

# Lesson 15

## Today's topics

- Compressed NFTs
- Debugging Solana Programs
- Pyth Oracle Network
- Eclipse
- Anchor
  - PDA and CPI
  - Further examples
- Anchor v 0.30.0
- Identity Solutions
- MEV
- Commitment Levels in Transactions

## News

Hacker jailed for stealing from Solana DEXs

See [article](#)

---

## Compressed NFTs

See [Docs](#)

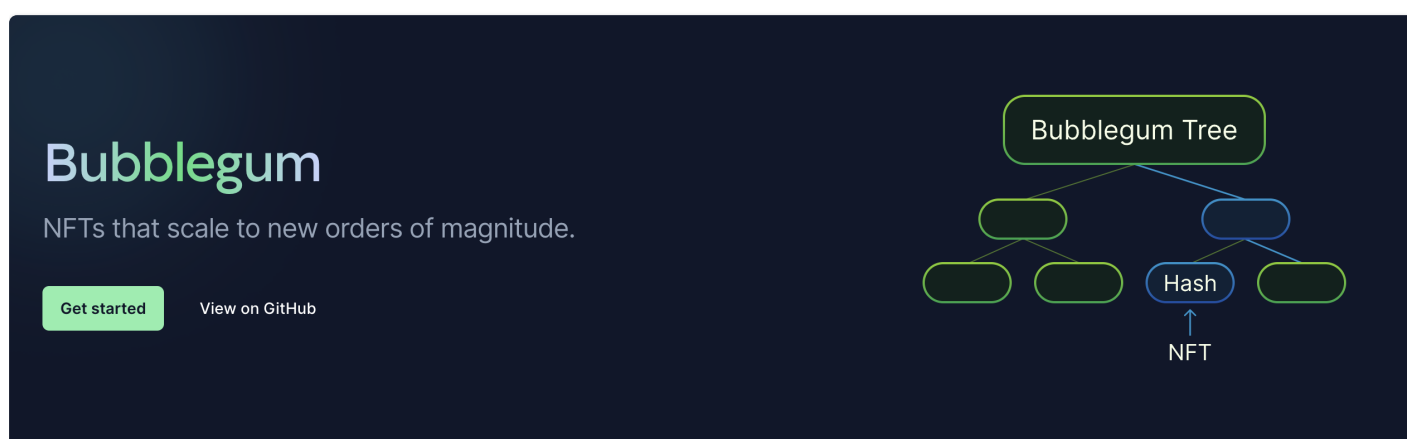
We have seen that NFTs with their metadata is stored on chain.

This can be very expensive if you need to mint large numbers of NFTs

Compressed NFTs were introduced to reduced this cost, they do so by storing the metadata off chain.

They use state [compression](#) and merkle trees to store data off chain in a special ledger, but it is still made available for consensus.

Metaplex have created the Bubblegum project to handle compressed NFTs



The image shows a screenshot of the Bubblegum project website on the left and a diagram of a Bubblegum Tree on the right. The website has a dark blue background with the 'Bubblegum' logo in green and white, the tagline 'NFTs that scale to new orders of magnitude.', and two buttons: 'Get started' and 'View on GitHub'. The diagram on the right illustrates a 'Bubblegum Tree' structure. It is a binary tree where the root node is labeled 'Bubblegum Tree'. It branches into two nodes, which further branch into four nodes. The third node from the left is labeled 'Hash' and is highlighted in blue. An arrow points from the text 'NFT' below to the 'Hash' node, indicating that the NFT's metadata is stored off-chain and its hash is stored on-chain.

Compressed NFTs store all of their metadata in the [ledger](#), instead of in traditional [accounts](#) like uncompressed NFTs, so indexing services are needed to retrieve the metadata.

This indexing service is available from RPC providers such as

- Helius
- Triton
- SimpleHash

#### Creating a compressed NFT

- create an NFT collection (or use an existing one)
- create a [concurrent merkle tree](#) (using the `@solana/spl-account-compression` SDK)
- mint compressed NFTs into your tree (to any owner's address you want)

#### Transferring the NFT

1. get the NFT "asset" information (from the indexer)
2. get the NFT's "proof" (from the indexer)
3. get the Merkle tree account (from the Solana blockchain)
4. prepare the asset proof (by parsing and formatting it)
5. build and send the transfer instruction

For a walkthrough of this process see [Docs](#)

Using compressed NFTs can reduce the cost of minting by up to 99%

---

## Debugging Solana Programs

For a guide to debugging Solana programs see [Cookbook](#)

---

## Pyth Oracle Network

See [site](#)



Pyth Network is an oracle project built on Solana. It focuses on bringing high fidelity market data from trust worthy data provider in a high speed manner.

Market data, as well as a confidence level of that market price are submitted by data provider to be aggregated on Solana chain. It also has cross chain compatibility using Wormhole as a bridge between ERC20 and SPL tokens.

Example data [feed](#)

[Integrating a data feed](#)

See [Docs](#)

Pyth on Solana [Docs](#)

Developers should integrate Pyth into both their on-chain and off-chain code:

1. On-chain programs should read prices from the Pyth program deployed on the same chain
2. Off-chain frontends and jobs should include Pyth price updates alongside (or within) their application-specific transactions.
3. Pyth provides ecosystem-specific SDKs to assist with both the on- and off-chain pieces of the integration. The easiest way to use Pyth price feeds is to integrate the appropriate SDKs into your application.

Pyth on Solana

Example Anchor [code](#) and [lib.rs](#)

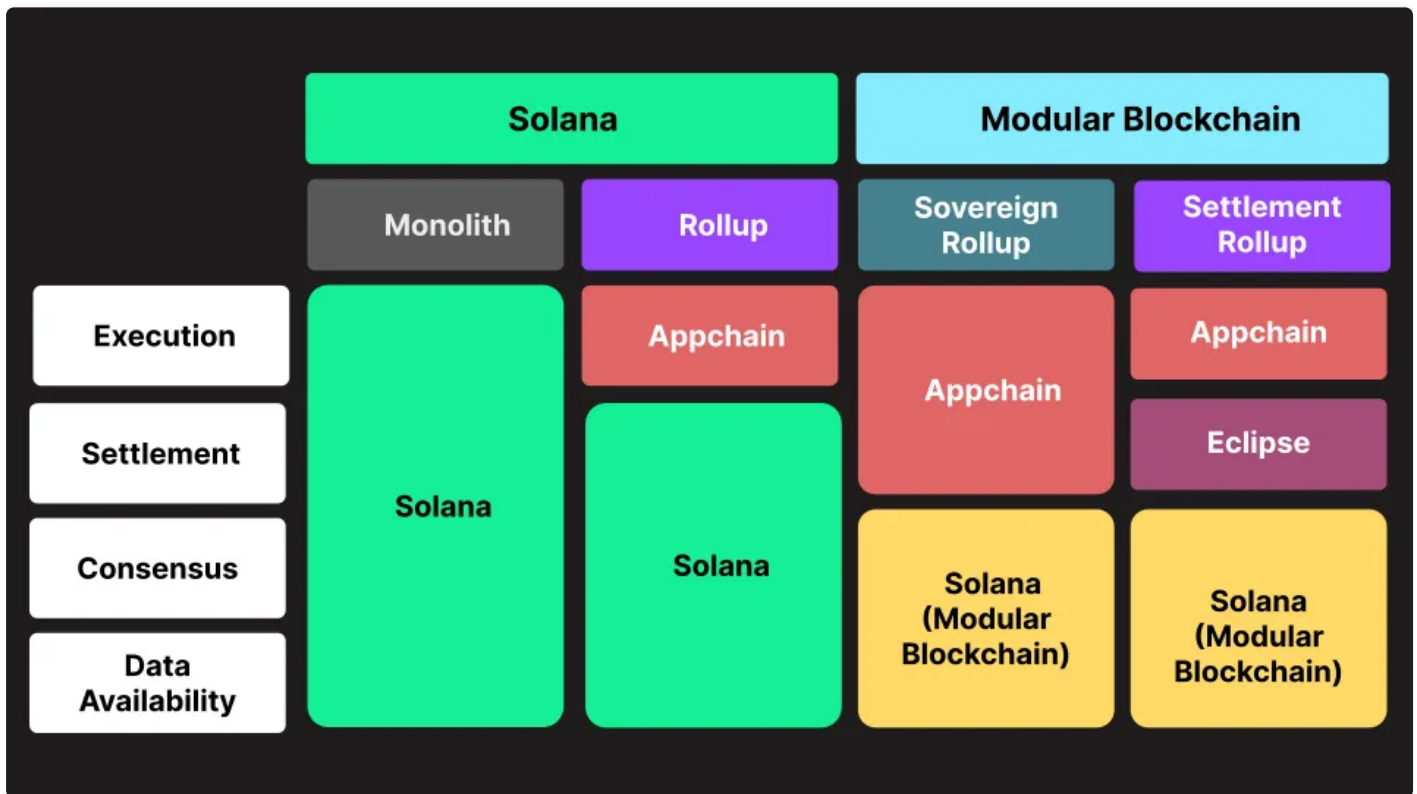
---

## Eclipse and App specific rollups

See [Eclipse Docs](#)

See [Article](#)

See [Bankless Podcast](#)



Eclipse is a rollup as a service, as an IBC-enabled rollup on Celestia, which is the data availability layer.

Eclipse allows you to spin up your own rollup



with the Sealevel VM with a base layer provided by [CelestiaOrg](#) , [Avail](#) , or [Eigen Layer](#).

There are also options to run an EVM or the Move VM.

#### Advantages to using Eclipse

1. **Maximum customisability:** Any dApp can adjust its block times, subsidise gas, or restrict who can use your blockchain.
  2. **Shared security:** dApps don't have to worry about managing infrastructure or bootstrapping validators, because it borrows security from the underlying Layer 1.
  3. **Scale horizontally:** This means that dApps can always spin up another execution layer, giving the power of Solana all to themselves.
  4. **Cheaper running costs:** The data availability layer doesn't need to support execution, and therefore the cost of running a node is cheaper promoting decentralisation.
-

## **PDA in Anchor**

See Anchor book [explanation](#)

## **CPI in Anchor**

See Anchor book [explanation](#)

---

## Further Anchor examples

Election program with Anchor

Example program in Anchor to run an election

See [Article](#)

Staking NFT

See [article](#)

---

## Anchor v 0.30.0

See [release notes](#) and [Changelog](#)

How to upgrade

First update the avm

```
cargo install --git  
https://github.com/coral-xyz/anchor --tag  
v0.30.0 avm --locked
```

Then update the anchor-cli

```
avm install latest
```

Update any dependencies to version 0.30.0

IDL generation change

The IDL type specification and generation has been rewritten, see [details](#)

`idl-build` feature is now required in your program's `Cargo.toml` definition in order for the IDL generation to work.

**Constraints for Token Extension program**

See examples [here](#)

## Test a specific program

A single program can be tested in a multi program workspace with the `--program-name` (`-p`) argument

```
anchor test --program-name example
```

---

# Identity Standards

## Decentralised Identifiers

Decentralised Identity standards exist in the form of

- DIDs
- Verifiable credentials

These are generic standards and not dependent on blockchain / zkp.

There is a W3 [standard](#) for DIDs

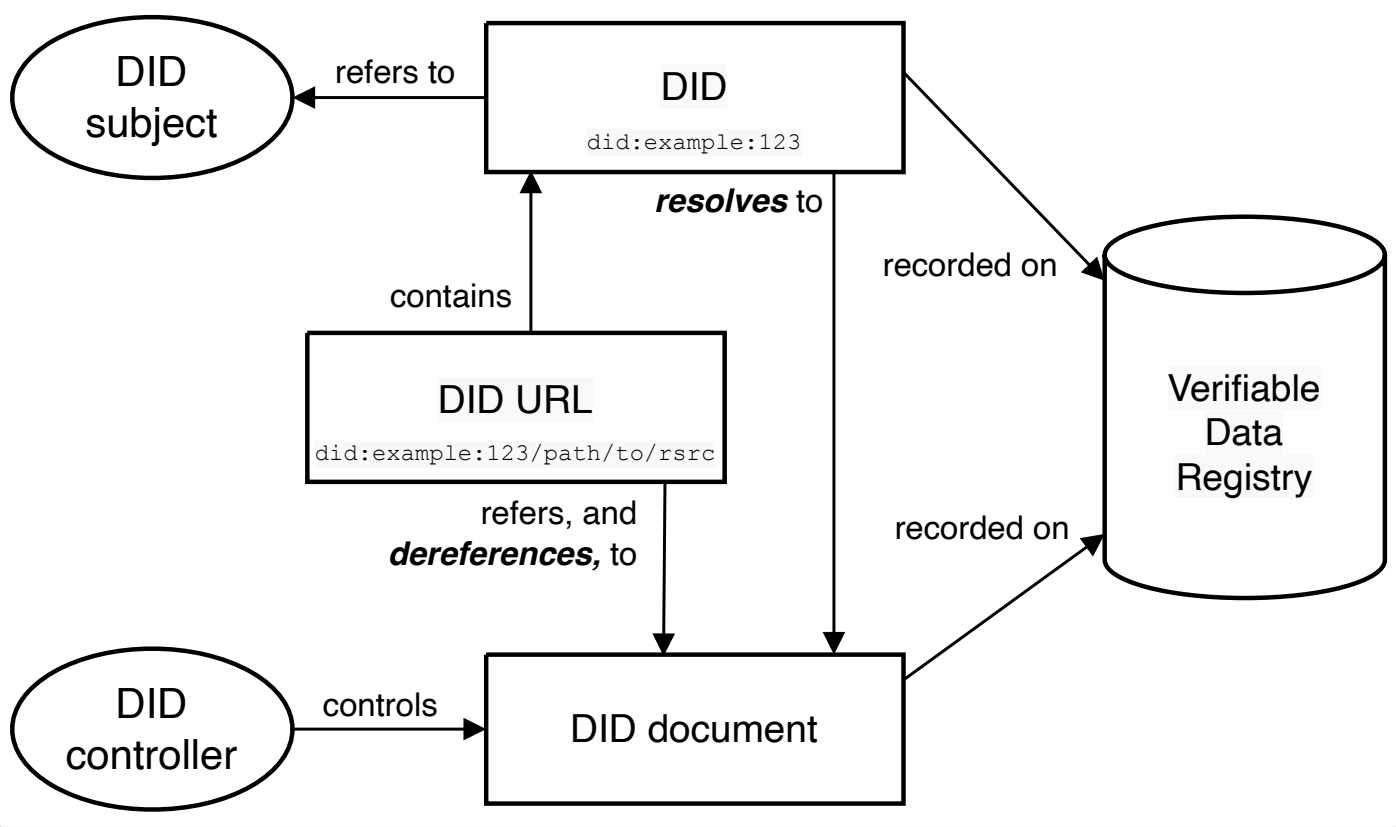
The DID consists of 3 parts

1. the `did` URI scheme identifier,
2. the identifier for the [DID method](#), and
3. the DID method-specific identifier.

Scheme

`did:example:123456789abcdefghi`

`example` is the DID Method  
`123456789abcdefghi` is the DID Method-Specific Identifier

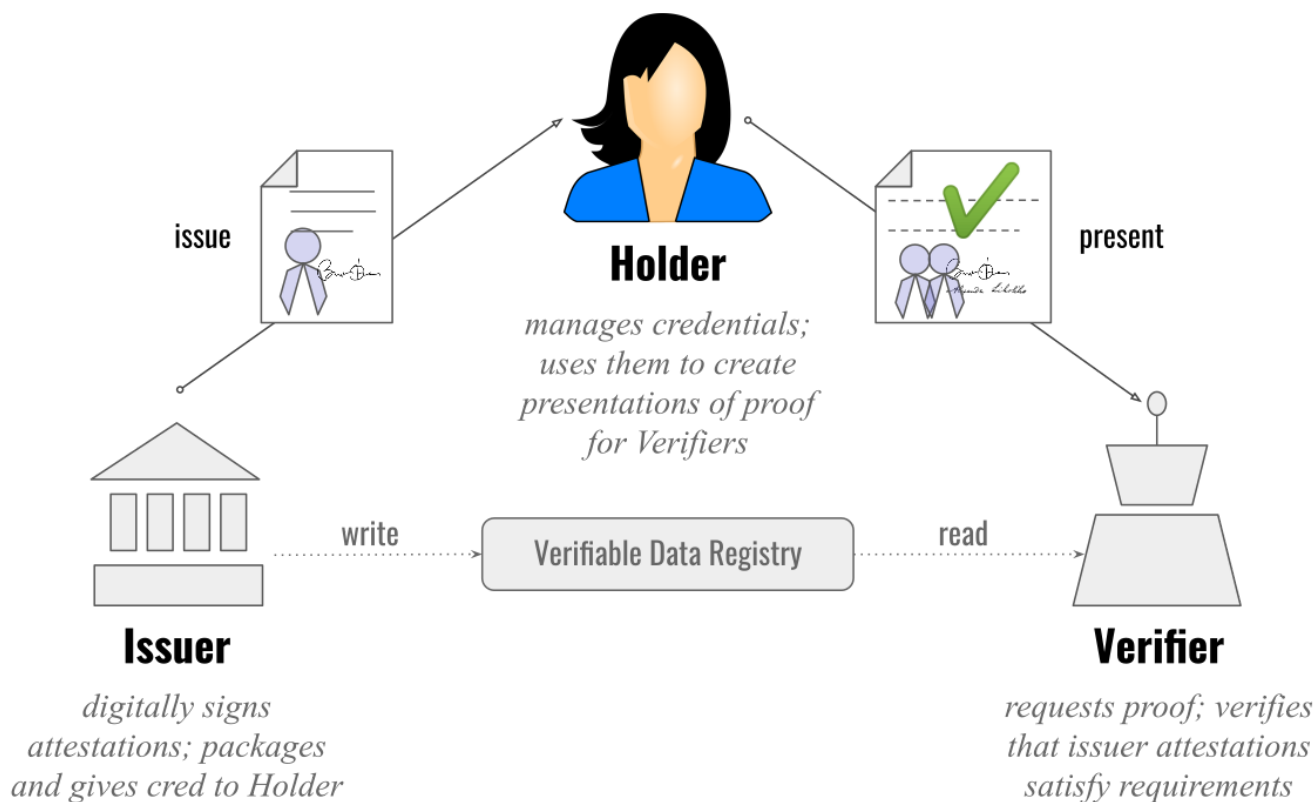


DIDs follow the [Privacy by Design](#) principle, with privacy as the default setting and privacy embedded in the design.

## Verifiable Credentials

See [Standard](#)

A verifiable credential is a tamper-evident credential that has authorship that can be cryptographically verified. Verifiable credentials can be used to build [verifiable presentations](#), which can also be cryptographically verified.





# Solana Identity Solutions

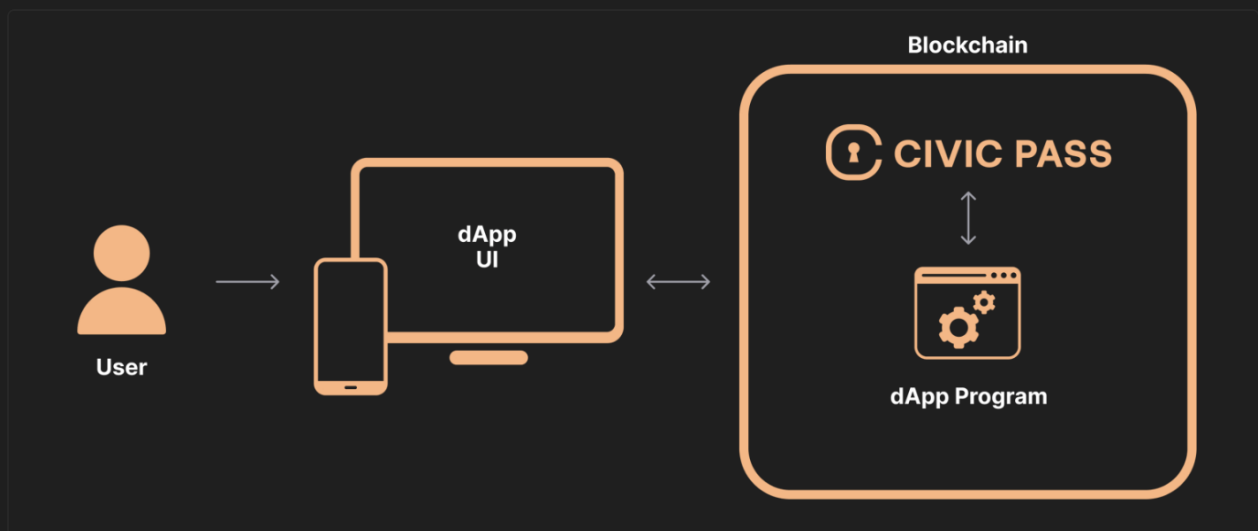
See [overview](#) from Alchemy

Most solutions are available on multiple chains

Civic

See [Docs](#)

Civic Pass is a multi-chain, wallet-agnostic identity and access management solution (IAM) for smart contracts, dApps, and Web2 companies entering blockchain.



Civic Pass allows you to manage access to your protocol or services, based on verifications you perform or with inputs from Civic's Identity Verification (IDV) Services.

An active Civic Pass proves that a user's wallet was verified according to your dApp's requirements.

## Features

- Sybil and AI resistance for
  - Access control
  - Gaming
  - NFTs
  - Online communities

- Compliance for:
  - CeFi
  - DeFi
  - Real World Assets (RWAs)
  - Gambling platforms

## BlockPass

See [announcement](#)

Blockpass is a user-centric digital identity verification system that enables a one-click compliance gateway to financial services and other regulated industries.

## Lit Protocol

See [Docs](#)

Lit is an internet native security and identity network. With Lit, you can create and control keys and run code for virtually any application that can be made private, immutable, and / or autonomous.

---

# MEV on Solana

## Introduction

Maximal (formerly Miner) Extractable Value is the value that can be extracted from a blockchain by any agent without special permissions.

Considering this permissionless nature, any agent with transaction ordering rights will be in a privileged position to perform the extraction.

There are features of public blockchains that allow front running

1. All transactions are available in a public mempool before they are mined
2. All transaction data is public
3. Transactions can be cloned

For Solana however we don't have a conventional mempool as commonly found on other chains such as Ethereum.

## Main Points

- Validators hold power in deciding transaction order and inclusion.

- This power can be leveraged for personal gain through transaction ordering or selling the right to order transactions.
- This creates vulnerabilities in the consensus layer because incentives are misaligned with the core protocol.
- Ethereum is susceptible to MEV due to its public mempool, public transaction data, transaction cloning, slow block time, and complex DeFi ecosystem.
- MEV can also exist on other blockchains such as Bitcoin, but is generally proportional to the blockchain's complexity of application-layer behaviour.

For reference, [~\\$24 million in profits](#) (according to EigenPhi) have been made from sandwiching alone on Ethereum over the past 30 days. When users set a maximum slippage (the amount of variation around a certain value that the user agrees to before sending the transaction), they were almost always getting filled at that price. In other words, a user's expected slippage was almost always equal to their max slippage, should the order become filled.

From [article](#)

- **NFT Minting:** MEV from NFT minting occurs when participants attempt to secure rare or valuable non-fungible tokens (NFTs) during public minting events (both for “blue-chip” and long-tail NFTs). The nature of NFT minting events is characterized by sudden increases, with block  $x-1$  having no NFT MEV opportunity and block  $x$  having a large MEV opportunity. Block  $x$  here refers to the block when the mint went live.

These NFT mint/IDO mechanics were one of the first sources of large-scale congestion spikes that caused Solana to temporarily halt block production in 2021/2022.

- **Liquidations:** When borrowers fail to maintain the required [collateralization](#) ratio for their loans, their positions become eligible for liquidation.

Searchers scan the blockchain for such undercollateralized positions and execute liquidations to repay part or all of the debt,

receiving a portion of the collateral as a reward. Liquidations occur in protocols that utilize both tokens and NFTs as collateral. Liquidations are necessary for protocols to stay solvent and are beneficial for the broader ecosystem.

- **Arbitrage:** Arbitrage involves taking advantage of price discrepancies of the same asset across different markets or platforms.

These arbitrage opportunities exist intra-chain, inter-chain, and between CEXes and DEXes. Intra-chain arbitrage is currently the only form of arbitrage that guarantees atomicity as both legs execute on the same chain, as intra-chain arbitrage requires additional trust assumptions.

## Shreds

Shreds are fractions of blocks that are made by a block producer on Solana.

Shreds are constantly being emitted as a validator produces a block to create a block stream.

Other nodes on the network receive these

shreds and replay them on-the-fly, meaning state is always changing on Solana as opposed to Ethereum where state is paused intra-slot.

Jito MEV Auction

See [introduction](#)

On January 10, 2024, a searcher tipped a validator 890 SOL, one of the largest tips in Jito's history:

Bundle Details	
Bundle Id	7d2f02a0542fd3950d90c9bd8ca84d233e28f0298d9f002c7e3cc0959b72b24f
Timestamp	61 days ago ( 2024-01-10T22:20:40+00:00)
Slot	241056629
Landed	Yes
Validator Tip	890.42 SOL
Signer	2Fast1RAxFtcXipCucpJhFQACiksuySKa8E9S9ckwVYY
Validator Identity	Fd7btgySsrjuo25CJCj7oE7VPMyezDhnx7pZkj2v69Nk
Included Transactions	3G8jGDM8y828S53poCfov5PnMV4VAxAF6u6hifddToZyxQC8PihykzxFSLAkk1aHSrD5KLiKmeoaJzarRfocVz6C 5qYuZ9ZLSHLB1MuV83xHRcTgVA9A5pUajjnQUUcPbk3bf 4tiBb51sbUsQshoNU8VbzaiiXxnBucJ6po9tY3BQ2VvVGPSNTAgAXuXSdRSC8z2J9rHtjbrS1174ghZf96p1pqde 2Fast1RAxFtcXipCucpJhFQACiksuySKa8E9S9ckwVYY (tipper)

The week up to March 12, 2024, saw Solana validators earn in excess of \$7 million in Jito tips for blockspace inclusion on Solana. Today, over 50% of Solana transactions are on failed arbitrages.

[Jito](#) is an out-of-protocol blockspace auction for partial blocks, unlike MEV-boost where full blocks are built (Jito and mev-geth are similar in spirit but greatly differ in implementation).

Jito offers off-chain inclusion guarantees of a specific set of transactions called [bundles](#).

Bundles are executed sequentially and atomically – all-or-nothing.

Searchers submit bundles with guaranteed on-chain execution should they win the auction and pay the minimum tip of 10,000 lamports.

Jito tips exist out-of-protocol and are separate from in-protocol [priority fees](#).

Jito runs multiple Block Engines around the world that receive shreds directly from leaders running Jito-Solana. It also receives retransmitted shreds from these leaders.

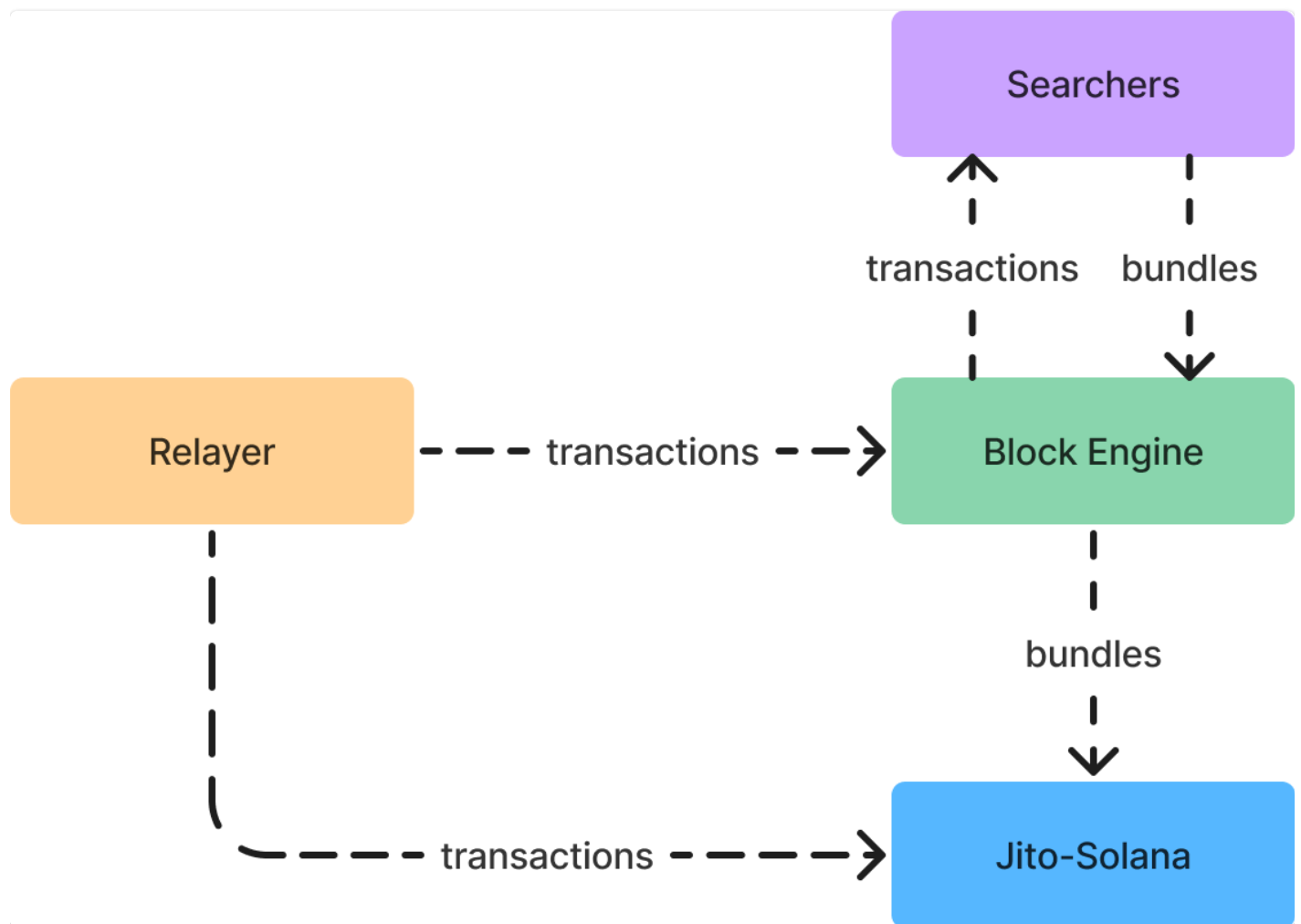
The block engines verify these shreds before distributing them to hundreds of servers around the world.

This approach aims to reduce [spam](#) and enhance the efficiency of Solana's compute resources by running the auctions off-chain,



only posting the single winner of the auction into the block via a guaranteed bundle. A searcher may use a bundle for one or both of the following properties:

- fast, guaranteed inclusion and
  - bidding for frontrun/backrun opportunities.
- This is particularly important considering that a significant portion of the network's compute resources are currently consumed by unsuccessful transactions.



Jito Foundation has several overarching priorities for the evolution of MEV on Solana:

- Minimize negative externalities: Solana has experienced network congestion from disorganized MEV trading.

Network performance needs to improve so users have a good experience. If not, users may be less active on Solana.

Jito-Solana's auction mechanism is designed to ensure the network's reliability.

- Prevent centralisation

MEV tends to encourage centralisation over time as there are economies of scale.

The best defense against this is

- transparency,
- low barriers to entry and
- healthy competition.

In addition to open source tools, Jito will be publishing dashboards and other data to provide transparency into the MEV market.

#### Reward Distribution

MEV can be highly profitable.

The Foundation's work enables transparent distribution of a portion of this revenue across the ecosystem.

MEV does not exist without regular users nor a successful validator community.

These parties should get some share of profits for the network to maintain a healthy balance.

Also see [Jito MEV Auction explanation](#)

## Components

### Relayers

Relayers on Solana are different from Ethereum. On Solana, relayers are responsible for relaying incoming transactions, performing limited TPU operations such as packet deduplication and signature verification. Relayers forward packets to both the Block Engine and the validator.

The equivalent is not necessary on Ethereum because Ethereum has a mempool, while Solana does not.

### Block Engine

The Block Engine simulates transaction combinations and runs the off-chain blockspace auction. MEV-maximizing bundles are then forwarded to the leader running the Jito-Solana client.

## Searchers

Searchers seek to exploit price discrepancies and other opportunities by inserting their own transactions into a given block. They can utilise sources such as Jito's [ShredStream](#) or source their own up-to-date information.

## Validators

Validators build and produce blocks. Jito-Solana blocks are built with the scheduler reserving 3M CUs for transactions routed via Jito for the first 80% of the block.

## Shredstream

Shredstream provides the lowest latency shreds to any RPC or validator on the network. This service can save hundreds of milliseconds when trading on Solana and provides a redundant shred path for servers in remote locations.

---

## Commitment Levels

See [article](#) from Omnia

### Transaction lifecycle

#### Client side

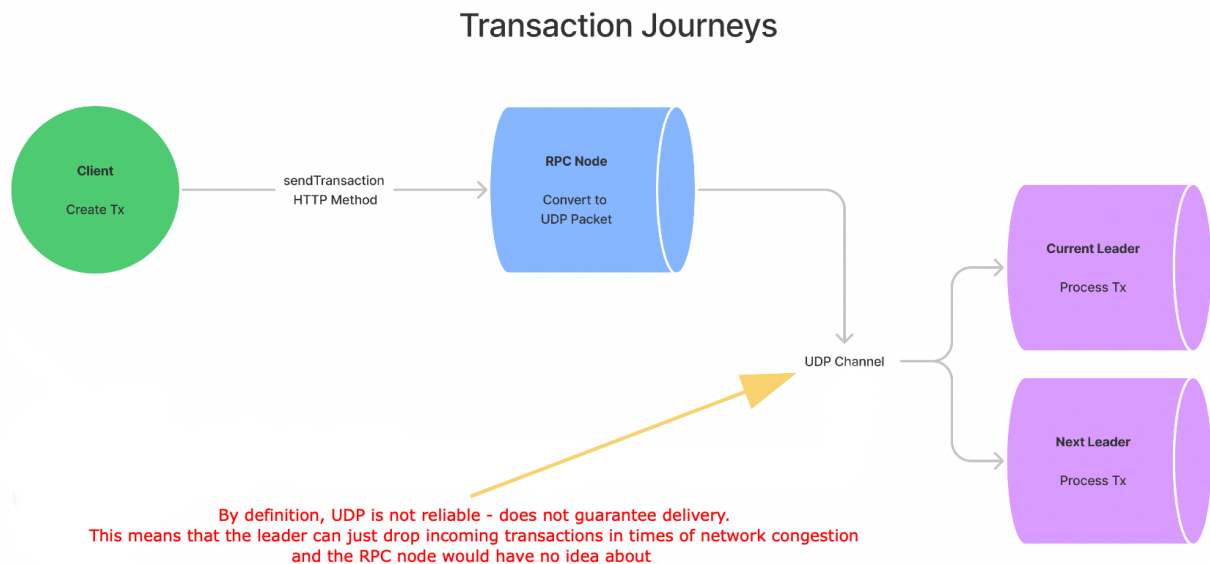
1. Client creates a list of instructions along with the list of accounts that instructions need to read and write
2. Fetch a recent blockhash and use it to prepare a transaction message.  
Here we also chose the commitment level.
3. Simulate the transaction (client side, i.e. within web3 library) to ensure it behaves as expected
4. Prompt user to sign the prepared transaction message with their private key

#### Network side

1. Send the transaction to an RPC node which attempts to forward it to the current block producer
2. Hope that a block producer does not drop the transactions but also that validates and commits the transaction into their produced block

### 3. Confirm the transaction has either been included in a block or detect when it has expired

#### Diagram from Omnia



### Transaction Expiration

By default, all Solana transactions will expire if not committed to a block (referenced via blockhash used in step 2 from above transaction lifecycle) in a certain amount of time.

Transactions expire to help validators avoid processing the same transaction twice.

When fetching a recent blockhash the client chooses the commitment level desired.

```
getLatestBlockhash(commitmentOrConfig?): Promise<Readonly<{  
    blockhash: string;  
    lastValidBlockHeight: number;  
}>>
```

```
Commitment: "processed" | "confirmed" | "finalized" | "recent" | "single" | "singleGossip" | "root" | "max"
```

### Commitment levels

The commitment level measures the network confirmation for a specific block.

There are three specific commitment statuses:

**finalised** – This is the highest level of commitment and transactions are irreversible. The RPC node will query the most recent block confirmed by the supermajority of the cluster as having reached maximum lockout, meaning the cluster has recognised this block as finalised.

**confirmed** – This commitment level is usually used for transactions that have been included in a block, but have not yet been confirmed by the network. The node will query the most recent block that has been voted on by the supermajority of the cluster (optimistic confirmation)

**processed** – Here, the node will query its most recent block but the block may still be skipped by the cluster if the transaction fails. This is the least secure of all the three levels and transactions in this level are usually used for testing and development purposes.

The official Solana recommendation is to use **confirmed** commitment when fetching recent blockhash.

This will provide a better (extended with aprox. 13s) time window as compared to **finalized** commitment, thus reducing the risk of transaction expiration.