Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.082384

ARTICLE

DSPT: Distributed Similar Payload Traceback Based on Bloom Filter

Changsheng Hou[1], Xionglve Li[2], Bingnan Hou[2], Zhiping Cai[2], Jingtao Hu[1,*], Shuai Ye[1] and Hao Li[1]

[1]Academy of Military Sciences, Beijing, China
[2]College of Computer, National University of Defense Technology, Changsha, China
*Corresponding Author: Jingtao Hu. Email: hujingtao17@nudt.edu.cn

Received: 15 March 2026; Accepted: 13 July 2026; Published: 13 August 2026

ABSTRACT: Malicious network attacks pose severe threats to cyberspace, and efficient post-incident traceback and forensics techniques are urgently demanded. Existing payload attribution methods mainly support exact matching, while similar-payload schemes suffer from low efficiency and excessive overhead; most are single-node solutions that fail against IP spoofing and stepping-stone attacks, and the distributed Topology-aware Single Packet IP Traceback System (TOPO) relies on full-node cooperation and flooding forwarding, leading to huge overhead and a nearly 100% false positive rate. To mitigate these issues, we propose Distributed Similar Payload Traceback (DSPT), a distributed system that achieves hop-by-hop traceback via upstream cooperative notice without flooding, and uses packet caching and non-shingling to improve the accuracy of malicious traffic and variant tracing. Extensive experiments on real topologies and campus traffic show that DSPT supports efficient traceback for excerpts of different lengths, reduces the false positive rate to below 26% even in similar-payload scenarios, and achieves much lower average false positives and query time than TOPO.

KEYWORDS: IP traceback; similar traffic detection; similar traffic traceback; distributed similar traffic traceback; attack attribution; network security; bloom filter

1 Introduction

Cybercrimes constantly threaten computer networks. Proactive prevention often fails against emerging threats, including undetected worm outbreaks and internal sensitive data leaks. Thus, we need reliable post-incident analysis methods alongside traditional defense solutions. These methods trace malicious traffic and assist forensic analysis based on traffic features. The Payload Attribution System (PAS) enables long-term continuous packet capture and network traffic recording. As such, payload attribution technology serves as a core foundation for network forensics and cybercrime investigations [1–4].

Researchers continuously optimize current payload attribution systems to boost data compression ratio and lower false positive rates [5,6]. Still, these systems ignore detection for excerpts and their similar variants. Most existing methods only support exact excerpt matching. They suffer severe performance loss when handling similar payload variants. Hosseini et al. adopt a Digital Signal Processing (DSP)-based method to enable similar payload queries [7]. However, this approach suffers from low processing speed. Moreover, as a flow-based solution, it stores full payloads for all network flows on network nodes. This design causes excessive storage consumption and heavy computational overhead.

In addition, most existing payload attribution techniques are single-node schemes and cannot defend against IP spoofing and stepping-stone attacks. Attackers that compromise adjacent nodes of the deployed PAS device will mislead traceback procedures and generate invalid or wrong node results, as illustrated in Fig. 1. Zhang and Guan proposed TOPO, a distributed topology-aware traceback system that performs traceback using router upstream information [8]. However, this approach assumes all network routers are trusted, controllable, and deployable on arbitrary nodes, which is an unrealistic premise. Zhang and Guan further proposed a partial deployment strategy, while it still requires non-deployed nodes to cooperate and forward requests via flooding. Such full-cooperation assumptions remain impractical. Furthermore, even with full cooperation, flooding incurs additional overhead and increases false positives. These unreasonable deployment constraints and inefficiencies severely limit TOPO's practical applicability. Therefore, it is necessary to design an effective distributed method for detecting and tracing similar traffic to counter IP spoofing attacks and stepping-stone attacks, thereby locating malicious sources quickly and accurately.

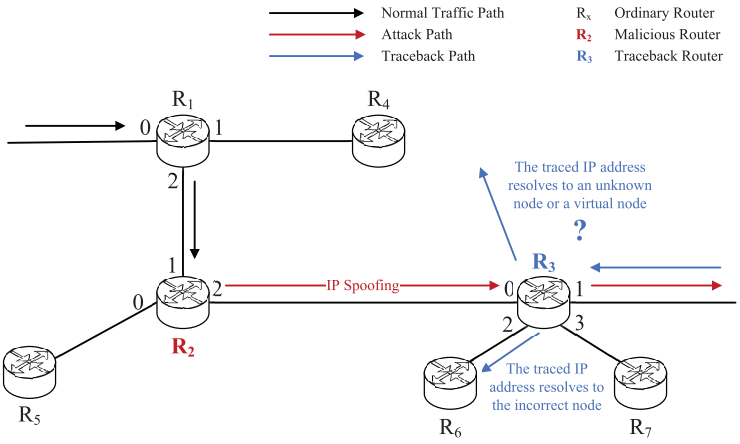


Figure 1: The impact of IP spoofing on malicious traffic traceback.

This paper presents Distributed Similar Payload Traceback (DSPT), a distributed system for detecting and tracing similar traffic based on upstream cooperative node notices. DSPT is deployed on trusted cooperative nodes to monitor passing network flows. It generates compact traffic digests and maintains upstream node information in packet headers. Upon detecting malicious traffic, the victim initiates a hop-by-hop traceback toward upstream cooperative nodes without network-wide flooding. The system identifies the traffic source by querying the Bloom Filter and checking recorded upstream information or ingress interfaces. To improve the accuracy of tracing malicious traffic and its variants, DSPT adopts packet caching and non-shingling mechanisms. Extensive experiments on real topologies and campus traffic show that DSPT supports efficient traceback for excerpts of different lengths, reduces the false positive rate to below 26% even in similar-payload scenarios, and achieves much lower average false positives and query time than the state-of-the-art method TOPO.

Unlike single-node payload attribution schemes such as BIFM, which only support isolated local payload detection, DSPT integrates an upstream cooperative notice mechanism to realize distributed multi-hop traceback against IP spoofing and stepping-stone attacks. Different from distributed traceback approaches (e.g., TOPO), constrained to exact payload matching, DSPT adopts relaxation-constrained matching to identify variant and obfuscated payloads. Rather than a simple incremental improvement over existing studies, DSPT constructs a collaborative digest architecture that unifies similar payload

detection and lightweight distributed traceback, providing a distinctive and practical solution for network security forensics.

The main contributions are as follows:

1. We propose DSPT, a distributed similar traffic detection and traceback system based on upstream cooperative node notices.
2. We present a traceback mechanism based on interface information notice among upstream cooperative nodes, which avoids flooding traceback requests across the entire network, improves query efficiency, and reduces the false positive rate of traceback.
3. We conduct comprehensive experiments on real topologies and campus traffic. Results validate that DSPT achieves superior performance over existing schemes in terms of false positive rate, the average number of false positives, and query time.

The rest of this paper is organised as follows. Section 2 reviews related work, including approximate matching, single-node payload attribution, and distributed payload attribution approaches. A comparative table summarises the key features of existing methods to illustrate the research gap targeted by DSPT. Section 3 details the complete design of DSPT: we define the threat model, present the two-stage system workflow, elaborate on the distributed network flow digest algorithm and similar payload query algorithm, and further introduce the practical deployment strategy of DSPT. Section 4 provides a theoretical analysis of DSPT's detection accuracy as well as time and space complexity. Section 5 presents comprehensive experimental evaluations based on two real network topologies, covering comparison with TOPO, parameter sensitivity analysis, overhead and scalability analysis, and a dedicated discussion on the inherent limitations of DSPT and the real-world implications on DSPT. Section 6 concludes this paper and points out future research directions.

2 Related Work

Network attack forensics and malicious source tracing are essential components of cyberspace security defense. Related research can be divided into three mainstream technical branches: approximate matching, single-node payload attribution, and distributed payload attribution.

2.1 Approximate Matching

In network attacks, adversaries commonly obfuscate malicious payloads via tampering, splitting, or disrupting to bypass security detection mechanisms. Since exact matching algorithms fail to identify attack variants, approximate matching techniques have been developed to enable fuzzy retrieval of similar files or fragments to support network forensic investigation and data leakage tracing [9–12].

Breitinger an Baggili presented mrsh-net [13], a Bloom Filter-based approximate matching solution that splits traffic payloads into blocks and builds a digest database for similar content detection. Subsequent work, mrsh-cf [14], further replaced Bloom Filter with Cuckoo Filter to improve matching efficiency and reduce false positives. Nevertheless, these methods are originally designed for file identification and cannot be directly adapted to similar traffic detection with stable performance.

2.2 Payload Attribution

Approximate matching merely supports offline content comparison. In contrast, PAS is designed for long-term storage and high-speed querying of full-network traffic, enabling real-time forensic tracing subsequent to security breaches.

Early representative work of PAS splits payloads into fixed-size blocks and maps blocks into a Bloom Filter for fast excerpt matching. Subsequent optimization studies focus on balancing compression ratio and detection accuracy. Winnowing Block Shingling (WBS) [6] introduces shingling overlapping and winnowing-based variable block partitioning to avoid exhaustive offset traversal, and Winnowing Multi-Hashing (WMH) [6] improves robustness via multi-parameter collaborative filtering. CBID [5] combines compressed bitmap indexing and traffic downsampling to reduce query overhead, but only supports exact matching. On this basis, BIFM [15] introduces fuzzy matching and non-shingling mechanisms to support variant payload detection, while it is limited to single-node deployment.

Character Dependent Multi-Bloom Filters (CMBF) [16] enable wildcard query by fingerprint modulo operation, but suffer from exponential overhead growth as unknown bytes increase. DSP-based Payload Attribution System (DSPAS) [7] adopts the discrete cosine transform and frequency-domain quantization to implement approximate payload detection. However, it relies on flow-level complete payload recording, resulting in excessive storage cost and low query efficiency.

2.3 Distributed Payload Attribution

All single-node payload attribution schemes cannot complete multi-hop cross-network traceback. When attackers launch IP spoofing or stepping-stone attacks, single-node systems will return the wrong source information. Distributed traceback frameworks are proposed to solve this limitation, yet most existing distributed solutions rely on blind flooding and only support exact payload matching.

Some distributed traceback schemes try to record upstream neighbor information at routers. Renukuntla and Rawat approach [17] stores interface-level mapping information, but still trusts raw packet header fields and is vulnerable to header forgery. As a typical topology-aware distributed traceback framework, TOPO [8] jointly digests payload blocks, flow IDs, and neighbor identifiers into Bloom Filters. However, its fully deployed mode requires full-network cooperative nodes, which is unrealistic, and the partially deployed mode inevitably generates flooding requests on non-deployment nodes, leading to high overhead and unstable detection performance.

Existing distributed traceback methods, such as TOPO, focus on exact payload matching and rely on flooding or full-node cooperation, leading to excessive overhead and extremely high false positive rates. By contrast, single-node similar payload detection approaches, including BIFM and DSPAS, support fuzzy matching, yet they are incapable of conducting distributed traceback to defend against IP spoofing and stepping-stone attacks.

The proposed DSPT integrates the merits of the above two research branches. It inherits the distributed traceback framework from TOPO and introduces similar payload identification mechanisms inspired by BIFM and DSPAS. Equipped with upstream cooperative node notice, packet caching, and non-shingling mechanisms, DSPT seamlessly unifies distributed traceback and similar payload detection into a lightweight, robust, and practical framework. Therefore, DSPT constitutes a reasonable and progressive improvement over current mainstream traceback solutions. Table 1 summarizes a comparison of existing methods.

Table 1: Comparison of existing methods.

Method	Deployment	Similar Query	Wildcard Query	Distributed Traceback
mrsh-net [13]/mrsh-cf [14]	Single-node	✓	×	×
WBS [6]/WMH [6]/CBID [5]	Single-node	×	×	×
BIFM [15]	Single-node	✓	×	×
CMBF [16]	Single-node	×	✓	×
DSPAS [7]	Single-node	✓	✓	×
TOPO [8]	Distributed	×	×	✓
DSPT (Ours)	**Distributed**	✓	×	✓

Note: Bold entries denote our proposed method DSPT and its features.

3 The Design of DSPT

DSPT focuses on distributed similar traffic detection and traceback. It aims to summarize passing network traffic. When it detects malicious activities, it traces malicious traffic and its similar variants hop by hop to approach the attacker's actual location.

3.1 Threat Model

We formally define the threat model and security assumptions.

3.1.1 Trusted Entities

The victim host, critical routers (e.g., backbone routers, gateway routers), and nodes with high protection levels are trusted. These nodes perform standard network functions correctly and do not tamper with valid packet header information or payloads. They are allowed to use unused header fields to carry notice messages.

3.1.2 Untrusted Entities

Ordinary hosts and routers are untrusted. These nodes may be controlled by attackers or compromised and deceived.

3.1.3 Adversary Capabilities

The attacker can forge packet headers or modify payload content, specifically including the following capabilities.

- IP Spoofing: The attacker can forge the source IP address of attack packets to evade traceback.
- Traffic Injection: The attacker can send malicious packets to frame innocent hosts that have not established valid sessions with the victim.
- Stepping-Stone Attack: The attacker can compromise poorly protected routers or hosts and use a chain of compromised hosts (stepping stones) to launch attacks, intercept, and modify packets.
- Attack Path Forgery: The attacker can manipulate path information to mislead the traceback.
- Payload Obfuscation: The attacker can split and disrupt critical malicious content in attack packets to avoid detection.

3.1.4 Possible Attacks

- Command Injection: The attacker constructs malicious packets to inject system or application commands, such as system command injection, SQL injection, expression injection, or Trojan upload.
- Unauthorized Data Access/Modification: The attacker modifies request parameters to access or tamper with sensitive data, e.g., identity hijacking, application data tampering, or internal data leakage.
- Malware Propagation: The attacker embeds worms, Trojans, viruses, mining scripts, or espionage payloads to spread malicious programs.

3.1.5 Collusion Attacks

Untrusted nodes may collude to launch collusion attacks. Collusion attackers can jointly forge attack paths, resulting in a failed traceback. They can also perform Distributed Denial of Service Attacks (DDoS), such as flooding attacks, to crash trusted nodes or disable traffic digest/traceback services.

3.1.6 Security Assumptions

The victim node is honest and trusted.

DSPT is deployed on trusted nodes, referred to as cooperative nodes. Cooperative nodes are typically deployed in secure, managed environments such as institutional gateways or backbone routers, which are well protected against unauthorized access. Thus, cooperative nodes are trusted, unmodified, and execute DSPT correctly.

3.1.7 Scope and Limitations

Under the above assumptions, DSPT can effectively trace attacks such as command injection, unauthorized access, and malware propagation, and defend against IP spoofing, traffic injection, stepping-stone attacks, path forgery, and payload obfuscation. However, DSPT has the following limitations.

- Compromised Cooperative Nodes: Since DSPT fully trusts cooperative nodes, it cannot perform correct traceback if a cooperative node is compromised.
- Denial of Service Attacks (DoS): DoS attacks may crash cooperative nodes or disable DSPT services, interrupting traceback.
- Compression and Encryption: If payloads are compressed or encrypted, DSPT only supports querying the compressed or encrypted content.

3.2 Overview of DSPT

The DSPT system is deployed on trusted cooperative network nodes. Its workflow includes two core stages: network traffic digest and similar payload query, as shown in Fig. 2. In the network traffic digest stage, DSPT compresses and stores incoming packets to generate traffic digests. It also sends local node information to downstream cooperative nodes to support subsequent hop-by-hop traceback and investigation.

In the similar payload query stage, the victim node sends a traceback request with the target excerpt to its nearest cooperative node. DSPT extracts the excerpt and runs a local query. If the query reveals that the excerpt might come from an upstream node, DSPT forwards the traceback request to that node. If not, it checks the peer IPs of all local interfaces. Upon a successful match, DSPT locates the malicious traffic source and sends a traceback confirmation packet to the victim node. DSPT supports queries for malicious traffic and its similar variants.

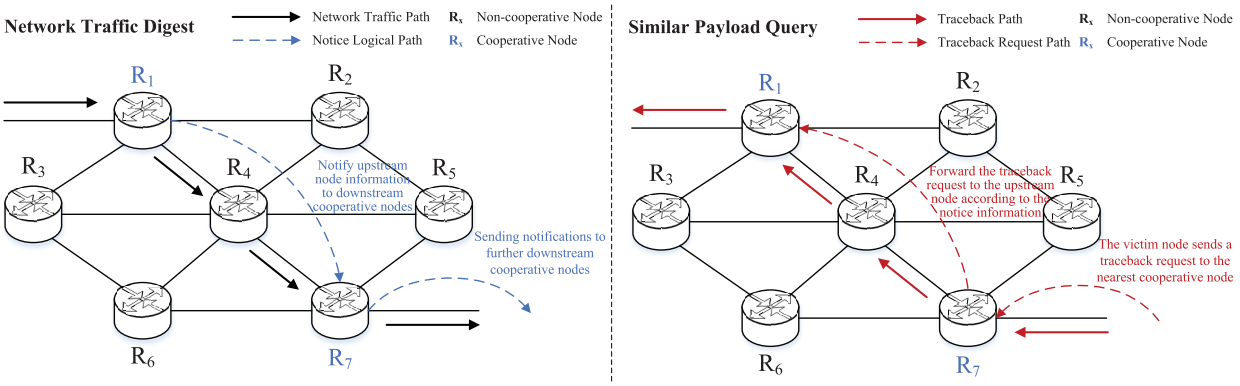


Figure 2: The framework of DSPT.

Unlike BIFM, DSPT is a distributed scheme for similar traffic detection and traceback. DSPT nodes only trust and store data from trusted cooperative nodes and key protected targets. Even if adjacent nodes tamper with malicious traffic sources, DSPT can forward traceback requests to nodes closer to the attacker for accurate tracing. DSPT only requires control over cooperative nodes, rather than full-network management privileges. More widely distributed cooperative nodes further enhance its traceback capability.

For the convenience of subsequent explanations, we first define symbols in Table 2.

Table 2: Notation table.

Notation	Description
S_{Packet}	Incoming continuous packet sequence (algorithm input)
P	Single network packet
P_{Req}	Traceback request packet
P_{Ret}	Traceback confirmation packet
$Flow\_ID$	Flow identifier, composed of source IP and destination IP
$Payload$	Original payload content of a network packet
$Excerpt$	Query payload excerpt for similar traffic detection
$Payload_{comp}$, $Excerpt_{comp}$	Composite content of payload/excerpt and peer IP address
BF	Bloom Filter for network traffic digest storage
L_C	Local flow information cache list
$L_{Notified}$	List of notified upstream node IP addresses
$L_{RemoteIP}$	List of peer IP addresses of local ingress interfaces
N_{Flow}	Total number of recorded network flows
$N_{Notified}$	Total number of notified IP entries
Idx	Index of a specified flow in L_C
$NotifiedIP$	IP address of notified upstream cooperative node
$RemoteIP$	Peer IP address of local ingress network interface
$Cache$	Cached trailing fixed-length payload content for flow continuity
$isNotified$	Boolean flag indicating whether the flow carries upstream notice information
$Count_{raw}$	Raw block hit count of original excerpt query
$Count_{comp}$	Block hit count of composite content query

(Continued)

Table 2 (continued)

Notation	Description
$Query(\cdot)$	Bloom Filter matching and block hit counting function
C_{loosen}	Relaxation coefficient (relaxation threshold)
TH_{hit}	Block hit threshold for preliminary matching judgment
R	Final boolean result of similar traffic detection

3.3 Network Traffic Digest

The network traffic digest stage consists of three steps, as shown in Fig. 3.

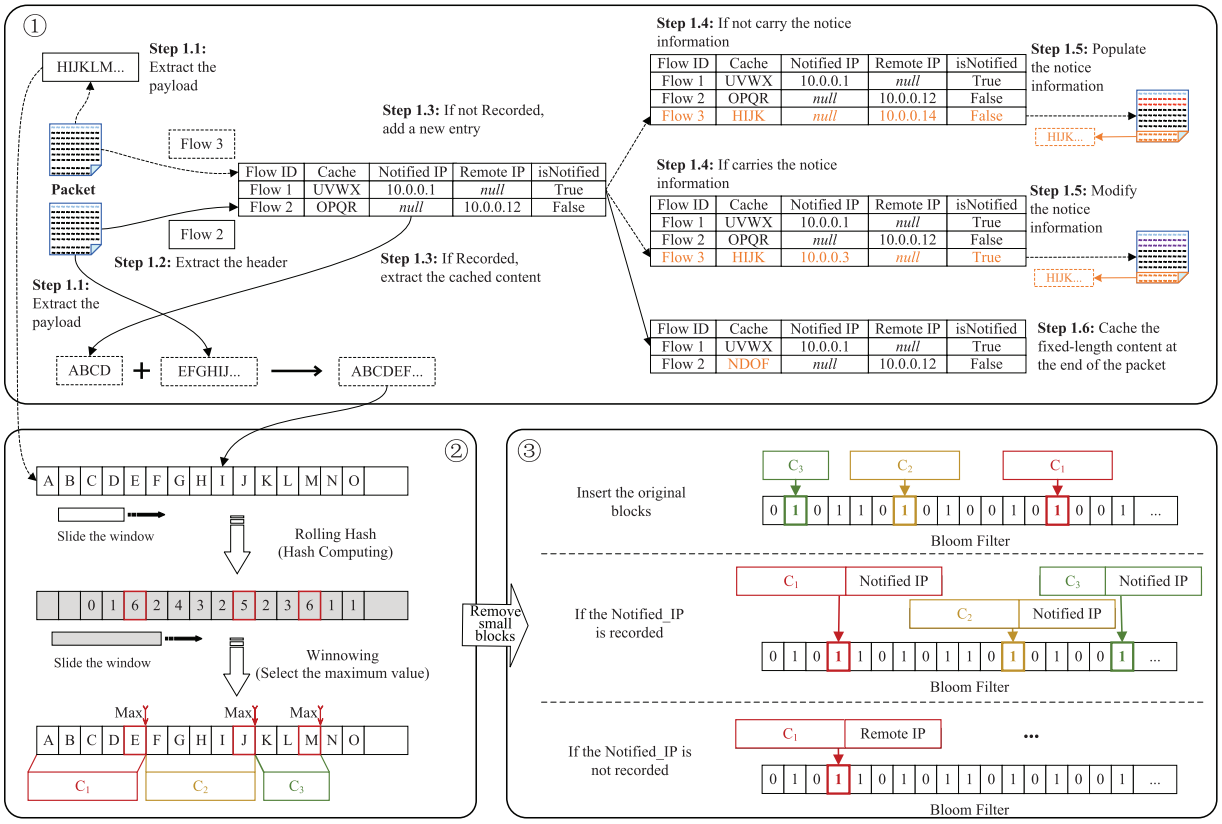


Figure 3: The process of network traffic digest

3.3.1 Step 1: Receiving and Modifying Notice Information

DSPT adopts a packet caching mechanism to guarantee consistent chunking of identical payloads in both traffic digestion and query processes. Upon packet arrival, DSPT extracts the flow identifier from the packet header, then searches the flow information list using this identifier. Each list entry records the cached flow content, the upstream cooperative node IP or ingress interface peer IP, and a flag for received notice information. DSPT checks whether the flow already exists in the list. If present, it retrieves cached content and appends it to the new packet payload; otherwise, it creates a new entry for this flow.

After creating a new flow list entry, DSPT checks whether the packet carries upstream notice information. Such information refers to the egress interface IP. Similar to packet marking methods [18–20], DSPT

stores the notified IP in an unused header field, such as the optional field. Only the first packet of each flow carries this notice.

If the incoming packet carries upstream notice information, DSPT records the upstream cooperative node's IP in the **Notified IP** field of the new entry, and sets the **isNotified** flag to **True**. Meanwhile, DSPT modifies the packet's notice information to the local egress interface IP for subsequent forwarding. If the packet contains no notice information, the local node acts as the first trusted node of the flow. DSPT records the ingress interface peer IP into the **Remote IP** field and sets the **isNotified** flag to **False**. Meanwhile, DSPT embeds the local egress interface IP into the first packet of the new flow as notice information. Whether the packet belongs to a new flow or not, DSPT always caches a fixed-length tail segment of each packet and saves it in the flow information list. It then processes the extracted and concatenated payload.

3.3.2 Step 2: Chunking Payload

Payload chunking consists of three sub-steps. First, a window slides over the payload to calculate rolling hash values. Another winnowing window slides over these hashes, marks the maximum value in each window, and generates block boundaries right after that position. DSPT splits the payload into blocks by these boundaries and eliminates blocks below the preset size threshold via downsampling [5]. DSPT uses a non-shingling mechanism to better support similar payload queries.

3.3.3 Step 3: Mapping and Storing Payload Blocks

This step maps payload blocks into a Bloom Filter for storage.

The Bloom Filter is a space-efficient probabilistic bit-array data structure widely adopted for fast membership checking. Structurally, a standard Bloom Filter consists of a fixed-length bit array initialized to 0 and a group of independent hash functions. It supports two fundamental operations: insertion and query.

- Insertion: During the insertion operation, a target element is mapped to multiple bit positions in the bit array through predefined hash functions. All corresponding bits are set to 1 to record the existence of the element.
- Query: During the query operation, the element is hashed using the same hash functions to locate the corresponding bit positions. If all queried bits are 1, the element is judged to be possibly present; if any bit is 0, the element is definitely not present.

Due to its probabilistic nature, the Bloom Filter may produce false positive results but eliminates false negatives. The false positive rate is determined by the bit array size, the number of hash functions, and the volume of inserted elements, which can be constrained by a reasonable parameter configuration.

In this step, the Bloom Filter compresses and stores two types of blocks: **original blocks** and **composite blocks**. As shown in Fig. 3, an original block is directly generated from payload chunking, while a composite block combines an original block with its flow identifier.

3.3.4 Distributed Network Flow Digest Algorithm

Algorithm 1 details the distributed network flow digest algorithm. It takes continuous incoming packets as input and outputs the traffic digest BF and the notified IP list $L_{Notified}$. DSPT-enabled cooperative nodes run traffic digestion periodically, and upload the generated digest and IP list to persistent storage at the end of the cycle.

Algorithm 1: Distributed network flow digest algorithm

Input: S_{Packet}
Output: BF, $L_{Notified}$

1: Initialize L_C, N_{Flow}, BF, $L_{Notified}$, $N_{Notified}$
2: **for** $P \in S_{Packet}$ **do**
3: $Flow\_ID =< p.srcIP, P.dstIP >$
4: $Payload = P.data$
5: **if** $Flow\_ID \in L_C$ **then**
6: Retrieve the index Idx of the flow $Flow\_ID$
7: $Payload = L_C[Idx].Cache + Payload$
8: **else**
9: Add a new entry to L_C, $N_{Flow}+ = 1$
10: $L_C[N_{Flow}].ID = Flow\_ID$
11: **if** P carries the notice information of the upstream node **then**
12: $NotifiedIP = P.NotifiedIP$
13: $L_C[N_{Flow}].NotifiedIP = NotifiedIP$
14: $L_C[N_{Flow}].isNotified = True$
15: Add a new entry to $L_{Notified}$, $N_{Notified}+ = 1$
16: $L_{Notified}[N_{Notified}] = NotifiedIP$
17: Modify the notice information $P.NotifiedIP$ of packet P to the egress interface IP address
18: **else**
19: Obtain the peer IP address $RemoteIP$ of the ingress interface for the packet P
20: $L_C[N_{Flow}].RemoteIP = RemoteIP$
21: $L_C[N_{Flow}].isNotified = False$
22: Modify P to carry the notice information, and set $P.NotifiedIP$ to the egress interface IP address
23: **end if**
24: **end if**
25: Chunking $Payload$ into blocks and map them into BF for storage
26: Retrieve the index Idx of the flow $Flow\_ID$
27: **if** $L_C[Idx].isNotified == True$ **then**
28: $Payload_{comp} = Payload + L_C[Idx].NotifiedIP$
29: **else**
30: $Payload_{comp} = Payload + L_C[Idx].RemoteIP$
31: **end if**
32: Map and store the composite block $Payload_{comp}$ into BF
33: Store the fixed-length content at the end of $Payload$ into $L_C[Idx].Cache$
34: **end for**
35: **return** BF, $L_{Notified}$

At the start of each period, DSPT initializes the flow information list L_C, Bloom Filter BF, and the notified IP list $L_{Notified}$ (Line 1). DSPT processes incoming packets in sequence. It generates a flow ID from packet IPs and extracts the payload (Lines 3–4). If the flow exists in L_C, DSPT merges cached data with the new payload (Lines 6–7); otherwise, it creates a new entry and configures fields according to packet notice content (Lines 9–23). For notified packets, DSPT refreshes the notice to the local egress IP (Line 17). For non-notified ones, it inserts the local notice into the spare header field (Line 22). Then, DSPT chunks the payload and inserts original blocks into the Bloom Filter (Line 25). According to the $isNotified$ flag, it assembles

payload blocks with $NotifiedIP$ or $RemoteIP$ into composite blocks, which are also stored in the Bloom Filter (Lines 26–32). DSPT caches the fixed-length tail payload into the corresponding flow entry (Line 33). After periodic digestion, DSPT uploads BF, and $L_{Notified}$ to persistent storage (Line 35).

3.4 Similar Payload Query

Similar payload query traces malicious flows carrying target excerpts and identifies obfuscated or modified similar traffic variants. This stage consists of three steps, as shown in Fig. 4.

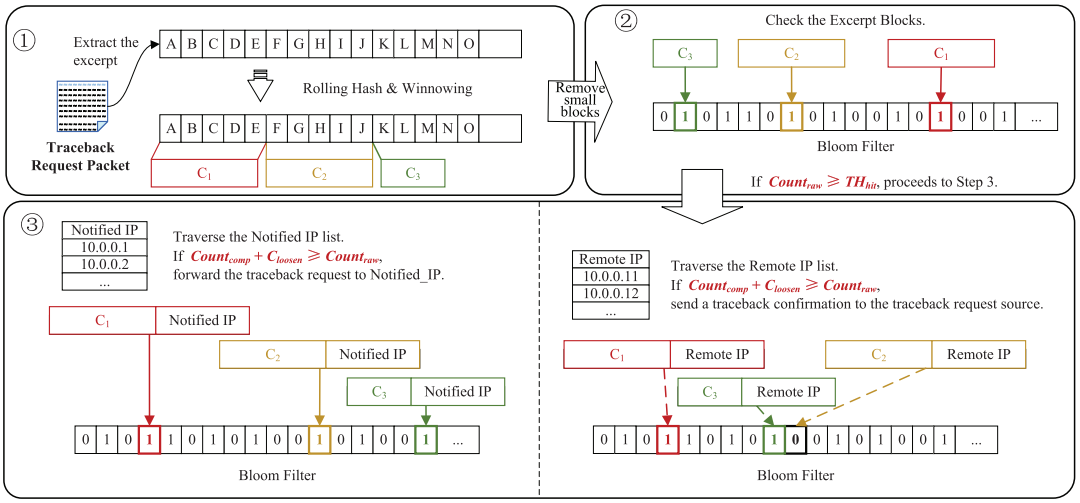


Figure 4: The process of similar payload query

3.4.1 Step 1: Chunking the Excerpt

DSPT first extracts and chunks the target excerpt from the traceback request, following the same chunking rule used in traffic digestion. Since an excerpt is only a partial payload segment, preceding content may mislead most block boundaries in the first winnowing window, except the last one. To avoid inconsistency between digestion and query, DSPT only trusts the final boundary of the first window, discards unreliable front blocks, and starts processing from the first valid boundary.

3.4.2 Step 2: Mapping and Checking the Excerpt Blocks

This step verifies whether the excerpt has appeared locally. DSPT maps excerpt blocks to the Bloom Filter and checks if all corresponding bits are set to 1; if so, the payload block is marked as a hit. If the excerpt's matched block count reaches or exceeds the predefined threshold, the system proceeds to Step 3.

3.4.3 Step 3: Traceback Confirmation or Traceback Request Forwarding

Step 2 confirms the appearance of the excerpt, and Step 3 further locates its source. DSPT traverses the notified and peer IP lists, combines each IP with payload blocks into composite blocks, and checks their matching status.

Attackers may split or disrupt with key payloads to evade detection. As a probabilistic structure, the Bloom Filter may falsely mark split payload blocks as hits, while their corresponding composite blocks still fail to match. This leads to composite block hits being fewer than original block hits, letting real malicious flows evade detection. To address this, a relaxation coefficient is introduced when determining flow sources

to tolerate original block false positives. For instance, if the relaxation coefficient is set to 2, it is assumed that up to 2 false positives may exist among the original block queries.

When traversing the notified IP list, DSPT forwards the traceback request to the suspected upstream node if matched. It keeps the source IP unchanged while updating the destination IP, ensuring upstream query results return promptly to the victim. If no upstream node is found, it checks all local interface peer IPs to locate potential malicious flow sources. Any successful match triggers the immediate delivery of a traceback confirmation packet to the victim.

3.4.4 Distributed Similar Traffic Detection Algorithm

Algorithm 2 illustrates the distributed similar traffic detection algorithm. It takes the traceback request packet P_{Req}, periodic traffic digest BF, notified IP list $L_{Notified}$, relaxation threshold C_{loosen} and block hit threshold TH_{hit} as inputs, and outputs the detection result R.

DSPT initializes the query result and extracts the target excerpt (Lines 1–2), then chunks the excerpt and queries BF to obtain $Count_{raw}$ (Line 3). If $Count_{raw} \geq TH_{hit}$, it traverses $L_{Notified}$ and $L_{RemoteIP}$ (Lines 4–21). Each IP is combined with payload blocks into composite blocks and queried against BF. If $Count_{comp} + C_{loosen} \geq Count_{raw}$, the IP is confirmed as the source. Notified IPs receive the forwarded traceback request; peer IPs trigger a confirmation packet and a positive result. The algorithm finally returns the detection result R (Line 22).

Algorithm 2: Distributed similar traffic detection algorithm

Input: P_{Req}, BF, $L_{Notified}$, C_{loosen}, TH_{hit}
Output: R

1: $R = False$
2: $Excerpt = P.data$
3: $Count_{raw} = Query(BF, Excerpt)$
4: **if** $Count_{raw} \geq TH_{hit}$ **then**
5: **for** $NotifiedIP \in L_{Notified}$ **do**
6: $Excerpt_{comp} = Excerpt + NotifiedIP$
7: $Count_{comp} = Query(BF, Excerpt_{comp})$
8: **if** $Count_{comp} + C_{loosen} \geq Count_{raw}$ **then**
9: Forward the traceback request P_{Req} to $NotifiedIP$
10: **end if**
11: **end for**
12: Query the peer IP addresses of all interfaces to form the list $L_{RemoteIP}$
13: **for** $RemoteIP \in L_{RemoteIP}$ **do**
14: $Excerpt_{comp} = Excerpt + RemoteIP$
15: $Count_{comp} = Query(BF, Excerpt_{comp})$
16: **if** $Count_{comp} + C_{loosen} \geq Count_{raw}$ **then**
17: Send the traceback confirmation P_{Ret} to the traceback request source
18: $R = True$
19: **end if**
20: **end for**

(Continued)

Algorithm 2 (continued)

21: **end if**
22: **return** R

3.5 Practical Deployment of DSPT

DSPT adapts to managed networks such as enterprise intranets, campus networks, and industrial control networks with high demands for attack traceback and lightweight deployment. It focuses on distributed similar payload traceback against IP spoofing and stepping-stone attacks, and can be deployed on existing routing or forwarding devices without full-network hardware replacement.

The deployment cost of DSPT rises with coverage. We adopt an incremental deployment strategy to balance overhead and performance. Initially, we deploy cooperative nodes preferentially on backbone locations: core routers, traffic aggregation points, and nodes near key assets. This maximizes coverage at low cost. Later, we add nodes incrementally to expand coverage, steadily improving detection and traceback performance without network reconstruction or excessive overhead. This cost-effective, scalable deployment enables seamless DSPT integration into real networks and supports sustained iterative optimization.

4 Theoretical Analysis

We provide a theoretical analysis of DSPT, including detection accuracy, space/time complexity, and communication overhead for distributed traceback, offering mathematical guarantees of effectiveness and scalability.

4.1 Detection Accuracy

We adopt a block-based detection mechanism, combined with the Bloom Filter and a relaxation coefficient, to identify malicious traffic and its variants. We first define the notation, then derive the false positive rate for non-malicious excerpts and the false negative rate for malicious excerpts.

- p: per-block false positive probability of the Bloom Filter.
- N_{normal}: total number of blocks in a non-malicious excerpt.
- $N_{malicious}$: total number of blocks in a malicious excerpt (including variants).
- R: number of unmodified original blocks in a malicious excerpt.
- $V = N_{malicious}$: number of modified blocks in a malicious excerpt.

4.1.1 False Positive Rate of the Bloom Filter

Let the Bloom Filter have m bits, k independent hash functions, and store n distinct blocks. The per-block false positive probability is:

$$p = \left(1 - e^{-\frac{kn}{m}}\right)^k$$

4.1.2 False Positive Rate

False Positive (FP) means a non-malicious excerpt is misclassified as malicious.

If $N_{normal} < TH_{hit}$, $FP = 0$.

If $N_{normal} \geq TH_{hit}$, $FP = \sum_{i=TH_{hit}}^{N_{normal}} p^i \cdot p^{i-C_{loosen}} = \dfrac{p^{2TH_{hit}-C_{loosen}} - p^{2N_{normal}-C_{loosen}+2}}{1-p^2}$.

4.1.3 False Negative Rate

False Negative (FN) means a malicious excerpt is misclassified as non-malicious.

If $R < TH_{hit}$, $TP = \sum_{i=TH_{hit}}^{N_m} p^{i-R} \cdot p^{i-R-C_{loosen}} = \frac{p^{2TH_{hit}-2R-C_{loosen}} - p^{2N_m-2R-C_{loosen}+2}}{1-p^2}$. Then, $FN = 1 - TP = 1 - \frac{p^{2TH_{hit}-2R-C_{loosen}} - p^{2N_m-2R-C_{loosen}+2}}{1-p^2}$.

If $R \geq TH_{hit}$ and $V < C_{loosen}$, $FN = 0$.

If $R \geq TH_{hit}$ and $V \geq C_{loosen}$, $FN = \sum_{i=C_{loose}+1}^{V} p^i (1-p)^{i-C_{loose}} = \frac{p^{C_{loose}+1}(1-p) - p^{V+1}(1-p)^{V-C_{loose}+1}}{1-p+p^2}$.

4.2 Time and Space Complexity

Each cooperative node maintains a Bloom Filter and a flow information list. The per-node space complexity is $O(m + N_{flow})$.

In the digest stage, per-packet processing includes rolling hash, chunking, and Bloom Filter insertion. The per-node time complexity is $O(L_{\text{payload}} + k \cdot N_{\text{payload}})$.

In the query stage, per-excerpt processing includes rolling hash, chunking, and Bloom Filter query. The per-node time complexity is $O(L_{\text{excerpt}} + k \cdot N_{\text{excerpt}})$.

Communication Overhead

DSPT performs hop-by-hop traceback without flooding. Let H be the number of cooperative nodes on the attack path. The communication overhead is $O(H)$.

5 Experimental Evaluation

We evaluate the performance of DSPT and perform a comparative analysis with TOPO [8]. We did not compare with single-node payload attribution systems such as BIFM and DSPAS for the following reasons.

- Both BIFM and DSPAS are single-node similar traffic traceback systems capable of local detection of malicious content, yet they do not support multi-hop traceback. In contrast, TOPO is a distributed traceback system oriented to exact matching. This scheme adopts the flooding forwarding mechanism, which incurs substantial network overhead and suffers from a high false positive rate. As a distributed similar payload traceback system, DSPT mainly targets payload tampering attacks (including command injection, unauthorized access, malware propagation, etc.), enables hop-by-hop traceback, and can effectively defend against IP spoofing and stepping-stone attacks.
- Traffic traceback is mainly divided into two phases: detecting malicious traffic and its similar variants, and tracing back the malicious sources. BIFM detects malicious traffic via fuzzy matching and performs traceback based on flow information recorded in the bitmap index table; DSPAS identifies malicious traffic through frequency-domain correlated signals and completes traceback by combining recorded flow information. Both these methods heavily hinge on the premise that information delivered by adjacent nodes is completely credible, making them vulnerable to IP spoofing or stepping-stone attacks. In the first phase, DSPT adopts a fuzzy matching method similar to that of BIFM, while it differs fundamentally from BIFM in the second phase. It accomplishes traceback relying on notice information from trusted upstream cooperative nodes, supports hop-by-hop traceback, and can effectively prevent IP spoofing and stepping-stone attacks. Single-node solutions such as BIFM and DSPAS directly trust source IP addresses and packet forwarding paths. Once confronted with IP spoofing or stepping-stone attacks, their traceback results become completely invalid, with the traceback accuracy of 0% and the

error rate of 100%. Accordingly, it is infeasible to compare the performance of BIFM/DSPAS and DSPT in distributed scenarios.

- DSPT focuses on distributed similar payload traceback, with payload tampering attacks as its core defense target. Apart from research like TOPO, there are few distributed traceback schemes designed to counter such attacks. Most existing distributed traffic traceback systems are optimized and improved versions of packet marking schemes, which are mainly developed to defend against DDoS attacks rather than payload content tampering attacks, and thus fall outside the research scope of this paper.

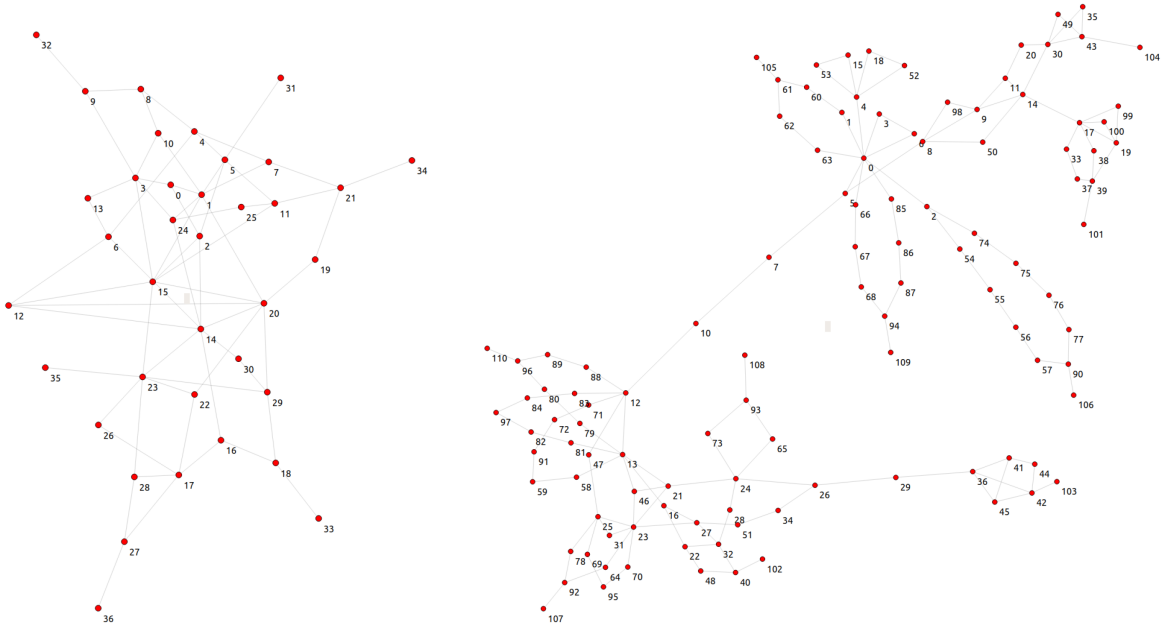
First, we describe the constructed distributed simulation environment and introduce the datasets employed in the experiments. We then elaborate in detail on the parameter settings and evaluation metrics. All experiments were conducted on an Ubuntu host equipped with an Intel Core i9-9900KF processor and 32 GB of RAM, and both DSPT and TOPO are implemented in C++.

5.1 Experimental Environment and Dataset

5.1.1 Experimental Environment

Based on topology data from the Rocketfuel [21] and CAIDA ITDK projects, we build a distributed traffic traceback simulation environment using ns-3. As a discrete-event network simulator, ns-3 is widely adopted in network research. Rocketfuel provides ISP router-level topologies, while ITDK offers global Internet connectivity and routing data for router-level topology research.

We adopt two experimental topologies: Rocketfuel AS3967 and CAIDA ITDK (March 8, 2023), to evaluate DSPT under small-scale and large-scale network scenarios, respectively. Degree-1 source/destination nodes are removed, as they do not forward traffic and contribute little to the simulation. Ultimately, the AS3967 topology retains 37 nodes, including 31 core nodes and 6 source–destination nodes, as shown in Fig. 5a. The ITDK topology retains 111 nodes, including 101 core nodes and 10 source–destination nodes, as shown in Fig. 5b.



(a) AS3967 **(b)** ITDK

Figure 5: The topology of AS3967 and ITDK.

The two topologies are built in ns-3 to establish the distributed traffic traceback simulation environment. In this environment, source nodes generate and send network traffic to destination nodes, while destination nodes receive flow data and send traceback requests to the nearest cooperative node based on malicious flow excerpts. To evaluate the performance of DSPT and TOPO, forwarding nodes in the simulation environment are equipped with the DSPT system and the TOPO system, respectively.

5.1.2 Dataset

Source nodes generate simulated traffic from a real-world dataset captured at a campus core switch in Nanjing. The traffic was collected during peak hours from 20:00 to 20:30 on November 26, 2020.

Due to strict privacy policies, publicly available network traffic datasets usually remove packet payloads to protect user privacy. Under such constraints, we use the T5 corpus [22] to reconstruct packet payloads. This enables performing payload digest and query operations. The reconstructed dataset maintains realistic traffic characteristics, including packet length distribution, flow statistics, and temporal patterns, which ensures the effectiveness of experimental evaluation. Furthermore, the T5 corpus is collected from publicly available U.S. government websites and has been adopted in previous studies to evaluate approximate matching algorithms. The content characteristics of the T5 corpus can represent typical sensitive data theft attacks, in which attackers exfiltrate confidential information via normal traffic. Although the reconstructed payloads cannot cover all network attack behaviors, they are sufficiently representative to verify the performance of DSPT in distributed similar traffic traceback.

We padded the first 16 s of IP-trace data with T5 corpus files to form a 3.38 GB dataset (25,688 flows, 1.6M packets). We select unique excerpts to test query performance and construct split and disrupt datasets. Experiments evaluate DSPT under three scenarios: “unsplit” (exact match), “split”, and “split and disrupt” (similar payload queries), enabling DSPT to return malicious traffic and its variants’ sources.

5.2 Parameter Configurations

We compare DSPT with TOPO [8] on the AS3967 and ITDK topologies. TOPO generates no notice messages during traffic digestion. For traceback, its cooperative nodes forward requests to neighbors or reply with confirmation packets, while non-cooperative nodes relay packets via flooding. TOPO lacks packet caching and non-shingling mechanisms. In AS3967, we deploy 4 source nodes, 2 destination nodes, and 5 distributed cooperative nodes to guarantee traffic coverage. In ITDK, we configure 6 source nodes, 4 destination nodes, and 10 dispersed cooperative nodes.

Experiments adopt a 40:1 data reduction ratio for traffic digestion. We select excerpts of 300, 400, 500, and 600 bytes from 25,688 flows, yielding 7630, 6756, 6225, and 5286 unique excerpts, respectively.

We detail the parameter settings for DSPT and TOPO. Both adopt a rolling hash window of 7, six Bloom Filter hash functions, and a winnowing window of 32. For excerpt lengths of 300, 400, 500, and 600 bytes, the block hit thresholds are set to 4, 8, 11, 14 for the AS3967 topology and 5, 7, 11, 13 for the ITDK topology.

The block hit thresholds are set to ensure that the malicious payload is detectable across “unsplit”, “split”, and “split and disrupt” scenarios while minimizing the false positive rate. Table 3 shows the minimum original block hit counts for query excerpts of different lengths across the three scenarios, and these minimum values are adopted as the final thresholds. This section mainly focuses on verifying the similar traffic traceback performance of DSPT and the effectiveness of upstream cooperative node notices, packet caching, and non-shingling. How to select an appropriate block hit threshold in different network environments to further balance the detection accuracy of malicious flows and the false positive rate remains another challenging issue, which will be explored in future work.

Table 3: Minimum original block hit counts of query excerpts under different scenarios.

Topology	Scenarios	300	400	500	600
AS3967	Unsplit	8	11	14	17
	Split	4	8	11	15
	Split and Disrupt	6	9	11	14
ITDK	Unsplit	8	11	14	17
	Split	5	8	12	13
	Split and Disrupt	5	7	11	14

The Bloom Filter size is derived from the target data reduction ratio. For 3.38 GB of traffic flows digested at a data reduction ratio of 40:1, the memory overhead of the Bloom Filter is 86.5 MB, corresponding to $m = 725849473$. Based on Table 3, the number of blocks for a packet of length L_{Packet} is approximately $0.03L_{\text{Packet}} - 1$. For the 3.38 GB dataset containing 1.6M packets, the total number of inserted blocks is about $n = 107277421$. Using the formula in Section 4.1.1, the false positive rate for querying a single block is 4.13%. When querying payloads with more than 4 blocks, the overall false positive rate drops below 2.92×10^{-6}.

5.3 Metrics

We evaluate DSPT and TOPO using seven metrics:

1. **Accuracy:** Ratio of traceback requests successfully locating the real malicious source to all requests, regardless of additional false sources.
2. **Error Rate:** Ratio of requests tracing solely to incorrect sources.
3. **False Negative Rate:** Ratio of requests failing to find any source. Here, we only consider the false negative rate of exact queries.
4. **False Positive Rate:** Ratio of requests containing incorrect sources among those that successfully identify the true source.
5. **Average Number of False Positives:** Average false sources returned per traceback request.
6. **Average Query Count:** Average maximum forwarding times per request.
7. **Average Query Time:** Average source tracing time per excerpt.

5.4 Experimental Results

5.4.1 Comparison between DSPT and TOPO

We compare the performance of DSPT and TOPO for excerpts of different lengths.

We first compare the traceback accuracy, error rate, and false negative rate of DSPT and TOPO. Tables 4 and 5 list results on the AS3967 and ITDK topologies. The accuracy, error rate, and false negative rate of DSPT and TOPO differ slightly on both small and large topologies. These three metrics are mainly affected by the data compression ratio of each cooperative node. In the "Unsplit" scenario, both DSPT and TOPO achieve 100% accuracy, 0% error rate, and 0% false negative rate, indicating that both DSPT and TOPO can fully traceback malicious traffic. In the "Split" and "Split and Disrupt" scenarios, the accuracy falls below 100% and fluctuates with excerpt length, as do error rate and false negative rate, with the two methods alternately performing the best. Though DSPT integrates packet caching and non-shingling mechanisms to support similar payload queries and should theoretically outperform TOPO, payload block false hits and bounded relaxation coefficients may mislead request forwarding. A false hit on the original payload block

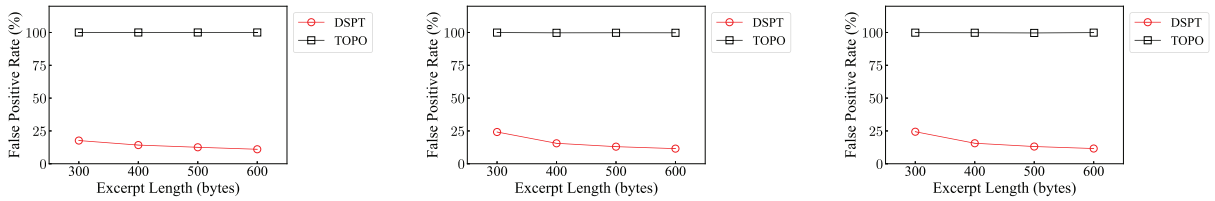
with a small relaxation coefficient may fail to identify the correct upstream node, while a composite block combining a wrong upstream IP and the original payload may exceed the hit threshold, diverting traceback requests. Subsequent false hits at misdirected nodes cause either traceback errors or false negatives. Thus, DSPT occasionally underperforms TOPO slightly, yet their differences in the three metrics are within 0.2%. By contrast, TOPO suffers an impractically high false positive rate, as shown in Figs. 6 and 7.

Table 4: Accuracy, error rate, and false negative rate of DSPT and TOPO for similar payload query on the AS3967 topology.

Scenarios		Unsplit				Split				Split and Disrupt			
Metrics	**Length**	**300**	**400**	**500**	**600**	**300**	**400**	**500**	**600**	**300**	**400**	**500**	**600**
Accuracy	DSPT	100.00%	100.00%	100.00%	100.00%	99.82%	99.79%	99.84%	99.81%	99.80%	99.82%	99.82%	99.85%
	TOPO	100.00%	100.00%	100.00%	100.00%	99.91%	99.73%	99.78%	99.75%	99.86%	99.81%	99.65%	99.87%
Error Rate	DSPT	0.00%	0.00%	0.00%	0.00%	0.18%	0.18%	0.16%	0.19%	0.20%	0.18%	0.18%	0.09%
	TOPO	0.00%	0.00%	0.00%	0.00%	0.08%	0.25%	0.22%	0.25%	0.14%	0.18%	0.32%	0.11%
False Negative Rate	DSPT	0.00%	0.00%	0.00%	0.00%	0.00%	0.03%	0.00%	0.00%	0.00%	0.00%	0.00%	0.06%
	TOPO	0.00%	0.00%	0.00%	0.00%	0.01%	0.01%	0.00%	0.00%	0.00%	0.01%	0.03%	0.02%

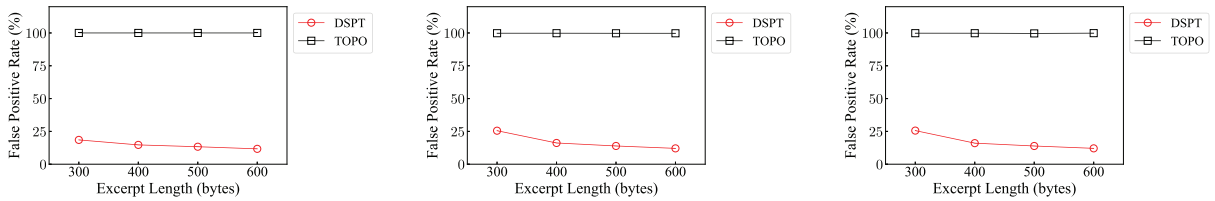
Table 5: Accuracy, error rate, and false negative rate of DSPT and TOPO for similar payload query on the ITDK topology.

Scenarios		Unsplit				Split				Split and Disrupt			
Metrics	**Length**	**300**	**400**	**500**	**600**	**300**	**400**	**500**	**600**	**300**	**400**	**500**	**600**
Accuracy	DSPT	100.00%	100.00%	100.00%	100.00%	99.86%	99.81%	99.65%	99.77%	99.83%	99.82%	99.71%	99.77%
	TOPO	100.00%	100.00%	100.00%	100.00%	99.80%	99.81%	99.76%	99.75%	99.83%	99.81%	99.66%	99.85%
Error Rate	DSPT	0.00%	0.00%	0.00%	0.00%	0.14%	0.19%	0.32%	0.21%	0.17%	0.15%	0.24%	0.23%
	TOPO	0.00%	0.00%	0.00%	0.00%	0.18%	0.19%	0.24%	0.25%	0.13%	0.19%	0.32%	0.15%
False Negative Rate	DSPT	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.03%	0.02%	0.00%	0.03%	0.05%	0.00%
	TOPO	0.00%	0.00%	0.00%	0.00%	0.01%	0.00%	0.00%	0.00%	0.04%	0.00%	0.02%	0.00%



(a) Unsplit **(b)** Split **(c)** Split and Disrupt

Figure 6: False positive rate of DSPT and TOPO for similar payload query on the AS3967 topology.



(a) Unsplit **(b)** Split **(c)** Split and Disrupt

Figure 7: False positive rate of DSPT and TOPO for similar payload query on the ITDK topology.

We next compare the false positive rates of DSPT and TOPO. Figs. 6 and 7 present results on the AS3967 and ITDK topologies, showing consistent trends across network scales. DSPT substantially outperforms TOPO under all three scenarios. In the "Unsplit" scenario, TOPO achieves a 100% false positive rate across all excerpt lengths, making it impractical for real deployment. By contrast, DSPT's false positive rate stays below 19% and declines with excerpt length to a minimum of 11%. DSPT leverages upstream notice to avoid network-wide request flooding, lowering false positives while boosting efficiency. Although TOPO accurately forwards requests at cooperative nodes, non-cooperative nodes adopt blind flooding. Coupled with inherent Bloom Filter false positives, this floods victim nodes with spurious traceback confirmations and severely degrades TOPO's false positive performance. In the "Split" and "Split and Disrupt" scenarios, DSPT's false positive rate rises moderately to a peak near 26%, yet remains far below TOPO's false positive rate of over 99%. Similar payload queries reduce valid block matches, raising misidentification of false sources. DSPT's false positive rate steadily drops with longer excerpts, as more payload blocks reduce erroneous traceback. TOPO's rate fluctuates slightly, dominated by its inherent traceback mechanism. Detailed numerical results are given in Tables 6 and 7.

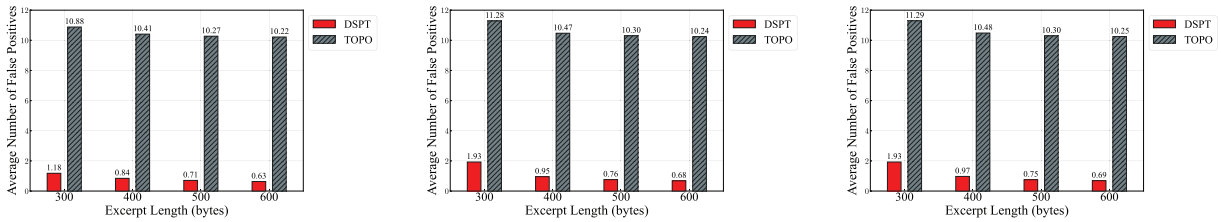
Table 6: False positive rate of DSPT and TOPO for similar payload query on the AS3967 topology.

Scenarios	Unsplit				Split				Split and Disrupt			
Length	**300**	**400**	**500**	**600**	**300**	**400**	**500**	**600**	**300**	**400**	**500**	**600**
DSPT	17.67%	14.22%	12.55%	11.03%	24.18%	15.57%	13.01%	11.52%	24.39%	15.62%	13.11%	11.60%
TOPO	100.00%	100.00%	100.00%	100.00%	99.91%	99.73%	99.78%	99.75%	99.86%	99.79%	99.65%	99.87%

Table 7: False positive rate of DSPT and TOPO for similar payload query on the ITDK topology.

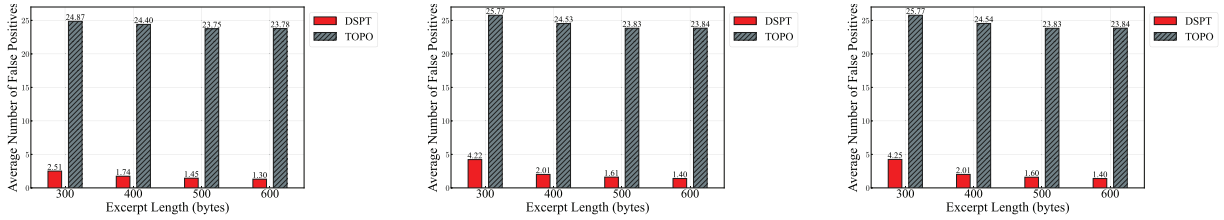
Scenarios	Unsplit				Split				Split and Disrupt			
Length	**300**	**400**	**500**	**600**	**300**	**400**	**500**	**600**	**300**	**400**	**500**	**600**
DSPT	18.52%	14.71%	13.32%	11.71%	25.53%	16.18%	13.90%	12.07%	25.61%	16.04%	13.88%	12.09%
TOPO	100.00%	100.00%	100.00%	100.00%	99.80%	99.81%	99.76%	99.75%	99.83%	99.81%	99.66%	99.85%

We further compare the average number of false positives of DSPT and TOPO. Figs. 8 and 9 show results on the AS3967 and ITDK topologies. The average false positives on the larger ITDK topology are nearly twice those on AS3967, owing to its more deployed cooperative nodes, which induce more false upstream nodes and spurious sources. Across all three scenarios, DSPT yields far fewer average false positives than TOPO, attributed to its notice-based traceback mechanism. Similar payload queries also produce slightly higher false positives than exact matching, as fewer original payload blocks are matched and incorrect sources are more probable. For both methods, longer excerpts reduce the average false positives, since more blocks lower the chance of misidentifying traceback sources.



(a) Unsplit **(b)** Split **(c)** Split and Disrupt

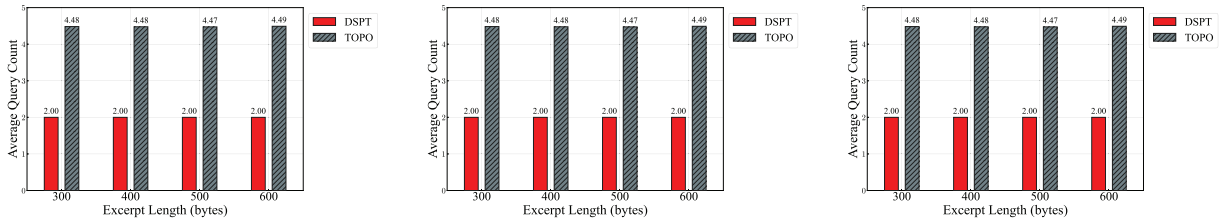
Figure 8: Average number of false positives of DSPT and TOPO for similar payload query on the AS3967 topology.



(a) Unsplit **(b)** Split **(c)** Split and Disrupt

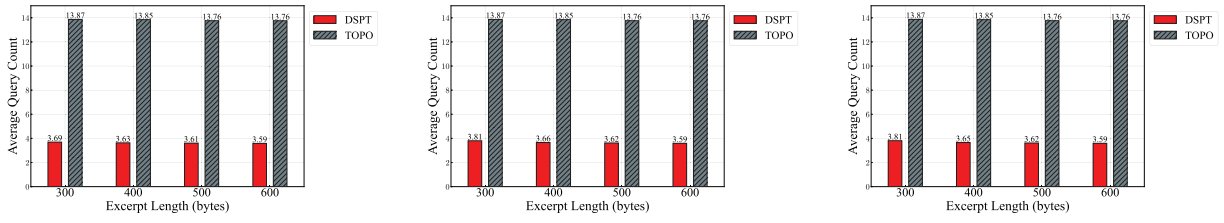
Figure 9: Average number of false positives of DSPT and TOPO for similar payload query on the ITDK topology.

In addition, we compare the average query count of DSPT and TOPO. Figs. 10 and 11 show the results on the AS3967 and ITDK topologies. The larger ITDK topology presents a markedly higher query count than AS3967, due to its more cooperative nodes enabling farther upstream traceback. Across all three scenarios, DSPT achieves a much lower average query count than TOPO. TOPO forces non-cooperative nodes to flood traceback packets, incurring heavy forwarding overhead counted in query rounds. Such flooding also makes cooperative nodes receive requests from multiple interfaces, triggering unnecessary queries to distant wrong nodes and further raising the query count. Besides, exact and similar payload queries yield nearly identical query counts, which remain stable across excerpt lengths. This metric is primarily determined by the network distribution of cooperative nodes.



(a) Unsplit **(b)** Split **(c)** Split and Disrupt

Figure 10: Average query count of DSPT and TOPO for similar payload query on the AS3967 topology.



(a) Unsplit **(b)** Split **(c)** Split and Disrupt

Figure 11: Average query count of DSPT and TOPO for similar payload query on the ITDK topology.

Finally, we compare the average query time of DSPT and TOPO. Tables 8 and 9 present the results on the AS3967 and ITDK topologies. The larger topology yields longer query time due to more cooperative nodes enabling farther upstream traceback. Across all three scenarios, DSPT outperforms TOPO in average query time. The gap is 0.002 s on AS3967 and at least 0.02 s on ITDK. Though DSPT introduces a slight extra overhead by traversing notified IP lists, its notice-based traceback accurately locates upstream nodes and eliminates blind flooding, greatly reducing overall time, with more prominent advantages in the large-scale

topology. For both methods, average query time rises gradually with excerpt length, as longer excerpts generate more blocks and higher processing overhead. Exact and similar payload queries incur nearly identical query time, since DSPT processes excerpts uniformly in both cases.

Table 8: Average query time of DSPT and TOPO on the AS3967 topology (seconds).

Scenarios	Unsplit				Split				Split and Disrupt			
Length	**300**	**400**	**500**	**600**	**300**	**400**	**500**	**600**	**300**	**400**	**500**	**600**
DSPT	0.0183	0.0186	0.0189	0.0192	0.0184	0.0186	0.0189	0.0192	0.0184	0.0186	0.0189	0.0192
TOPO	0.0201	0.0206	0.021	0.0215	0.0201	0.0206	0.021	0.0215	0.0201	0.0206	0.021	0.0215

Table 9: Average query time of DSPT and TOPO on the ITDK topology (seconds).

Scenarios	Unsplit				Split				Split and Disrupt			
Length	**300**	**400**	**500**	**600**	**300**	**400**	**500**	**600**	**300**	**400**	**500**	**600**
DSPT	0.0398	0.0399	0.0404	0.0408	0.0409	0.0402	0.0406	0.0409	0.0409	0.0402	0.0406	0.0409
TOPO	0.0615	0.0628	0.0637	0.0649	0.0615	0.0628	0.0637	0.0649	0.0615	0.0628	0.0637	0.0649

5.4.2 Sensitivity Analysis

We analyze DSPT's parameter sensitivity and the effects of its notice, packet caching, and non-shingling mechanisms.

We first analyze the effect of relaxation coefficients on DSPT. Tables 10 and 11 present DSPT's accuracy, error rate, and false negative rate on the AS3967 and ITDK topologies. The relaxation coefficient exhibits consistent effects across both small and large topologies. It has no influence on the three metrics in the "Unsplit" scenario. In the "Split" and "Split and Disrupt" scenarios, higher coefficients improve accuracy while reducing error and false negative rates. At a relaxation coefficient of 4, DSPT achieves 100% accuracy and 0% error rate for all excerpt lengths; a relaxation coefficient of 3 yields a 0% false negative rate. For similar payload queries, a larger relaxation coefficient helps candidate sources satisfy matching criteria and better include the true source. However, this also raises the false positive rate.

Table 10: Accuracy, error rate, and false negative rate of DSPT for similar payload query under different relaxation coefficients on the AS3967 topology.

Scenarios		Unsplit				Split				Split and Disrupt			
Metrics	**Length**	**300**	**400**	**500**	**600**	**300**	**400**	**500**	**600**	**300**	**400**	**500**	**600**
Accuracy	$C_{loosen} = 0$	100.00%	100.00%	100.00%	100.00%	93.20%	93.31%	93.48%	93.57%	93.67%	93.10%	93.46%	93.74%
	$C_{loosen} = 1$	100.00%	100.00%	100.00%	100.00%	98.23%	98.37%	98.39%	98.47%	98.32%	98.30%	98.23%	98.81%
	$C_{loosen} = 2$	100.00%	100.00%	100.00%	100.00%	99.82%	99.79%	99.84%	99.81%	99.80%	99.82%	99.82%	99.85%
	$C_{loosen} = 3$	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	99.96%	99.97%	100.00%	99.95%	100.00%
	$C_{loosen} = 4$	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
Error Rate	$C_{loosen} = 0$	0.00%	0.00%	0.00%	0.00%	2.16%	2.07%	1.65%	1.53%	2.04%	1.98%	1.85%	1.53%
	$C_{loosen} = 1$	0.00%	0.00%	0.00%	0.00%	1.57%	1.36%	1.27%	1.08%	1.52%	1.35%	1.45%	0.70%
	$C_{loosen} = 2$	0.00%	0.00%	0.00%	0.00%	0.18%	0.18%	0.16%	0.19%	0.20%	0.18%	0.18%	0.09%
	$C_{loosen} = 3$	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.04%	0.03%	0.00%	0.05%	0.00%
	$C_{loosen} = 4$	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
False Negative Rate	$C_{loosen} = 0$	0.00%	0.00%	0.00%	0.00%	4.64%	4.62%	4.87%	4.90%	4.29%	4.91%	4.69%	4.73%
	$C_{loosen} = 1$	0.00%	0.00%	0.00%	0.00%	0.20%	0.27%	0.34%	0.45%	0.16%	0.36%	0.32%	0.49%
	$C_{loosen} = 2$	0.00%	0.00%	0.00%	0.00%	0.00%	0.03%	0.00%	0.00%	0.00%	0.00%	0.00%	0.06%
	$C_{loosen} = 3$	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	$C_{loosen} = 4$	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%

Table 11: Accuracy, error rate, and false negative rate of DSPT for similar payload query under different relaxation coefficients on the ITDK topology.

Scenarios		Unsplit				Split				Split and Disrupt			
Metrics	**Length**	**300**	**400**	**500**	**600**	**300**	**400**	**500**	**600**	**300**	**400**	**500**	**600**
Accuracy	$C_{loosen} = 0$	100.00%	100.00%	100.00%	100.00%	91.52%	91.59%	91.13%	91.71%	91.09%	91.40%	91.16%	91.92%
	$C_{loosen} = 1$	100.00%	100.00%	100.00%	100.00%	97.89%	97.65%	97.72%	98.34%	97.98%	97.90%	97.80%	97.99%
	$C_{loosen} = 2$	100.00%	100.00%	100.00%	100.00%	99.86%	99.81%	99.65%	99.77%	99.83%	99.82%	99.71%	99.77%
	$C_{loosen} = 3$	100.00%	100.00%	100.00%	100.00%	100.00%	99.97%	99.98%	99.96%	100.00%	99.99%	99.97%	100.00%
	$C_{loosen} = 4$	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
Error Rate	$C_{loosen} = 0$	0.00%	0.00%	0.00%	0.00%	2.48%	2.21%	2.18%	1.91%	2.48%	2.22%	2.17%	2.02%
	$C_{loosen} = 1$	0.00%	0.00%	0.00%	0.00%	1.74%	1.78%	1.67%	1.14%	1.66%	1.70%	1.61%	1.49%
	$C_{loosen} = 2$	0.00%	0.00%	0.00%	0.00%	0.14%	0.19%	0.32%	0.21%	0.17%	0.15%	0.24%	0.23%
	$C_{loosen} = 3$	0.00%	0.00%	0.00%	0.00%	0.00%	0.03%	0.02%	0.04%	0.00%	0.01%	0.03%	0.00%
	$C_{loosen} = 4$	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
False Negative Rate	$C_{loosen} = 0$	0.00%	0.00%	0.00%	0.00%	6.00%	6.20%	6.68%	6.38%	6.44%	6.38%	6.67%	6.05%
	$C_{loosen} = 1$	0.00%	0.00%	0.00%	0.00%	0.37%	0.58%	0.61%	0.53%	0.35%	0.40%	0.59%	0.51%
	$C_{loosen} = 2$	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.03%	0.02%	0.00%	0.03%	0.05%	0.00%
	$C_{loosen} = 3$	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	$C_{loosen} = 4$	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%

Tables 12 and 13 list DSPT's false positive rate, average number of false positives, and average query count on the AS3967 and ITDK topologies. The relaxation coefficient exhibits consistent trends across both network scales. In all three scenarios, higher coefficients gradually increase the false positive rate and average number of false positives, as relaxed matching admits more misclassified traffic sources. In the large ITDK topology, the average query count also rises with the relaxation coefficient, since more distant false sources are involved in traceback. In the small AS3967 topology, its simple structure and sparse cooperative node deployment make the coefficient's impact on query count negligible.

Table 12: False positive rate, average number of false positives, and average query count of DSPT for similar payload query under different relaxation coefficients on the AS3967 topology.

Scenarios		Unsplit				Split				Split and Disrupt			
Metrics	**Length**	**300**	**400**	**500**	**600**	**300**	**400**	**500**	**600**	**300**	**400**	**500**	**600**
False Positive Rate	$C_{loosen} = 0$	7.79%	6.87%	6.46%	6.11%	6.50%	5.28%	4.98%	4.82%	6.53%	5.27%	4.85%	4.71%
	$C_{loosen} = 1$	13.60%	11.52%	10.39%	9.23%	14.23%	11.12%	9.61%	8.34%	14.47%	11.12%	9.43%	8.82%
	$C_{loosen} = 2$	17.67%	14.22%	12.55%	11.03%	24.18%	15.57%	13.01%	11.52%	24.39%	15.62%	13.11%	11.60%
	$C_{loosen} = 3$	23.43%	17.14%	14.80%	12.79%	41.45%	20.35%	16.16%	13.55%	40.72%	20.72%	16.03%	13.66%
	$C_{loosen} = 4$	35.33%	20.91%	16.85%	14.68%	66.91%	27.83%	18.60%	15.59%	66.51%	27.68%	18.83%	15.65%
Average Number of False Positives	$C_{loosen} = 0$	0.3967	0.3357	0.3002	0.2995	0.4815	0.3853	0.3335	0.3362	0.4750	0.3778	0.3369	0.3269
	$C_{loosen} = 1$	0.7861	0.6380	0.5584	0.5110	0.9861	0.7096	0.5872	0.5293	0.9982	0.7044	0.5953	0.5302
	$C_{loosen} = 2$	1.1796	0.8431	0.7065	0.6311	1.9275	0.9541	0.7579	0.6809	1.9273	0.9703	0.7526	0.6886
	$C_{loosen} = 3$	1.8581	1.1018	0.8868	0.7594	4.0815	1.4150	0.9875	0.8069	3.9712	1.4332	0.9878	0.8191
	$C_{loosen} = 4$	3.4556	1.4868	1.0763	0.9083	7.5984	2.2772	1.2406	0.9856	7.5482	2.2759	1.2580	0.9767
Average Query Count	$C_{loosen} = 0$	2	2	2	2	2	2	2	2	2	2	2	2
	$C_{loosen} = 1$	2	2	2	2	2	2	2	2	2	2	2	2
	$C_{loosen} = 2$	2	2	2	2	2	2	2	2	2	2	2	2
	$C_{loosen} = 3$	2	2	2	2	2	2	2	2	2	2	2	2
	$C_{loosen} = 4$	2	2	2	2	2	2	2	2	2	2	2	2

Table 13: False positive rate, average number of false positives, and average query count of DSPT for similar payload query under different relaxation coefficients on the ITDK topology.

Scenarios		Unsplit				Split				Split and Disrupt			
Metrics	Length	300	400	500	600	300	400	500	600	300	400	500	600
False Positive Rate	$C_{loosen} = 0$	8.13%	7.31%	6.88%	6.53%	6.59%	5.48%	4.88%	4.65%	6.38%	5.45%	5.03%	4.58%
	$C_{loosen} = 1$	14.19%	12.14%	11.13%	9.99%	14.91%	11.23%	9.98%	9.16%	15.12%	11.41%	10.17%	8.80%
	$C_{loosen} = 2$	18.52%	14.71%	13.32%	11.71%	25.53%	16.18%	13.90%	12.07%	25.61%	16.04%	13.88%	12.09%
	$C_{loosen} = 3$	24.39%	17.72%	15.68%	13.68%	42.63%	21.14%	16.93%	14.32%	42.53%	21.28%	16.96%	14.40%
	$C_{loosen} = 4$	36.29%	21.63%	17.73%	15.49%	67.93%	28.77%	19.81%	16.42%	67.60%	29.06%	19.68%	16.50%
Average Number of False Positives	$C_{loosen} = 0$	0.7988	0.6719	0.5963	0.5972	1.0028	0.7848	0.6748	0.6737	0.9832	0.7795	0.6869	0.6645
	$C_{loosen} = 1$	1.6132	1.3261	1.1393	1.0488	2.0684	1.4444	1.2027	1.0846	2.1180	1.4698	1.2238	1.0823
	$C_{loosen} = 2$	2.5134	1.7429	1.4514	1.3046	4.2229	2.0117	1.6121	1.3992	4.2506	2.0133	1.5964	1.3963
	$C_{loosen} = 3$	4.0978	2.2996	1.8397	1.5821	9.2785	3.0434	2.0953	1.7217	9.2765	3.0459	2.1036	1.6905
	$C_{loosen} = 4$	7.7727	3.2337	2.2606	1.8950	17.2391	5.0970	2.6747	2.0492	17.0971	5.1328	2.6664	2.0651
Average Query Count	$C_{loosen} = 0$	3.5398	3.5292	3.5202	3.5117	3.5261	3.5121	3.4989	3.4916	3.5268	3.5140	3.4986	3.4885
	$C_{loosen} = 1$	3.6155	3.5919	3.5765	3.5579	3.6421	3.5909	3.5752	3.5529	3.6469	3.5965	3.5795	3.5503
	$C_{loosen} = 2$	3.6936	3.6329	3.6082	3.5866	3.8056	3.6591	3.6217	3.5934	3.8111	3.6549	3.6193	3.5936
	$C_{loosen} = 3$	3.7906	3.6881	3.6448	3.6129	4.1166	3.7426	3.6664	3.6252	4.1085	3.7383	3.6649	3.6235
	$C_{loosen} = 4$	4.0093	3.7506	3.6810	3.6453	4.5727	3.8660	3.7131	3.6583	4.5661	3.8712	3.7118	3.6587

Tables 14 and 15 report DSPT's average query time on the AS3967 and ITDK topologies. The relaxation coefficient shows consistent effects across both network scales. In all scenarios, larger relaxation coefficients gradually increase query time, owing to traceback to more distant false sources.

Table 14: Average query time of DSPT under different relaxation coefficients on the AS3967 topology (seconds).

Scenarios	Unsplit				Split				Split and Disrupt			
Length	300	400	500	600	300	400	500	600	300	400	500	600
$C_{loosen} = 0$	0.0181	0.0184	0.0188	0.0191	0.0181	0.0184	0.0187	0.0191	0.0181	0.0184	0.0187	0.0191
$C_{loosen} = 1$	0.0182	0.0185	0.0188	0.0192	0.0182	0.0185	0.0188	0.0192	0.0182	0.0185	0.0188	0.0192
$C_{loosen} = 2$	0.0183	0.0186	0.0189	0.0192	0.0184	0.0186	0.0189	0.0192	0.0184	0.0186	0.0189	0.0192
$C_{loosen} = 3$	0.0184	0.0186	0.0189	0.0193	0.0188	0.0187	0.0189	0.0193	0.0188	0.0187	0.0189	0.0193
$C_{loosen} = 4$	0.0187	0.0187	0.0190	0.0193	0.0194	0.0188	0.019	0.0193	0.0194	0.0189	0.019	0.0193

Table 15: Average query time of DSPT under different relaxation coefficients on the ITDK topology (seconds).

Scenarios	Unsplit				Split				Split and Disrupt			
Length	300	400	500	600	300	400	500	600	300	400	500	600
$C_{loosen} = 0$	0.0383	0.0389	0.0395	0.0400	0.0381	0.0387	0.0392	0.0398	0.0381	0.0387	0.0392	0.0397
$C_{loosen} = 1$	0.0390	0.0395	0.0401	0.0405	0.0393	0.0395	0.0401	0.0405	0.0393	0.0395	0.0401	0.0404
$C_{loosen} = 2$	0.0398	0.0399	0.0404	0.0408	0.0408	0.0402	0.0406	0.0409	0.0409	0.0402	0.0406	0.0409
$C_{loosen} = 3$	0.0407	0.0405	0.0408	0.0411	0.0439	0.0410	0.0411	0.0413	0.0438	0.041	0.0410	0.0413
$C_{loosen} = 4$	0.0428	0.0411	0.0412	0.0415	0.0483	0.0423	0.0416	0.0416	0.0482	0.0423	0.0415	0.0417

Overall, we recommend setting the relaxation coefficient to 2. With this configuration, DSPT maintains over 99% accuracy, an error rate below 0.32%, a false negative rate under 0.06%, and a false positive rate no

more than 26% across all scenarios and excerpt lengths. A larger relaxation coefficient improves accuracy but pushes the false positive rate above 40%, which is impractical.

We next analyze the effects of the notice, packet caching, and non-shingling mechanisms by comparing full DSPT with a version disabling these mechanisms. Tables 16 and 17 list accuracy, error rate, and false negative rate on the AS3967 and ITDK topologies. The three mechanisms have no impact on the three metrics in the "Unsplit" scenario. In the "Split" and "Split and Disrupt" scenarios, the system without the notice mechanism achieves higher accuracy and lower error rate and false negative rate. This is because DSPT's notice mechanism may route requests to incorrect upstream nodes; subsequent false hits at these nodes cause traceback errors or false negatives, slightly degrading accuracy. Nevertheless, Tables 18 and 19 show that the system without the notice mechanism suffers an extremely high false positive rate. Moreover, the version without packet caching and non-shingling mechanisms yields lower accuracy and higher error and false negative rates. By increasing original block match hits in similar payload queries, packet caching and non-shingling effectively improve DSPT's overall performance.

Table 16: Impact of the notice, packet caching, and non-shingling mechanism on the accuracy, error rate, and false negative rate of dspt for similar payload query on the AS3967 topology.

Scenarios		Unsplit				Split				Split and Disrupt			
Metrics	**Length**	**300**	**400**	**500**	**600**	**300**	**400**	**500**	**600**	**300**	**400**	**500**	**600**
Accuracy	DSPT	100.00%	100.00%	100.00%	100.00%	99.82%	99.79%	99.84%	99.81%	99.80%	99.82%	99.82%	99.85%
	Without Notice	100.00%	100.00%	100.00%	100.00%	99.96%	99.94%	99.92%	99.96%	99.95%	99.96%	99.94%	99.96%
	Without Packet Caching	100.00%	100.00%	100.00%	100.00%	99.67%	99.45%	99.55%	99.32%	99.63%	99.63%	99.29%	99.53%
	Shingling	100.00%	100.00%	100.00%	100.00%	99.80%	99.69%	99.73%	99.81%	99.78%	99.73%	99.74%	99.79%
Error Rate	DSPT	0.00%	0.00%	0.00%	0.00%	0.18%	0.18%	0.16%	0.19%	0.20%	0.18%	0.18%	0.09%
	Without Notice	0.00%	0.00%	0.00%	0.00%	0.04%	0.06%	0.08%	0.04%	0.05%	0.04%	0.06%	0.04%
	Without Packet Caching	0.00%	0.00%	0.00%	0.00%	0.30%	0.49%	0.43%	0.59%	0.37%	0.34%	0.66%	0.32%
	Shingling	0.00%	0.00%	0.00%	0.00%	0.20%	0.28%	0.26%	0.17%	0.22%	0.27%	0.26%	0.15%
False Negative Rate	DSPT	0.00%	0.00%	0.00%	0.00%	0.00%	0.03%	0.00%	0.00%	0.00%	0.00%	0.00%	0.06%
	Without Notice	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	Without Packet Caching	0.00%	0.00%	0.00%	0.00%	0.03%	0.06%	0.02%	0.09%	0.00%	0.03%	0.05%	0.15%
	Shingling	0.00%	0.00%	0.00%	0.00%	0.00%	0.03%	0.02%	0.02%	0.00%	0.00%	0.00%	0.06%

Table 17: Impact of the notice, packet caching, and non-shingling mechanism on the accuracy, error rate, and false negative rate of DSPT for similar payload query on the ITDK topology.

Scenarios		Unsplit				Split				Split and Disrupt			
Metrics	**Length**	**300**	**400**	**500**	**600**	**300**	**400**	**500**	**600**	**300**	**400**	**500**	**600**
Accuracy	DSPT	100.00%	100.00%	100.00%	100.00%	99.86%	99.81%	99.65%	99.77%	99.83%	99.82%	99.71%	99.77%
	Without Notice	100.00%	100.00%	100.00%	100.00%	99.91%	99.93%	99.89%	99.91%	99.93%	99.91%	99.89%	99.92%
	Without Packet Caching	100.00%	100.00%	100.00%	100.00%	99.66%	99.50%	99.20%	99.41%	99.53%	99.53%	99.18%	99.41%
	Shingling	100.00%	100.00%	100.00%	100.00%	99.67%	99.67%	99.53%	99.70%	99.78%	99.66%	99.65%	99.72%
Error Rate	DSPT	0.00%	0.00%	0.00%	0.00%	0.14%	0.19%	0.32%	0.21%	0.17%	0.15%	0.24%	0.23%
	Without Notice	0.00%	0.00%	0.00%	0.00%	0.09%	0.07%	0.11%	0.09%	0.07%	0.09%	0.11%	0.08%
	Without Packet Caching	0.00%	0.00%	0.00%	0.00%	0.31%	0.46%	0.67%	0.49%	0.46%	0.38%	0.71%	0.51%
	Shingling	0.00%	0.00%	0.00%	0.00%	0.30%	0.31%	0.40%	0.28%	0.22%	0.31%	0.31%	0.28%
False Negative Rate	DSPT	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.03%	0.02%	0.00%	0.03%	0.05%	0.00%
	Without Notice	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	Without Packet Caching	0.00%	0.00%	0.00%	0.00%	0.03%	0.04%	0.13%	0.09%	0.01%	0.09%	0.11%	0.08%
	Shingling	0.00%	0.00%	0.00%	0.00%	0.03%	0.01%	0.06%	0.02%	0.00%	0.03%	0.05%	0.00%

Table 18: Impact of the notice, packet caching, and non-shingling mechanism on the false positive rate, average number of false positives, and average query count of DSPT for similar payload query on the AS3967 topology.

Scenarios		Unsplit				Split				Split and Disrupt			
Metrics	**Length**	**300**	**400**	**500**	**600**	**300**	**400**	**500**	**600**	**300**	**400**	**500**	**600**
False Positive Rate	DSPT	17.67%	14.22%	12.55%	11.03%	24.18%	15.57%	13.01%	11.52%	24.39%	15.62%	13.11%	11.60%
	Without Notice	100.00%	100.00%	100.00%	100.00%	99.96%	99.94%	99.92%	99.96%	99.95%	99.96%	99.94%	99.96%
	Without Packet Caching	17.68%	14.19%	12.53%	11.05%	25.78%	15.65%	13.06%	11.10%	26.15%	15.94%	12.67%	11.45%
	Shingling	17.48%	13.99%	12.40%	10.88%	25.19%	15.44%	12.76%	11.35%	25.16%	15.56%	12.87%	11.31%
Average Number of False Positives	DSPT	1.1796	0.8431	0.7065	0.6311	1.9275	0.9541	0.7579	0.6809	1.9273	0.9703	0.7526	0.6886
	Without Notice	10.876	10.4113	10.2702	10.2134	11.1442	10.448	10.2885	10.2274	11.1398	10.4553	10.289	10.2316
	Without Packet Caching	1.1780	0.8421	0.7054	0.6313	2.1422	0.9853	0.7812	0.6936	2.1624	1.0181	0.7665	0.7069
	Shingling	1.1726	0.8356	0.7028	0.6283	2.054	0.9645	0.7606	0.6804	2.0370	0.9825	0.7531	0.6848
Average Query Count	DSPT	2	2	2	2	2	2	2	2	2	2	2	2
	Without Notice	4.4832	4.4775	4.4741	4.4902	4.4832	4.4775	4.4741	4.4902	4.4832	4.4775	4.4741	4.4902
	Without Packet Caching	2	2	2	2	2	2	2	2	2	2	2	2
	Shingling	2	2	2	2	2	2	2	2	2	2	2	2

Table 19: Impact of the notice, packet caching, and non-shingling mechanism on the false positive rate, average number of false positives, and average query count of DSPT for similar payload query on the ITDK topology.

Scenarios		Unsplit				Split				Split and Disrupt			
Metrics	**Length**	**300**	**400**	**500**	**600**	**300**	**400**	**500**	**600**	**300**	**400**	**500**	**600**
False Positive Rate	DSPT	18.52%	14.71%	13.32%	11.71%	25.53%	16.18%	13.90%	12.07%	25.61%	16.04%	13.88%	12.09%
	Without Notice	100.00%	100.00%	100.00%	100.00%	99.91%	99.93%	99.89%	99.91%	99.93%	99.91%	99.89%	99.92%
	Without Packet Caching	18.47%	14.68%	13.30%	11.71%	27.30%	16.33%	13.75%	11.73%	27.10%	16.25%	13.57%	11.92%
	Shingling	18.24%	14.55%	13.22%	11.56%	26.03%	16.00%	13.62%	11.94%	26.51%	15.90%	13.69%	11.92%
Average Number of False Positives	DSPT	2.5134	1.7429	1.4514	1.3046	4.2229	2.0117	1.6121	1.3992	4.2506	2.0133	1.5964	1.3963
	Without Notice	24.8448	24.36	23.7285	23.7579	25.4075	24.4369	23.7796	23.787	25.4067	24.4442	23.7733	23.7896
	Without Packet Caching	2.5085	1.7423	1.4514	1.3044	4.7948	2.1459	1.6698	1.4316	4.8095	2.1479	1.6414	1.4430
	Shingling	2.5043	1.7370	1.4455	1.2995	4.4882	2.0407	1.6116	1.4123	4.5920	2.0472	1.6035	1.3931
Average Query Count	DSPT	3.6936	3.6329	3.6082	3.5866	3.8056	3.6591	3.6217	3.5934	3.8111	3.6549	3.6193	3.5936
	Without Notice	13.8651	13.8397	13.7569	13.7516	13.8645	13.8394	13.7568	13.7514	13.8645	13.8393	13.7561	13.751
	Without Packet Caching	3.6927	3.6329	3.6085	3.5866	3.8402	3.6656	3.6230	3.5944	3.8462	3.6612	3.6203	3.5941
	Shingling	3.6904	3.6319	3.6056	3.5836	3.8167	3.6588	3.6181	3.5913	3.8290	3.6562	3.6167	3.591

Tables 18 and 19 present the false positive rate, average number of false positives, and average query count over the AS3967 and ITDK topologies. Across all three scenarios, the system without the notice mechanism has an extremely high false positive rate (over 99%), with the average number of false positives roughly 10 times that of DSPT and a larger query count. This stems from pure flooding traceback across the entire network. In the "Unsplit" scenario, the system using the shingling mechanism outperforms DSPT in lower false positives and query overhead, as consecutive block verification reduces mis-tracing to remote false sources. The system without packet caching performs comparably to DSPT, with fluctuating advantages on either side. This is because false positives arise from hash collisions, and packet caching has indeterminate effects on collisions and query counts. In the "Split" and "Split and Disrupt" scenarios, short excerpts yield higher false positives and query counts for the system without packet caching and using the shingling mechanism. As the excerpt length increases, its performance gradually approaches DSPT. Without packet caching or with shingling enabled, fewer original block matches occur in similar payload queries, raising misidentification probability and further increasing false positives and query counts.

Tables 20 and 21 list the average query time on the AS3967 and ITDK topologies. Across all three scenarios, the system without the notice mechanism exhibits a longer average query time: approximately 0.002 s longer than DSPT on AS3967 and over 0.02 s on ITDK. Lacking precise upstream node guidance

causes request flooding and higher time overhead. By contrast, the system without packet caching and using the shingling mechanism has nearly the same query time as DSPT. Their false positive gaps are minor and do not induce extra remote mis-tracing, leading to comparable query time.

Table 20: Impact of the notice, packet caching, and non-shingling mechanism on the average query time of DSPT on the AS3967 topology (seconds).

Scenarios	Unsplit				Split				Split and Disrupt			
Length	**300**	**400**	**500**	**600**	**300**	**400**	**500**	**600**	**300**	**400**	**500**	**600**
DSPT	0.0183	0.0186	0.0189	0.0192	0.0184	0.0186	0.0189	0.0192	0.0184	0.0186	0.0189	0.0192
Without Notice	0.0201	0.0206	0.021	0.0215	0.0201	0.0206	0.021	0.0215	0.0201	0.0206	0.021	0.0215
Without Packet Caching	0.0183	0.0186	0.0189	0.0192	0.0185	0.0186	0.0189	0.0192	0.0185	0.0186	0.0189	0.0192
Shingling	0.0183	0.0186	0.0189	0.0192	0.0185	0.0186	0.0189	0.0192	0.0185	0.0186	0.0189	0.0192

Table 21: Impact of the notice, packet caching, and non-shingling mechanism on the average query time of DSPT on the ITDK topology (seconds).

Scenarios	Unsplit				Split				Split and Disrupt			
Length	**300**	**400**	**500**	**600**	**300**	**400**	**500**	**600**	**300**	**400**	**500**	**600**
DSPT	0.0398	0.0399	0.0404	0.0408	0.0409	0.0402	0.0406	0.0409	0.0409	0.0402	0.0406	0.0409
Without Notice	0.0614	0.0628	0.0637	0.0649	0.0614	0.0628	0.0637	0.0649	0.0614	0.0628	0.0637	0.0649
Without Packet Caching	0.0398	0.0399	0.0404	0.0408	0.0412	0.0403	0.0406	0.0409	0.0413	0.0402	0.0406	0.0409
Shingling	0.0398	0.0399	0.0404	0.0408	0.0410	0.0402	0.0406	0.0409	0.0411	0.0402	0.0405	0.0409

5.4.3 Overhead and Scalability

With a data reduction ratio of 40:1, the Bloom Filter introduces a memory overhead of 86.5 MB. The flow information list contains 25,688 entries, each occupying 6 bytes, resulting in a total memory usage of 150.5 KB. Therefore, the total memory overhead of a single cooperative node is 86.6 MB.

During traceback, taking query excerpts of length 300 as an example, a standard UDP packet consists of a 28-byte header and a 300-byte excerpt, yielding a total packet size of 328 bytes.

To evaluate the practical deployment performance of DSPT, we compare its overhead and scalability against TOPO in both the AS3967 and ITDK topologies. Table 22 presents the memory overhead and control message overhead (querying 300-byte excerpts).

Table 22: Overhead comparison between DSPT and TOPO.

Topology	Scenarios	Total Memory Overhead	Number of Traceback Packets	Control Message Overhead
AS3967	TOPO	432.64 MB	60,780	19.01 MB
	DSPT	433.37 MB	15,260	4.77 MB
ITDK	TOPO	865.28 MB	14,536,639	4547.14 MB
	DSPT	866.75 MB	29,079	9.10 MB

As shown in Table 22, DSPT achieves significantly lower control message overhead than TOPO, accompanied by only a slight increase in memory overhead. In larger network topologies, more cooperative

nodes must be deployed to enable traceback closer to the attack source. As the number of cooperative nodes increases, both memory and control overhead rise; however, DSPT's advantage in control message overhead becomes increasingly prominent.

As expected, memory overhead increases linearly with the number of cooperative nodes (5 nodes in AS3967, 10 nodes in ITDK). In contrast, control message overhead exhibits drastically different trends. TOPO relies on flooding-based forwarding, leading to explosive growth in control message overhead (approximately 4.44 GB) and poor scalability. By comparison, DSPT maintains well-controlled control message overhead even with an increasing number of nodes. Although DSPT requires slightly more complex deployment and maintenance due to its flow information list, it achieves substantially superior scalability.

In small networks with few cooperative nodes, both TOPO and DSPT are applicable. In large topologies where more cooperative nodes are needed to accurately trace attacks to their sources, TOPO becomes infeasible due to its rapidly growing control message overhead. In contrast, DSPT's memory and control overhead grow linearly with network scale, making it highly suitable for large-scale deployment.

5.5 System Limitations

First, DSPT relies on several idealized assumptions that may restrict its practical deployment.

1. The scheme assumes that cooperative nodes are fully trusted. In partially trusted or compromised network environments, malicious nodes may tamper with upstream notice information and composite block digests, which degrades traceback accuracy and introduces misleading forwarding paths.
2. DSPT requires minor packet header modification and available unused header fields to carry neighbor notice data. In heterogeneous, rigid, or legacy network devices that forbid header rewriting, the upstream notice mechanism cannot work properly, and DSPT will degrade to single-node payload matching without distributed multi-hop traceback capability.
3. The relaxation-constrained fuzzy matching improves the detection of obfuscated payloads, but it also raises the false positive rate under complex network conditions.

Second, although DSPT achieves lower false positives than the TOPO scheme, its false positive rate remains considerable for practical forensic scenarios. Excessive false matches generate redundant traceback paths, raise manual auditing costs, and impair the accuracy of attack source localization in high-precision security investigations.

Third, the experimental evaluation is limited in scalability verification. The network topologies and test scales adopted in experiments are limited in size. Large-scale network scenarios are not fully verified, which restricts the comprehensive analysis of system scalability and real-world large-network applicability.

Fourth, the Bloom Filter for storing traffic digest requires trade-offs among memory overhead, false positive rate, and parameter settings. DSPT adopts the standard Bloom Filter in the current implementation. Larger filter sizes and extra hash functions reduce false positives at the cost of higher memory consumption, while lightweight parameters cut resource usage but elevate false match risks. Herein, Bloom Filter parameters are empirically configured. Besides, various optimized Bloom Filter variants can provide better performance in reducing the false positive rate. In-depth parameter sensitivity analysis and comparisons with optimized filter structures will be explored in future work.

5.6 Real-World Implications

The practical applicability of the DSPT is constrained by real-world network behaviors.

First, compromised cooperative nodes represent a typical real-world constraint. In practical network deployments, once trusted gateway or backbone cooperative nodes are compromised and manipulated

by adversaries, the locally stored upstream notice messages will be falsified. The hop-by-hop traceback mechanism built upon trusted node cooperation consequently yields spurious forwarding paths, rendering DSPT incapable of pinpointing the genuine attack source. Such vulnerabilities cannot be mitigated under the current DSPT architecture. Accordingly, DSPT is only viable for network scenarios where core nodes are fully isolated and subject to rigorous access administration.

Second, cooperative nodes are commonly targeted by DDoS flooding attacks in real cyber threats. Floods of illegitimate traffic will rapidly exhaust the storage and computational resources of deployed nodes, suspend payload digest generation services, and disrupt the entire traceback pipeline. In this case, DSPT fails to capture valid payload features, which results in missing forensic records. This indicates that DSPT requires core nodes to be equipped with fundamental anti-DDoS access control mechanisms to guarantee stable operation.

Third, encrypted or compressed payloads are prevalent in mainstream production traffic. DSPT generates digests directly from incoming payload data. Specifically, digests generated from encrypted payloads only enable matching for encrypted excerpts, whereas digests built on compressed payloads are limited to compressed excerpts. As DSPT relies on block-based matching, encryption and compression drastically alter the underlying bit sequences of payloads. As a result, DSPT achieves reliable query performance only when the queried payloads share the identical encrypted or compressed format used for digest generation.

The aforementioned three real-world network conditions constrain DSPT's applicability. Nevertheless, under specific security assumptions (as described in Section 3.1), DSPT can effectively trace attacks such as command injection, unauthorized access, and malware propagation, and defend against IP spoofing, traffic injection, stepping-stone attacks, path forgery, and payload obfuscation. However, DSPT has the following limitations.

6 Conclusion

We propose DSPT, a distributed similar traffic detection and traceback system based on upstream cooperative node notice. As network flow passes through cooperative nodes, upstream node information is notified downstream. Upon detecting malicious traffic, the victim node initiates traceback via critical excerpts to upstream cooperative nodes, which perform hop-by-hop tracing using notice records to pinpoint the nearest attacker position. DSPT further adopts packet caching and non-shingling mechanisms to improve traceback accuracy for malicious traffic and its variants. Experiments show DSPT supports effective distributed traceback over excerpts of different lengths. DSPT reduces the false positive rate to below 26% even in similar-payload scenarios, and achieves much lower average false positives and query time than TOPO.

In future work, we intend to advance DSPT along three key directions. First, we will further optimize system performance, with a particular focus on lowering the false positive rate to meet the stringent demands of practical network forensics. Second, we will conduct extensive experiments on larger, more realistic network topologies and implement targeted structural optimizations to strengthen DSPT's scalability. Third, we will develop more robust and effective solutions to enable DSPT to function reliably in non-ideal, uncontrolled network conditions.

Acknowledgement: None.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: The authors confirm contribution to the paper as follows: Conceptualization, Changsheng Hou and Bingnan Hou; methodology, Changsheng Hou; software, Xionglve Li; validation, Changsheng Hou, Xionglve Li

and Jingtao Hu; formal analysis, Xionglve Li; investigation, Jingtao Hu; resources, Zhiping Cai; data curation, Shuai Ye; writing—original draft preparation, Changsheng Hou and Bingnan Hou; writing—review and editing, Changsheng Hou; visualization, Jingtao Hu; supervision, Hao Li; project administration, Hao Li. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are available from the Corresponding Author, Jingtao Hu, upon reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: Given his role as Editorial Board Member of this journal, Zhiping Cai had no involvement in the peer review of this article and had no access to information regarding its peer review. Full responsibility for the editorial process for this article was delegated to another journal editor. The authors declare no other conflicts of interest.

References

1. Asassfeh M, Al-Mousa MR, Al-Dweikat H, Al-Mashagbeh MH, Afaneh S, Samara G, et al. An overview of tools and techniques in network forensics. In: Proceedings of the 2024 25th International Arab Conference on Information Technology (ACIT); 2024 Dec 10–12; Zarqa, Jordan. p. 1–7.
2. Sikos LF. Packet analysis for network forensics: a comprehensive survey. Forensic Sci Int Digit Investig. 2020;32(5):200892. doi:10.1016/j.fsidi.2019.200892.
3. Khan S, Gani A, Wahab AWA, Shiraz M, Ahmad I. Network forensics: review, taxonomy, and open challenges. J Netw Comput Appl. 2016;66:214–35.
4. Al Boloushi AG, Al-Meer FA, Nawshin F, Unal D. Network forensics: techniques, challenges, and incident response. In: Bhateja V, Reza Khan Z, Simic M, Sharma DK, editor. Information system design: communication networks and Internet of Things. Singapore: Springer Nature Singapore; 2026. p. 51–62.
5. Hosseini SM, Jahangir AH. An effective payload attribution scheme for cybercriminal detection using compressed bitmap index tables and traffic downsampling. IEEE Trans Inf Forensics Secur. 2018;13(4):850–60.
6. Ponec M, Giura P, Brönnimann H, Wein J. Highly efficient techniques for network forensics. In: Proceedings of the ACM Conference on Computer and Communications Security, CCS '07. New York, NY, USA: Association for Computing Machinery; 2007. p. 150–60.
7. Mohammad Hosseini S, Jahangir AH, Kazemi M. Digesting network traffic for forensic investigation using digital signal processing techniques. IEEE Trans Inf Forensics Secur. 2019;14(12):3312–21. doi:10.1109/tifs.2019.2915190.
8. Zhang L, Guan Y. TOPO: a topology-aware single packet attack traceback scheme. In: Proceedings of the 2006 Securecomm and Workshops; 2006 Aug 28–Sep 1; Baltimore, MD, USA. p. 1–10.
9. Uhlig F, Struppek L, Hintersdorf D, Göbel T, Baier H, Kersting K. Combining AI and AM—improving approximate matching through transformer networks. Forensic Sci Int Digit Investig. 2023;45:301570. doi:10.1016/j.fsidi.2023.301570.
10. Flynn R, Olukoya O. Using approximate matching and machine learning to uncover malicious activity in logs. Comput Secur. 2025;151(1):104312. doi:10.1016/j.cose.2025.104312.
11. Kim D, Jang H, Shin Y. FuzzyBin: enhanced border binary identification by leveraging fuzzy hashing algorithms. IEEE Access. 2025;13:49659–71.
12. He D, Yu X, Zhu S, Chan S, Guizani M. Fuzzy hashing on firmwares images: a comparative analysis. IEEE Internet Comput. 2023;27(2):45–50.
13. Breitinger F, Baggili I. File detection on network traffic using approximate matching. J Digit Forensics Secur Law. 2014;9(2):23–36.
14. Gupta V, Breitinger F. How cuckoo filter can improve existing approximate matching techniques. In: James JI, Breitinger F, editors. Digital forensics and cyber crime. Berlin/Heidelberg, Germany: Springer; 2015. p. 39–52.
15. Yang Y, Hou C, Hu L, Li X, Hou B, Cai Z. BIFM: an effective similar payload attribution approach for cybercriminal detection using bitmap index table and fuzzy matching. Comput J. 2026;3:bxag022.

16. Haghighat MH, Tavakoli M, Kharrazi M. Payload attribution via character dependent multi-bloom filters. IEEE Trans Inf Forensics Secur. 2013;8(5):705–16. doi:10.1109/tifs.2013.2252341.
17. Renukuntla SSB, Rawat S. Optimization of excerpt query process for packet attribution system. In: Proceedings of the 2014 10th International Conference on Information Assurance and Security; 2014 Nov 28–30; Okinawa, Japan. p. 41–6.
18. Subash A, Arvin Danny CS, Vijayalakshmi M. An enhanced hybrid scheme for IP traceback. In: Proceedings of the 2023 4th International Conference on Innovative Trends in Information Technology (ICITIIT); 2023 Feb 11–12; Kottayam, India. p. 1–5.
19. Mohamed H, Ouldmohamed Y, Nacera B. A single-packet IP traceback: combating DoS-DDoS attacks. EDPACS. 2022;66(4):1–12.
20. Pasupathi S, Kumar R, Pavithra LK. Proactive DDoS detection: integrating packet marking, traffic analysis, and machine learning for enhanced network security. Clust Comput. 2025;28(3):210.
21. Rocketfuel: an ISP topology mapping engine. [cited 2024 Jul 1]. Available from: https://research.cs.washington.edu/networking/rocketfuel/.
22. Roussev V. An evaluation of forensic similarity hashes. Digit Investig. 2011;8(7):S34–41. doi:10.1016/j.diin.2011.05.005.