HEPFS: Homomorphic Encryption-based Privacy Forwarding Scheme for Efficient Ciphertext-Domain Routing *

Jianbang Chen, Yizhong Hu, Jianfeng Guan†

Beijing University of Posts and Telecommunications, Beijing, China
{jbchen, yzhu, jfguan}@bupt.edu.cn

Abstract

Privacy-preserving packet forwarding is fundamental for securing Internet communication against eavesdropping and traffic analysis. While existing solutions like Tor and Crowds rely on multi-hop encapsulation and relay mechanisms, they often suffer from significant latency overhead and suboptimal path lengths. The emerging Homomorphic Routing (HR) paradigm, which performs routing computations directly on ciphertexts, presents a promising alternative, while its implementations faces practical bottlenecks such as inflexible forwarding path determination and significant overhead from packet encapsulation processes. To overcome these limitations, we propose the Homomorphic Encryption-based Privacy Forwarding Scheme (HEPFS) which introduces the Homomorphic Tree (HTree), a novel encrypted data structure that enables efficient longest-prefix matching on ciphertexts. HTree allows intermediate routers to determine next-hop addresses without decrypting packet headers, thus preserving privacy while operating within standard path selection frameworks like BGP or OSPF. Consequently, HEPFS introduces minimal packet overhead and remains compatible with existing internet routing infrastructure. Theoretical analysis and experimental evaluations demonstrate that HEPFS demonstrates an advantage in routing length and routing match latency over anonymous communication systems such as Tor and Crowds, and maintains near-native packet delivery ratios, underscoring its practicality for high-speed networks.

1 Introduction

During data transmission, IP addresses in packets are typically transmitted in plaintext, revealing location and identity attributes[9]. Consequently, cyber attackers can employ techniques such as traffic analysis, eavesdropping, and packet sniffing to steal private data based on users' communication metadata[18, 19, 32], thus silently monitoring users' online activities[2, 4]. For instance, in January 2024, malicious attacker compromised Microsoft's email infrastructure, resulting in the exfiltration of a significant quantity of email correspondence within the Federal Civilian Executive Branch (FCEB) and with Microsoft.

To protect data transmission privacy, Anonymous Communication (AC) offers a technological solution that conceals the identities of communicating parties, as well as the time, location, and content of the communication. This effectively mitigates the risks associated with communication surveillance[3, 15, 27]. Current AC schemes employ various methods, including source-routed approaches like the onion router (Tor), which conceals communication paths through multiple encryption layers; hop-by-hop routed methods such as Crowds, which ensure

*Proceedings of the 9th International Conference on Mobile Internet Security (MobiSec'25), Article No. 41, December 16-18, 2025, Sapporo, Japan.

†Corresponding author

anonymity via random node forwarding; broadcast techniques, exemplified by the Babel protocol, which simultaneously sends data to multiple nodes to thwart attacker control of the communication path; and Homomorphic Routing (HR), an emerging AC technology that enables endpoint communication without disclosing source or destination addresses. These technologies have been rigorously evaluated and have demonstrated their effectiveness in various scenarios.

Recent studies in AC have concentrated on enhancing existing systems [7, 14, 20, 24], advancing packet data protection [21], and choosing the suitable security point[11, 10]. Despite these efforts, each method has inherent limitations and current AC technologies face several significant challenges.

1. Nested multilayer packet encapsulation can decrease payload efficiency, leading to increased packet fragmentation, reduced processing performance, and a heavier transmission burden on the network.

2. Fixed forwarding paths restrict nodes to a single optimal routing path, thereby increasing communication delays and complicating the maintenance of stable data transmission.

3. The additional entities utilized for routing and forwarding are vulnerable to risks such as route hijacking or single point of failure, potentially leading to service instability or even compromising user privacy.

To address these challenges, this paper introduces a homomorphic encryption-based privacy forwarding scheme (HEPFS) and develops a homomorphic-based routing architecture to facilitate efficient route lookup and forwarding in the ciphertext domain. The key contributions of this work are summarized as follows:

- HEPFS supports IP packets, and ensures seamless interoperability while maintaining the integrity of the original packet format

- HEPFS does not interfere with packet forwarding in the network, ensuring the shortest path for packets and thus improving data transmission performance.

- HEPFS operates on the premise of un-trusted routers and integrates corresponding security-focused design elements.

The remainder of this paper is structured as follows: section 2 reviews related research on anonymous communication; section 3 outlines the fundamental principles of homomorphic encryption, HTree, and attack models; section 4 offers a comprehensive description of the HEPFS design and workflow; section 5 assesses the HEPFS and analyzes the findings; and section 6 concludes the paper.

2 Related Work

The concept of Anonymous Communication (AC) originated with the anonymous email service proposed by *Chaum* in 1981[1]. Over the years, AC technology has been extended to various application domains, including anonymous voting [17], Private Information Retrieval (PIR) [5], censorship resistance [30, 31], anonymous web browsing [8], and hidden web services [28]. Currently, AC techniques can be categorized into source-routing schemes, hop-by-hop routing schemes, broadcast schemes [27], and the recently proposed cipher-text forwarding-based HR.

Source-Routing Schemes. Source-routing is an anonymous communication method where the sender specifies the entire route for data transmission, selecting a sequence of relay nodes who is aware only of its immediate predecessor and successor, thereby enhancing privacy. A significant advantage is that the sender can select trusted nodes, reducing the risk from malicious actors. For instance, Mixnet and Tor [6] employ source-routing to protect anonymity. However, this method faces challenges such as route replay and path prediction attacks.

Hop-by-Hop Routing Schemes. In hop-by-hop routing, the sender designates only the initial relay, after which each intermediate node autonomously selects the next hop. This approach offers flexibility and adaptability, as nodes make localized decisions that enhance security. A notable example is the Crowds protocol [25], which diminishes route predictability through randomized selection of the next hop. Although hop-by-hop routing facilitates scalability and dynamism in extensive networks, it also exposes vulnerabilities such as predecessor attacks.

Broadcast Schemes. Broadcast transmits data to multiple nodes simultaneously without a predefined route. This method is beneficial for applications such as emergency alerts, where path control is not crucial. The Babel protocol [13] exemplifies this approach, where messages are broadcasted, and nodes independently determine whether to forward them. Although effective in preventing path manipulation by attackers, broadcast can result in redundant data transmission and significant bandwidth usage in extensive networks. Furthermore, it raises privacy issues as broadcast messages might reveal communication patterns.

Homomorphic Routing. HR [29] is a private routing scheme that employs homomorphic encryption to safeguard source and destination addresses. Unlike tunneling or onion routing, HR offers a hop-by-hop solution compatible with BGP, eliminating the need for routers to maintain flow state. Performance evaluations indicate that HR reduces inter-domain routing rules to approximately 5% of IPv6 size and achieves logarithmic-time matching.

While existing research has proposed various AC routing solutions, each involves trade-offs between security and resource efficiency. In this paper, we aim to develop a homomorphic encryption-based privacy forwarding mechanism for low-overhead routing.

3 Prior Work

3.1 Paillier homomorphic cryptosystem

Homomorphic Encryption (HE) enables computations on encrypted data without the need for decryption. In 2009, Craig Gentry introduced the first Fully Homomorphic Encryption (FHE) scheme, allowing any computation on encrypted data. Despite its potential, FHE's performance and practical applications remain limited, leading to the continued widespread use of the Paillier encryption scheme. This paper presents an improved version of the Paillier algorithm to reduce decryption costs and enhance suitability for routing computations.

In the original Paillier cryptosystem, the system selects two large prime numbers p and q that meet $gcd(pq, (p-1)(q-1)) = 1$. After that, it calculates modulus $n = p \times q$ and the private key parameter $\lambda = lcm(p-1, q-1)$(Note that lcm is least common multiple). Select an integer $g_p \in \mathbb{Z}^*_{n^2}$ as a generator and compute $\mu = (L(g^\lambda \mod n^2))^{-1} \mod n$ in which auxiliary function $L(x) = \frac{x-1}{n}$. A common simplification is to set $g = n + 1$, then μ can be computed as the modular inverse of λ modulo n, i.e., $\mu = \lambda^{-1} \mod n$. Finally, it forms the public key (n, g_p) and private key (λ, μ).

- **Encryption**: To encrypt a message m where $0 \leq m < n$, select a random integer r where $0 < r < n$ and $\gcd(r, n) = 1$, and compute the ciphertext c

$$c = g_p^m \cdot r^n \mod n^2 \tag{1}$$

  The use of random r ensures probabilistic encryption, where identical plaintexts encrypt to different ciphertexts, providing semantic security.

- **Decryption**: Compute the plaintext m

$$m = L(c^\lambda \mod n^2) \cdot \mu \mod n \tag{2}$$

  In the simplified variant with $g_p = n + 1$, Eq. 2 simplifies to $m = L(c^\lambda \mod n^2) \cdot \lambda^{-1} \mod n$.

The Paillier cryptosystem exhibits two fundamental homomorphic properties:

- **Homomorphic Addition**: The product of two ciphertexts decrypts to the sum of their corresponding plaintexts:

$$D(E(m_1) \cdot E(m_2) \mod n^2) = m_1 + m_2 \mod n \tag{3}$$

- **Homomorphic Multiplication**: A ciphertext raised to a constant power decrypts to the product of the plaintext and the constant:

$$D(E(m_1)^k \mod n^2) = k \cdot m_1 \mod n \tag{4}$$

However, μ can be made public since decryption using only μ is computationally difficult. Consequently, the new public and private keys are (n, g_p, μ) and λ, respectively. Following this modification, anyone can decrypt using (n, g_p, μ), whereas encryption requires λ. This adjustment shifts the majority of the computational load to the encryption process, thereby making decryption more efficient [22].

It is worth noting that the modified Paillier cryptosystem remains fully functional for all homomorphic encryption operations. This modification merely shifts the computational burden from decryption to encryption. The subsequent equation illustrates the Paillier algorithm with lightweight decryption:

$$E(m, r, \lambda, g_p, n) = g_p^{m\lambda} \cdot r^{n\lambda} \mod n^2 = c \tag{5}$$

$$D(c) = L(c \mod n^2) \cdot \mu \mod n = m \tag{6}$$

Eq. 5 and Eq. 6 represent the encryption and decryption, respectively. Here, r is a random number, and the auxiliary function L is defined as $L(x) = \frac{x-1}{n}$. In this situation, the public key holder can decrypt each encrypted value individually. In HEPFS, the router is designed to forward packets by comparing the difference between the routing table and the packet's destination address. Consequently, the router holding the public key must not be able to decrypt the destination address directly.

To ensure this, we introduce the parameters g_p^t and g_p^{-t} into the encryption process. This ensures that during route lookup, the router can only decrypt the result of the matching computation, without accessing the plaintext value of the destination IP address.

$$E'(m_1, r, \lambda, g_p, n, t) = g_p^t E(m_1, r, \lambda, g_p, n) = c_1 \tag{7}$$

$$D(c_1) = L(c_1 \mod n^2) \cdot \mu \mod n = m_1 \tag{8}$$

$$E'(-m_2, r, \lambda, g_p, n, t) = g_p^{-t} E(m_2', r, \lambda, g_p, n) = c_2 \tag{9}$$

$$D(c_2) = L(c_2 \mod n^2) \cdot \mu \mod n = -m_2 \tag{10}$$

The introduction of parameters g_p^t and g_p^{-t} disrupts the original ciphertext ordering, thereby preventing the router from decrypting c_1 and c_2 individually. However, the underlying computational logic of homomorphic addition remains based on the multiplication of two ciphertexts:

According to Eq. 3, we can get that:

$$\begin{aligned} D(c_1 \cdot c_2) &= D(E(m_1) \cdot E(-m_2)) \\ &= m_1 - m_2 \end{aligned} \tag{11}$$

The multiplication of g_p^t with g_p^{-t} neutralizes, enabling the router to decrypt the result of the homomorphic addition, which represents the difference of $m_1 - m_2$.

While Paillier encryption does not inherently support subtraction, the negative of a value m can be represented as: $-m \equiv n - m \pmod{n}$. Given that the maximum value of the operation in this paper is 255, a negative number such as $-m_2$ in Eq.9 can be expressed as $m_2' = 255 - m_2$ to simplify the operation. Consequently, after decrypting $D(c_1 \cdot c_2)$, the actual plaintext difference obtained is: $m_1 + m_2'$. Subtracting 255 from this result yields the true difference: $m_1 - m_2 = m_1 + m_2' - 255$.

3.2 Homomorphic Tree

The prevalent approach for route matching involves employing a prefix tree-like structure for per-bit matching, achieving a time complexity of $O(k)$, where k represents the number of IP address binary bits. However, this conventional method is applicable only when the IP address is in plaintext. In contrast, HEPFS mandates that IP addresses must not appear in plaintext within backbone networks. Furthermore, due to the introduction of a random parameter r in homomorphic encryption, each encryption yields a distinct ciphertext, thereby precluding the use of per-bit matching for routing encrypted IP addresses.

To address the dynamic ciphertext matching problem, we have developed a specialized table structure known as the Homomorphic Tree (HTree). Fig.1 illustrates the HTree structure, which elucidates the lookup process for ciphertext IPv4 addresses.

3.2.1 Data Structure

HTree is a multi-level tree structure. As depicted in Fig.1, using an IPv4-generated HTree as an example, the IPv4 address is divided into four segments for homomorphic processing, resulting in a four-layer hierarchical structure (*Level* 0 to *Level* 3) arranged from top to bottom. Its overall architecture is similar to a $B+$ tree, where the IP matching process resembles querying a database index. However, each node in HTree may also function as a leaf node, storing the next hop's data. The content of each HTree node primarily includes the node array ($array$), the offset variable ($offset$), a priority queue ($Pqueue$), and a termination flag (end).

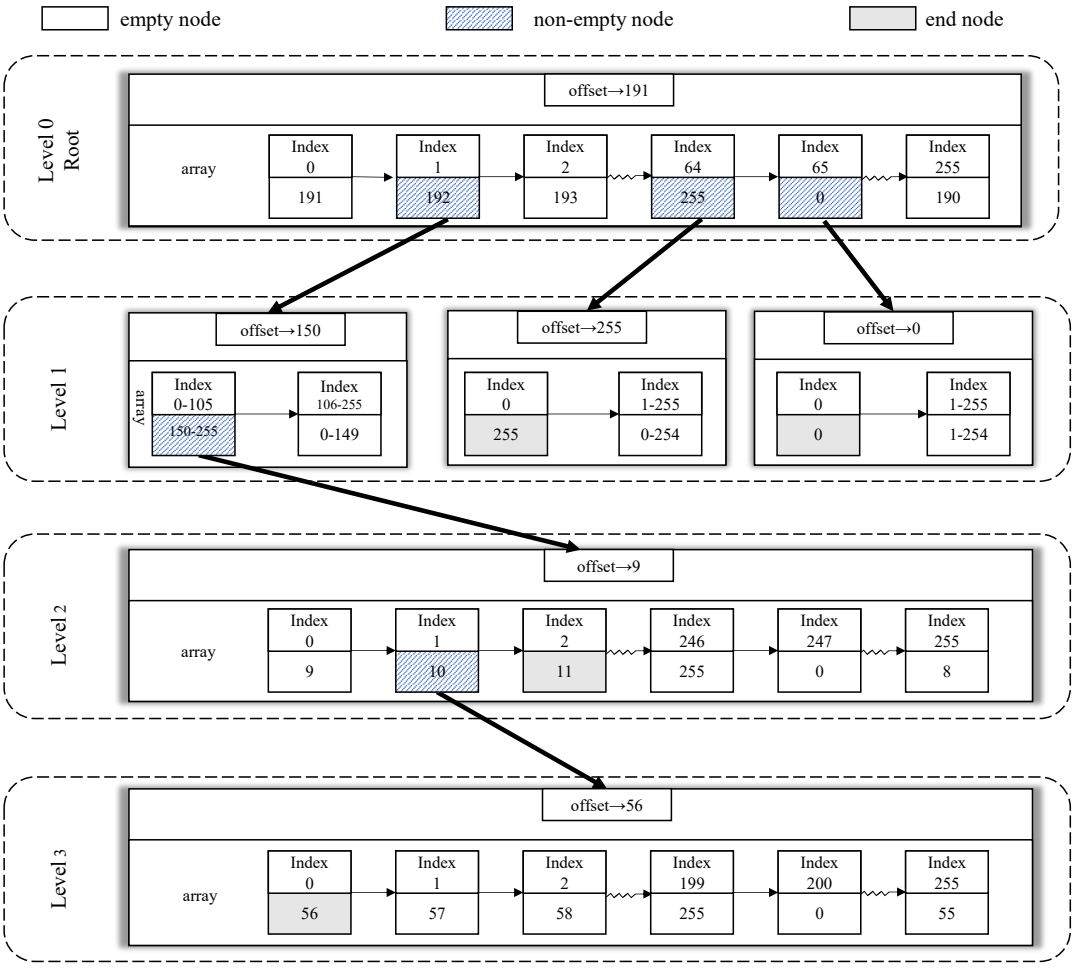


Figure 1: Example of HTree structure

- **array.** The array comprises 256 elements, each representing a value within the range [0, 255], corresponding to an 8-bit IP segment. Each element in the *array* as a distinct HTree node.

- **offset.** The $offset$ results from applying homomorphic negative encryption to a randomly selected number, *random*, within the range [0, 255], as detailed in Section 3.1. Negative values are assigned to *random* to simplify difference calculations through homomorphic addition. Specifically, *random* represents the IP segment denoted by the 0th element of the *array*. For instance, if $random = -198$, the $array[0, 1, 2, ..., 255]$ is circularly shifted left by 198 positions, yielding $[198, 199, ..., 255, 0, 1, ..., 197]$. Here, the first element of the array signifies that the IP segment value is 198, not 0.

- **Pqueue.** During the construction of the HTree, certain elements in the *array* may remain uninitialized. When the router employs this HTree for route matching, there is a possibility that the corresponding array element may be empty. To resolve this issue, we

introduce a priority queue. When constructing the HTree, the IP segment value of each element can be easily determined due to the sequential structure of the *array*. Utilizing this property, the HTree calculates the length of the common prefix between the current element and the IP segments of the subsequent elements, thus forming the common prefix array. To reduce traversal time when the router selects the highest-priority element, the HTree optimizes the common prefix array into a priority queue. This queue is sorted by the length of the common prefix, with its elements being the indices of the array elements corresponding to each prefix length.

- **end.** The nature of Classless Inter-Domain Routing (CIDR) enables IP addresses to have variable-length prefixes, allowing them to terminate at any level within the HTree. The *end* marker indicates the termination point of an IP address in the HTree, specifying if a particular node signifies the end of the IP. During the route matching process, the *end* marker helps determine whether the routing entry has been successfully identified.

3.2.2 Constructing the HTree

As depicted in Fig.1, the two construction scenarios of the HTree are illustrated using the routed IP addresses 192.168.10.56/28 and 192.168.11.0/24, with the assumption that both IP addresses will be processed sequentially within the HTree.

HTree also employs the prefix aggregation feature of CIDR. IP prefixes with lengths in the range $[1-8]$ terminate at *level* 0, those in $[9-16]$ at *level* 1, and so forth. The IP prefix length determines the level at which the IP address terminates in the HTree. For instance, the IP address 192.168.10.56/28 has its next hop stored at *level* 3 due to its prefix length of 28. Conversely, the IP address 192.168.11.0/24 has its next hop stored at *level* 2, given its prefix length of 24.

The next hop data is stored at *level* 3. The IP address 192.168.10.56/28 is encrypted into four ciphertexts, denoted as c_0, c_1, c_2, c_3. These ciphertexts are structured as tree nodes at *levels* 0, 1, 2, *and* 3, respectively. At *level* 0 (root), an offset value $offset = E(-198)$ is generated according to Eq. 9. The ciphertext c_0 is homomorphically added to the offset to produce the ciphertext difference $E(diff) = E(c_0 \cdot offset)$. This is then homomorphically decrypted to obtain the plaintext difference $diff$. The value $diff$ is used as a subscript to locate the ciphertext difference of the IP address 192 in the array position at level 0. The index of the array is given in Eq.12.

$$index = \begin{cases} diff, & \text{if } diff \geq 0 \\ diff + 256, & \text{if } diff < 0 \end{cases} \tag{12}$$

Create a new HTree node at *index* and proceed with the same process for constructing nodes $c_1 \sim c_3$ at *level* $1 \sim$ *level* 3. Mark the leaf node at *index* 0 in *level* 3 with an end marker to signify the termination of route matching.

The next hop data is stored at *level* 2. The IP address 192.168.11.0/24 is encrypted into four ciphertexts c_0', c_1', c_2', c_3'. Similarly, this IP address is constructed into HTree nodes step by step. Since both IP addresses share the common prefix 192.168, during the construction of nodes at *level* 0 and *index* 1, the existing nodes can be directly matched without regeneration. At *level* 2, the homomorphic addition of c_2' and $offset' = E(-9)$ yields the ciphertext difference $Encrypt(diff) = E(c_2' \cdot offset')$, which is then homomorphically decrypted to obtain the plaintext difference $diff'$. The *array* index $index'$ corresponding to $diff'$ is used to create the new HTree node. Because the CIDR prefix of IP $\rightarrow$ 192.168.11.0/24 is 24 bits (i.e., the subnet

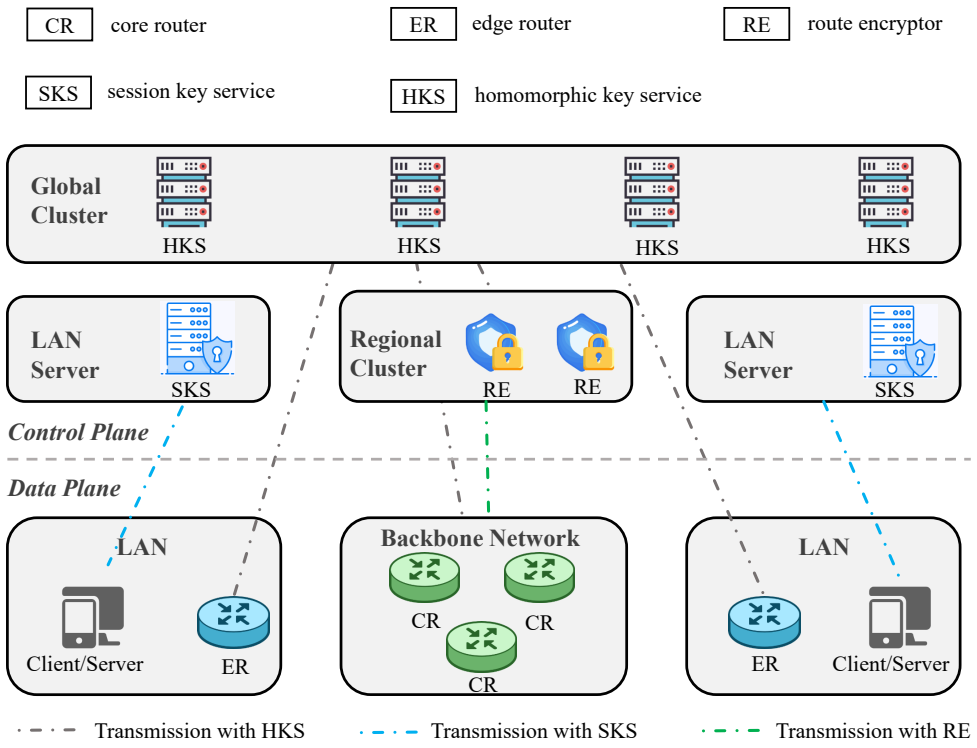


Figure 2: HEPFS architecture

mask is 255.255.255.0), *level* 2 represents the endpoint of the IP construction. Therefore, the node with $index' = 2$ at *level* 2 should be marked as the endpoint with *end*.

3.3 Threat Model

We categorize attack models into two main types and make the following assumptions:

- The primary attacker can deploy eavesdropping in the backbone network to analyze packets and deduce access patterns. They may also control routers to log packets and perform in-depth traffic analysis using SQL queries.
- Other attackers can execute common attacks such as traffic analysis, timing attacks, and malicious node attacks. These attacks enable them to monitor traffic characteristics, manipulate nodes for active or passive attacks, amplify traffic through their nodes, and significantly compromise network integrity and anonymity.

4 The HEPFS Design

4.1 System Components and Architecture

HEPFS mainly consist of backbone network and access network functionality, as illustrated in Fig. 2. HEPFS enhances traditional Internet services by introducing new components and

redefining router roles: the Route Encryptor (RE) applies homomorphic encryption to dynamic and static FIB tables, generating matchable encrypted routing tables; the Homomorphic Key Service (HKS) functions as a Key Distribution Center, authenticating devices and distributing encryption parameters; the Session Key Service (SKS) uses the PISKES protocol [26] for intra-domain key derivation to prevent source address exposure, while employing lightweight symmetric encryption locally for optimized protection; the Edge Router (ER) encrypts and decrypts the addresses of the packets originating from an Autonomous System (AS) or destined for an AS; the Core Router (CR) integrates a homomorphic function and supports private forwarding.

4.2 Initialization Phase

During the initialization of the HEPFS network system, network entities must acquire homomorphic parameters and cryptographic keys to enable homomorphic forwarding functionality. The HKS designs and provides a series of interfaces to obtain the homomorphic public key (μ, n, g_p), the private key (λ), and the homomorphic parameters (t).

The HKS maintains a list of trusts, so that only REs and ERs that are recognized as honest by the HEPFS system can obtain the private key and the public key to be used in cryptographic operations, whereas other non-trusted devices, such as the CRs, are only able to obtain the public key information. It should be noted that the parameter (t), as key variables in the computation process, is available to all devices performing homomorphic operations.

4.3 Connection Phase

4.3.1 Packet Design

HEPFS packets are compatible with both IPv4 and IPv6 protocols, while this paper focuses primarily on the IPv4 implementation scenario. Fig.3 illustrates the structure in HEPFS packet. Generally, the Ethernet frame's *EtherType* field requires a dedicated value to indicate the HEPFS packet protocol (e.g. IPv4: 0x0800, IPv6: 0x86DD). In this paper, we use the 0X8870 denote the HEPFS packets. Take IPv4 for example, to maintain compatibility with IPv4, we repurpose the Type of Service (ToS) field in the IPv4 header, and define the 2nd and 3rd bit of ToS field, named FIN and SYN, respectively. Both of them are used to clarify the current communication state and guide the corresponding operations of the communicating entities with the ER:

- **SYN:** Indicates the first packet sent by both communicating parties, usually initiated by client to server.
- **FIN:** Indicates that the server already knows who is communicating with it.

The HEPFS packets are divided into $INIT_{HEPFS}$ packet and $Data_{HEPFS}$ packet according to the communication phase, with SYN=1 for $INIT_{HEPFS}$ packet and FIN=1 for $Data_{HEPFS}$ packet. The source and destination addresses of a packet overwrite the original address after encryption. If the ciphertext address length needs to be increased for enhanced security, additional options can be used to store the encrypted address information.

4.3.2 Establish Connection

As shown in Fig.4, the client needs to carry out a homomorphic connection similar to the TCP handshake three times when formally communicating with the server, which is designed

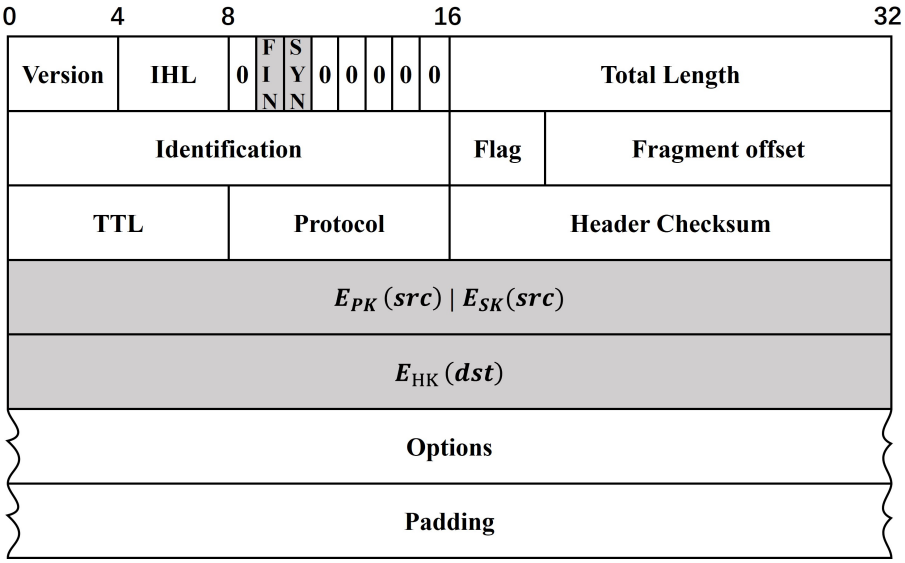


Figure 3: Data Packets in HEPFS.

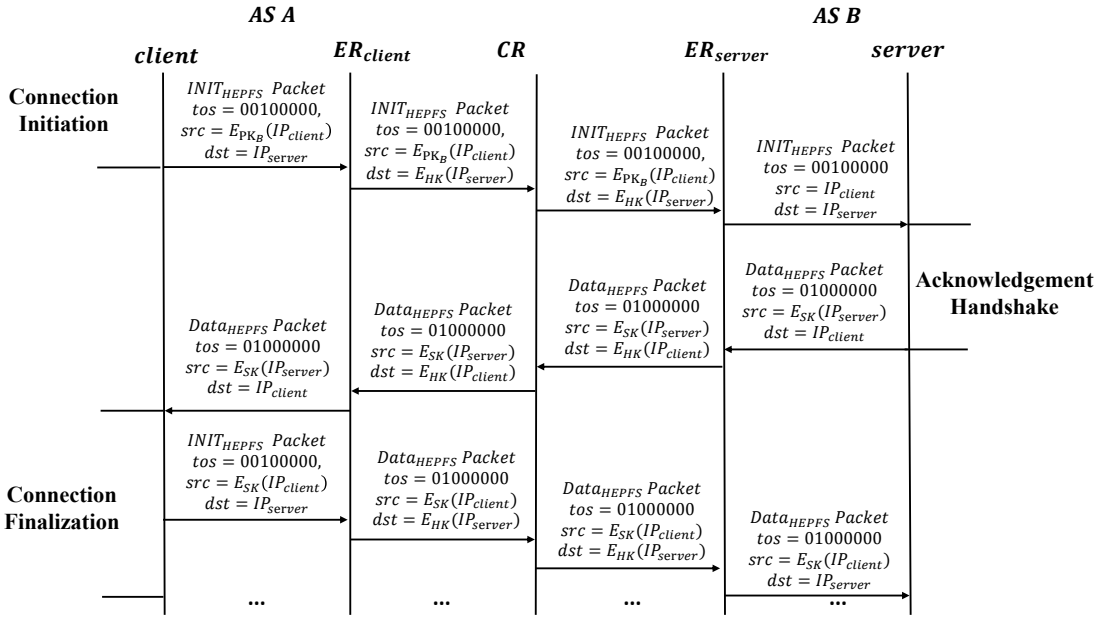


Figure 4: The connection establish procedure between Client and Server.

to verify and confirm the identity of the two communicating parties and, at the same time, avoid disclosing the information of the two communicating parties as much as possible.

Connection Initiation . The client constructs the $INIT_{HEPFS}$ packet (SYN=1), with the destination address in plaintext and the source address encrypted using the public key PK_B of AS_B. Upon detecting SYN=1 in the $INIT_{HEPFS}$ packet, the ER_{client} homomorphically

encrypts the destination address and forwards the packet. As the packet traverses the backbone, the CR will conduct homomorphic matching exclusively on packets with SYN/FIN flags set to 1, without altering the packet's content. Upon receiving an $INIT_{HEPFS}$ packet into AS_B, ER_{server} verifies if SYN=1. If true, it decrypts the packet's source and destination addresses and forwards the decrypted packet to the destination server.

Acknowledgment Handshake . The server receives the packet and upgrades the $INIT_{HEPFS}$ packet with SYN=1 to the $Data_{HEPFS}$ packet with FIN=1. At this point, the server is already aware of the client's identity. Subsequently, the server sends the identities of both communicating parties as a key parameter to the SKS to obtain the session key SK. The server employs SK to encrypt the source address of the $Data_{HEPFS}$ packet and responds to the client. ER_{server} identifies the FIN flag of the packet, homomorphically encrypts the destination address, and forwards it. The ER_{client} receives the $Data_{HEPFS}$ packet and will homomorphically decrypt the destination address before sending it to the client.

Connection Finalization . The client receiving the $Data_{HEPFS}$ packet, the client confirms that the connection has been established. Subsequently, homomorphic communication ensures that each subsequent $Data_{HEPFS}$ packet adheres to the following rules: i) the client and server encrypt the source address using their shared session key (SK); ii) the destination address is encrypted using ER homomorphic encryption; and iii) the FIN flag is set to 1.

4.4 Transmission Phase

4.4.1 Route computation.

In HEPFS, the router sends the FIB routing entries to the RE for encryption. Upon receipt of the FIB table, the RE applies segmented homomorphic encryption to the destination IP addresses, encrypting each IP field individually. The RE subsequently builds an HTree, serializes the structure, and returns it to the router. The router deserializes the data back into an HTree and disseminates it to the forwarding plane.

4.4.2 Intra-domain route forwarding.

HEPFS does not utilize homomorphic matching within its domain. Packets transmitted from endpoints feature encrypted source addresses, whereas destination addresses remain in plaintext. Consequently, routers within the domain can forward packets using standard procedures.

4.4.3 Inter-domain route forwarding.

In the core network, the CR determines the routing logic for route matching based on the SYN and FIN flag values. If either SYN=1 or FIN=1, the CR divides the destination address (dst) of the packet into four segments, each an 8-bit segment, and sequentially matches the $Enc_{HK}(dst)$ packet with the HTree. The operational process involves:

1. The initial ciphertext segment undergoes homomorphic addition with the offset at HTree *level* 0, resulting in the ciphertext difference C_1. This is subsequently decrypted to obtain the plaintext difference M_1.

2. Based on M_1, the CR aims to match the element at position M_1 in the *level* 0 array. If the element is empty, proceed to step 3. Otherwise, continue matching the remaining ciphertext IP segments from step 1 until an element marked with *end* is encountered. Then proceed to step 4.

3. If the element is empty, the CR retrieves the highest-priority value from the associated queue. It then identifies the array element that best matches the IP segment. If this element is non-empty, the CR continues to match the remaining ciphertext IP segments from step 1. If the element remains empty, the CR fetches the next value from the queue and repeats the process until a non-empty element is located.

4. If the current element contains an *end* tag, the CR extracts the routing entry data from the element and forwards the $INIT_{HEPFS}$ *Packet* according to the next hop information.

When SYN and FIN are both 0, the router uses standard routing procedures to match the packet against the explicit routing table, determining the next hop for forwarding.

5 Evaluation

5.1 Security Analysis

5.1.1 HEPFS Security Analysis

Traffic Analysis. An attacker may capture traffic data, primarily metadata, within a target network using sniffing techniques to deduce the identities and intentions of the communicating parties. In HEPFS, however, users employ data encryption protocols such as SSL/TLS, ensuring that packet contents and addresses remain in a homomorphic ciphertext state. Furthermore, HEPFS packets are indistinguishable from regular packets. Consequently, it becomes challenging for the attacker to extract user information in this encrypted state.

Timing Attacks. Attackers leverage temporal variations in system operations to infer sensitive information, such as passwords, cryptographic keys, and tokens. HEPFS utilizes homomorphic encryption in its routing forward algorithm, providing strong resistance to brute-force attacks on modern computing systems. Additionally, the security of homomorphic algorithms can be bolstered through dynamic adjustment of encryption parameters.

Router Node Protection. Routers are considered untrusted entities that perform only encrypted matching and forwarding. Compromised routers are unable to decrypt communications.

5.1.2 HTree Security Analysis

If a malicious router attempts to profit by analyzing routing forwarding information, such as access patterns of critical institutions, it would be thwarted by HEPFS's dual-layer defenses.

Homomorphic Encryption Barrier. Both HTree routing tables and packet addresses are encrypted. Brute-forcing either demands excessive computational resources.

Preventing Reverse Derivation. Even when routers derive HTree arrangement patterns from plaintext RIB tables, the randomness of homomorphic encryption prevents the inference of keys. Additionally, route aggregation ensures that packet IPs only partially match routing entries. Consequently, attackers cannot reconstruct full plaintext IPs through ciphertext-plaintext pairs or full-text matching.

5.2 Packet Structure Analysis

This subsection offers a comparative analysis of the communication overhead of HEPFS and compare it with the HR, Tor and Crowds.

An HR session involves two types of IP packets: $INIT_{HR}$ *Packet* and $Data_{HR}$ *Packet*. The $INIT_{HR}$ *Packet* contains two destination addresses: $\mathrm{Enc}_{HE}(\mathrm{dst})$ and $\mathrm{Enc}_{PK}(\mathrm{dst})$. Additionally, it constructs the encrypted path ESDP and EDSP as the IP packets are forwarded through the network. The $Data_{HR}$ *The packet* includes complete path information, which is utilized for path selection during data transmission, and it also incorporates session ID and hop counter to track the transmission path.

The HR packet contains at least five more fields than the HEPFS packet. As the session path lengthens, the ESDP and EDSP data sizes also increase. Additionally, HR packets include numerous custom fields, which require more logical judgments by the router. This complicates the design of the forwarding pipeline and increases the overall processing time.

5.3 Modeling Analysis

This section employs a straightforward modeling approach to evaluate the differences in path lengths and forwarding delays among HEPFS, Tor, and Crowds across various network topologies. To determine the path lengths for different topology sizes, we utilized the anonymous communication system model developed by Yong Guan [12] to simulate Tor and Crowds. To model forward delay, we applied the router model proposed by Umit Y. Ogras [23].

5.3.1 Models of Anonymous Communication Systems

Without considering the relationship between length and factors such as defensiveness and latency (because it is more complex and there is no suitable model that can accurately define the relationship), this paper takes the expected value of path length for Tor and Crowds. Assuming that the length of the packet l follows a uniform distribution over the interval $[a, b]$, its expected value L is:

$$L = E(l) = \frac{a+b}{2} \tag{13}$$

5.3.2 Router Performance Modeling

This model represents a novel on-chip router performance framework, supporting various network topologies and deterministic routing. It enables the analysis of performance metrics such as average delay and throughput. This routing model incorporates numerous fine-grained parameters, such as traffic rate and packet arrival rate. However, this paper focuses solely on the impact of path length and packet size on routing performance. Consequently, only the relevant parameters are extracted, as listed in Table 1.

Table 1: Symbol Description of Router Model

Input	Description
S	Size of data packet (unit: bit)
W	Network channel bandwidth (unit: bit/sec)
H_s	Service time of header flit
T_s	Service time of data packet

The packet service time defined by the model is:

$$T_s = H_s + \lceil S/W \rceil \tag{14}$$

Here, H_s is associated with the router and includes the traversal time through routers and links. Since this parameter is common in the current evaluation and does not affect the comparative results, it is set to $H_s = 0$ for simplicity of analysis.

Based on the model derivation, the average delay for a packet transmitted from source node s to destination node d is expressed as:

$$T_{sd} = T_q + \sum_{(i,j)\in\Pi_{sd}} (T_{ij} + T_s) \tag{15}$$

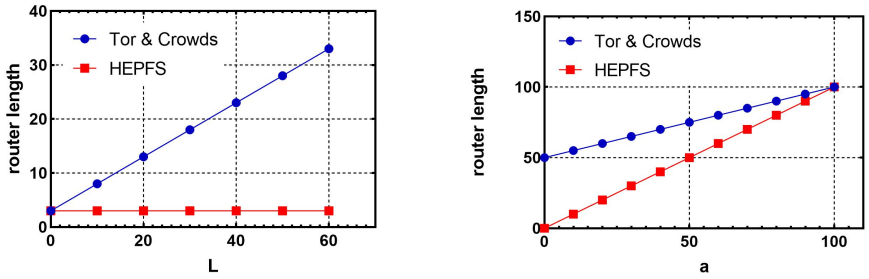
Here, T_q denotes the queuing delay at the source, T_{ij} represents the queuing delay at router i in channel j, and T_s is the router service time. Since only T_s in Eq. 15 relates to packet size, the remaining parameters can be treated as constants. To facilitate calculation and analysis in the graphical plots, the irrelevant variables T_q and T_{ij} are set to zero. The final simplified formula for calculating L_{sd} becomes:

$$T_{sd} = L \cdot T_s \tag{16}$$

In this equation, L denotes the path length from source s to destination d. It is important to note that L includes the source node s itself.

5.3.3 Path length comparison

The experimental parameters are: i) the lower path limit a; ii) the upper path limit b; iii) and the difference L between the shortest and longest paths.

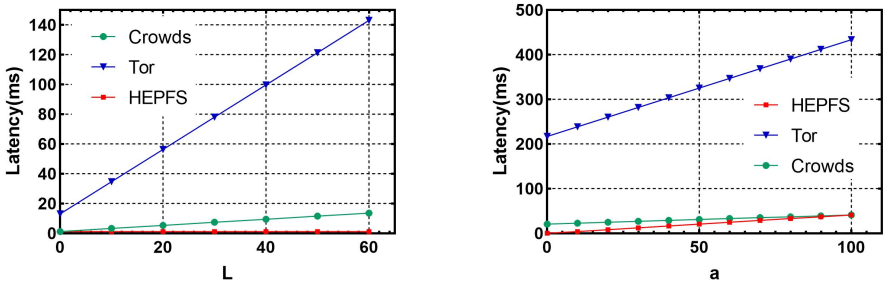


(a) Path lengths under a = 3. (b) Path lengths under b = 100.

Figure 5: Path length comparison of HEPFS with Tor and Crowds.

Figure 5 illustrates the differences in path length between HEPFS, Tor, and Crowds with variable path lengths. This paper examines two scenarios: one where a is fixed and L increases, and another where b is fixed and a increases.

The findings indicate that the privacy features of Tor and Crowds cause packets to consistently deviate from the shortest path during forwarding. Moreover, the longer the topology's longest path, the greater the forwarding path expectation. While this impact can be reduced by increasing the shortest path length, anonymous communication systems must also consider the long path effect, which increases latency and resource consumption without enhancing anonymity. Consequently, routing policies in practical scenarios will not permit this. In comparison, HEPFS demonstrates an advantage in path selection over anonymous communication systems such as Tor and Crowds.



(a) Forwarding delays under a = 3. (b) Forwarding delay under b = 100.

Figure 6: Forwarding Latency Comparison of HEPFS with Tor and Crowds.

5.3.4 Delay Comparison

The current simulation does not incorporate anonymity features such as relay nodes, jondo nodes, or homomorphic routing forwarding as parameters, as these features currently lack sufficient modeling/theoretical foundations to enable reasonable computational implementation.

This paper elucidates the packet structures of anonymous communication systems. HEPFS and Crowds do not modify packet structures, while Tor introduces cells within TLS records. In Torv4, a cell consists of a 4-byte CircID, a 1-byte Command field, and a 509-byte body. The current network packet omits the application layer and payload, maintaining a fixed size for the remaining components.

$$TCP(20\ bytes) + IPv4(20\ bytes) + MAC(14\ bytes) = 54\ bytes \tag{17}$$

Without calculating the load and using IPv4 and TCP protocols, the packet size is 54 bytes (432 bits) for HEPFS and Crowds, and 568 bytes (4544 bits) for Tor.

The experiments in this subsection employ Yong Guan's model with integrated path variables, while simultaneously utilizing Umit Y. Ogras's router model for latency simulation. The experimental parameters are as follows: i) packet size; ii) fixed model parameters H_S, T_q, T_{ij} set to 0 (not a factor in this evaluation); iii) variable model parameter $W = 1Mbps$.

The results from merging the two model simulations are depicted in Fig. 6, illustrating a similar trend in forwarding delay as observed in the path model. Introducing the *packet size* variable reveals that Tor exhibits suboptimal latency due to redundant packet nesting overhead. In contrast, Crowds and HEPFS perform better than Tor, as they do not involve extended packets. Furthermore, HEPFS surpasses Crowds in latency performance due to its optimized routing paths.

5.4 Experimental Analysis

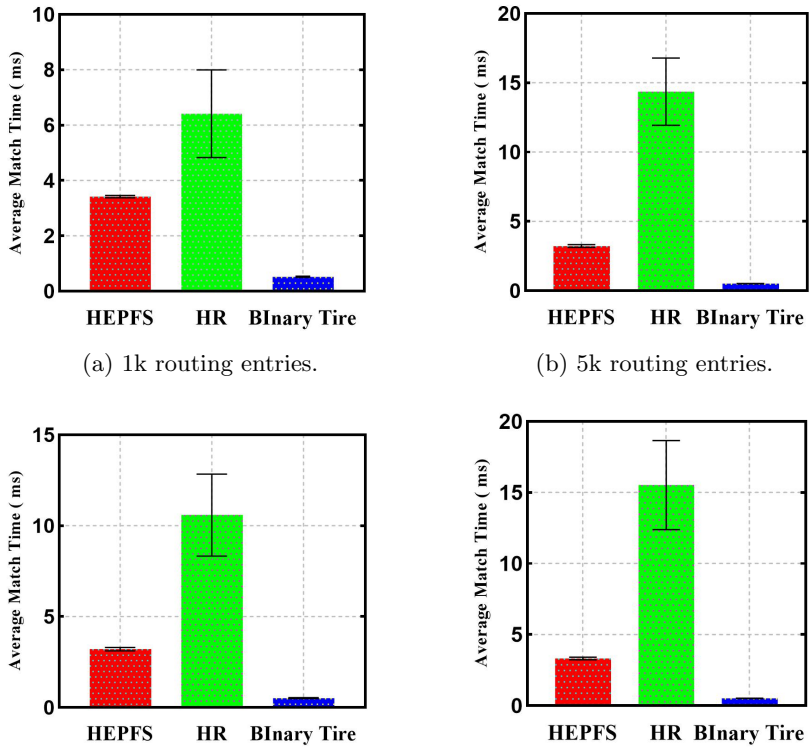
We assessed the complexity and performance of route matching algorithms for HEPFS, HR, and a benchmark algorithm (Binary Trie), focusing on the communication overhead between HEPFS and HR. We implemented the routing algorithms for HEPFS, HR, and Binary Trie, and utilized mobile operator routing tables sourced from the web as our test dataset.

To assess the practical matching performance of routing table architectures at different scales of routing entries, we randomly selected various IP entry quantities from the dataset to create routing tables. Each table underwent 1000 iterations of routing match testing[16].

In this subsection, we examine the performance benefits of HEPFS compared to HR and highlight the distinctions between HEPFS and Binary Trie across three dimensions: algorithmic complexity, practical matching performance, and stability. It is important to note that, as HR is primarily tailored for IPv6 network architecture, the reproduction of HR for this study has been adapted to IPv4 to ensure a consistent basis for comparing the matching performance of HEPFS, HR, and Binary Trie in IPv4 networks.

Table 2: Comparison of algorithm complexities

Algorithm	Complexity of lookup	Complexity of match	Stability of match
HEPFS	$O(\frac{w}{4})$	$O(\frac{w*hp}{4})$	Strong
HR	$O(\log_2 n)$	$O(\log_2(n*hp))$	Low
Binary Trie	$O(\log_2 32)$	$O(\log_2 32)$	Strong



(a) 1k routing entries.

(b) 5k routing entries.

(c) 10k routing entries.

(d) 100k routing entries.

Figure 7: Matching Performance Comparison Across Varying Routing Entry Counts.

Algorithmic Complexity. Table 2 outlines the algorithmic complexity for the three routing matching mechanisms. Here, w denotes the average prefix length in the HEPFS routing table, n represents the height of the binary tree in the HR routing table, and hp indicates a homomorphic lookup operation. In terms of algorithmic complexity, both HEPFS and Binary Trie

demonstrate a query complexity of $O(1)$ under all conditions, which is more efficient than HR, whose complexity increases with the size of the routing table. In practical matching scenarios, the primary performance bottleneck stems from homomorphic operations. The integration of homomorphic encryption in HEPFS leads to marginally slower query times compared to Binary Trie, which has the same algorithmic complexity. Nonetheless, HEPFS still surpasses HR in query speed.

Actual Matching Performance. Fig. 7 demonstrates the matching performance of the three techniques. The matching times for HEPFS and Binary Trie remain stable at approximately 3 ms and 0.5 ms, respectively, while HR exhibits a peak matching time of around 15 ms as the number of routing entries grows. The performance disparity between HEPFS and Binary Trie in matching time can be largely attributed to the time-intensive homomorphic operations in HEPFS.

Stability. The test results for HEPFS indicate a consistent time of approximately 3 ms for 1000 IP matches. In contrast, HR demonstrates significant instability in actual matching, with a difference of over 5 ms between the worst and best-case scenarios. Binary Trie, which does not involve homomorphic operations, exhibits minimal fluctuation in matching time regardless of the number of queries, stabilizing at around 0.5 ms.

6 Conclusion

The HEPFS framework mitigates the risk of CR dishonesty by safeguarding user intent (session orientation) against interception or analysis through plaintext IP addresses in backbone networks. Compared to HR, a recently proposed privacy forwarding scheme with a comparable technical foundation, HEPFS demonstrates substantial advancements in route matching efficiency, packet complexity, and overall network overhead. Future efforts will concentrate on optimizing the underlying algorithms and enhancing the architectural design to further improve the stability and performance of HEPFS.

7 Acknowledgement

This research was supported by the National Key R&D Program of China under Grant No. 2022YFB3102304, and in part by National Natural Science Foundation of China Grants (62394323, U22B2031, 62225105).

References

[1] Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM **24**(2), 84–90 (Feb 1981), http://dx.doi.org/10.1145/358549.358563

[2] Cherubin, G., Jansen, R., Troncoso, C.: Online website fingerprinting: Evaluating website fingerprinting attacks on tor in the real world. In: 31st USENIX Security Symposium (USENIX Security 22). pp. 753–770. USENIX Association, Boston, MA (Aug 2022), https://www.usenix.org/conference/usenixsecurity22/presentation/cherubin

[3] Das, D., Meiser, S., Mohammadi, E., Kate, A.: Anonymity trilemma: Strong anonymity, low bandwidth overhead, low latency - choose two. In: 2018 IEEE Symposium on Security and Privacy (SP). p. 108–126. IEEE (May 2018), http://dx.doi.org/10.1109/SP.2018.00011

[4] Deng, X., Yin, Q., Liu, Z., Zhao, X., Li, Q., Xu, M., Xu, K., Wu, J.: Robust multi-tab website fingerprinting attacks in the wild. In: 2023 IEEE Symposium on Security and Privacy (SP). p. 1005–1022. IEEE (May 2023), http://dx.doi.org/10.1109/SP46215.2023.10179464

[5] Dingledine, R., Freedman, M.J., Molnar, D.: The free haven project: Distributed anonymous storage service. In: Federrath, H. (ed.) Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 25-26, 2000, Proceedings. Lecture Notes in Computer Science, vol. 2009, pp. 67–95. Springer (2000), https://doi.org/10.1007/3-540-44702-4_5

[6] Dingledine, R., Mathewson, N., Syverson, P.: Tor: The Second-Generation onion router. In: 13th USENIX Security Symposium (USENIX Security 04). USENIX Association, San Diego, CA (Aug 2004), https://www.usenix.org/conference/13th-usenix-security-symposium/tor-second-generation-onion-router

[7] Geddes, J., Schliep, M., Hopper, N.: Abra cadabra: Magically increasing network utilization in tor by avoiding bottlenecks. In: Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society. CCS'16, ACM (Oct 2016), http://dx.doi.org/10.1145/2994620.2994630

[8] Goldschlag, D.M., Reed, M.G., Syverson, P.F.: Hiding routing information. In: Anderson, R.J. (ed.) Information Hiding, First International Workshop, Cambridge, UK, May 30 - June 1, 1996, Proceedings. Lecture Notes in Computer Science, vol. 1174, pp. 137–150. Springer (1996), https://doi.org/10.1007/3-540-61996-8_37

[9] Guan, J., Liu, K., Yao, S., Hu, X., Qin, Y., Liu, J., Fu, S., Gao, X., Xu, K.: PIPE: Identity-aware privacy-enhanced source and path verification for strengthened network accountability. In: 2025 IEEE 33rd International Conference on Network Protocols (ICNP). pp. 1–11 (2025), https://doi.org/10.1109/ICNP65844.2025.11192382

[10] Guan, J., Wei, Z., You, I.: GRBC-based network security functions placement scheme in SDS for 5G security. J. Netw. Comput. Appl. **114**, 48–56 (2018), https://doi.org/10.1016/j.jnca.2018.03.013

[11] Guan, J., Yan, Z., Yao, S., Xu, C., Zhang, H.: GBC-based caching function group selection algorithm for SINET. J. Netw. Comput. Appl. **85**, 56–63 (2017). https://doi.org/10.1016/J.JNCA.2016.12.004, https://doi.org/10.1016/j.jnca.2016.12.004

[12] Guan, Y., Fu, X., Bettati, R., Zhao, W.: An optimal strategy for anonymous communication protocols. In: Proceedings 22nd International Conference on Distributed Computing Systems. p. 257–266. ICDCSW-02, IEEE Comput. Soc (2002), http://dx.doi.org/10.1109/ICDCS.2002.1022263

[13] Gulcu, C., Tsudik, G.: Mixing e-mail with babel. In: Proceedings of Internet Society Symposium on Network and Distributed Systems Security. pp. 2–16. NDSS-96, IEEE Comput. Soc. Press (1996), http://dx.doi.org/10.1109/NDSS.1996.492350

[14] Hogan, K., Servan-Schreiber, S., Newman, Z., Weintraub, B., Nita-Rotaru, C., Devadas, S.: Shortor: Improving tor network latency via multi-hop overlay routing. In: 2022 IEEE Symposium on Security and Privacy (SP). p. 1933–1952. IEEE (May 2022), http://dx.doi.org/10.1109/SP46214.2022.9833619

[15] Hu, X., Guan, J., Liu, K., Liu, J., Qin, Y., Yao, S.: HIDE: anonymous hop-by-hop identity authentication based on dynamic address label embedding. In: Security and Privacy in Communication Networks - 21st EAI International Conference, SecureComm 2025, Xiantan, China. pp. 1–19. Springer (2025)

[16] ISPIP: CMCC dataset (Oct 2024), https://ispip.clang.cn/cmcc.html, Accessed: Oct. 20, 2024

[17] Jakobsson, M., Juels, A., Rivest, R.L.: Making mix nets robust for electronic voting by randomized partial checking. In: 11th USENIX Security Symposium (USENIX Security 02). USENIX Association, San Francisco, CA (Aug 2002), https://www.usenix.org/conference/11th-usenix-security-symposium/making-mix-nets-robust-electronic-voting-randomized

[18] Jia, J., Meng, H., Hu, Y., Liu, K., Guan, J.: MENTOR: malicious network traffic detection through

optimized LLM-based recognition. In: Huang, D., Zhang, C., Zhang, Q., Pan, Y. (eds.) Advanced Intelligent Computing Technology and Applications - 21st International Conference, ICIC 2025, Ningbo, China, July 26-29, 2025, Proceedings, Part XV. Lecture Notes in Computer Science, vol. 15856, pp. 149–161. Springer (2025), https://doi.org/10.1007/978-981-96-9914-8_13

[19] Kulshrestha, A., Mayer, J.: Estimating incidental collection in foreign intelligence surveillance: Large-Scale multiparty private set intersection with union and sum. In: 31st USENIX Security Symposium (USENIX Security 22). pp. 1705–1722. USENIX Association, Boston, MA (Aug 2022), https://www.usenix.org/conference/usenixsecurity22/presentation/kulshrestha

[20] Lin, S., Xin, R., Goel, A., Yang, X.: Invicloak: An end-to-end approach to privacy and performance in web content distribution. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. p. 1947–1961. CCS '22, ACM (Nov 2022), http://dx.doi.org/10.1145/3548606.3559336

[21] Majeed, A., Lee, S.: Anonymization techniques for privacy preserving data publishing: A comprehensive survey. IEEE Access **9**, 8512–8545 (2021), http://dx.doi.org/10.1109/ACCESS.2020.3045700

[22] Nabeel, M., Appel, S., Bertino, E., Buchmann, A.P.: Privacy preserving context aware publish subscribe systems. In: López, J., Huang, X., Sandhu, R.S. (eds.) Network and System Security - 7th International Conference, NSS 2013, Madrid, Spain, June 3-4, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7873, pp. 465–478. Springer (2013), https://doi.org/10.1007/978-3-642-38631-2_34

[23] Ogras, U.Y., Marculescu, R.: Analytical router modeling for networks-on-chip performance analysis. In: 2007 Design, Automation & Test in Europe Conference & Exhibition. p. 1–6. IEEE (Apr 2007). https://doi.org/10.1109/date.2007.364440, http://dx.doi.org/10.1109/DATE.2007.364440

[24] Rahimi, M.: Larmix++: Latency-aware routing in mix networks with free routes topology. In: Kohlweiss, M., Pietro, R.D., Beresford, A.R. (eds.) Cryptology and Network Security - 23rd International Conference, CANS 2024, Cambridge, UK, September 24-27, 2024, Proceedings, Part I. Lecture Notes in Computer Science, vol. 14905, pp. 187–211. Springer (2024), https://doi.org/10.1007/978-981-97-8013-6_9

[25] Reiter, M.K., Rubin, A.D.: Crowds: anonymity for web transactions. ACM Transactions on Information and System Security **1**(1), 66–92 (Nov 1998), http://dx.doi.org/10.1145/290163.290168

[26] Rothenberger, B., Roos, D., Legner, M., Perrig, A.: Piskes: Pragmatic internet-scale key-establishment system. In: Proceedings of the 15th ACM Asia Conference on Computer and Communications Security. p. 73–86. ASIA CCS '20, ACM (Oct 2020), http://dx.doi.org/10.1145/3320269.3384743

[27] Shirazi, F., Simeonovski, M., Asghar, M.R., Backes, M., Diaz, C.: A survey on routing in anonymous communication protocols. ACM Computing Surveys **51**(3), 1–39 (Jun 2018), http://dx.doi.org/10.1145/3182658

[28] The Tor Project: The Tor Project – Privacy and Anonymity Online (2022), https://www.torproject.org, Accessed: June 2023

[29] Tusa, F., Griffin, D., Rio, M.: Homomorphic routing: Private data forwarding in the internet. In: Proceedings of the 2nd ACM SIGCOMM Workshop on Future of Internet Routing & Addressing. p. 1–7. ACM SIGCOMM '23, ACM (Sep 2023), http://dx.doi.org/10.1145/3607504.3609287

[30] Waldman, M., Mazières, D.: Tangler: a censorship-resistant publishing system based on document entanglements. In: Proceedings of the 8th ACM conference on Computer and Communications Security. p. 126–135. CCS01, ACM (Nov 2001), http://dx.doi.org/10.1145/501983.502002

[31] Waldman, M., Rubin, A.D., Cranor, L.F.: Publius: A robust, Tamper-Evident, Censorship-Resistant, and Source-Anonymous web publishing system. In: 9th USENIX Security Symposium (USENIX Security 00). USENIX Association, Denver, CO

(Aug 2000), https://www.usenix.org/conference/9th-usenix-security-symposium/publius-robust-tamper-evident-censorship-resistant-and

[32] Zohaib, A., Sheffey, J., Houmansadr, A.: Investigating traffic analysis attacks on apple icloud private relay. In: Proceedings of the ACM Asia Conference on Computer and Communications Security. p. 773–784. ASIA CCS '23, ACM (Jul 2023), http://dx.doi.org/10.1145/3579856.3595793