YYSFold System Walkthrough

A Complete Guide to the Dashboard and Pipeline

YYSFold Team

December 5, 2025

Abstract

This document provides a comprehensive walkthrough of the YYSFold blockchain fingerprinting system. It explains what you see on the live dashboard, how each component works, and the complete data pipeline from raw blockchain data to behavioral fingerprints.

Contents

1 What Is YYSFold?

YYSFold is a **blockchain behavioral fingerprinting system**. It watches blockchain activity in real-time, compresses each block into a compact "fingerprint," identifies unusual patterns (hot-zones), and can predict what the next block might look like—all while generating cryptographic proofs that the analysis was done correctly.

Live Demo

URL: `https://yysfold.ngrok.io`

2 Dashboard Overview

This section walks through each element of the dashboard from top to bottom.

2.1 Header & Navigation

The header contains:

- **YYSFold logo** — Links to home
- **Atlas** — 2D/3D visualization of the vector space
- **Chat** — AI assistant for querying block data

2.2 Latest Block Fingerprint Card

Latest Behavioral Fingerprint

Chain & Block:	sol · #383106448
Timestamp:	Nov 28, 08:33
Status:	[Live] Updated 7s ago

Tags: ASSET SOLANA PROGRAM HIGH THROUGHPUT MIXED ACTIVITY BRIDGE ACTIVITY

Block hash:	EyU8uUGPYrKC8NZk...
Artifacts:	[raw block] [proof]
Regime:	Bridge Activity + Lending Activity
Anomaly:	0.37 (Typical)
Metrics:	Density: 1058183.24 · PQ: 6.279 · Tags: 5

2.2.1 Anomaly Labels

2.3 Predicted Next Block Card

Predicted Next Block

Chain:	SOL
Confidence:	24%
Predicted Tags:	MIXED ACTIVITY
ETA:	~10s · 9:10:41 AM
Reason:	Steady state

Table 1: Latest Block Card Elements

Element	Description
sol · #383106448	Chain (Solana) and block height
[Live]	Real-time SSE connection active
Updated 7s	Time since last heartbeat
Tags (pills)	Semantic behaviors detected in this block
Block hash	Truncated block identifier
[raw block]	Link to download the raw block JSON
[proof]	Link to the ZK proof artifact
Regime	Dominant behavioral pattern combination
Anomaly: 0.37	Score from 0–1 (higher = more unusual)
Density	Peak hotzone density (KDE value)
PQ	Average PQ reconstruction error

Score Range	Label	Meaning
< 0.45	Typical	Normal activity
$0.45 - 0.75$	Unusual	Notable patterns
> 0.75	Rare	Potential anomaly

How predictions work:

1. Keep a rolling history of recent mempool snapshots
2. Compute trends: gas slope, transaction delta, tag frequency changes
3. If trends are stable → MIXED_ACTIVITY with low confidence
4. If gas spiking + DEX activity → HIGH_FEE + DEX_ACTIVITY with higher confidence

2.4 Mempool Ticker (Live Feed)

Live Mempool

```
08:33:12 ETH ^ High 142 txs | 45 gwei | 2.1 ETH
                      DEX surge, gas climbing

08:33:08 SOL * Normal 89 txs | 0.00025 SOL
                      Steady state

08:33:02 AVAX v Low 23 txs | 28 nAVAX
                      Light activity
```

- **Timestamp** — When the snapshot was taken
- **Chain** — Which blockchain
- **Pressure indicator** — △ High / • Normal / ▽ Low
- **Stats** — Pending transaction count, gas price, total value
- **Highlights** — Notable patterns detected

Important

This is **NOT** showing confirmed blocks—it's showing the **mempool** (pending transactions waiting to be included).

2.5 Block Detail Page

When you click on a block, you see:

- **Hotzones** — Up to 16 density clusters, each with tags
- **Hypergraph Visualization** — 3D force-directed graph of hotzone relationships
- **Metrics** — Transaction count, folded vectors, PQ residuals, commitment hash

Hotzones are regions of high-density activity in the compressed vector space. Each hotzone represents a cluster of similar transaction behaviors.

2.6 Atlas Page

The Atlas shows **all hotzones from recent blocks** projected into 2D or 3D space.

- **Position** — 2D/3D coordinates from dimensionality reduction
- **Size** — Proportional to density
- **Color** — Mapped from dominant semantic tag
- **Edges** — Hyperedges connecting related hotzones

This visualization lets you see:

- Clusters of similar behavioral patterns
- How current activity compares to historical norms
- Outliers that don't fit normal patterns

3 The Pipeline

This section explains what happens behind the scenes when a new block arrives.

3.1 Step 1: Block Ingestion

```
RPC Node --> watchIngest.js --> Raw Block JSON
```

The **ingestion worker** polls blockchain RPC endpoints every few seconds:

- **Ethereum** — via ETH_RPC_URLS
- **Solana** — via Solana RPC
- **Avalanche** — via AVAX RPC

For each new block, it downloads:

- Block header (height, timestamp, hash)
- All transactions
- Execution traces (if available)
- State changes

3.2 Step 2: Vectorization

```
Raw Block --> vectorizeBlock() --> Vectors
```

Each transaction becomes a **16-dimensional vector**:

```
Transaction {amount, fee, gas, ...}
    |
    v
Vector [0.72, 0.15, 0.33, 0.5, 0.001, ...]
```

What gets encoded:

- Normalized amounts (0–1 scale)
- Fee levels
- Gas usage
- Contract type (hashed to bucket)
- Asset/token (hashed)
- Sender/receiver patterns
- Timestamp position in day

3.3 Step 3: Folding

```
[1000+ vectors] --> foldVectors() --> [6 vectors]
```

We compress all transaction vectors into just **6 summary vectors**:

1. **Mean vector** — Average of all transactions
2. **Variance vector** — How spread out the values are
3. **Components 1–4** — Principal directions of variation

This reduces a block with 1000 transactions from ∼16,000 numbers to just 96 numbers.

3.4 Step 4: Product Quantization (PQ)

```
[6 vectors] --> pqEncode() --> [6 codes] + residuals
```

Each folded vector is further compressed using a **codebook**:

```
Vector [0.72, 0.15, 0.33, 0.5]
    |
    v
Split into 4 subvectors
    |
    v
Find nearest centroid for each
    |
    v
Code: [142, 87, 203, 56] (4 bytes instead of 16 floats)
```

The **residual** is how much error this compression introduces. We track this to ensure quality.

3.5 Step 5: Hotzone Detection (KDE)

```
PQ codes --> pqDecode() --> Vectors --> KDE --> Hotzones
```

We reconstruct the vectors and run **Kernel Density Estimation**:

```
For each vector:
    density = sum of Gaussian kernels from all other vectors

If density > threshold:
    This is a HOTZONE (cluster of similar activity)
```

Hotzones get **semantic tags** based on which vector components are high:

- High component 0 → HIGH_VALUE
- High component 2 → DEX_ACTIVITY
- High component 7 → AML_ALERT

3.6 Step 6: Hypergraph Construction

```
Hotzones --> buildHypergraph() --> Graph
```

We connect hotzones that are:

- Close in vector space (small distance)
- Both high density

This creates a **graph structure** showing relationships between behavioral clusters.

3.7 Step 7: Anomaly Scoring

```
(density, residuals, tags) --> computeAnomalyScore() --> 0.37
```

The anomaly score combines three signals:

Component	**Weight**	**What it measures**
Density	50%	How concentrated activity is
Residuals	35%	How well PQ compression fits
Tags	15%	Presence of suspicious tags

3.8 Step 8: Cryptographic Commitment

```
Fingerprint --> BLAKE3 hash --> Commitment
```

We hash everything to create tamper-proof commitments:

- **Folded commitment** — Hash of the 6 folded vectors
- **PQ commitment** — Hash of the PQ codes
- **Codebook root** — Hash of the compression codebook

These go into the ZK proof.

3.9 Step 9: ZK Proof Generation

```
Witness + Public Inputs --> Halo2 Prover --> Proof
```

If Halo2 is configured, we generate a zero-knowledge proof that:

1. The vectorization was done correctly
2. The folding math is right
3. The PQ encoding respects the error bound
4. The commitments match

Current Status

Currently: Running mock proofs (placeholder). Real proofs require compiled Halo2 binaries.

3.10 Step 10: Storage & Display

```
Summary + Proof --> SQLite + JSON files --> Dashboard
```

Everything is saved to:

- **SQLite database** — For queries (by chain, tag, time)
- **JSON files** — Raw artifacts in `/artifacts/`

The dashboard reads from these to display the fingerprints.

4 Services Architecture

4.1 Local Development (via ngrok)

Service	Port	Purpose
Next.js Dashboard	3000	Web UI
ngrok tunnel	—	Public URL (yysfold.ngrok.io)
Ingest Worker	—	Background: fetches blocks
Mempool Worker	—	Background: watches pending txs

4.2 Cloud Deployment (Render + Vercel)

Service	Platform	Purpose
API	Render	Serves data to frontend
Ingest Worker	Render	Background block ingestion
Mempool Worker	Render	Background mempool watching
Dashboard	Vercel	Static Next.js frontend

File	Purpose
`scripts/ingestBlocks.ts`	Main ingestion pipeline
`scripts/mempoolWatch.ts`	Mempool polling + predictions
`folding/vectorize.ts`	Transaction → Vector
`folding/fold.ts`	Vectors → Folded Block
`folding/pq.ts`	Product Quantization
`analytics/hotzones.ts`	KDE + hotzone detection
`analytics/hypergraph.ts`	Graph construction
`zk/halo2Backend.ts`	ZK proof interface
`dashboard/app/page.tsx`	Main dashboard page

5 Key Files Reference

6 Quick Explanation Script

For Verbal Presentations

"YYSFold takes raw blockchain blocks and compresses them into behavioral fingerprints. Each block with thousands of transactions gets reduced to just 6 summary vectors using statistical folding and product quantization.
We then run kernel density estimation to find 'hotzones'—clusters of similar transaction patterns—and tag them semantically (DEX activity, bridge transfers, potential AML alerts, etc.).
The anomaly score tells us if a block looks normal or unusual compared to baseline. The mempool feed shows pending transactions in real-time, and we use trend analysis to predict what the next block might look like.
Everything is cryptographically committed with BLAKE3 hashes, and we can optionally generate zero-knowledge proofs that the analysis was done correctly—so you can verify the fingerprint without re-running all the math yourself."

7 Glossary

Term	Definition
Fingerprint	Compact representation of a block's behavior
Folding	Compressing many vectors into few summary vectors
PQ	Product Quantization—vector compression using a codebook
KDE	Kernel Density Estimation—finding dense regions
Hotzone	A cluster of similar transactions
Hypergraph	Graph where edges can connect 2+ nodes
Anomaly Score	0–1 measure of how unusual a block is
Mempool	Pool of pending (unconfirmed) transactions
ZK Proof	Cryptographic proof of correct computation
Commitment	Hash that binds data without revealing it
SSE	Server-Sent Events—real-time data streaming
RPC	Remote Procedure Call—API to blockchain nodes