets below.



Metadata

chainId	0x1 for Ethereum
type	new contract (0x0) or other (0x2)
hash	hash of the transaction
blockNumber	block containing txn - num
blockHash	block containing txn - hash
transactionIndex	block position of txn
from	address of the txn generator
nonce	index of txn (generator)
to	address of the txn receiver
value	ETH transfered (WEI)
gas	gas used by the txn
gasPrice	price (WEI) paid per unit of gas
maxFeePerGas	max bid per unit of gas (WEI)
maxPriorityFee	max tip per unit of gas (WEI)
(r, s, v)	values for authentication



```
{
  "accessList":[ ••• ],
  "blockHash":"0x479c9dca8a806183261d7b3c2c69844a1a5cb3eae7e10b4d8298f3c6cf207346",
  "blockNumber":15499910,
  "chainId":"0x1",
  "from":"0x1ecc89fd4fc4ded8543204854ab4596aec69eb47",
  "gas":134434,
  "gasPrice":149358907014,
  "hash":"0x6582df4448ce1eb37b5c3365fe869ce43282eda92d78f2a6e0e7ad065deea081",
  "input":"0x00000001000000000000000000000000000000000000000000000000000000d006824c000
  "maxFeePerGas":154096481318,
  "maxPriorityFeePerGas":138636083893,
  "nonce":4205,
  "r":"0x423ff6d0f848e83b7b46572956e28a4b72ceb8b10f6f68d9b378e0e0de9f1b94",
  "s":"0x712e01d03c25d8f75179e9232b56d45f943a05f7f51ee318b7ad1946806ada4",
  "to":"0xbeefbabeea323f07c59926295205d3b7a17e8638",
  "transactionIndex":2,
  "type":"0x2",
  "v":"0x0",
  "value":15499910
}
```

(6/18) chainId - introduced by EIP-155 to protect against an $ETC replay attack

type - there are two types: a new contract (0x0) and all others (0x2). EIP-2718 introduced a wrapper (0x2) that allows for many more types without affecting the core @ethereum specs

(7/18) nonce - number of transactions sent from a given address. Once imprinted on a block, the wallet's nonce is increased. Protects against replay attacks

to - address (wallet or smart contract) the transaction is being sent to

(8/18) value - amount of $ETH being transferred. Note - this is ONLY for $ETH (no other token)

gas - units of gas used by the transaction

maxFeePerGas - maximum amount (WEI per gas) the user who created the transaction is willing to pay. Inclusive of base fee and priority fee

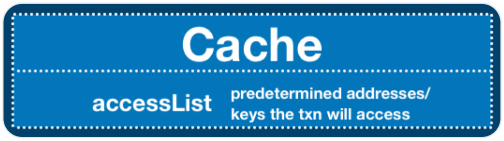
(9/18) maxPriorityFeePerGas - maximum amount (WEI per gas) above the base fee the user who created the transaction is willing to pay. This fee will be paid directly to the miner/validator as a tip to incentive inclusion.

gasPrice - cost per unit of gas paid by this transaction

(10/18) (r, s, v) - three values that form the signature of the user who created the transaction. They can be used to verify that the user authorized the transaction before it was executed in the EVM

For more information, see: Elliptic Curve Digital Signature Algorithm (ECDSA)

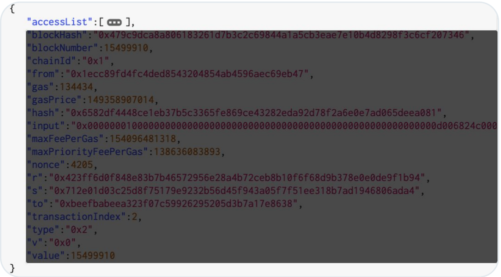
(11/18) Cache - contains the accessList, a list of addresses and keys the transaction anticipates using. The transaction will still be able to use resources off this list, but at a higher cost.



Cache

accessList

predetermined addresses/ keys the txn will access



```
{
   "accessList":[ ••• ],
   "blockHash":"0x479c9dca8a806183261d7b3c2c69844a1a5cb3eae7e10b4d8298f3c6cf207346",
   "blockNumber":15499910,
   "chainId":"0x1",
   "from":"0x1ecc89fd4fc4ded8543204854ab4596aec69eb47",
   "gas":134434,
   "gasPrice":149358907014,
   "hash":"0x6582df4448ce1eb37b5c3365fe869ce43282eda92d78f2a6e0e7ad065deea081",
   "input":"0x000000010000000000000000000000000000000000000000000000000000d006824c000
   "maxFeePerGas":154096481318,
   "maxPriorityFeePerGas":138636083893,
   "nonce":4205,
   "r":"0x423ff6d0f848e83b7b46572956e28a4b72ceb8b10f6f68d9b378e0e0de9f1b94",
   "s":"0x712e01d03c25d8f75179e9232b56d45f943a05f7f51ee318b7ad1946806ada4",
   "to":"0xbeefbabeea323f07c59926295205d3b7a17e8638",
   "transactionIndex":2,
   "type":"0x2",
   "v":"0x0",
   "value":15499910
}
```

(12/18) The accessList was added by EIP-2929, allowing clients to fetch/cache data to be used during the transaction.

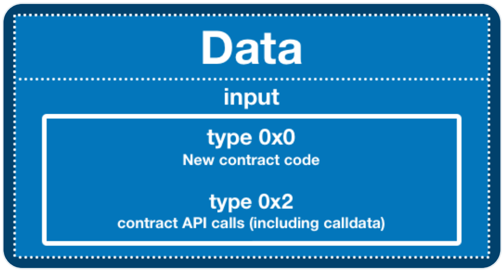
Today, the discount for using addresses & keys in the accessList is ~10%. However, this will increase in the future as @ethereum moves to support light clients.

(13/18) Data - the data payload being delivered by the transaction. This can be used in 3 ways:

- $ETH transfer - empty

- smart contract API call - name of function and parameters

- new smart contract - code of the smart contract

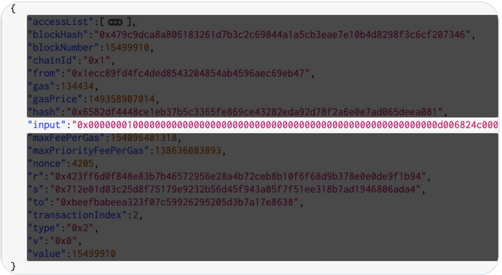


Data

input

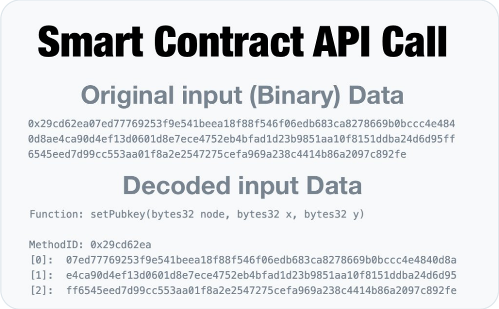
type 0x0
New contract code

type 0x2
contract API calls (including calldata)



```
{
   "accessList":[ ••• ],
   "blockHash":"0x479c9dca8a806183261d7b3c2c69844a1a5cb3eae7e10b4d8298f3c6cf207346",
   "blockNumber":15499910,
   "chainId":"0x1",
   "from":"0x1ecc89fd4fc4ded8543204854ab4596aec69eb47",
   "gas":134434,
   "gasPrice":149358907014,
   "hash":"0x6582df4448ce1eb37b5c3365fe869ce43282eda92d78f2a6e0e7ad065deea081",
   "input":"0x000000010000000000000000000000000000000000000000000000000000d006824c000
   "maxFeePerGas":154096481318,
   "maxPriorityFeePerGas":138636083893,
   "nonce":4205,
   "r":"0x423ff6d0f848e83b7b46572956e28a4b72ceb8b10f6f68d9b378e0e0de9f1b94",
   "s":"0x712e01d03c25d8f75179e9232b56d45f943a05f7f51ee318b7ad1946806ada4",
   "to":"0xbeefbabeea323f07c59926295205d3b7a17e8638",
   "transactionIndex":2,
   "type":"0x2",
   "v":"0x0",
   "value":15499910
}
```

(14/18) Data in the input field is recorded in binary, but can be translated back to a human readable form.



Smart Contract API Call

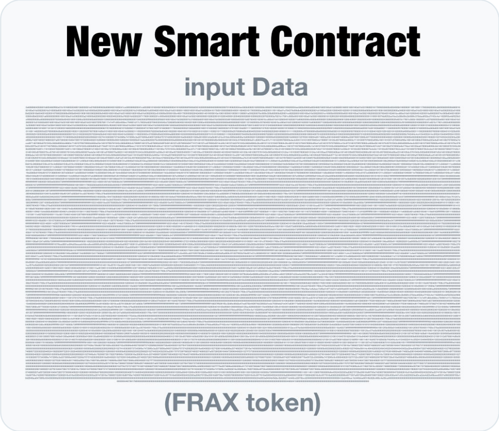
Original input (Binary) Data

```
0x29cd62ea07ed77769253f9e541beea18f88f546f06edb683ca8278669b0bccc4e484
0d8ae4ca90d4ef13d0601d8e7ece4752eb4bfad1d23b9851aa10f8151ddba24d6d95ff
6545eed7d99cc553aa01f8a2e2547275cefa969a238c4414b86a2097c892fe
```

Decoded input Data

```
Function: setPubkey(bytes32 node, bytes32 x, bytes32 y)

MethodID: 0x29cd62ea
[0]:  07ed77769253f9e541beea18f88f546f06edb683ca8278669b0bccc4e4840d8a
[1]:  e4ca90d4ef13d0601d8e7ece4752eb4bfad1d23b9851aa10f8151ddba24d6d95
[2]:  ff6545eed7d99cc553aa01f8a2e2547275cefa969a238c4414b86a2097c892fe
```



New Smart Contract

input Data

(FRAX token)

(15/18) The input field exists on-chain, but is not part of the EVM state. It simply provides data for the contract to use during the transaction, it is not tracked by @ethereum nor used in consensus.

The EVM can only use data supplied in that transaction; it cannot look back.

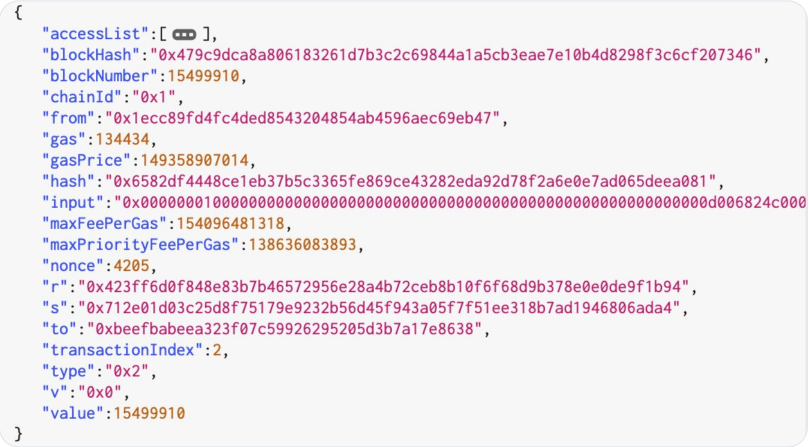
(16/18) This property becomes useful for applications that want to write historical data to the @ethereum blockchain (eg for manual retrieval later) but don't care about having direct EVM access.

Rollups are the first category of applications to truly leverage this idea.

(17/18) We will cover rollups another time. For now, rollups rely on the fact that writing data into the input field is cheaper than writing directly into the @ethereum state.

This allows rollups to execute much more efficiently while still posting a record of all txns on-chain.

(18/18) And there you have it! That's an @ethereum transaction!



```
{
   "accessList":[ ••• ],
   "blockHash":"0x479c9dca8a806183261d7b3c2c69844a1a5cb3eae7e10b4d8298f3c6cf207346",
   "blockNumber":15499910,
   "chainId":"0x1",
   "from":"0x1ecc89fd4fc4ded8543204854ab4596aec69eb47",
   "gas":134434,
   "gasPrice":149358907014,
   "hash":"0x6582df4448ce1eb37b5c3365fe869ce43282eda92d78f2a6e0e7ad065deea081",
   "input":"0x000000010000000000000000000000000000000000000000000000000000000d006824c000
   "maxFeePerGas":154096481318,
   "maxPriorityFeePerGas":138636083893,
   "nonce":4205,
   "r":"0x423ff6d0f848e83b7b46572956e28a4b72ceb8b10f6f68d9b378e0e0de9f1b94",
   "s":"0x712e01d03c25d8f75179e9232b56d45f943a05f7f51ee318b7ad1946806ada4",
   "to":"0xbeefbabeea323f07c59926295205d3b7a17e8638",
   "transactionIndex":2,
   "type":"0x2",
   "v":"0x0",
   "value":15499910
}
```

Like what you read? Help me spread the word by retweeting the thread (linked below).

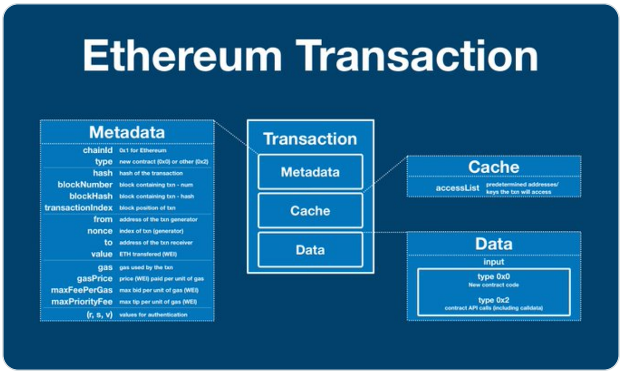
Follow me for more explainers and as much alpha as I can possibly serve.

Haym Salomon
@SalomonCrypto · **Follow**

(1/18) **@ethereum** Fundamentals: Transactions

Sent **$ETH**? LP'ed into an AMM? Deployed a new contract? Everything you do on the World Computer leaves an on-chain record. Ever wonder what's inside your transactions?

A field-by-field guide to the atomic unit of Ethereum computing


Ethereum Transaction
Metadata
chainId 0x1 for Ethereum
type new contract (0x0) or other (0x2)
hash hash of the transaction
blockNumber block containing txn - num
blockHash block containing txn - hash
transactionIndex block position of txn
from address of the txn generator
nonce index of txn (generator)
to address of the txn receiver
value ETH transfered (WEI)
gas gas used by the txn
gasPrice price (WEI) paid per unit of gas
maxFeePerGas max bid per unit of gas (WEI)
maxPriorityFee max tip per unit of gas (WEI)
(r, s, v) values for authentication
Transaction
Metadata
Cache
Data
Cache
accessList predetermined addresses/ keys the txn will access
Data
input
type 0x0
New contract code
type 0x2
contract API calls (including calldata)

4:22 AM · Sep 9, 2022

Read the full conversation on Twitter

♡ 81 Reply Copy link

Read 3 replies

• • •