From Scores to Queues: Operationalizing Cross-Chain Obfuscation Signals for Smart-Contract Audits

Yao Zhao, Zhang Sheng[†], Shengchen Duan, Shen Wang, Daoyuan Wu

***Abstract*—Obfuscation substantially increases the interpretation cost of smart-contract auditing, while the comparability and transferability of obfuscation signals across chains remain unclear. We present HOBFNET as an efficient surrogate of OBFS_TOOL (ObfProbe), enabling fast cross-chain scoring at scale. The model aligns well with tool outputs on Ethereum (PCC 0.9158, MAPE 8.20%) and achieves 8–9 ms per contract, a 2.3k–5.2k× speedup over second-level OBFS_TOOL runs, enabling million-scale scoring. On large BSC, Polygon, and Avalanche corpora, we find systematic score drift: fixed-threshold transfer inflates/deflates candidate queues, motivating *within-chain* main/extreme thresholds (p99/p99.9) and an actionable queueing strategy. The high-score tail exhibits rare selectors, external-call opcode enrichment, and low signature density; a proxy indicator is enriched in the BSC high-score queue, enabling secondary triage. Cross-chain reuse analysis shows tail enrichment and directional diffusion, with traceable same-hash cases across chains. In publicly alignable incident samples, all fall into the p99 queue; Transit Swap DEX Hack and New Free DAO Flash Loan exhibit cross-chain spillover, indicating real-world hit and prioritization value. We deliver a two-tier audit queue and cross-chain linkage workflow to support practical multi-chain security operations.**

***Index Terms*—blockchain, smart-contract obfuscation, cross-chain analysis, security measurement**

I. INTRODUCTION

Obfuscation has become a central obstacle to understanding fund transfers in smart contracts. Here, obfuscation refers to deliberate transformations of contract bytecode that preserve functionality while making transfer logic harder to inspect or analyze. The recent Ethereum study by Sheng Z. et al. [1] provides the first systematic characterization of obfuscated fund transfers. Their large-scale analysis shows that obfuscation is routine but risk concentrates in the most heavily obfuscated contracts; within the heavily obfuscated Ethereum subset, they identify 463 high-risk cases including MEV bots, Ponzi schemes, fake decentralization, and extreme centralization, placing roughly $100M at risk, and obfuscated scams exhibit higher peak losses than non-obfuscated ones. They also show that obfuscation can cripple existing detectors; for example, SourceP recall drops from 0.80 to 0.12, indicating that attackers can hide intent through obfuscation. Complementary work on closed-source contracts shows that obfuscation can conceal critical vulnerabilities rather than provide protection [2]. At the same time, advances in bytecode analysis such as decompilation and function recovery enable large-scale measurement [3], [4]. These results establish obfuscation as a security-critical signal and expose a practical audit challenge: security operations at million-scale often rely on high-risk queues for prioritization, where obfuscation scores can serve as an actionable ranking signal.

EVM ecosystems are now multi-chain, and contracts are routinely redeployed across BSC, Polygon, and Avalanche. A direct audit question emerges: *can Ethereum's high-obfuscation cutoff be reused as a universal screening threshold for other chains?* This is not merely a statistical issue—if the cutoff fails to transfer, cross-chain audit queues will either overflow or miss high-risk candidates, leading to alert fatigue or blind spots. Code reuse is pervasive on Ethereum [5], [6], multi-chain analyses reveal copy-and-paste deployments with non-trivial behavioral mismatches [7], and malicious templates such as sniper bots migrate across chains [8]. Cross-chain tracing studies and industry reports further show that cross-chain fund migration and laundering are widespread [9], [10]. These observations imply that obfuscation signals may propagate across chains, yet their thresholds may not.

Even if we can form a main audit queue, "high score" alone is insufficient for triage—we must understand what the queue actually contains. If the tail is dominated by standard ERC/NFT or proxy templates, the audit strategy should differ from one dominated by low-visibility, specialized logic. Thus we need a structural profile of the high-score tail and executable features for secondary triage.

Moreover, whether high-score templates diffuse across chains determines whether auditing must be coordinated across ecosystems: if high-score templates reuse and spread directionally, risk can spill over across chains and require cross-chain linkage. Cross-chain interoperability and fund migration further complicate incident pathways: vulnerabilities may occur on a single chain while funds move cross-chain for laundering or redistribution; meanwhile, the same template can be redeployed across chains. Therefore, for obfuscation scores to be audit-relevant, they must be validated against real incidents rather than remain purely distributional signals.

Answering these questions requires scoring large corpora across multiple chains. However, existing obfuscation analysis

Yao Zhao is with the Department of Aeronautical and Aviation Engineering, The Hong Kong Polytechnic University, Hong Kong, China.

Zhang Sheng is with The State Key Laboratory of Blockchain and Data Security, Zhejiang University (e-mail: dcszhang@foxmail.com). [†]Corresponding author.

Shengchen Duan is with Singapore Management University.

Shen Wang is with Final Round AI.

Daoyuan Wu is with The State Key Laboratory of Blockchain and Data Security, Zhejiang University.

tools are expensive to run at scale. Based on our measurements, Sheng et al.'s ObfProbe (denoted as OBFS_TOOL in this paper) can take weeks of runtime for a single chain and consume substantial compute [1], making cross-chain analysis slow and difficult to iterate. This efficiency bottleneck motivates a scalable cross-chain scoring pipeline.

To overcome it, we train a hierarchical attention model HOBFNET on an Ethereum obfuscation dataset as a fast surrogate for OBFS_TOOL, enabling rapid inference of obfuscation scores for large BSC, Polygon, and Avalanche corpora. With cross-chain scores in hand, we answer the following research questions:

1) **RQ1 (Audit Thresholds):** Can Ethereum's high-obfuscation cutoff be directly reused on other chains? If not, how should within-chain thresholds be set to keep audit queues actionable?
2) **RQ2 (Tail Profiling):** What structural features define the high-score tail, and how can they be used for executable secondary triage?
3) **RQ3 (Cross-Chain Diffusion):** Do high-score templates reuse across chains and exhibit directional diffusion from smaller to larger ecosystems?
4) **RQ4 (Real-World Hits):** Do high-score queues cover real cross-chain incident contracts, thereby validating practical audit value?

Our contributions are fivefold: (1) We build and validate HOBFNET as an efficient surrogate of OBFS_TOOL: ablation and error analyses demonstrate reliability (MAPE 8.20%, PCC 0.9158), and inference reaches 8–9 ms per contract, a 2.3k–5.2k× speedup over second-level ObfProbe runs, enabling million-scale scoring. (2) We provide *within-chain* main/extreme thresholds and queue sizes (p99/p99.9): ETH 18.07/22.69, BSC 16.82/19.74, Polygon 18.72/20.51, Avalanche 19.18/20.67, and show fixed-threshold transfer yields 0.48%–2.32% candidate inflation/deflation, yielding actionable audit queueing. (3) We characterize the high-score tail structure: rare selectors (lift 10–50×), external-call opcode enrichment, and low signature density; the BSC proxy indicator is enriched in the tail (0.25%→1.03%), supporting secondary triage. (4) We show directional cross-chain reuse: tail Jaccard is about 1.5–2.0× higher than overall and diffusion is asymmetric (small→large); we provide traceable same-hash cases. (5) Under publicly available evidence, aligned incident samples all hit p99; Transit Swap DEX Hack and New Free DAO Flash Loan serve as cross-chain spillover cases, indicating real-world hit value of the high-score queue. We have open-sourced the model: https://github.com/dcszhang/HObfNET.

II. Related Work

Obfuscation measurement and impact. Sheng et al. propose ObfProbe to quantify transfer-path obfuscation in Ethereum bytecode, defining a feature-based taxonomy and using a Z-score to rank obfuscation strength, and they show that heavy obfuscation concentrates risk and degrades detector effectiveness [1]. Yang et al. further demonstrate that closed-source and control-flow obfuscation can conceal critical vulnerabilities in MEV bots, motivating deobfuscation-aware analysis [2]. From an adversarial perspective, source-level obfuscation and bytecode-level obfuscation have been shown to substantially reduce the success of existing analyzers and decompilers [11], [12]. BiAn further systematizes source-level obfuscation and substantially increases decompilation and detection difficulty [13]. Together, these studies establish obfuscation as a security-relevant signal, yet they are largely Ethereum-centric and do not address whether obfuscation signals transfer across chains.

Bytecode analysis and decompilation. Large-scale bytecode analysis is enabled by advances in decompilation and function recovery. Gigahorse introduced a declarative decompiler that lifts EVM bytecode to higher-level IRs and supports scalable analyses [3]. Elipmoc improves recovery for complex control flow and optimized bytecode, while Shrnkr further increases scalability with shrinking context sensitivity [14], [15]. Neural-FEBI improves function identification directly on EVM bytecode, enabling more precise downstream analyses [4], [16]–[19]. These tools make obfuscation measurement feasible at scale and enable reuse analysis, but they are not designed to study cross-chain drift and have not been systematically applied to cross-chain distribution and reuse measurement.

Learning-based program analysis and surrogates. Recent work applies machine learning to EVM bytecode for function identification and vulnerability detection [4], [20]. In parallel, transaction language models and graph-based representations have been used for Ethereum fraud detection and blockchain phishing detection [21], [22], highlighting the effectiveness of learned signals for security tasks. For vulnerability detection, SymGPT combines symbolic execution with LLMs [23], VASCOT applies Transformers to bytecode sequences [24], and MANDO-HGT uses heterogeneous graph Transformers to capture control/data-flow semantics [25], [26]. These approaches show that learned representations can reduce reliance on costly analysis pipelines, but they are not designed to score obfuscation or to support multi-chain distribution comparison. Our surrogate model follows this learning-based paradigm to enable fast obfuscation scoring at cross-chain scale.

Code reuse and clone detection. Clone detection studies reveal pervasive code reuse in Ethereum, both at the contract level and the function level [5], [27]. Chen et al. analyze code reuse patterns and show that standardized templates and shared components are prevalent in smart contract development [6]. Recent work benchmarks clone detectors and compares their effectiveness on smart contracts [28], and interpretable similarity tools have been proposed to identify reuse families [29]. Clone propagation at the platform level has also been studied; BlockScope identifies cloned vulnerabilities across forked blockchains [30]. These works imply that code reuse can propagate vulnerabilities and design patterns, yet they focus on single-chain ecosystems and do not quantify reuse families in cross-chain settings [31].

Cross-chain reuse and migration. In multi-chain contexts, EquivGuard reports that copy-and-paste reuse can introduce EVM-inequivalent code smells, highlighting semantic drift even among reused contracts [7]. Empirical studies of sniper bots across Ethereum and BNB Smart Chain show that adversarial templates migrate between chains, suggesting cross-chain propagation of risky patterns [8]. Cross-chain imitation attacks further demonstrate real-world threats from rapid cross-chain copying [32]. These findings motivate our focus on cross-chain reuse families in high-obfuscation tails and on the transferability of obfuscation thresholds.

Cross-chain security context. Cross-chain interoperability and bridge systems are major risk concentrations [33]–[35]. Industry reports and tracing studies further show that cross-chain fund migration and laundering are widespread [9], [10], [36], [37]. Public enforcement actions document multi-chain scam deployments, such as the Forsage case [38]. These risks compound with cross-chain reuse, which raises the stakes for auditing and triage. These works provide incident and ecosystem context, but they do not analyze contract-level obfuscation; our work aligns high-obfuscation signals with such incidents to validate practical relevance.

Positioning and comparison. Compared with ObfProbe and other Ethereum-centric obfuscation studies [1], [2], we focus on cross-chain transferability by quantifying distribution shifts and tail composition across multiple EVM chains rather than a single-chain prevalence analysis. Relative to bytecode analysis and decompilation work [3], [4], [14], [15], our contribution is not a new lifter but a scalable measurement pipeline that uses learned scoring to accelerate cross-chain analysis. In contrast to clone and reuse studies that remain largely single-chain [5], [6], [27]–[29], we explicitly trace reuse families across chains and examine whether high-obfuscation tails are enriched by cross-chain reuse. Finally, while bridge security studies and incident reports quantify cross-chain risk [33], [34], [38], our work links those risks to obfuscation signals, offering a measurement-based view of where cross-chain auditing should prioritize attention.

III. Model Construction and Evaluation

A. Task Definition

Let $\mathcal{B} = \{\text{Ethereum}, \text{BSC}, \text{Polygon}, \text{Avalanche}\}$ be the chain set. For each chain $b \in \mathcal{B}$, we denote its contract corpus as $\mathcal{C}_b$. Each contract $c \in \mathcal{C}_b$ is represented by a bytecode sequence x_c, and our goal is to predict an obfuscation score $s_c \in \mathbb{R}$. We set $s_c = s_c^{\text{tool}}$, where s_c^{tool} is produced by ObfProbe.

Our supervision signal comes from Sheng et al.'s ObfProbe (denoted as Obfs_Tool) on Ethereum. The tool outputs a scalar score s_c^{tool} and a feature vector $F_c \in \mathbb{R}^K$ (transfer-path obfuscation features). We use the tool-defined obfuscation score as the learning target.

B. Training Data and Preprocessing

We use Ethereum contracts for training and validation because only Ethereum has tool-derived labels. Contracts from BSC, Polygon, and Avalanche are used only for cross-chain inference in Section IV.

Bytecode normalization. We strip the `0x` prefix and remove compiler metadata and constructor artifacts to obtain a canonical bytecode sequence.

Segmentation. The bytecode is mapped to integer tokens (0–255) and sliced into N fixed-length segments of size L. We reserve a dedicated padding token (256) and apply attention masks so padding does not contribute to attention or pooling; `0x00` (STOP) remains a normal opcode. We maintain a validity mask $M \in \{0, 1\}^N$ to distinguish real segments from padding. Contracts longer than $L \times N$ are truncated at the end of the runtime bytecode; we quantify length sensitivity in Section III-F. We report L and N in Section III-F.

Leakage control. Within Ethereum, we perform train/validation/test splits at the bytecode-family level (near-duplicate clusters) to reduce clone leakage. Families are built using canonicalized bytecode fingerprints and clustering with a fixed similarity threshold to keep the split reproducible. We use a 7:2:1 split at the family level.

C. Supervision Signals

ObfProbe provides K feature signals and a Z-score based obfuscation measure. We treat s_c^{tool} (Z-score) as the primary regression target and reconstruct F_c as an auxiliary objective. The Z-score is defined as:

$$s_c^{\text{tool}} = \sum_{k=1}^{K} \frac{F_{c,k} - \mu_k}{\sigma_k}, \tag{1}$$

where (μ_k, σ_k) are computed on the Ethereum training set to avoid information leakage.

D. Model Architecture

1) Overview: Figure III-D1 illustrates our model HObfNet, a Hierarchical Attention Network (HAN) designed for obfuscation scoring. The model follows a divide-and-conquer principle to handle extreme bytecode length and to capture both local opcode patterns and global contract structure. We report the number of layers and attention heads in Section III-F.

2) Local Encoder (local patterns): We embed each segment $x_i \in \mathbb{R}^L$ into a 256-dimensional space with positional embeddings to preserve opcode order. The local encoder captures short-range obfuscation artifacts (e.g., instruction-level patterns):

$$H_{local}^{(i)} = \text{Transformer}_{enc}(\text{Embed}(x_i) + P_{local}). \tag{2}$$

3) Global Encoder (long-range dependencies): The global encoder models cross-segment dependencies that reflect contract-level obfuscation strategies. We apply multi-head attention with chunk-level positional encodings P_{chunk} and allow bidirectional visibility.

4) Masked Mean Pooling (padding control): To avoid padding artifacts, we aggregate only valid segments using a masked mean:

$$v_{contract} = \frac{\sum_{i=1}^{N} (H_{global}^{(i)} \cdot M_i)}{\sum_{i=1}^{N} M_i + \epsilon}. \tag{3}$$

This prevents length imbalance from biasing the contract representation.

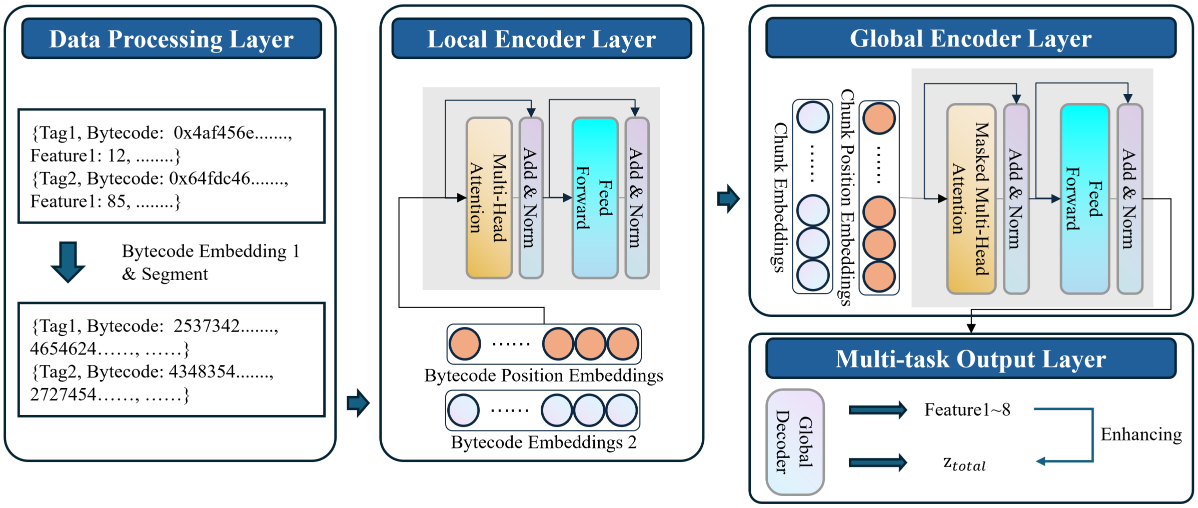

Data Processing Layer
{Tag1, Bytecode: 0x4af456e......., Feature1: 12, ...}
{Tag2, Bytecode: 0x64fdc46......., Feature1: 85, ...}
Bytecode Embedding 1 & Segment
{Tag1, Bytecode: 2537342......., 4654624……, ……}
{Tag2, Bytecode: 4348354......., 2727454……, ……}
Local Encoder Layer
Multi-Head Attention
Add & Norm
Feed Forward
Add & Norm
Bytecode Position Embeddings
Bytecode Embeddings 2
Global Encoder Layer
Chunk Embeddings
Chunk Position Embeddings
Masked Multi-Head Attention
Add & Norm
Feed Forward
Add & Norm
Multi-task Output Layer
Global Decoder
Feature1~8
z_total
Enhancing

Fig. 1. The overall architecture of HObfNET: data processing, local encoder, global encoder, and a multi-task output layer that reconstructs domain features.

5) Multi-task Enhancement (noise control and interpretability): We reconstruct domain features to regularize the latent space:

$$\hat{F} = \text{MLP}_{rec}(v_{contract}). \tag{4}$$

We compute an auxiliary score $\hat{s}^{\text{tool}}$ by applying the tool's Z-score formula to $\hat{F}$:

$$\hat{s}^{\text{tool}} = \sum_{k=1}^{K} \frac{\hat{F}_k - \mu_k}{\sigma_k}. \tag{5}$$

The final prediction fuses the deep representation, reconstructed features, and the auxiliary score:

$$\hat{s} = \text{MLP}_{head}([v_{contract} \oplus \hat{F} \oplus \hat{s}^{\text{tool}}]). \tag{6}$$

E. Training Objective and Optimization

We train end-to-end with a joint loss:

$$\mathcal{L}_{total} = \lambda_s \mathcal{L}_{MSE}(s_c^{\text{tool}}, \hat{s}) + \lambda_{aux} \mathcal{L}_{MSE}(s_c^{\text{tool}}, \hat{s}^{\text{tool}}) \tag{7}$$

$$+ \lambda_{feature} \sum_{k=1}^{K} \mathcal{L}_{MSE}(F_{gt}^{(k)}, \hat{F}_k), \tag{8}$$

with $\lambda_s = 1$, $\lambda_{aux} = 0.1$, and $\lambda_{feature} = 0.01$. We use AdamW with learning rate 5×10^{-4} and weight decay 1×10^{-4} on an NVIDIA A100 [39], [40].

F. Experimental Setup

We evaluate reliability from three angles: accuracy (MAPE/MAE/MSE and Pearson correlation on the held-out Ethereum set), robustness (ablation and error analysis), and efficiency (throughput on GPU). We compare the full model with controlled variants (Standard Transformer, GRU-HAN, HAN without multi-task) to validate architectural choices. The training corpus follows the prior Ethereum dataset (1.04M contracts, 2022-06 to 2024-08) with a 7:2:1 train/validation/test split at the family level. We set $L = 512$, $N = 32$, vocab size $= 257$, $D_{\text{MODEL}} = 256$, NUM_LAYERS $= 2$, N_HEAD $= 4$, and dropout $= 0.1$. Training uses AdamW (lr 5×10^{-4}, weight decay 10^{-4}), batch size $= 24$, 20 epochs, and gradient clipping at 0.5 on an NVIDIA A100. Inference uses batch size $= 200$ on GPU. Incident alignment is address- and transaction-based (no explicit time window), and the sources are reported in RQ4.

G. Evaluation Metrics

To evaluate predictive accuracy and correlation, we report mean absolute percentage error (MAPE), mean absolute error (MAE), mean squared error (MSE), and Pearson correlation coefficient (PCC):

$$\begin{aligned} \mathcal{L}_{MAPE} &= \frac{1}{N} \sum_{i=1}^{N} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \times 100\%, \\ \mathcal{L}_{MAE} &= \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i|, \\ \mathcal{L}_{MSE} &= \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2, \\ \mathcal{L}_{PCC} &= \frac{\sum_{i=1}^{N} (y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})}{\sqrt{\sum_{i=1}^{N} (y_i - \bar{y})^2} \sqrt{\sum_{i=1}^{N} (\hat{y}_i - \bar{\hat{y}})^2}}, \end{aligned} \tag{9}$$

where N is the number of test samples, y_i and $\hat{y}_i$ are the tool score and prediction for sample i, and $\bar{y}, \bar{\hat{y}}$ are their means.

H. Ablation Study

To explicitly validate the effectiveness of the proposed hierarchical architecture and the domain-guided multi-task mechanism, we conducted an ablation study by comparing the full HAN model with three variants. The configuration of these variants is described as follows:

- Standard Transformer: This variant removes the "divide-and-conquer" strategy. Instead of processing bytecode in chunks, it treats the entire smart contract as a single, flat sequence of bytes fed directly into a standard Transformer.

- GRU-HAN without Multi-Task: The Gated Recurrent Unit (GRU) mechanism is employed instead of the attention block to extract and expand the features.
- HAN without Multi-Task: This variant retains the hierarchical encoder structure (Local + Global Encoder) but removes the Multi-task module.

We report MAPE/MAE/MSE/PCC for HObfNET in Table I. First, the standard Transformer exhibits significantly lower performance compared to hierarchical models. This performance drop can be attributed to forced truncation, which leads to the loss of critical logic located at the end of large contracts. This confirms that the hierarchical approach mitigates truncation effects relative to a flat Transformer, although contracts longer than $L \times N$ are still truncated. Second, GRU-HAN underperforms the Transformer-based global encoder, indicating that attention captures long-range dependencies more effectively than recurrent aggregation. The performance of HAN without the multi-task module indicates that removing auxiliary supervision leads to a noticeable increase in error rates. This suggests that latent features derived solely from raw bytecode are insufficient to fully capture high-level obfuscation patterns. Feature reconstruction and tool-consistent auxiliary scoring provide regularization, aligning representations with the tool-defined obfuscation metric.

TABLE I
Ablation Study Results. Comparison between the full model and its variants. Lower MAPE/MAE/MSE and higher PCC indicate better performance.

Model	MAPE (↓)	MAE (↓)	MSE (↓)	PCC (↑)
Standard Transformer	16.29	0.9521	2.7147	0.8466
HAN (GRU)	14.28	0.8794	2.4511	0.8484
HAN (without multi-task)	13.02	0.8359	2.3371	0.8619
HAN (Full Model)	8.20	0.6341	1.4477	0.9158

I. Efficiency Analysis

Beyond accuracy, computational efficiency is critical for large-scale smart contract analysis. As a baseline, ObfProbe's SSA pipeline on Ethereum reports 93.55% first-pass success under a 20s timeout; the first pass averages 19.66s (median 19.9s) [1]. Failed contracts are rerun without a timeout cap, where the average runtime is about 45s (median 41s) and large/complex bytecodes often fall in the 20–80s range. Training HObfNET is performed on Ethereum; each epoch takes about 6.13 hours, with an additional 0.52 hours for validation on the same A100 server. For inference, we score BSC, Polygon, and Avalanche at scale.

TABLE II
Efficiency Analysis. ObfProbe vs. HAN inference time per contract.

System / Chain	Contracts	Avg. time / contract (ms)
ObfProbe (ETH, SSA)	1042923	19660
HObfNET (BSC)	2308899	8.67
HObfNET (Polygon)	288611	8.68
HObfNET (Avalanche)	96173	8.69

As shown in Table II, per-contract latency is stable (about 8–9 ms) across chains, indicating that the hierarchical encoder adds minimal overhead beyond bytecode normalization and chunked batching. For reference, ObfProbe reruns without a timeout cap average 45s (median 41s) and heavy bytecodes often fall in the 20–80s range. Compared with ObfProbe's SSA runtime on Ethereum, our per-contract inference is approximately 2.3k–5.2k× faster.

J. Error Analysis

We analyze prediction errors on the Ethereum validation set to understand when the surrogate deviates from the tool. Overall, the model attains MAPE 8.20%, MAE 0.6341, MSE 1.4477, and PCC 0.9158, consistent with Table I. The absolute error distribution is heavy-tailed: median 0.3008, p90 1.3340, p95 2.1702, and p99 5.4801.

Length sensitivity. We bin contracts by original bytecode length quartiles (pre-truncation) and observe monotonic degradation as length grows (Table III): MAE/MAPE are 0.3747/8.18% (0–4,802 bytes), 0.4829/9.52% (4,802–11,244), 0.5952/10.97% (11,244–19,186), and 1.0836/14.97% (19,186–32,768). This pattern reflects both intrinsic length complexity and truncation for contracts longer than $L \times N$.

TABLE III
Length-binned errors. Quartile bins of bytecode length and corresponding MAE/MAPE.

Length bin (bytes)	MAE	MAPE
0–4,802	0.3747	8.18%
4,802–11,244	0.4829	9.52%
11,244–19,186	0.5952	10.97%
19,186–32,768	1.0836	14.97%

Tail robustness. We use two tail levels: p99 ($\tau_{eth}^{1\%} = 18.07$, about 10,430 contracts) and an extreme tail p99.9 ($\tau_{eth}^{0.1\%} = 22.69$, about 1,043 contracts). The reported errors are for the extreme tail (p99.9): MAE 2.0045 and MAPE 9.84%, indicating that performance degrades but remains stable enough for tail-focused analysis.

Plain summary. This chapter builds HObfNET, a fast surrogate of ObfProbe for bytecode obfuscation scoring, shows it works through ablations and error analysis, and demonstrates that it is thousands of times faster than the tool baseline. These results justify using the model for large-scale cross-chain measurements in the next chapter.

IV. Cross-Chain Obfuscation Measurements and Findings

This chapter uses the HObfNET model trained in Section III-F to score deduplicated runtime bytecode on BSC, Polygon, and Avalanche. We obtain cross-chain obfuscation scores s and answer RQ1–RQ4. For non-Ethereum chains, s comes from model predictions; since the training objective aligns with the tool score, we treat them as comparable signals for distribution, tail, and reuse analyses.

TABLE IV
CROSS-CHAIN TOOL ALIGNMENT (10K SAMPLE PER CHAIN).

Chain	N	MAE	RMSE	MAPE	PCC
BSC	10,000	0.45	0.93	5.7%	0.97
Polygon	10,000	0.47	0.95	5.9%	0.96
Avalanche	10,000	0.45	0.91	5.3%	0.97

A. Data and Measurement Setup

Scale and time span. We normalize and deduplicate runtime bytecode (canonicalized runtime hash) and obtain 2,308,899 contracts on BSC, 288,611 on Polygon, and 96,173 on Avalanche. The coverage window is 2022-05 to 2024-10, referring to the collection/indexing period when we crawled and organized contract addresses and runtime bytecode from explorers/APIs.

Score definition and tool features. The Ethereum training/validation file provides s and tool-side features (feature1–feature7). We confirm that s is the sum of these (standardized) feature terms; thus we treat it as the *tool-aggregated scalar score* and use it as the aligned target across chains.

Cross-chain sanity check. We perform a lightweight cross-chain spot-check by sampling 200k contracts from BSC and Polygon (and using all Avalanche contracts) and computing correlations between s and bytecode length / signature counts. Correlations are consistently positive (0.46–0.72 for length; 0.10–0.27 for signature counts), indicating that the surrogate model preserves stable structural monotonicities across chains even without tool labels.

Cross-chain tool alignment (10k per chain). To directly validate score comparability, we run ObfProbe on 10k contracts per chain and align tool scores with model predictions (Table IV). The agreement is strong (PCC 0.96–0.97, MAE 0.45–0.47), supporting s as a comparable cross-chain signal for distribution and tail analysis.

Feature definitions and heuristics. We extract 4-byte selectors from `PUSH4` immediates in runtime bytecode; the number of unique selectors is the *signature count*. *Signature density* is defined as $\#\text{unique selectors}/(\text{bytecode size in KB})$. ERC20/721 labels are assigned when bytecode contains a core selector set (ERC20: `totalSupply`, `balanceOf`, `transfer`, `approve`, `transferFrom`; ERC721: `balanceOf`, `ownerOf`, `approve`, `setApprovalForAll`, `transferFrom`, `safeTransferFrom`), used as a heuristic indicator rather than a definitive classification. The *proxy indicator* marks contracts whose runtime bytecode exhibits delegatecall-based forwarding patterns (e.g., presence of `DELEGATECALL` with short forwarding dispatch), and is intentionally broader than strict EIP-1167 matching. For selector/opcode enrichment, we compute $\text{lift}(x) = p_{\text{tail}}(x)/p_{\text{all}}(x)$ with a small ϵ smoothing term in the denominator, and we only report patterns above a minimum occurrence threshold to avoid unstable lift ratios.

Two-level tail definition. We adopt two tail levels: (i) *main tail* as chain-level p99 (Top 1%) for stable statistics; (ii) *extreme tail* as p99.9 (Top 0.1%) for robustness analysis. The corresponding score cutoffs are: main tail (ETH 18.07, BSC 16.82, Polygon 18.72, Avalanche 19.18) and extreme tail (ETH 22.69, BSC 19.74, Polygon 20.51, Avalanche 20.67).

We also test direct transfer of the Ethereum cutoff 18.07 and compute $\Pr(s \geq 18.07)$ on other chains (Table VI).

B. RQ1: Distribution Transferability and Threshold Drift

Question (starting point). We begin with the most actionable audit question: *are obfuscation-score distributions consistent across chains, and can Ethereum's high-score cutoff be reused as a universal screening threshold?* This section provides a definitive answer and sets up why differences must be explained.

1) Cross-chain distributions and quantiles: Table V reports score quantiles. Medians are similar (around 5.0–5.4), but high quantiles diverge: BSC p90/p95/p99 = 9.60/12.45/16.82, notably lower than Polygon (13.50/15.66/18.72) and Avalanche (14.09/16.30/19.18). This indicates a lighter tail on BSC and heavier tails on Polygon/Avalanche.

2) What happens if we transfer the Ethereum cutoff (18.07)?: Table VI reports tail shares under the Ethereum cutoff (18.07): BSC 0.48%, Polygon 1.58%, and Avalanche 2.32%. Relative to Ethereum's 1.00% baseline, BSC shrinks while Polygon/Avalanche inflate, showing that a fixed threshold does not translate consistently across chains.

From an operational perspective, this creates immediate audit distortion. With the ETH cutoff (18.07), BSC would flag only 11,079 contracts (0.48%) instead of its chain-specific baseline of $\sim$23,089, *missing about 12k high-obfuscation candidates*. In contrast, Polygon and Avalanche inflate to 4,553 (+58%) and 2,229 (+132%) candidates, respectively, dramatically increasing review load. Here, "high-obfuscation candidates" refers to the *within-chain Top1% (p99) candidate set*. Thus threshold drift is not a statistical curiosity—it translates directly into *missed coverage* or *alert overload.*

Queue length and staffing cost. Let the manual review throughput be r contracts/day (or average review time t minutes per contract). The candidate gap Δn induced by threshold transfer directly converts to labor cost: $\Delta\text{days} = \Delta n/r$ or $\Delta\text{hours} = \Delta n{\cdot}t/60$. For BSC, reusing the ETH cutoff reduces the queue by $\sim 1.2 \times 10^4$ relative to the chain's Top1% baseline, i.e., $\approx 1.2 \times 10^4/r$ days of review workload, but at the cost of excluding a comparable number of high-percentile candidates (potential misses).

Actionable two-tier triage. In practice, we recommend using s as a triage signal rather than a single hard cutoff: a main queue for routine audits and continuous monitoring, and an emergency queue for the most extreme cases that warrant immediate reverse engineering and fund-flow tracing. Accordingly, we recommend *chain-specific, actionable* screening cutoffs: main screening scores of ETH 18.07, BSC 16.82, Polygon 18.72, and Avalanche 19.18 to maintain stable candidate volumes; and extreme-priority cutoffs of ETH 22.69, BSC 19.74, Polygon 20.51, and Avalanche 20.67 when prioritizing the most severe cases.

Practical reminder. Known incident contracts do not necessarily fall into the Top1% tail (see RQ4 percentiles). In monitoring scenarios, we therefore suggest maintaining a wider

TABLE V
SCORE QUANTILES ACROSS CHAINS.

Chain	N	p50	p90	p95	p99	p99.9
Ethereum	1,042,923	5.03	9.37	12.44	18.07	22.69
BSC	2,308,899	5.13	9.60	12.45	16.82	19.74
Polygon	288,611	5.37	13.50	15.66	18.72	20.51
Avalanche	96,173	5.21	14.09	16.30	19.18	20.67

TABLE VI
ABSOLUTE-THRESHOLD TRANSFER (ETH CUTOFF = 18.07).

Chain	Tail count	Tail share
Ethereum	10,430	1.00%
BSC	11,079	0.48%
Polygon	4,553	1.58%
Avalanche	2,229	2.32%

"watch band" (e.g., p95–p99) for continuous observation, while using the main/extreme cutoffs for resource-constrained prioritization.

Therefore, for cross-chain audit dashboards, it is often more practical to expose *within-chain percentiles* as a unified scale: map each contract's score to its position in the chain-specific distribution, and trigger alerts at fixed percentiles (e.g., p99/p99.9).

3) Threshold-transfer sensitivity across chain-specific cutoffs: We compare cross-chain transfers of each chain's main cutoff. Using the BSC cutoff (16.82) yields 3.27% (Polygon) and 4.20% (Avalanche), while using the Avalanche cutoff (19.18) yields only 0.17% on BSC. Even among chain-specific main cutoffs, cross-chain transfer causes substantial shifts.

4) Extreme tail robustness: Transferring the Ethereum extreme cutoff (22.69) yields zero hits on all three chains, suggesting stronger chain-specificity in the extreme tail (or more conservative cross-chain scoring). Note that s on non-Ethereum chains is model-predicted rather than tool-derived; thus "zero hits" may also reflect calibration/extrapolation uncertainty under cross-chain drift, further motivating within-chain percentiles for comparison. This does not mean other chains lack risky contracts; rather, absolute extreme thresholds do not transfer well. We therefore use the chain-specific main cutoffs as the primary tail definition and keep extreme cutoffs for robustness.

Finding 1 (RQ1). The Ethereum threshold cannot serve as a universal cross-chain screening line: differing chain distributions cause systematic queue shrinkage or inflation, leading to missed coverage or queue overload. Audit practice should therefore rely on chain-specific percentile thresholds (main queue p99: 18.07/16.82/18.72/19.18; emergency queue p99.9: 22.69/19.74/20.51/20.67) or explicit calibration mappings, rather than a single fixed value.

This matters operationally: an over-wide threshold inflates review queues and audit budgets, while an over-strict threshold suppresses alerts and increases missed coverage. Based on our analysis, we provide chain-specific obfuscation audit score references for the four chains to support practical screening.

C. RQ2: Tail Structure

Following RQ1. If distributions differ, we ask *what drives those differences*: what actually dominates the high-obfuscation tail—standard ERC/NFT contracts, proxy templates, or low-visibility bespoke logic? We combine ERC labels, a proxy indicator, and signature-density evidence to build an interpretable profile.

1) ERC20/ERC721 presence in the tail: Table VII compares overall vs. main-cutoff tail ERC shares. Overall ERC20 shares are 2.88% (BSC), 0.35% (Polygon), and 0.84% (Avalanche), while overall ERC721 shares are 0.07%, 3.42%, and 0.55%, respectively. In the tail, ERC20/ERC721 shares drop sharply: BSC ERC20 falls from 2.88% to 0.17 ($\sim$17×), and Polygon/Avalanche ERC721 drops to $< 0.01\%$ (order-of-magnitude decline).

This indicates that the high-obfuscation tail is not dominated by standard ERC contracts. Given high sighash missingness and lower ABI visibility in obfuscated contracts, ERC labels are likely undercounted; we thus interpret this as "tail contracts are less standard and less ABI-visible." From an audit perspective, focusing only on ERC asset contracts would systematically miss the high-obfuscation tail. The structural cues and exemplar checks below further support this interpretation.

2) Proxy (heuristic indicator) prevalence: **Scope note.** We distinguish two notions: (i) *EIP-1167 minimal proxy* (strict template matching, extremely rare overall and in the tail, serving as a lower bound), and (ii) a broader *proxy indicator* (heuristic patterns covering transparent/UUPS/custom forwarders, potentially over/under-counting). Table VIII reports (ii); strict minimal proxy is only a lower-bound reference and should not be interpreted as equivalent to EIP-1167 prevalence. We mark contracts with a proxy indicator and report prevalence in overall and tail sets (Table VIII). The indicator is based on heuristic proxy bytecode patterns and forwarding behaviors, and is intentionally broad rather than a strict EIP-1167 detector; as such it may over/under-count. Proxy indicators remain rare overall (sub-1%); under the main cutoff, BSC shows clear enrichment (0.25%→1.03%) while Polygon/Avalanche are flat or slightly lower. This suggests proxy patterns are not the dominant tail component, but can

TABLE VII
ERC20/ERC721 SHARES OVERALL VS. MAIN-CUTOFF TAIL.

Chain	ERC20%	ERC721%	Tail ERC20%	Tail ERC721%
BSC	2.88	0.07	0.17	0.04
Polygon	0.35	3.42	< 0.01	< 0.01
Avalanche	0.84	0.55	0.21	< 0.01

TABLE VIII
PROXY INDICATOR PREVALENCE (OVERALL VS. MAIN-CUTOFF TAIL).

Chain	Overall Proxy indicator%	Tail Proxy indicator%(main cutoff)
BSC	0.25	1.03
Polygon	0.69	0.52
Avalanche	0.90	0.62

be active in certain chains' high-score regions. Therefore, ERC/Proxy alone still cannot explain the tail, motivating structural cues below.

3) Selector/opcode enrichment: structural cues of the tail: **Enrichment definition.** For each chain, we compute selector/opcode frequencies in the tail and overall sets, and define lift as $\text{lift}(x) = \frac{p_{\text{tail}}(x)}{p_{\text{all}}(x)}$. To avoid inflation from tiny counts, we only report patterns with counts above a minimum threshold (and apply ϵ smoothing to denominators); opcodes are counted by total occurrences. We further compare selector and opcode distributions between the tail and the overall population, and extract the most enriched patterns (shown below). The tail is dominated by *rare 4-byte selectors* with very high lift (10–50×), rather than standard ERC interfaces, indicating non-standard or specialized logic. On the opcode side, tail contracts show clear enrichment of stack operations and external-call plumbing (e.g., DUP8–DUP11, RETURNDATASIZE/RETURNDATACOPY, GAS, STATICCALL), consistent with obfuscation and template-like logic.

BSC: selectors `0xfe9179cc` (45.0×), `0xd3e1c284` (34.2×), `0xfa483e72` (33.0×); opcodes: DUP10 (2.50×), RETURNDATASIZE (2.21×), GAS (2.19×).

Polygon: selectors `0xfce8337f` (52.6×), `0xef590ca5` (52.6×), `0xe6a43905` (32.6×); opcodes: STATICCALL (2.26×), RETURNDATACOPY (2.08×), GAS (2.04×).

Avalanche: selectors `0xdc7e0ce8` (15.2×), `0x37d20fff` (13.1×), `0x61d027b3` (9.9×); opcodes: DUP11 (1.97×), DUP10 (1.82×), STATICCALL (1.63×).

In addition, the most frequent tail selectors still include `0xf2fde38b` (transferOwnership) and `0x8da5cb5b` (owner), indicating that tail contracts mix governance/permission entry points with non-standard logic. Many high-lift selectors do not map to public standard interfaces, so we treat them as structural cues rather than functional labels.

4) Manual inspection of top tail exemplars: To provide illustrative exemplars, we manually inspect the highest-scoring contract from each chain. These contracts consistently exhibit *very low signature density* (about 0.25/KB), consistent with the rare-selector and external-call enrichment signals, indicating low-visibility, template-like logic.

Consistency with high-risk audit cues. The above signals match common "high explanation-cost" audit cues: rare selectors and low signature density imply sparse, opaque interfaces; enriched external-call and return-data handling suggest more complex inter-contract orchestration; and cross-chain template reuse indicates conditions for bulk deployment and rapid propagation. We therefore treat the high-obfuscation tail as a *higher-priority audit queue*.

Finding 2 (RQ2). The high-score tail can be translated into an *actionable audit profile*: *rare selectors* + *external-call enrichment* + *low signature density* jointly point to hidden-logic, template-driven families. These structural signals can be used directly to triage the RQ1 queue and prioritize likely fast-spreading high-risk templates.

Audit playbook: secondary triage within the tail queue. Given the RQ1 chain-specific screening queue, prioritize contracts that: (1) satisfy *low signature density* + *high external-call enrichment* + *rare selectors*; (2) belong to cross-chain reuse clusters (same bytecode hash on multiple chains); (3) expose `owner/transferOwnership` entry points with unusually sparse selector sets; (4) hit the proxy indicator while implementation logic is opaque (trace the implementation). This triage reduces alert fatigue and concentrates analyst time on templates with the highest explanation cost and propagation potential.

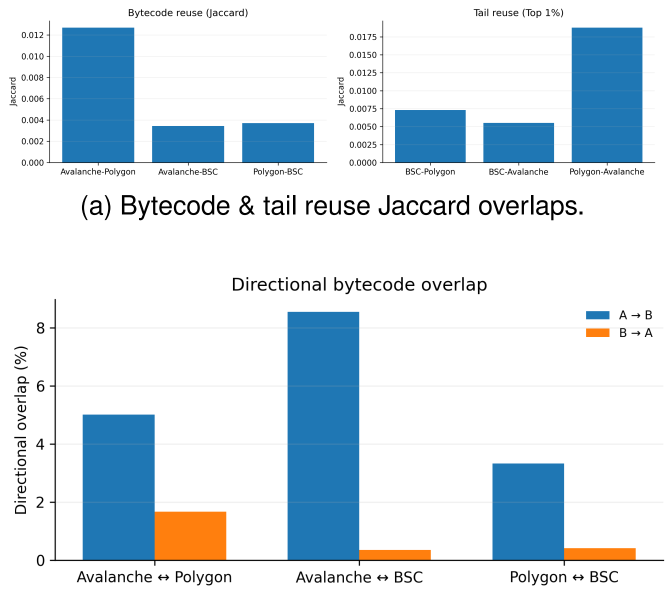
D. RQ3: Cross-Chain Reuse and Propagation

Following RQ2. RQ2 shows that the tail is dominated by a profile of "rare selectors + external-call enrichment + low signature density." The key next question is whether these structural signals are driven by the *same reusable templates propagating across chains*. If so, high-obfuscation risk exhibits cross-chain diffusion and calls for coordinated auditing. We test the existence, directionality, and amplification of cross-chain reuse in the high-obfuscation tail.

1) Overall reuse and directional asymmetry: We use deduplicated runtime bytecode hashes as reuse identifiers and measure cross-chain overlap via Jaccard and overlap coefficients (Table IX). Jaccard values are low (0.003–0.013), indicating that each chain has a largely distinct contract ecosystem. Figure 2 contrasts overall vs. tail reuse; "tail Jaccard" is computed after extracting tail sets (Top1%) within each chain and then computing hash Jaccard across tail sets, while "overall Jaccard" is computed on full deduplicated sets. Tail Jaccard is clearly higher than overall (roughly 0.0055–0.0188 vs 0.0034–0.0127), a 1.5–2.0× enrichment, with Polygon–Avalanche most pronounced; in some pairs it approaches a 2× increase (Fig. 2). Directionally, overlap is much higher from small→large chains than the reverse (e.g., Avalanche→BSC

TABLE IX
OVERALL CROSS-CHAIN BYTECODE REUSE.

Pair	Jaccard	Small→Large	Large→Small
Avalanche–Polygon	0.0127	5.01%	1.67%
Avalanche–BSC	0.0034	8.55%	0.36%
Polygon–BSC	0.0037	3.33%	0.42%



(a) Bytecode & tail reuse Jaccard overlaps.

(b) Directional overlap (small→large chains are higher).

Fig. 2. Cross-chain reuse overlap summaries.

8.55% vs 0.36%, Polygon→BSC 3.33% vs 0.42%), where overlap$(A \rightarrow B) = \frac{|H_A \cap H_B|}{|H_A|}$ and H_A is the deduplicated hash set of chain A. This asymmetry suggests cross-chain reuse is closer to diffusion from smaller ecosystems to larger ones (e.g., copying templates into larger markets for greater visibility and interaction opportunities). Together with the RQ2 profile, this supports the mechanism that template-like, low-visibility contracts are more reusable and thus amplified in cross-chain tails.

2) High-obfuscation templates reuse across chains: We analyze reuse clusters (by bytecode hash) and select high-mean-s clusters. These clusters tend to have long bytecode, low signature counts, and multi-chain coverage, aligning with the "low ABI visibility" profile from RQ2. The statistics show mean-s shifting upward with greater chain coverage; both 2-chain and 3-chain clusters contain high-s outliers (near ∼20), indicating strong cross-chain *reusability/replicable deployment*: the same template reappears across chains with consistently high mean-s.

3) Cross-chain reuse cases: We observe identical bytecode hashes reappearing across multiple chains, and some cases even show identical addresses across chains. These patterns align with template-driven deployment or deterministic address strategies (e.g., CREATE2), providing traceable evidence of cross-chain reuse. Identical-address reuse further suggests deterministic deployment or unified scripting; defensively, this enables *cross-chain tracing and preemptive alerting* once a high-risk template is identified on any single chain.

Finding 3 (RQ3). Overall reuse is low, but tail reuse is markedly enriched (Jaccard up by ∼1.5–2.0×) and directionally asymmetric (small→large is far higher than the reverse). This suggests reuse is closer to diffusion from smaller ecosystems to larger ones, where high-score templates are copied into bigger markets and create cross-chain spillovers.

Audit Action (RQ3). Once a high-score reuse cluster is detected on any chain, immediately search and trace the same bytecode hash across other chains to reduce lagged discovery costs.

E. RQ4: Incident Alignment and Investigative Value

Following RQ3. Since high-score templates are reused across chains, RQ4 asks one practical question: *do real cross-chain incidents fall into our high-score screening queue?* If yes, the queue has real-world hit value; if no, it is only a weak clue. We assess this via incident alignment and case-based inspection.

1) Alignment overview: We extract incident-related addresses from public reports and tx/log evidence and match them against deduplicated contracts. Coverage varies by chain (about 31% on BSC, lower on Polygon/Avalanche); tx/log-resolved evidence dominates. While alignment is incomplete, it is sufficient to test whether high-score queues hit real incidents.

2) Percentile localization: real incidents hit the high-score queue: Table X reports aligned samples and their within-chain percentiles (cross-chain spillover cases). During screening, we identified two cross-chain incident samples that fall into the p99 tail but not the p99.9 extreme tail. The picture is consistent: p99 acts as the practical main audit queue, while p99.9 is best reserved for the most urgent, minimal queue. In practice, s should be used for prioritization: review p99 tail first and combine with reuse clusters and low-signature evidence for deeper triage.

3) Case studies of cross-chain spillovers (excerpt): **Transit Swap DEX Hack (2022-10-02, BSC, tx_resolved, p99.74):** public analyses and on-chain fund flows indicate cross-chain fund migration, i.e., a *cross-chain deployment + cross-chain fund migration* spillover type. Its p99 hit shows the high-score queue captures real cross-chain spillover cases.

New Free DAO Flash Loan (2022-09-08, BSC, direct, p99.21): public analyses indicate part of the proceeds moved cross-chain for laundering, i.e., a *single-chain exploit + cross-chain laundering* spillover type; it lands in p99 but not p99.9, reinforcing p99 as the practical main queue.

TABLE X
ALIGNED INCIDENTS (EXCERPT) WITH PERCENTILE LOCALIZATION. A SINGLE INCIDENT MAY APPEAR MULTIPLE TIMES WITH DIFFERENT EVIDENCE TYPES.

Incident	Chain	Evidence	Percentile	In tail (p99/p99.9)
Transit Swap DEX Hack	BSC	tx_resolved	p99.74	Yes / No
New Free DAO Flash Loan	BSC	direct	p99.21	Yes / No

Finding 4 (RQ4). The aligned cross-chain incident samples fall into the p99 tail but not p99.9, indicating that the high-score queue has *real-world hit value* and that p99 is the practical main audit queue. Transit Swap and New Free DAO provide cross-chain spillover evidence in public analyses, showing that high-obfuscation templates can leak across chains in practice. s is therefore a usable prioritization signal to narrow down large contract populations, while p99.9 is best reserved for the most urgent, minimal queue.

F. Summary

This chapter builds a coherent audit narrative across RQ1–RQ4. RQ1 establishes systematic cross-chain drift in obfuscation scores, showing that a single cutoff is not transferable and motivating within-chain percentile queues (p99 as the main queue and p99.9 as the emergency queue). RQ2 then translates the high-score queue into an interpretable structural profile—rare selectors, external-call enrichment, and low signature density—enabling secondary triage within the queue. RQ3 demonstrates directional template reuse across chains, closer to diffusion from smaller ecosystems to larger ones, making cross-chain joint auditing necessary. Finally, RQ4 closes the loop with incident alignment and spillover evidence, showing that the high-score queue has real-world hit value and can serve as a practical audit-prioritization signal. Overall, the chapter connects "distribution drift → structural profile → cross-chain propagation → real-world hits," yielding actionable thresholds and a workflow for cross-chain auditing.

V. DISCUSSION

A. Operational Implications and Audit Workflow

Our results transfer cross-chain obfuscation measurements into an *actionable audit workflow*. RQ1 provides a two-tier queueing strategy: p99 as the main queue for stable coverage, and p99.9 as an emergency queue for urgent triage. RQ2's structural profile (rare selectors, external-call enrichment, low signature density) enables secondary prioritization within the main queue, focusing scarce analyst effort on contracts with the highest interpretation cost. RQ3 motivates cross-chain joint auditing: once a high-score reuse cluster is found on one chain, the same bytecode hash should be searched on other chains. RQ4's incident alignment and case studies show that the p99 queue has real-world hit value and can be used as a practical *prioritization signal*.

Minimal SOP (Audit Workflow)
Step 1 (Queueing): use within-chain p99 for the main queue and p99.9 for the emergency queue.
Step 2 (Secondary triage): re-rank the p99 queue by "low signature density + rare selectors + external-call enrichment."
Step 3 (Cross-chain linkage): for high-score reuse clusters (same bytecode hash), search and aggregate hits on other chains.
Step 4 (Evidence closure): if incident signals or tx/log evidence is found, escalate to manual audit and reporting.

B. Mechanistic Interpretation of Drift and Spillover

Score drift and tail-structure differences are driven by ecosystem-specific contract composition and template reuse. The enrichment of low-visibility logic and complex external-call patterns in the tail suggests that high-score regions are dominated by template-like, hard-to-interpret contracts. Directional reuse indicates that high-score templates diffuse across chains, more often from smaller ecosystems to larger ones, which aligns with a spillover risk channel. This mechanism explains why a single global threshold fails and why cross-chain coordination is needed. **H1 (Template Diffusion Hypothesis):** high-obfuscation templates are more likely to be reused across chains and to diffuse toward larger ecosystems (evidence: tail Jaccard enrichment and directional overlap).
H2 (Low-Visibility Structure Hypothesis): the tail is driven by low-visibility logic and complex call chains (evidence: selector/opcode enrichment and low signature density).

C. Scope and Risk Boundaries

s is a screening and ranking signal rather than a direct "maliciousness" label. The extreme threshold (p99.9) is best viewed as an emergency queue; relying solely on it can miss real incidents. Incident alignment provides evidence of real-world hit value but remains constrained by incomplete public reports and evidence-chain availability. Thus, our contribution is a *prioritization workflow*, not a one-shot classifier. In addition, cross-chain comparison should not directly reuse absolute scores from another chain; and scores should be combined with reuse, visibility, and call-structure cues, otherwise alert fatigue and audit-resource contention can occur.

D. Limitations and Future Work

The surrogate inherits noise from OBFS_TOOL labels and does not directly capture semantic risk. Cross-chain calibration can be further refined. Future work can proceed in two concrete directions: (i) cross-chain calibration via quantile mapping or monotonic calibration functions that map s into unified risk bands; (ii) semantic enhancement by integrating verified source code, known exploit labels, and audit reports into a multi-task "obfuscation + risk semantics" model. More fine-grained template families and interaction graphs are also promising to improve interpretability and cross-chain early warning.

VI. Conclusion

We present HObfNet, a fast surrogate of Obfs_Tool whose inference cost and speed support million-scale, cross-chain scoring. We analyze four chains at scale (ETH 1,042,923; BSC 2,308,899; Polygon 288,611; Avalanche 96,173) and provide operational, *within-chain* thresholds (p99/p99.9; e.g., ETH 18.07/22.69, BSC 16.82/19.74, Polygon 18.72/20.51, Avalanche 19.18/20.67). On top of Ethereum supervision, we answer four research questions: (i) scores drift across chains; (ii) the high-score tail exhibits a structural profile with rare selectors, external-call enrichment, and low signature density, supporting secondary triage; (iii) high-score templates show directional cross-chain reuse, with tail Jaccard about 1.5–2.0× higher than overall; and (iv) incident alignment on the available cross-chain cases (all in p99, none in p99.9) shows real-world hit value of the high-score queue within the aligned evidence set. Collectively, these results yield an actionable cross-chain audit workflow and prioritization queues for security operations.

Future work includes cross-chain calibration that maps s into unified risk bands and semantic enhancement by integrating verified source code/verified contracts, known exploit labels, and audit reports to improve interpretability and early warning.

References

[1] S. Zhang, T. K. Quang, S. Wang, S. Duan, K. Li, and Y. Duan, "Understanding and characterizing obfuscated funds transfers in ethereum smart contracts," *arXiv*, 2025, arXiv:2505.11320.

[2] S. Yang, K. Qin, A. Yaish, and F. Zhang, "Insecurity through obscurity: Veiled vulnerabilities in closed-source contracts," *arXiv*, 2025, arXiv:2504.13398.

[3] N. Grech, L. Brent, B. Scholz, and Y. Smaragdakis, "Gigahorse: Thorough, declarative decompilation of smart contracts," in *Proceedings of the 41st International Conference on Software Engineering (ICSE)*, 2019.

[4] J. He, S. Li, X. Wang, S.-C. Cheung, G. Zhao, and J. Yang, "Neural-febi: Accurate function identification in ethereum virtual machine bytecode," *Journal of Systems and Software*, vol. 199, p. 111627, 2023.

[5] N. He, L. Wu, H. Wang, Y. Guo, and X. Jiang, "Characterizing code clones in the ethereum smart contract ecosystem," in *Financial Cryptography and Data Security*, 2020.

[6] X. Chen, P. Liao, Y. Zhang, and Y. Huang, "Understanding code reuse in smart contracts," in *2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2021, pp. 470–481.

[7] Z. Wang, J. Chen, T. Zhang, Y. Zhang, W. Zhang, Y. Feng, and Z. Zheng, "Copy-and-paste? identifying evm-inequivalent code smells in multi-chain reuse contracts," *Proceedings of the ACM on Software Engineering*, vol. 2, no. ISSTA, pp. 1031–1053, 2025.

[8] F. Cernera, M. La Morgia, A. Mei, A. M. Mongardini, and F. Sassi, "The blockchain warfare: Investigating the ecosystem of sniper bots on ethereum and bnb smart chain," *ACM Transactions on Internet Technology*, vol. 25, no. 3, pp. 15:1–15:29, 2025.

[9] Y. Lin *et al.*, "Track and trace: Automatically uncovering cross-chain transactions in the multi-blockchain ecosystem," *arXiv*, 2025, arXiv:2504.01822.

[10] T. Labs, "Illicit crypto ecosystem report 2023," urlhttps://www.trmlabs.com/reports-and-whitepapers/the-illicit-crypto-economy-2023, 2023, accessed 2026-01-24.

[11] M. Zhang, P. Zhang, X. Luo, and F. Xiao, "Source code obfuscation for smart contracts," in *2020 27th Asia-Pacific Software Engineering Conference (APSEC)*, 2020, pp. 513–514.

[12] Q. Yu, P. Zhang, H. Dong, Y. Xiao, and S. Ji, "Bytecode obfuscation for smart contracts," in *2022 29th Asia-Pacific Software Engineering Conference (APSEC)*, 2022, pp. 566–567.

[13] P. Zhang *et al.*, "Bian: Smart contract source code obfuscation," *IEEE Transactions on Software Engineering*, vol. 49, no. 9, pp. 4456–4468, 2023.

[14] N. Grech, S. Lagouvardos, I. Tsatiris, and Y. Smaragdakis, "Elipmoc: Advanced decompilation of ethereum smart contracts," *Proceedings of the ACM on Programming Languages*, vol. 6, no. OOPSLA1, pp. 1–27, 2022.

[15] S. Lagouvardos, Y. Bollanos, N. Grech, and Y. Smaragdakis, "The incredible shrinking context... in a decompiler near you," *Proceedings of the ACM on Software Engineering*, vol. 2, no. ISSTA, pp. 1350–1373, 2025.

[16] G. Feist, G. Grieco, and A. Groce, "Slither: A static analysis framework for smart contracts," *arXiv*, 2019, arXiv:1908.09878.

[17] P. Tsankov, A. Dan, D. Drachsler-Cohen, A. Gervais, F. Bünzli, and M. Vechev, "Securify: Practical security analysis of smart contracts," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2018.

[18] S. Kalra, S. Goel, M. Dhawan, and S. Sharma, "Zeus: Analyzing safety of smart contracts," in *Proceedings of the Network and Distributed System Security Symposium (NDSS 2018)*, 2018.

[19] M. di Angelo, T. Durieux, J. F. Ferreira, and G. Salzer, "Evolution of automated weakness detection in ethereum bytecode: A comprehensive study," *Empirical Software Engineering*, vol. 29, p. 41, 2024.

[20] J. Huang, S. Han, W. You, W. Shi, B. Liang, J. Wu, and Y. Wu, "Hunting vulnerable smart contracts via graph embedding based bytecode matching," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2144–2156, 2021.

[21] J. Sun, Y. Jia, Y. Wang *et al.*, "Ethereum fraud detection via joint transaction language model and graph representation learning," *Information Fusion*, vol. 120, p. 103074, 2025.

[22] Z. Sheng, L. Song, and Y. Wang, "Dynamic feature fusion: Combining global graph structures and local semantics for blockchain phishing detection," *IEEE Transactions on Network and Service Management*, vol. 22, no. 5, pp. 5202–5214, 2025.

[23] Y. Xia *et al.*, "Symgpt: Auditing smart contracts via combining symbolic execution with large language models," *arXiv*, 2025, arXiv:2502.07644.

[24] E. Balci *et al.*, "Examining the effectiveness of transformer-based smart contract vulnerability scan," *Journal of Systems and Software*, vol. 231, p. 112593, 2026.

[25] Y. Liao, H. H. Nguyen, N. M. Nguyen, C. Xie, Z. Ahmadi, D. Kudendo, T.-N. Doan, and L. Jiang, "Mando-hgt: Heterogeneous graph transformers for smart contract vulnerability detection," in *Proceedings of the 20th International Conference on Mining Software Repositories (MSR 2023)*, 2023, pp. 334–346.

[26] J. Yang, S. Liu, S. Dai, Y. Fang, K. Xie, and Y. Lu, "Byteeye: A smart contract vulnerability detection framework at bytecode level with graph neural networks," *Automated Software Engineering*, vol. 33, p. 24, 2026.

[27] F. Khan, I. David, D. Varro, and S. McIntosh, "Code cloning in smart contracts on the ethereum platform: An extended replication study," *IEEE Transactions on Software Engineering*, vol. 49, no. 4, pp. 2006–2019, 2023.

[28] Z. Wang, Z. Wan, Y. Chen, Y. Zhang, D. Lo, D. Xie, and X. Yang, "Clone detection for smart contracts: How far are we?" *Proceedings of the ACM on Software Engineering*, vol. 2, no. FSE, pp. 1249–1269, 2025.

[29] Z. Liu, L. Ma, Z. Mu, C. Wei, X. Xu, Y. Jiao, and K. Ren, "I know who clones your code: Interpretable smart contract similarity detection," *arXiv*, 2025, arXiv:2509.09630.

[30] J. Yi *et al.*, "Blockscope: Detecting and investigating propagated vulnerabilities in forked blockchain projects," in *Proceedings of the Network and Distributed System Security Symposium (NDSS 2023)*, 2023.

[31] S. Duan, Y. Xu, S. Zhang, S. Wang, and Y. Duan, "Pdlogger: Automated logging framework for practical software development," *arXiv*, 2025, arXiv:2507.19951.

[32] K. Qin *et al.*, "The blockchain imitation game," in *Proceedings of the 32nd USENIX Security Symposium (USENIX Security 2023)*, 2023.

[33] A. Augusto, R. Belchior, M. Correia, A. Vasconcelos, L. Zhang, and T. Hardjono, "Sok: Security and privacy of blockchain interoperability," in *Proceedings of the 45th IEEE Symposium on Security and Privacy (SP 2024)*, 2024, pp. 3840–3865.

[34] N. Belenkov, V. Callens, A. Murashkin, K. Bak, M. Derka, J. Gorzny, and S.-S. Lee, “Sok: A review of cross-chain bridge hacks in 2023,” *arXiv*, 2025, arXiv:2501.03423.

[35] C. Team, “Vulnerabilities in cross-chain bridge protocols emerge as top security risk,” urlhttps://www.chainalysis.com/blog/cross-chain-bridge-hacks-2022/, 2022, accessed 2025-01-01.

[36] T. Yan, C. Huang, and C. J. Tessone, “Tracing cross-chain transactions between evm-based blockchains: An analysis of ethereum-polygon bridges,” *Ledger*, vol. 10, pp. 113–134, 2025.

[37] N. Shadab, F. Houshmand, and M. Lesani, “Cross-chain transactions,” in *Proceedings of the 2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2020, pp. 1–9.

[38] U. Securities and E. Commission, “Sec charges eleven individuals in 300 million crypto pyramid scheme,” urlhttps://www.sec.gov/newsroom/press-releases/2022-134, 2022, accessed 2026-01-18.

[39] S. Zhang, L. Duan, and H. Jiang, “Accelerating gaussian beam tracing method with dynamic parallelism on graphics processing units,” *Computer Physics Communications*, vol. 315, p. 109722, 2025.

[40] S. Zhang and H. Zhu, “Optimizing acoustic field rendering through heterogeneous computing,” in *Fifth International Conference on Signal Processing and Computer Science (SPCS 2024)*, vol. 13442. SPIE, 2025, pp. 316–321.