YYSFold: Mathematical Foundations

Blockchain Fingerprinting via Vector Folding,
Product Quantization, and Zero-Knowledge Proofs

YYSFold Team

December 5, 2025

Abstract

This document provides a comprehensive mathematical treatment of the YYSFold blockchain fingerprinting system. We detail the transformation pipeline from raw blockchain data through vectorization, folding, and product quantization to produce compact behavioral fingerprints. We then describe the kernel density estimation approach for hotzone detection, hypergraph construction for structural analysis, and the anomaly scoring framework. Finally, we specify the zero-knowledge proof system that provides cryptographic verifiability of the entire pipeline.

Contents

1 Introduction and System Overview

YYSFold is a blockchain fingerprinting system that compresses high-dimensional transaction data into compact, semantically meaningful representations while preserving cryptographic verifiability through zero-knowledge proofs.

1.1 Pipeline Architecture

The system processes blockchain data through the following stages:

1. **Vectorization**: Raw transactions, execution traces, and witness data are mapped to fixed-dimensional vectors.
2. **Folding**: Block-level vectors are aggregated into a compact representation via statistical summarization and linear projection.
3. **Product Quantization**: Folded vectors are compressed using a learned codebook with bounded reconstruction error.
4. **Density Analysis**: Kernel density estimation identifies regions of concentrated activity (hotzones).
5. **Hypergraph Construction**: Relationships between hotzones form a structural signature.
6. **Anomaly Scoring**: A composite score flags unusual behavioral patterns.
7. **Cryptographic Commitment**: BLAKE3 hashes bind all transformations.
8. **Zero-Knowledge Proof**: A Halo2-based proof attests to correct computation.

Definition 1.1 (Behavioral Fingerprint)**.** A *behavioral fingerprint* of a blockchain block B is the tuple:

$$\mathcal{F}(B) = (F_B, \mathcal{C}, \mathcal{H}, \mathcal{G}, S, \pi) \tag{1}$$

where F_B is the folded block, $\mathcal{C}$ is the PQ code, $\mathcal{H}$ is the hotzone set, $\mathcal{G}$ is the hypergraph, S is the anomaly score, and π is the zero-knowledge proof.

2 Vectorization Layer

2.1 Transaction Vectorization

Each transaction tx_i in a block is mapped to a fixed-dimensional vector $\mathbf{x}_i \in \mathbb{R}^{d_{tx}}$ where $d_{tx} = 16$.

Definition 2.1 (Transaction Vector Map)**.** The vectorization function $\phi_{tx} : \text{Tx} \to \mathbb{R}^{16}$ is defined as:

$$\mathbf{x}_i = \phi_{tx}(tx_i) = \begin{pmatrix} \nu(\text{amount}, M_{\text{amt}}) \\ \nu(\text{fee}, M_{\text{fee}}) \\ \nu(\text{gasUsed}, M_{\text{gas}}) \\ \nu_{\text{idx}}(\text{index}, n) \\ \nu(\text{height}, 10^7) \\ \tau(\text{timestamp}) \\ h(\text{type}, 64) \\ h(\text{asset}, 256) \\ \nu(\text{nonce}, 10^6) \\ \mathbb{1}[\text{status} = \text{success}] \\ \nu(\text{slot}, 10^5) \\ \nu(\text{chainId}, 10^3) \\ \nu(\text{priorityFee}, M_{\text{fee}}) \\ h(\text{selector}, 1024) \\ h(\text{sender}, 10000) \\ h(\text{receiver}, 10000) \end{pmatrix} \tag{2}$$

2.2 Normalization Functions

Definition 2.2 (Linear Normalization)**.** For a value v with maximum M:

$$\nu(v, M) = \text{clamp}\left(\frac{v}{M}, -1, 1\right) \tag{3}$$

where $\text{clamp}(x, a, b) = \min(\max(x, a), b)$.

Definition 2.3 (Index Normalization)**.** For an index i in a collection of size n:

$$\nu_{\text{idx}}(i, n) = \begin{cases} 0 & \text{if } n \leq 1 \\ \frac{i}{n-1} & \text{otherwise} \end{cases} \tag{4}$$

Definition 2.4 (Cyclic Timestamp Normalization)**.** For timestamp t with scale $s = 86400$ (seconds per day):

$$\tau(t) = \frac{t \mod (10s)}{10s} \tag{5}$$

Definition 2.5 (Hash Bucketing)**.** For a value v and bucket count B:

$$h(v, B) = \frac{\text{hash}(v) \mod B}{B - 1} \tag{6}$$

where the hash function is the polynomial rolling hash:

$$\text{hash}(s) = \sum_{i=0}^{|s|-1} s[i] \cdot 31^i \mod 2^{32} \tag{7}$$

2.3 State and Witness Vectors

Definition 2.6 (State Vector)**.** Each execution trace trace_j maps to $\mathbf{s}_j \in \mathbb{R}^{12}$:

$$\mathbf{s}_j = \phi_{\text{state}}(\text{trace}_j) \tag{8}$$

with components for balance delta, storage operations, log events, contract hash, gas consumed, etc.

Definition 2.7 (Witness Vector)**.** Each witness bundle bundle_k maps to $\mathbf{w}_k \in \mathbb{R}^8$:

$$\mathbf{w}_k = \phi_{\text{witness}}(\text{bundle}_k) \tag{9}$$

encoding constraint count, degree, gate count, and prover metadata.

3 Folding Operator

3.1 Vector Collection

Given a block B, we collect all vectors:

$$\mathcal{V} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{n_{tx}}\} \cup \{\mathbf{s}_1, \ldots, \mathbf{s}_{n_s}\} \cup \{\mathbf{w}_1, \ldots, \mathbf{w}_{n_w}\} \tag{10}$$

Each vector is resized to canonical dimension $d = 16$:

$$\tilde{\mathbf{v}}_i = \text{resize}(\mathbf{v}_i, d) \in \mathbb{R}^d \tag{11}$$

3.2 Statistical Aggregation

Definition 3.1 (Block Mean Vector)**.**

$$\boldsymbol{\mu} = \frac{1}{|\mathcal{V}|} \sum_{i=1}^{|\mathcal{V}|} \tilde{\mathbf{v}}_i \tag{12}$$

Definition 3.2 (Block Standard Deviation Vector)**.** For each component $j \in [d]$:

$$\sigma_j = \sqrt{\frac{1}{|\mathcal{V}| - 1} \sum_{i=1}^{|\mathcal{V}|} (\tilde{v}_{i,j} - \mu_j)^2} \tag{13}$$

3.3 Component Projection

Let $\mathbf{W} \in \mathbb{R}^{k \times d}$ be a fixed projection matrix with $k = 4$ rows (principal directions).

Definition 3.3 (Component Vector)**.** For row m of $\mathbf{W}$:

$$\mathbf{c}_m = \frac{1}{|\mathcal{V}|} \sum_{i=1}^{|\mathcal{V}|} \tilde{\mathbf{v}}_i \odot \mathbf{w}_m \tag{14}$$

where $\odot$ denotes element-wise multiplication.

Definition 3.4 (Normalized Component)**.**

$$\hat{\mathbf{c}}_m = \frac{\mathbf{c}_m}{\|\mathbf{c}_m\|_2 + \epsilon}, \quad \epsilon = 10^{-6} \tag{15}$$

3.4 Folded Block

Definition 3.5 (Folded Block)**.** The folded block is the set of $k + 2 = 6$ vectors:

$$F_B = \{\boldsymbol{\mu}, \boldsymbol{\sigma}, \hat{\mathbf{c}}_1, \hat{\mathbf{c}}_2, \hat{\mathbf{c}}_3, \hat{\mathbf{c}}_4\} \tag{16}$$

Remark 3.1. The folding operation reduces a block with n transactions from $O(n \cdot d_{tx})$ to a fixed $6 \cdot 16 = 96$ scalars, independent of block size.

4 Product Quantization

4.1 Codebook Structure

Definition 4.1 (PQ Codebook)**.** A Product Quantization codebook $\mathcal{C}$ partitions $\mathbb{R}^d$ into m subspaces, each with K centroids:

$$\mathcal{C} = \{C^{(1)}, C^{(2)}, \ldots, C^{(m)}\} \tag{17}$$

where $C^{(s)} = \{\mathbf{c}_1^{(s)}, \ldots, \mathbf{c}_K^{(s)}\} \subset \mathbb{R}^{d/m}$ for each subspace s.

Default Parameters:

- $m = 4$ subspaces
- $K = 256$ centroids per subspace
- $d/m = 4$ subvector dimension

4.2 Encoding Process

Algorithm 1 PQ Encoding

```
Require: Folded vector $\mathbf{f} \in \mathbb{R}^d$, Codebook $\mathcal{C}$
Ensure: PQ code $(q_1, \ldots, q_m) \in [K]^m$
1: Split $\mathbf{f} = [\mathbf{f}^{(1)}|\mathbf{f}^{(2)}|\cdots|\mathbf{f}^{(m)}]$
2: for $s = 1$ to $m$ do
3:     $q_s \leftarrow \arg\min_{k \in [K]} \left\|\mathbf{f}^{(s)} - \mathbf{c}_k^{(s)}\right\|_2$
4: end for
5: return $(q_1, \ldots, q_m)$
```

4.3 Reconstruction

Definition 4.2 (PQ Reconstruction)**.** Given code $(q_1, \ldots, q_m)$:

$$\hat{\mathbf{f}} = [\mathbf{c}_{q_1}^{(1)}|\mathbf{c}_{q_2}^{(2)}|\cdots|\mathbf{c}_{q_m}^{(m)}] \tag{18}$$

4.4 Residual Analysis

Definition 4.3 (Reconstruction Residual)**.**

$$r(\mathbf{f}) = \left\|\mathbf{f} - \hat{\mathbf{f}}\right\|_2 \tag{19}$$

Definition 4.4 (Error Bound Constraint)**.** The encoding enforces:

$$r(\mathbf{f}) \leq \varepsilon, \quad \varepsilon = 0.25 \text{ (default)} \tag{20}$$

Definition 4.5 (Residual Statistics)**.** For a block with residuals $\{r_1, \ldots, r_n\}$:

$$\bar{r} = \frac{1}{n}\sum_{i=1}^{n} r_i \qquad \text{(average)} \tag{21}$$

$$r_{\max} = \max_i r_i \qquad \text{(maximum)} \tag{22}$$

$$r_{95} = \text{percentile}(\{r_i\}, 0.95) \qquad \text{(95th percentile)} \tag{23}$$

5 Kernel Density Estimation and Hotzones

5.1 Gaussian Kernel

Definition 5.1 (Gaussian Kernel)**.** The radial basis function kernel with bandwidth h:

$$K(\mathbf{x}, \mathbf{y}; h) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|_2^2}{2h^2}\right) \tag{24}$$

Default bandwidth: $h = 0.15$.

5.2 Kernel Density Estimator

Definition 5.2 (KDE)**.** For decoded PQ vectors $\{\hat{\mathbf{f}}_1, \dots, \hat{\mathbf{f}}_n\}$, the density at point $\mathbf{x}$:

$$\hat{p}(\mathbf{x}) = \frac{1}{n \cdot (h\sqrt{2\pi})^d} \sum_{i=1}^{n} K(\mathbf{x}, \hat{\mathbf{f}}_i; h) \tag{25}$$

5.3 Hotzone Detection Algorithm

Algorithm 2 Hotzone Detection

```
Require: Decoded vectors $\{\hat{\mathbf{f}}_i\}$, threshold $\theta$, max zones $k$
Ensure: Hotzone set $\mathcal{H}$
1: for $i = 1$ to $n$ do
2:     $\rho_i \leftarrow \hat{p}(\hat{\mathbf{f}}_i)$                                  ▷ Compute density
3: end for
4: $\mathcal{H}_{\text{cand}} \leftarrow \{i : \rho_i \geq \theta\}$                     ▷ Threshold filter
5: $\mathcal{H} \leftarrow \text{top}_k(\mathcal{H}_{\text{cand}}, \rho)$                 ▷ Select top-$k$
6: for $j \in \mathcal{H}$ do
7:     $\text{tags}_j \leftarrow \text{deriveTags}(\hat{\mathbf{f}}_j)$
8:     Output hotzone $(\hat{\mathbf{f}}_j, \rho_j, r = 2h, \text{tags}_j)$
9: end for
```

5.4 Semantic Tag Derivation

Tags are derived from vector component thresholds:

6 Hypergraph Construction

6.1 Pairwise Weight Function

Definition 6.1 (Hotzone Weight)**.** For hotzones $H_a = (\mathbf{c}_a, \rho_a, r_a)$ and $H_b = (\mathbf{c}_b, \rho_b, r_b)$:

$$w(H_a, H_b) = \max(0, r_a + r_b - d_{ab}) \cdot \sqrt{\rho_a \cdot \rho_b} \tag{26}$$

where $d_{ab} = \|\mathbf{c}_a - \mathbf{c}_b\|_2$ is the center distance.

Index	Condition	Tag
0	> 0.65	HIGH_VALUE
1	> 0.55	FEE_INTENSIVE
2	> 0.55	DEX_ACTIVITY
3	> 0.55	NFT_ACTIVITY
4	> 0.45	BRIDGE_ACTIVITY
5	> 0.50	TIME_CLUSTER
6	> 0.50	LENDING_ACTIVITY
7	> 0.50	AML_ALERT
8	> 0.55	MEV_ACTIVITY
9	< -0.45	VOLATILITY_CRUSH

Table 1: Semantic tag derivation thresholds

6.2 Hyperedge Formation

Definition 6.2 (2-Edges).

$$E_2 = \{(i,j) : w(H_i, H_j) \geq \theta_2\}, \quad \theta_2 = 5 \times 10^{-5} \tag{27}$$

Definition 6.3 (3-Edges).

$$E_3 = \{(i,j,k) : \bar{w}_{ijk} \geq 1.5\theta_2\} \tag{28}$$

where:

$$\bar{w}_{ijk} = \frac{w(H_i, H_j) + w(H_j, H_k) + w(H_i, H_k)}{3} \tag{29}$$

Definition 6.4 (4-Edges).

$$E_4 = \{(a,b,c,d) : \bar{w}_{abcd} \geq 2\theta_2\} \tag{30}$$

where:

$$\bar{w}_{abcd} = \frac{1}{6} \sum_{\{p,q\} \subseteq \{a,b,c,d\}} w(H_p, H_q) \tag{31}$$

Definition 6.5 (Block Hypergraph).

$$\mathcal{G} = (\mathcal{H}, E_2 \cup E_3 \cup E_4) \tag{32}$$

7 Anomaly Scoring

7.1 Component Scores

Definition 7.1 (Density Component).

$$S_{\text{density}} = 1 - \min\left(1, \frac{\max_j \rho_j}{\rho_{\text{baseline}}}\right) \tag{33}$$

where $\rho_{\text{baseline}} = 1.1 \times 10^6$.

Definition 7.2 (Residual Component).

$$S_{\text{residual}} = \min\left(1, \frac{r_{95}}{0.5}\right) \tag{34}$$

Definition 7.3 (Tag Component).

$$S_{\text{tags}} = \min\left(1, \frac{\sum_{t \in \text{tags}} \pi(t)}{3}\right) \tag{35}$$

where $\pi(t)$ is the prior weight for tag t.

Tag	**Prior** $\pi(t)$
AML_ALERT	0.8
AML_RULE	0.7
HIGH_FEE	0.6
VOL_CRUSH	0.5
BRIDGE_ACTIVITY	0.45
LENDING_ACTIVITY	0.4
NFT_ACTIVITY	0.35
DEX_ACTIVITY	0.3

Table 2: Tag prior weights

7.2 Composite Anomaly Score

Theorem 7.1 (Anomaly Score)**.** *The final anomaly score is:*

$$S = 0.50 \cdot S_{density} + 0.35 \cdot S_{residual} + 0.15 \cdot S_{tags} \tag{36}$$

with $S \in [0, 1]$ *(clamped).*

Label Assignment:

$$\text{Label}(S) = \begin{cases} \text{Rare} & S \geq 0.75 \\ \text{Unusual} & 0.45 \leq S < 0.75 \\ \text{Typical} & S < 0.45 \end{cases} \tag{37}$$

8 Cryptographic Commitments

8.1 BLAKE3 Hash

All commitments use the BLAKE3 cryptographic hash function:

$$H : \{0,1\}^* \to \{0,1\}^{256} \tag{38}$$

8.2 Commitment Definitions

Definition 8.1 (Folded Block Commitment)**.**

$$C_{\text{fold}} = H(\text{JSON}(F_B)) \tag{39}$$

Definition 8.2 (PQ Code Commitment)**.**

$$C_{\text{PQ}} = H(\text{JSON}(\{(q_1^{(i)}, \ldots, q_m^{(i)})\}_{i=1}^{|F_B|})) \tag{40}$$

Definition 8.3 (Codebook Root)**.**

$$C_{\text{codebook}} = H(\text{JSON}(\mathcal{C}, \text{normalization}, \varepsilon)) \tag{41}$$

9 Zero-Knowledge Proof System

9.1 Public Inputs

9.2 Witness Structure

The ZK witness contains:

Signal	**Description**
prev_state_root	State root before block execution
new_state_root	State root after block execution
block_height	Block number
tx_merkle_root	Merkle root of transactions
C_{fold}	Folded block commitment
C_{PQ}	PQ code commitment
C_{codebook}	Codebook root

Table 3: ZK public inputs

- Full transaction list $\{tx_i\}$
- Execution traces $\{\text{trace}_j\}$
- Witness bundles $\{\text{bundle}_k\}$
- Intermediate folded vectors F_B
- PQ indices before hashing

9.3 Constraint System

Definition 9.1 (State Transition Gadget).

$$\text{Apply}(\text{prev\_state\_root}, \{tx_i\}) = \text{new\_state\_root} \tag{42}$$

Definition 9.2 (Vectorization Gadget).

$$\forall i : \mathbf{x}_i = \phi_{tx}(tx_i) \text{ (deterministic)} \tag{43}$$

Definition 9.3 (Folding Gadget).

$$F_B = \text{Fold}(\{\mathbf{x}_i, \mathbf{s}_j, \mathbf{w}_k\}) \tag{44}$$

$$H(F_B) = C_{\text{fold}} \tag{45}$$

Definition 9.4 (PQ Gadget).

$$\forall \mathbf{f} \in F_B : \left\| \mathbf{f} - \hat{\mathbf{f}} \right\|_2 \leq \varepsilon \tag{46}$$

$$H(\text{indices}) = C_{\text{PQ}} \tag{47}$$

9.4 Fixed-Point Arithmetic

Stage	**Format**	**Range**
Feature normalization	Q2.14	$[-4, 4)$
Statistics	Q8.24	$[-256, 256)$
Folding products	Q4.28	$[-16, 16)$
PQ error check	Q6.26	$[0, 64)$

Table 4: Fixed-point layouts for circuit arithmetic

9.5 Halo2 Implementation

The proof system uses Halo2 with:

- **IPA** (Inner Product Argument) commitment scheme
- **BN254 curve** for pairing operations
- **Poseidon hash** for in-circuit commitments
- **Lookup tables** for range checks and centroid access

10 Atlas Visualization

10.1 2D Projection

The Atlas projects hotzones to 2D for visualization:

$$\mathbf{p}_i = \text{Project}(\mathbf{c}_i) \in \mathbb{R}^2 \tag{48}$$

using PCA or t-SNE dimensionality reduction.

10.2 Visual Encoding

- **Position**: 2D coordinates from projection
- **Size**: Proportional to $\sqrt{\rho_i}$
- **Color**: Mapped from dominant semantic tag
- **Edges**: Hyperedges with $w > \theta$

10.3 3D Force-Directed Layout

For the hypergraph visualization:

- Node positions via force simulation with repulsion
- Edge springs with attraction proportional to $w(H_i, H_j)$
- Node size weighted by density

11 Summary of Parameters

References

1. Jégou, H., Douze, M., & Schmid, C. (2011). Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*

2. Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis.* Chapman & Hall.

3. Electric Coin Company. (2020). The Halo2 proving system. https://zcash.github.io/halo2/

4. O'Connor, J., et al. (2020). BLAKE3: One function, fast everywhere. https://github.com/BLAKE3-team/BLAKE3-specs

Parameter	**Symbol**	**Default**	**Description**
Transaction dim	d_{tx}	16	Features per transaction
Fold dim	d	16	Canonical vector dimension
Components	k	4	Principal directions
PQ subspaces	m	4	Codebook partitions
PQ centroids	K	256	Centroids per subspace
Error bound	ε	0.25	Max reconstruction error
KDE bandwidth	h	0.15	Gaussian kernel width
Density threshold	θ	0.02	Min hotzone density
Max hotzones	—	16	Top-k selection
Hyperedge threshold	θ_2	5×10^{-5}	Min edge weight

Table 5: System parameters