Preserving Both Privacy and Utility in Network Trace Anonymization

Meisam Mohammady
CIISE, Concordia University, Canada
m_ohamama@encs.concordia.ca

Lingyu Wang
CIISE, Concordia University, Canada
wang@ciise.concordia.ca

Yuan Hong
Illinois Institute of Technology
yuan.hong@iit.edu

Habib Louafi
Ericsson Research Security
habib.louafi@ericsson.com

Makan Pourzandi
Ericsson Research Security
makan.pourzandi@ericsson.com

Mourad Debbabi
CIISE, Concordia University, Canada
debbabi@encs.concordia.ca

ABSTRACT
As network security monitoring grows more sophisticated, there is an increasing need for outsourcing such tasks to third-party analysts. However, organizations are usually reluctant to share their network traces due to privacy concerns over sensitive information, e.g., network and system configuration, which may potentially be exploited for attacks. In cases where data owners are convinced to share their network traces, the data are typically subjected to certain anonymization techniques, e.g., CryptoPAn, which replaces real IP addresses with prefix-preserving pseudonyms. However, most such techniques either are vulnerable to adversaries with prior knowledge about some network flows in the traces, or require heavy data sanitization or perturbation, both of which may result in a significant loss of data utility. In this paper, we aim to preserve both privacy and utility through shifting the trade-off from between privacy and utility to between privacy and computational cost. The key idea is for the analysts to generate and analyze multiple anonymized views of the original network traces; those views are designed to be sufficiently indistinguishable even to adversaries armed with prior knowledge, which preserves the privacy, whereas one of the views will yield true analysis results privately retrieved by the data owner, which preserves the utility. We present the general approach and instantiate it based on CryptoPAn. We formally analyze the privacy of our solution and experimentally evaluate it using real network traces provided by a major ISP. The results show that our approach can significantly reduce the level of information leakage (e.g., less than 1% of the information leaked by CryptoPAn) with comparable utility.

CCS CONCEPTS
- Security and privacy → Privacy-preserving protocols;

KEYWORDS
Network trace anonymization, prefix-preserving anonymization, CryptoPAn, semantic attacks

1 INTRODUCTION
As the owners of large-scale network data, today’s ISPs and enterprises usually face a dilemma. As security monitoring and analytics grow more sophisticated, there is an increasing need for those organizations to outsource such tasks together with necessary network data to third-party analysts, e.g., Managed Security Service Providers (MSSPs) [1]. On the other hand, those organizations are typically reluctant to share their network trace data with third parties, and even less willing to publish them, mainly due to privacy concerns over sensitive information contained in such data. For example, important network configuration information, such as potential bottlenecks of the network, may be inferred from network traces and subsequently exploited by adversaries to increase the impact of a denial of service attack [2].

In cases where data owners are convinced to share their network traces, the traces are typically subjected to some anonymization techniques. The anonymization of network traces has attracted significant attention (a more detailed review of related works will be given in section 6). For instance, CryptoPAn replaces real IP addresses inside network flows with prefix preserving pseudonyms, such that the hierarchical relationships among those addresses will be preserved to facilitate analyses [3]. Specifically, any two IP addresses sharing a prefix in the original trace will also do so in the anonymized trace. However, CryptoPAn is known to be vulnerable to the so-called fingerprinting attack and injection attack [4–6]. In those attacks, adversaries either already know some network flows in the original traces (by observing the network or from other relevant sources, e.g., DNS and WHOIS databases) [7], or have deliberately injected some forged flows into such traces. By recognizing those known flows in the anonymized traces based on unchanged fields of the flows, namely, fingerprints (e.g., timestamps and protocols), the adversaries can extrapolate their knowledge to recognize other flows based on the shared prefixes [4]. We now demonstrate such an attack in details.

Example 1.1. In Figure 1, the upper table shows the original trace, and the lower shows the trace anonymized using CryptoPAn. In this example, without loss of generality, we only focus on source IPs. Inside each table, similar prefixes are highlighted through similar shading.
to send enough information to the third party analysts such that they can generate and analyze many different anonymized views of the original network trace; those anonymized views are designed to be sufficiently indistinguishable (which will be formally defined in subsection 2.4) even to adversaries armed with prior knowledge and performing the aforementioned attacks, which preserves the privacy; at the same time, one of the anonymized views will yield true analysis results, which will be privately retrieved by the data owner or other authorized parties, which preserves the utility. More specifically, our contributions are as follows.

(1) We propose a multi-view approach to the prefix-preserving anonymization of network traces. To the best of our knowledge, this is the first known solution that can achieve similar data utility as CryptoPAn does, while being robust against the so-called semantic attacks (e.g., fingerprinting and injection). In addition, we believe the idea of shifting the trade-off from between privacy and utility to between privacy and computational cost may potentially be adapted to improve other privacy solutions.

(2) In addition to the general multi-view approach, we detail a concrete solution based on iteratively applying CryptoPAn to each partition inside a network trace such that different partitions are anonymized differently in all the views except one (which yields valid analysis results that can be privately retrieved by the data owner). In addition to privacy and utility, we design the solution in such a way that only one seed view needs to be sent to the analysts, which avoids additional communication cost.

(3) We formally analyze the level of privacy guarantee achieved using our method, discuss potential attacks and solutions, and finally experimentally evaluate our solution using real network traces from a major ISP. The experimental results confirm that our solution is robust against semantic attacks with a reasonable computational cost.

The rest of the paper is organized as follows: Section 2 defines our models. Sections 3 introduces building blocks for our schemes. Section 4 details two concrete multi-view schemes based on CryptoPAn. Sections 5 presents the experimental results. Section Appendix 4.3 provides more discussions, and section 6 reviews the related work. Finally, section 7 concludes the paper.

# 2 MODELS

In this section, we describe models for the system and adversaries; we briefly review CryptoPAn; we provide a high level overview of our multi-view approach; finally, we define our privacy property. Essential definitions and notations are summarized in Table 1.

![Figure 1: An example of injection attack](image)

(1) Step 1: An adversary has injected three network flows, shown as the first three records in the original trace (upper table).

(2) Step 2: The adversary recognizes the three injected flows in the anonymized trace (lower table) through unique combinations of the unchanged attributes (Start Time and Src Port).

(3) Step 3: He/she can then extrapolate his/her knowledge from the injected flows to real flows as follows, e.g., since prefix 159.61 is shared by the second (injected), fifth (real) and sixth (real) flows, he/she knows all three must also share the same prefix in the original trace. Such identified relationships between flows in the two traces will be called matches from now on.

(4) Step 4: Finally, he/she can infer the prefixes or entire IPs of those anonymized flows in the original traces, as he/she knows the original IPs of his/her injected flows, e.g., the fifth and sixth last flows must have prefix 150.10, and the IPs of the fourth and last flows must be 10.1.1.10.

More generally, a powerful adversary who can probe all the subnets of a network using injection or fingerprinting can potentially de-anonymize the entire CryptoPAn output via a more sophisticated frequency analysis attack [4].

Most subsequent solutions either require heavy data sanitization or can only support limited types of analysis. In particular, the (k, j)-obfuscation method first groups together k or more flows with similar fingerprints and then bucketizes (i.e., replacing original IPs with identical IPs) j < k flows inside each group; all records whose fingerprints are not sufficiently similar to k − 1 others will be suppressed [2]. Clearly, both the bucketization and suppression may lead to significant loss of data utility. The differentially private analysis method first adds noises to analysis results and then publishes such aggregated results [17, 41, 42]. Although this method may provide privacy guarantee regardless of adversarial knowledge, the perturbation and aggregation prevent its application to analyses that demand accurate or detailed records in the network traces.

In this paper, we aim to preserve both privacy and utility by shifting the trade-off from between privacy and utility, as seen in most existing works, to between privacy and computational cost (which has seen a significant decrease lately, especially with the increasing popularity of cloud technology). The key idea is for the data owner...
2.1 The System and Adversary Model

Denote by $\mathcal{L}$ a network trace comprised of a set of flows (or records) $r_i$. Each flow includes a confidential multi-value attribute $A^P = \{IP_{src}, IP_{dst}\}$, and the set of other attributes $A = \{A_i\}$ is called the Fingerprint Quasi Identifier (fp-QI) [2]. Suppose the data owner would like the analyst to perform an analysis on $\mathcal{L}$ to produce a report $\Gamma$. To ensure privacy, instead of sending $\mathcal{L}$, an anonymization function $\mathcal{T}$ is applied to obtain an anonymized version $\mathcal{L}^\prime$. Thus, our main objective is to find the anonymization function $\mathcal{T}$ to preserve both the privacy, which means the analyst cannot obtain $\mathcal{T}$ or $\mathcal{L}$ from $\mathcal{L}^\prime$, and utility, which means $\mathcal{T}$ must be prefix-preserving.

In this context, we make following assumptions (similar to those found in most existing works [3–6]). i) The adversary is a honest-but-curious analyst (in the sense that he/she will exactly follow the approach) who can observe $\mathcal{L}^\prime$. ii) The anonymization function $\mathcal{T}$ is publicly known, but the corresponding anonymization key is not known by the adversary. iii) The goal of the adversary is to find all possible matches (as demonstrated in Example 1.1, an IP address may be matched to its anonymized version either through the fp-QI or shared prefixes) between $\mathcal{L}$ and $\mathcal{L}^\prime$. iv) Suppose $\mathcal{L}$ consists of $d$ groups each of which contains IP addresses with similar prefixes (e.g., those in the same subset), and among these the adversary can successfully inject or fingerprint $a \leq d$ groups (e.g., the demilitarized zone (DMZ) or other subnets to which the adversary has access). Accordingly, we say that the adversary has $\mathcal{SA}$ knowledge. v) Finally, we assume the communication between the data owner and the analyst is over a secure channel, and we do not consider integrity or availability issues (e.g., a malicious adversary may potentially alter or delete the analysis report).

2.2 The CryptoPAn Model

To facilitate further discussions, we briefly review the CryptoPAn [3] model, which gives a baseline for prefix-preserving anonymization.

**Definition 2.1.** Prefix-preserving Anonymization [3]: Given two IP addresses $a = a_1 a_2 \ldots a_{32}$ and $b = b_1 b_2 \ldots b_{32}$, and a one-to-one function $F_i : \{0, 1\}^{32} \rightarrow \{0, 1\}^{32}$, we say that
- $a$ and $b$ share a k-bit prefix ($0 \leq k \leq 32$), if and only if $a_1 a_2 \ldots a_k = b_1 b_2 \ldots b_k$, and $a_{k+1} \neq b_{k+1}$.
- $F$ is prefix-preserving, if, for any $a$ and $b$ that share a k-bit prefix, $F(a)$ and $F(b)$ also do so.

Given $a = a_1 a_2 \ldots a_{32}$ and $F(a) = a_1 a_2' \ldots a_{32}'$, the prefix-preserving anonymization function $F$ must necessarily satisfy the canonical form [3], as follows.

$$a_i' = a_i \oplus f_{i-1}(a_1 a_2 \ldots a_{i-1}), \quad i = 1, 2, \ldots, 32$$

where $f_i$ is a cryptographic function which, based on a 256/128-bit key $K$, takes as input a bit-string of length $i - 1$ and returns a single bit. Intuitively, the $i^{th}$ bit is anonymized based on $K$ and the preceding $i-1$ bits to satisfy the prefix-preserving property. The cryptographic function $f_i$ can be constructed as $L[R(P(a_1 a_2 \ldots a_{i-1}), K)]$ where $L$ returns the least significant bit, $R$ can be a block cipher such as Rijndael [30], and $P$ is a padding function that expands $a_1, a_2, \ldots, a_{i-1}$ to match the block size of $R$ [3]. In the following, $PP$ will stand for this CryptoPAn function and its output will be denoted by $a' = PP(a, K)$.

The advantage of CryptoPAn is that it is deterministic and allows consistent prefix-preserving anonymization under the same $K$. However, as mentioned earlier, CryptoPAn is vulnerable to semantic attacks, which will be addressed in next section.

2.3 The Multi-View Approach

We propose a novel multi-view approach to the prefix-preserving anonymization of network traces. The objective is to preserve both the privacy and the data utility, while being robust against semantic attacks. The key idea is to hide a prefix-preserving anonymized view, namely, the real view, among $N-1$ other fake views, such that an adversary cannot distinguish between those $N$ views, either using his/her prior knowledge or through semantic attacks. Our approach is depicted in Figure 2 and detailed below.

2.3.1 Privacy Preservation at the Data Owner Side.

**Step 1:** The data owner generates two CryptoPAn keys $K_0$ and $K_1$, and then obtains an anonymized trace using the anonymization function $PP$ (which will be represented by the gear icon inside this figure) and $K_0$. This initial anonymization step is designed to prevent the analyst from simulating the process as $K_0$ will never be given out. Note that this anonymized trace is still vulnerable to semantic attack and must undergo the remaining steps. Besides, generating this anonymized trace will actually be slightly more complicated due to migration as discussed later in Section 3.3.

**Step 2:** The anonymized trace is then partitioned (the partitioning algorithms will be detailed in Sections 3.2 and 4). Each partition is anonymized using $PP$ and key $K_1$, but the anonymization will be repeated, for a different number of times, on different partitions. For example, as the figure shows, the first partition is anonymized only once, whereas the second for three times, etc. The result of this step is called the seed trace. The idea is that, as illustrated by the different graphic patterns inside the seed trace, different partitions have been anonymized differently, and hence the seed trace in its entirety is no longer prefix-preserving, even though each partition is still prefix-preserving (note that this is only a simplified demonstration of the seed trace generator scheme which will be detailed in Section 4).

**Step 4:** The seed trace together with some supplementary parameters, including $K_1$, are outsourced to the analyst.

2.3.2 Utility Realization at the Data Analyst Side.

**Step 5:** The analyst generates totally $N$ views based on the received seed view and supplementary parameters. Our design will ensure one of those generated views, namely, the real view, will have all its partitions anonymized in the same way, and thus be prefix-preserving (detailed in Section 4), though the analyst (adversary) cannot tell which exactly is the real view.

**Step 6:** The analyst performs the analysis on all the $N$ views and generates corresponding reports.

**Step 7:** The data owner retrieves the analysis report corresponding to the real view following an oblivious random access memory (ORAM) protocol [23], such that the analyst cannot learn which view has been retrieved. Next, we define the privacy property for the multi-view solution.
2.4 Privacy Property against Adversaries

Under our multi-view approach, an analyst (adversary) will receive \( N \) different traces with identical fp-QI attribute values and different \( A^{IP} \) attribute values. Therefore, his/her goal now is to identify the real view among all the views, e.g., he/she may attempt to observe his/her injected or fingerprinted flows, or he/she can launch the aforementioned semantic attacks on those views, hoping that the real view might respond differently to those attacks. Therefore, the main objective in designing an effective multi-view solution is to satisfy the indistinguishability property which means the real view must be sufficiently indistinguishable from the fake views under semantic attacks. Motivated by the concept of Differential Privacy [37], we propose the \( \epsilon \)-indistinguishability property as follows.

**Definition 2.2.** \( \epsilon \)-Indistinguishable Views: A multi-view solution is said to satisfy \( \epsilon \)-Indistinguishability against an \( S_a \) adversary if and only if (both probabilities below are from the adversary’s point of view)

\[
\exists \epsilon \geq 0, \ s.t. \ \forall i \in \{1, 2, \ldots, N\} \Rightarrow \epsilon^{-\epsilon} \leq \frac{\Pr(\text{view } i \text{ may be the real view})}{\Pr(\text{view } r \text{ may be the real view})} \leq \epsilon^\epsilon \quad (2)
\]

In Definition 2.2, a smaller \( \epsilon \) value is more desirable as it means the views are more indistinguishable from the real view to the adversary. For example, an extreme case of \( \epsilon = 0 \) would mean all the views are equally likely to be the real view of the adversary (from now on, we call these views the real view candidates). In practice, the value of \( \epsilon \) would depend on the specific design of a multi-view solution and also on the adversary’s prior knowledge, as will be detailed in following sections.

Finally, since the multi-view approach requires outsourcing some supplementary parameters, we will also need to analyze the security/privacy of the communication protocol (privacy leakage in the protocol, which complements the privacy analysis in output of the protocol) in semi-honest model under the theory of secure multiparty computation (SMC) [43], [44] (see section 4.2.4).

3 THE BUILDING BLOCKS

In this section, we introduce the building blocks for our multi-view mechanisms, namely, the iterative and reverse CryptoPAn, partition-based prefix preserving, and CryptoPAn with IP-collision (migration).

3.1 Iterative and Reverse CryptoPAn

As mentioned in section 2.3, the multi-view approach relies on iteratively applying a prefix preserving function \( PP \) for generating the seed view. Also, the analyst will invert such an application of \( PP \) in order to obtain the real view (among fake views). Therefore, we first need to show how \( PP \) can be iteratively and reversely applied.

First, it is straightforward that \( PP \) can be iteratively applied, and the result also yields a valid prefix-preserving function. Specifically, denote by \( PP^j(a, K) \) \((j > 0) \) the iterative application of \( PP \) on IP address \( a \) using key \( K \), where \( j \) is the number of iterations, called the index. For example, for an index of two, we have \( PP^2(a, K) = PP(PP(a, K), K) \). It can be easily verified that given any two IP addresses \( a \) and \( b \) sharing a k-bit prefix, \( PP^j(a, K) \) and \( PP^j(b, K) \) will always result in two IP addresses that also share a k-bit prefix (i.e., \( PP^j \) is prefix-preserving). More generally, the same also holds for applying \( PP \) under a sequence of indices and keys (for both IPs), e.g., \( PP^j(PP^i(a, K_0), K_1) \) and \( PP^j(PP^i(b, K_0), K_1) \) will also share k-bit prefix. Finally, for a set of IP addresses \( S \), iterative \( PP \) using a single key \( K \) satisfies the following associative property:

\[
\forall S, K, \text{ and } i, j \in \mathbb{Z} \text{ (integers): } \quad PP^i(PP^j(S, K), K) = PP^{i+j}(S, K) \quad (3)
\]

On the other hand, when a negative number is used as the index, we have a reverse iterative CryptoPAn function \( RPP \) for short,
as formally characterized in Theorem 3.1 (the proof is in Appendix A.1).

**Theorem 3.1.** Given IP addresses \( a = a_1 a_2 \cdots a_{32} \) and \( b = PP(a, K) = b_1 b_2 \cdots b_{32} \), the function \( RPP(\cdot) : \{0, 1\}^{32} \rightarrow \{0, 1\}^{32} \) defined as

\[
RPP(b, K) = c = c_1 c_2 \cdots c_{32}
\]

where \( c_i = b_i \oplus f_{31-i}(c_1 \cdots c_{i-1}) \)

is the inverse of the PP function given in Equation 1, i.e., \( c = a \).

### 3.2 Partition-based Prefix Preserving

As mentioned in Section 2.3, the central idea of the multi-view approach is to divide the trace into partitions (Step 2), and then anonymize those partitions iteratively, but for different numbers of iterations (Step 3) in this subsection.

Given \( S \) as a set of IP addresses, we may divide \( S \) into partitions in various ways, e.g., forming equal-sized partitions after sorting \( S \) based on either the IP addresses or corresponding timestamps. The partitioning scheme will have a major impact on the privacy, and we will discuss two such schemes in next section.

Once the trace is divided into partitions, we can then apply PP on each partition separately, denoted by \( PP(P_i, K) \) for the \( i \)th partition. Specifically, given \( S \) divided as a set of \( m \) partitions \( [P_1, P_2, \ldots, P_m] \), we define a key vector \( V = [v_1, v_2, \ldots, v_m] \) where each \( v_i \) is a positive integer indicating the number of times PP should be applied to \( P_1 \), namely, the key index of \( P_1 \). Given a cryptographic key \( K \), we can then define the partition-based prefix preserving anonymization of \( S \) as \( PP(S, V, K) = \{PP^{v_1}(P_1, K), PP^{v_2}(P_2, K), \ldots, PP^{v_m}(P_m, K)\} \).

We can easily extend the associative property in Equation 3 to this case as the following (which will play an important role in designing our multi-view mechanisms in next section).

\[
PP[PP(S, V_1, K), V_2, K] = PP(S, (V_1 + V_2), K)
\]

### 3.3 IP Migration: Introducing IP-Collision into CryptoPAn

As mentioned in Section 2.3, once the analyst (adversary) receives the seed view, he/she would generate many indistinguishable views among which only one, the real view, will be prefix preserving across all the partitions, while the other (fake) views do not preserve prefixes across the partitions (Step 5). However, the design would have a potential flaw under a direct application of CryptoPAn. Specifically, since the original CryptoPAn design is collision resistant [3], the fact that similar prefixes are only preserved in the real view across partitions would allow an adversary to easily distinguish the real view from others.

**Example 3.1.** Figure 3 illustrates this flaw. The original trace includes three different addresses and has been divided into two partitions \( P_1 \) and \( P_2 \). As illustrated in the figure, the real view is easily distinguishable from the two fake views as the shared prefixes (159.61) between addresses in \( P_1 \) and \( P_2 \) only appear in the real view. This is because, since the partitions in fake views have different rounds of PP applied, and since the original CryptoPAn design is collision resistant [3], the shared prefixes will no longer appear. Hence, the adversary can easily distinguish the real view from others.

To address this issue, our idea is to create collisions between different prefixes in fake views, such that adversaries cannot tell whether the shared prefixes are due to prefix preserving in the real view, or due to collisions in the fake views. However, due to the collision resistance property of CryptoPAn [3], there is only a negligible probability that different prefixes may become identical even after applying different iterations of PP, as shown in the above example. Therefore, our key idea of IP migration is to first replace the prefixes of all the IPs with common values (e.g., zeros), and then fabricate new prefixes for them by applying different iterations of PP. This IP migration process is designed to be prefix-preserving (i.e., any IPs sharing prefixes in the original trace will still share the new prefixes), and to create collisions in fake views since the addition of key indices during view generation can easily collide. Next, we demonstrate this IP migration technique in an example.

\[
\text{Table 3: An example showing the shared prefixes (which allows it to be identified by adversaries) in the real view from others.}
\]

**Figure 3:** An example showing only the real view contains shared prefixes (which allows it to be identified by adversaries)

\[
\text{Figure 4: An example showing, by removing shared prefixes and fabricating them with the same rounds of PP, both fake view and real view may now contain fake or real shared prefixes (which makes them indistinguishable)}
\]

**Example 3.2.** In Figure 4, the first stage shows the same original trace as in Example 3.1. In the second stage, we “remove” the prefixes of all IPs and replace them with all zeros (by zeroing them with their own prefixes). Next, in the third stage, we fabricate new prefixes by applying different iterations of PP in a prefix preserving manner, e.g., the first two IPs still sharing a common prefix (11.215) different from that of the last IP. However, note that whether two IPs share the new prefixes only depends on their key indices now, e.g., 1 for first two IPs and 2 for the last IP. This is how we can create collisions in the next stage (the fake view) where the first and last IPs coincidentally share
the same prefix 95.24 due to their common key indices 2 (however, note these are the addition results of different key indices from the migration stage and the view generation stage, respectively). Now, the adversary will not be able to tell which of those views is real based on the existence of shared prefixes.

We now formally define the migration function in the following.

Definition 3.1. Migration Function: Let $S$ be a set of IP addresses consists of $d$ groups of IPs $S_1, S_2, \ldots, S_d$ with distinct prefixes $s_1, s_2, \ldots, s_d$ respectively, and $K$ be a random CryptoPAn key. Migration function $M : S \times C(\text{set of positive integers}) \rightarrow S^*$ is defined as

\[
S^* = M(S) = \{S_i^\prime | i \in \{1, 2, \ldots, d\}\}
\]

where $S_i^\prime = \{PP^r(s_i \oplus a_j, K), \forall a_j \in S_i\}$

(6)

where $C = PRNG(d, d) = \{c_1, c_2, \ldots, c_d\}$ is the set of $d$ non-repeating random key indices generated between $[1, d]$ using a cryptographically secure random number generator.

4 ε-INDISTINGUISHABLE MULTI-VIEW MECHANISMS

We first present a multi-view mechanism based on IP partitioning in Section 4.1. We then propose a more refined scheme based on distinct IP partitioning with key vector generator in Section 4.2.

4.1 Scheme I: IP-based Partitioning Approach

To realize the main ideas of multi-view anonymization, as introduced in Section 2.3, we need to design concrete schemes for each step in Figure 2. The key idea of our first scheme is the following. We divide the original trace in such a way that all the IPs sharing prefixes will always be placed in the same partition. This will prevent the attack described in Section 3.3, i.e., identifying the real view by observing shared prefixes across different partitions. As we will detail in Section 4.1.4, this scheme can achieve perfect indistinguishability without the need for IP migration (introduced in Section 3.3), although it has its limitations which will be addressed in our second scheme. Both schemes are depicted in Figure 5 and detailed below.

Specifically, our first scheme includes three main steps: privacy preservation (Section 4.1.1), utility realization (Section 4.1.2), and analysis report extraction (Section 4.1.3).

4.1.1 Privacy Preservation (Data Owner). The data owner performs a set of actions to generate the seed trace $L_0^*$ together with some parameters to be sent to the analyst for generating different views. These actions are summarized in Algorithm 1, and detailed in the following.

- Applying CryptoPAn using $K_0$: First, the data owner generates two independent keys, namely $K_0$ (for initial anonymization, which never leaves the data owner) and $K$ (key used for later anonymization steps). The data owner then generates the initially anonymized trace $L_0 = PP(L, K_0)$. This step is designed to prevent the adversary from simulating the scheme, e.g., using a brute-force attack to revert the seed trace back to the original trace in which he/she can recognize some original IPs. The leftmost block in Figure 5 shows an example of the initialized anonymized trace.

- Trace partitioning based on IP-value: The initially anonymized trace is partitioned based on IP values. Specifically, let $S$ be the set of IP addresses in $L_0$ consisting of $d$ groups of IPs $S_1, S_2, \ldots, S_d$ with distinct prefixes $s_1, s_2, \ldots, s_d$ respectively; we divide $L_0$ to $d$ partitions, each of which is the collection of all records containing one of these groups. For example, the upper part of Figure 5 depicts how our first scheme works. The set of three IPs are divided into two partitions where $P_1$ includes both IPs sharing the same prefix, 45.20.15.89 and 45.20.141.20, whereas the last IP 121.25.01.08 goes to $P_2$ since it does not share a prefix with others.

- Seed trace creation: The data owner in this step generates the seed trace using a $d$-size (recall that $d$ is the number of partitions) random key vector.

4.1.2 Network Trace Analysis (Analyst). The analyst generates the $N$ views requested by the data owner, which is summarized in Algorithm 2 in Appendix C and formalized below.

\[
L_t = PP(L_{t-1}, V, K), \quad i \in \{1, \ldots, N\}
\]

(7)

Since boundaries of partitions must be recognizable by the analyst to allow him/her to generate the views, we modify the time-stamp
of the records that are on the boundaries of each partition by changing the most significant digit of the time stamps which is easy to verify and does not affect the analysis as it can be reverted back to its original format by the analyst. Next, the analyst performs the requested analysis on all $N$ views and generates $N$ analysis reports $\Gamma_1, \Gamma_2, \ldots, \Gamma_N$.

### 4.1.3 Analysis Report Extraction (Data Owner)

The data owner is only interested in the analysis report that is related to the real view, which we denote by $\Gamma_r$. To minimize communication overhead, instead of requesting all the analysis reports $\Gamma_i$ of the generated views, the data owner can fetch only the one that is related to the real view $\Gamma_r$. He/she can employ the oblivious random accesses memory (ORAM) [23] to do so without revealing the information to the analyst (we will discuss alternatives in Section 6).

### 4.1.4 Security Analysis

We now analyze the level of indistinguishability provided by the scheme. Recall the indistinguishability property defined in Section 2; a multi-view mechanism is $\epsilon$-indistinguishable if and only if

$$
\exists \epsilon \geq 0, \text{ s.t. } \forall i \in \{1, 2, \ldots, N\} \Rightarrow
\epsilon^{-\epsilon} \leq \frac{Pr(\text{view } i \text{ may be the real view})}{Pr(\text{view } r \text{ may be the real view})} \leq \epsilon^\epsilon
$$

The statement inside the probability is the adversary’s decision on a view, declaring it as fake or a real view candidate, using his/her $S_a$ knowledge. Moreover, we note that generated views differ only in their IP values (ip-QI attributes are similar for all the views). Hence, the adversary’s decision can only be based on the published set of IP’s in each view through comparing shared prefixes among those IP addresses which he/she already know ($S_a$). Accordingly, in the following, we define a function to represent all the prefix relations for a set of IP’s.

**Lemma 4.1.** For two IP addresses $a$ and $b$, function $Q : \{0, 1\}^{82} \times \{0, 1\}^{32} \to \mathbb{N}$ returns the number of bits in the prefix shared between $a$ and $b$

$$
Q(a, b) = 31 - \lfloor \log_2 a \oplus b \rfloor
$$

where $\lfloor \cdot \rfloor$ denotes the floor function.

**Definition 4.1.** For a multiset of $n$ IP addresses $S$, the Prefixes Indicator Set (PIS) $\mathcal{R}(S)$ is defined as follows.

$$
\mathcal{R}(S) = \{Q(a_i, a_j) | \forall a_i, a_j \in S, i, j \in \{1, 2, \cdots, n\}\}
$$

Note that PIS remains unchanged when CryptoPAn is applied on $S$, i.e., $\mathcal{R}(PP(S, K)) = \mathcal{R}(S)$. In addition, since the multi-view solution keeps all the other attributes intact, the adversary can identify his/her pre-knowledge in each view and construct prefixes.

Figure 5: An example of a trace which undergoes multi-view schemes I, II
indicator sets out of them. Accordingly, we denote by $R_{ai}$ the PIS constructed by the adversary in view $i$.

**Definition 4.2.** Let $R_{i}$ be the PIS for the adversary’s knowledge, and $R_{ai}, i \in \{1, \ldots, N\}$ be the PIS constructed by the adversary in view $i$. A multi-view solution then generates $\epsilon$-indistinguishable views against an $S_a$ adversary if and only if

$$\exists \epsilon \geq 0, \text{ s.t. } \forall i \in \{1, 2, \ldots, N\} \Rightarrow$$

$$e^{-\epsilon} \leq \frac{Pr(R_{ai} = R_{i})}{Pr(R_{as} = R_{i})} \leq e^\epsilon$$  \hspace{1cm} (9)

**Lemma 4.2.** The indistinguishability property, defined in equation 9 can be simplified to

$$\exists \epsilon \geq 0, \text{ s.t. } \forall i \in \{1, 2, \ldots, N\} \Rightarrow$$

$$Pr(R_{ai} = R_{i}) \geq e^{-\epsilon}$$  \hspace{1cm} (10)

**Proof.** $Pr(R_{ai} = R_{i}) = 1$ as view $r$ is the prefix preserving output. Moreover, $\forall \epsilon \geq 0$ we have $e^\epsilon \geq 1$. \hfill $\square$

From the above, we only need to show $R_{ai} = R_{i}$ (each generated view $i$ is a real view candidate).

**Theorem 4.3.** Scheme I satisfies equation 10 with $\epsilon = 0$.

**Proof.** Scheme I divides the trace into $d$ (number of prefix groups) partitions containing all the records that have similar prefixes. Hence, for any partition $P_i (1 \leq i \leq d)$, any two IP addresses $a$ and $b$ inside $P_i$, and for any $m, n \leq N$, we have $R_m(a, b) = R_n(a, b)$ because $a$ and $b$ are always assigned with equal key indices. Moreover, for any two IP addresses $a$ and $b$ in any two different partitions and any $m, n \leq N$, we have $R_m(a, b) = R_n(a, b) = 0$ since they do not share any prefixes. \hfill $\square$

The above discussions show that scheme I produces perfectly indistinguishable views ($\epsilon = 0$). In fact, it is robust against the attack explained in Section 3.3 and thus does not require IP migration, because the partitioning algorithm already prevents addresses with similar prefixes from going into different partitions (the case in Figure 3). However, although adversaries cannot identify the real view, they may choose to live with this fact, and attack each partition inside any (fake or real) view instead, using the same semantic attack as shown in Figure 1. Note that our multi-view approach is only designed to prevent attacks across different partitions, and each partition itself is essentially still the output of CryptoPAn and thus still inherits its weakness.

Fortunately, the multi-view approach gives us more flexibility in designing specific schemes to further mitigate such a weakness of CryptoPAn. We present next scheme II which sacrifices some indistinguishability (in the sense of slightly less real view candidates) to achieve better protected partitions.

### 4.2 Scheme II: Multi-view Using N Key Vectors

To address the limitation of our first scheme, we propose the next scheme, which is different in terms of the initial anonymization step, IP partitioning, and key vectors for view generation. The data owner’s and the analyst’s actions are summarized in Algorithms 3, 4.

#### 4.2.1 Initial Anonymization with Migration

First, to mitigate the attack on each partition, we must relax the requirement that all shared prefixes go into the same partition. However, as soon as we do so, the attack of identifying the real view through prefixes shared across partitions, as demonstrated in Section 3.3, might become possible. Therefore, we modify the first step of the multi-view approach (initial anonymization) to enforce the IP migration technique. Figure 6 demonstrates this. The original trace is first anonymized with $K_0$, and then the anonymized trace goes through the migration process, which replaces the two different prefixes (97.17 and 75.91) with different iterations of PP, as discussed in Section 3.3.

![Figure 6: The updated initial anonymization (Step 1 in Figure 2) for enforcing migration](image)

#### 4.2.2 Distinct IP Partitioning and N Key Vectors Generation

For the scheme, we employ a special case of IP partitioning where each partition includes exactly one distinct IP (i.e., the collection of all records containing the same IP). For example, the trace shown in Figure 5 includes three distinct IP addresses 150.10.1.128, 10.10.1, and 150.10.20.0. Therefore, the trace is divided into three partitions. Next, the data owner will generate the seed view as in the scheme, we employ a special case of IP partitioning where each partition includes exactly one distinct IP (i.e., the collection of all records containing the same IP). For example, the trace shown in Figure 5 includes three distinct IP addresses 150.10.1.128, 10.10.1, and 150.10.20.0. Therefore, the trace is divided into three partitions. Next, the data owner will generate the seed view as in the scheme, although the key $V_0$ will be generated completely differently, as detailed below.

Let $S^* = \{S_1^*, S_2^*, \ldots, S_d^*\}$, be the set of IP addresses after the migration step. Suppose $S^*$ consists of $D$ distinct IP addresses. We denote by $C^*$ the multiset of totally $D$ migration keys for those distinct IPs (in contrast, the number of migration keys in $C$ is equal to the number of distinct prefixes, as discussed in Section 3.3). Also, let $PRNG(d, D, i)$ be the set of $D$ random number generated between $[1, d]$ using a cryptographically secure pseudo random number generator at iteration $i^{th}$. The data owner will generate $N + 1$ key vector $V_i$ as follows.

$$V_i = PRNG(d, D, i) - PRNG(d, D, i - 1),$$

$$\forall i \neq r \in \{1, 2, \ldots, N\} \hspace{1cm} (11)$$

and

$$V_0 = PRNG(d, D, 0) - C^*$$

$$V_r = C^* - PRNG(d, D, r - 1) \hspace{1cm} (12)$$

**Example 4.1.** In Figure 7, the migration and random vectors are $C^* = \{1 \ 2\ 2\}$, $PRNG(2, 3, 0) = \{1 \ 2 \ 2\}$, $PRNG(2, 3, 1) = \{1 \ 2 \ 1\}$, and...
PRNG(2, 3, 2) = [2 2 1], respectively. The corresponding key vectors will be \( V_0 = [0 0 0] \), \( V_1 = [0 0 1] \) and \( V_2 = [1 0 0] \) where only \( V_1 \) and \( V_2 \) are outsourced.

In this scheme, the analyst at each iteration \( i \) generates a new set of IP addresses \( S^*_i = \{ S^*_1, S^*_2, \ldots, S^*_d \} \) by randomly grouping all the distinct IP addresses into a set of \( d \) prefix groups. In doing so, each new vector \( V_i \) essentially cancels out the effect of the previous vector \( V_{i-1} \), and thus introduces a new set of IP addresses \( S^*_i \) consisting of \( d \) prefix groups. Thus, it is straightforward to verify that the \( i^{th} \) generated view will prefix preserving (the addresses are migrated back to their groups using \( C^* \)).

**Example 4.2.** Figure 7 shows that, in each iteration, a different set (but with an equal number of elements) of prefix groups will be generated. For example, in the seed view, IP addresses 150.10.20.0 and 128.10.10.1 are mapped to prefix group 11.215.

### 4.2.3 Indistinguishability Analysis

By placing each distinct IP in a partition, our second scheme is not vulnerable to semantic attacks on each partition, since such a partition contains no information about the prefix relationship among different addresses. However, compared with scheme I, as we show in the following, this scheme achieves a weaker level of indistinguishability (higher \( \epsilon \)). Specifically, to verify the indistinguishability of the scheme, we calculate \( \Pr(\mathcal{R}_i = \mathcal{R}_a) \) for scheme II in the following. First, the number of all possible outcomes of grouping \( D \) IP addresses into \( d \) groups with predefined cardinalities is:

\[
N_{\text{total}} = \frac{D!}{|S_1|!|S_2|! \cdots |S_d|!}
\]

where \( |S_i| \) denotes the cardinality of group \( i \). Also the number of all possible outcomes of grouping \( D \) IP addresses into \( d \) groups while still having \( \mathcal{R}_{a,i} = \mathcal{R}_a \) is:

\[
N_{\text{real view candidates}} = \frac{\alpha! (D - \alpha)!}{\sum_{i=1}^{\alpha} \left( \frac{D!}{|S_i|!|S_2|! \cdots |S_d|!} \cdot \frac{d!}{\Pi_{j=1}^{d} |S_{a_j}|} \right)}
\]

for some \( a_j \in \{ 1, 2, \cdots, d \} \). This equation gives the number of outcomes when a specific set of \( \alpha \) IP addresses \( (S_{a}) \) are distributed into \( \alpha \) different groups and hence keeping \( \mathcal{R}_{a,i} = \mathcal{R}_a \) (i.e., the adversary cannot identify collision). Note that term \( \sum_{i=1}^{\alpha} \left( \frac{d!}{\Pi_{j=1}^{d} |S_{a_j}|} \right) \) is all the combinations of choosing this \( \alpha \) groups for the numerator to model all the \((|S_{a}| - 1)!\) combinations. Finally, we have

\[
\forall i \leq N : \ Pr(\mathcal{R}_{a,i} = \mathcal{R}_a) = \frac{N_{\text{real view candidates}}}{N_{\text{total}}} = \mathcal{A}
\]

Thus, to ensure the \( \epsilon \)-indistinguishability, the data owner needs to satisfy the expression in equation 15 which is a relationship between the number of distinct IP addresses, the number of groups, the cardinality of the groups in the trace and the adversary’s knowledge.

**Theorem 4.4.** The indistinguishability parameter \( \epsilon \) of the generated views in scheme II is lower-bounded by

\[
\ln \left( \frac{D^\alpha}{\alpha!} \cdot \frac{\Pi_{i=0}^{\alpha-1}(D - i)}{\Pi_{i=0}^{\alpha-1}(D - i)} \right)
\]

**Proof.** Let \( b_1, b_2, \ldots, b_n \) be positive real numbers, and for \( k = 1, 2, \cdots, n \) define the averages \( M_k \) as follows:

\[
M_k = \sum_{1 \leq i_1 \leq i_2 \leq \cdots \leq i_k \leq n} b_{i_1} b_{i_2} \cdots b_{i_k} \quad \left( \begin{array}{c} k \\ \end{array} \right)
\]

By Maclaurin’s inequality [29], which is the following chain of inequalities:

\[
M_k \geq \sqrt[k]{M_k} \geq \sqrt[k]{M_k} \geq \cdots \geq \sqrt[k]{M_k} \quad \left( \begin{array}{c} k \\ \end{array} \right)
\]

where \( M_1 = \frac{\sum_{i=1}^{n} b_i}{n} \), we have

\[
\mathcal{A} = \frac{\alpha! \left( \frac{D!}{\Pi_{i=0}^{\alpha-1}(D - i)} \right) \Pi_{i=1}^{\alpha}(M_1)^{i-1}}{\Pi_{i=1}^{\alpha}(D - i)}
\]

and since \( M_1 = \frac{\sum_{i=1}^{n} |S_i|}{n} = \frac{D}{d} \), we have

\[
\mathcal{A} \leq \frac{D^\alpha}{\alpha!} \cdot \frac{\Pi_{i=0}^{\alpha-1}(D - i)}{\Pi_{i=0}^{\alpha-1}(D - i)}
\]

Figure 8(a) shows how the lower-bound in Equation 16 changes with respect to different values of fraction \( d/D \) and also the adversary’s knowledge. As it is expected, stronger adversaries have more power to weaken the scheme which results in increasing \( \epsilon \) or increasing the chance of identifying the real view. Moreover, as it is illustrated in the figure, when fraction \( d/D \) grows, \( \epsilon \) tends to converge to very small values. Hence, to decrease \( \epsilon \), the data owner may increase \( d/D \) in [0, 1] by grouping addresses based on a bigger number of bits in their prefixes, e.g., a certain combination of 3 octets would be considered as a prefix instead of one or two. Another solution could be aggregating the original trace with some other traces for which the cardinalities of each prefix group are small. We study this effect in our experiments in Section 5 where we illustrate the concept especially in Figures 10, 11.

Finally, Figure 8(b) shows how variance of the cardinalities affects the indistinguishability for a set of fixed parameters \( D, d, \alpha \). In fact, when the cardinalities of the prefix groups are close (small \( \sigma \)), \( \mathcal{A} \) grows to meet the lower-bound in Theorem 4.4. Hence, from
the data owner perspective, a trace with a lower variance of cardinalities and a bigger fraction $d/D$ has a better chance of misleading adversaries who wants to identify the real view.

4.2.4 Security of the communication protocol. We now analyze the security/privacy of our communication protocol in semi-honest model under the theory of secure multiparty computation (SMC) [43], [44].

**Lemma 4.5.** Scheme II only reveals the CryptoPan Key $K$ and the seed trace $L_0^*$ in semi-honest model.

**Proof.** Recall that our communication protocol only involves one-round communication between two parties (data owner to data analyst). We then only need to examine the data analyst’s view (messages received from the protocol), which includes (1) $N$: the number of views to be generated, (2) $K$: the outsourced key, (3) $L_0^*$: the seed trace, and (4) $V_1, V_2, \ldots, V_N$: the key vectors. As we discuss in section 4.2.3, the probability of identifying the real view by the adversary using all provided information (key and vectors) depends on the adversary knowledge and the trace itself which clearly implies that such “leakage” is trivial.

Indeed, each of $N$ and $V_1, V_2, \ldots, V_N$ can be simulated by generating a single random number from a uniform random distribution (which proves that they are not leakage in the protocol). Specifically, the number of generated views $N$ is integer which is bounded by $N_0$, where $N_0$ is the maximum number of views the data owner can afford and all the entries in $V_1, V_2, \ldots, V_N$ are in $[-d, d]$ where $d$ is the number of groups. First, given integer $0 < N \leq N_0$, the probability that $N$ is simulated in the domain would be $Pr[Simulator = N] = \frac{1}{N_0}$. Then, $N$ can be simulated in polynomial time (based on the knowledge data analyst already knew, i.e., his/her input and/or output of the protocol). Similarly, all the random entries in $V_1, V_2, \ldots, V_N$ can also be simulated in polynomial time using a similar simulator (only changing the bound). Thus, the protocol only reveals the outsourced key $K$ and the seed trace $L_0^*$ in semi-honest model.

Note that outsourcing the $L_0^*$ and the outsourced key are trivial leakage. The outsourced key can be considered as a public key and leakage of $L_0^*$ which is considered as the output of the protocol was studied earlier. Finally, we study the setup leakage and show that the adversary cannot exploit outsourced parameters to increase $\epsilon$ (i.e., decrease the number of real view candidates) by building his/her own key vector.

**Lemma 4.6.** (proof in Appendix A.2) For an $S_\alpha$ adversary, who wants to obtain the least number of real view candidates, if condition $(2d - 2)^D > N$ holds, the best approach is to follow scheme II, (scheme II returns the least number of real view candidates).

4.3 Discussion

In this section, we discuss various aspects and limitations of our approach.

1. **Application to EDB:** We believe the multi-view solution may be applicable to other related areas. For instance, processing on encrypted databases (EDB) has a rich literature including searchable symmetric encryption (SSE) [53], [54], fully-homomorphic encryption (FHE) [55], oblivious RAMs (ORAM) [44], functional encryption [56], and property preserving encryption (PPE) [57], [58]. All these approaches achieve different trade-offs between protection (security), utility (query expressiveness), and computational efficiency [60]. Extending and applying the multi-view approach in those areas could lead to interesting future directions.

2. **Comparing the Two Schemes:** As we discussed in the two schemes, scheme I achieves a better indistinguishability but less protected partitions in each view. Figure 14 compares the relative effectiveness of the two schemes on a real trace under 40% adversary knowledge. In particular, Figure 14(a), (b) demonstrate the fact that despite the lower number of real view candidates in scheme II compared with scheme I (30 vs 160 out of 160), the end result of the leakage in scheme II is much more appealing (3% vs 35%). Therefore, our experimental section has mainly focused on scheme II.

3. **Choosing the Number of Views $N$:** The number of views $N$ is an important parameter of our approach that determines both the privacy and computational overhead. The data owner could choose this value based on the level of trust on the analysts and the amount of computational overhead that can be afforded. Specifically, as it is implied by Equation 10 and demonstrated by our experimental results in section 5, the number of real view candidates is approximately $e^{-\epsilon} \cdot N$. The data owner should first estimate the adversary’s background knowledge $\alpha$ (number of prefixes known to the adversary) and then calculate $\epsilon$ either using Equation 15 or (approximately) using Equation 16. As it is demonstrated in Figures 8(a) and 9(b), a bigger $\alpha$ results in weaker indistinguishability and demands a larger number of views to be generated. An alternative solution is to increase the number of prefix groups ($D$) by sacrificing some prefix relations among IP’s, e.g., grouping them based on first 3 octets.

4. **Utility:** The main advantage of the multi-view approach is it can preserve the data utility while protecting privacy. In particular, we have shown that the data owner can receive an analysis report based on the real view ($\Gamma_r$) which is
prefix-preserving over the entire trace. This is more accurate than the obfuscated (through bucketization and suppression) or perturbed (through adding noise and aggregation) approaches. Specifically, in case of a security breach, the data owner can easily compute $L_2$ (migration output) to find the mapped IP addresses corresponding to each original address. Then the data owner applies necessary security policies to the IP addresses that are reported violating some policies in $\Gamma$. A limitation of our work is it only preserve the prefix of IPs, and a potential future direction is to apply our approach to other property-preserving encryption methods such that other properties may be preserved similarly.

(5) **Communicational/Computational Cost:** One of our contributions in this paper is to minimize the communication overhead by only outsourcing one (seed) view and some supplementary parameters. This is especially critical for large scale network data like network traces from the major ISPs. On the other hand, one of the key challenges to the multi-view approach is that it requires $N$ times computation for both generating the views and analysis.

Our experiments in Figure 11 shows that generating 160 views for a trace of 1 million packets takes approximately 4 minutes and we describe analytic complexity results in Tables 3 and 4. We note that the practicality of $N$ times computation will mainly depend on the type of analysis, and certainly may become impractical for some analyses under large $N$. How to enable analysts to more efficiently conduct analysis tasks based on multiple views through techniques like caching is an interesting future direction. Another direction is to devise more accurate measures for the data owner to more precisely determine the number of views required to reach a certain level of privacy requirement.

## 5 EXPERIMENTS

This section evaluates our multi-view scheme through experiments with real data.

### 5.1 Setup

To validate our multi-view anonymization approach, we use a set of network traces collected by a real ISP. We focus on attributes *Timestamp, IP Address, and Packet Size* in our experiments, and the meta-data are summarized in the table in Figure 9(a). In order to measure the security of the proposed approach, we implement the frequency analysis attack [60], [4]. This attack can compromise individual addresses protected by existing prefix-preserving anonymization in multi-linear time [4]. We stress that in the setting of EDBs (encrypted database systems), an attack is successful if it recovers even partial information about a single cell of the DB [60]. Accordingly, we define the information leakage metric to evaluate the effectiveness of our solution against the adversary’s semantic attacks. Several measures have been proposed in literature [3, 31] to evaluate the impact of semantic attacks. Motivated by [3], we model the information leakage (number of matches) as the number of records/packets, their original IP addresses are known by the adversary either fully or partially. More formally, **Information leakage metric** [3]: We measure $F_i$ defined as the total number of addresses that has at least $i$ most significant bits known, where $i \in \{1, 2, \ldots, 32\}$.

To model adversarial knowledge, we define a set of prefixes to be known by the adversary ranging from 10% up to 100% of all the prefixes in the trace. This knowledge is stored in a two dimensional vector that includes $d$ different addresses and their key indexes. Next, using our multi-view scheme, we generate all the $N$ views. However, before we apply the frequency analysis attack, we simulate how an adversary may eliminate some fake views from further consideration as follows. For each view, we check if two addresses from the adversary’s knowledge set with different prefixes now share prefixes in that view. If we find such a match in the key indices, the corresponding view will be discarded from the set of the real view candidates and will not be considered in our experiments since the adversary would know it is a fake view.

We validate the effectiveness of our scheme by showing the number of real view candidates and the percentage of the packets in the trace that are compromised (i.e., the percentage of IP packets whose addresses have at least eight most significant bits known). Each experiment is repeated more than 1,000 times and the end results are the average results of the frequency analysis algorithm applied to each of the real view candidates.

Moreover, evaluating the *utility* preservation and studying the scalability of using ORAM in our scheme are respectively discussed in Appendix B.2 and B.3.

We conduct all experiments on a machine running Windows with an Intel(R) Core(TM) i7-6700 3.40 GHz CPU, 4 GB Memory, and 500 GB storage.

### 5.2 Results

#### 5.2.1 Information Leakage Analysis

First, the numerical results of the indistinguishability parameter $\epsilon$ under different adversary’s knowledges are depicted in Figure 9(b). Those results correspond to three different cases, i.e., when addresses are grouped based on (1) only the first octet (136 groups), (2) the first and the second octets (417 groups), and (3) the first three octets (506 groups). As we can see from the results, $\epsilon$ decreases (meaning more privacy) as the number of prefix groups increases, and it increases as the amount of adversarial knowledge increases.
We next validate those numerical results through experiments in Figure 10. Specifically, we first analyze the behavior of our second multi-view scheme (introduced in Section 4.2) before comparing the two schemes in Appendix B. Figure 10 presents different facets of information leakage when our approach is applied in various grouping cases. The results in Figure 10 are for adversaries who has knowledge of no more than most 50% of the prefix groups (Figure 13 in Appendix B.1 presents the more extreme cases for the same experiments, i.e., up to 100% knowledge). The analysis of these figures is detailed in the following.

**Effect of the number of prefix groups:** As we discuss earlier, three different IP grouping cases are studied. Figures 10 (a) and (d) shows respectively the results of packet leakage and number of real view candidates when $d = 136$. As the numerical results in Figure 8 anticipates, because the fraction $d/D = 0.154$ is relatively low, the indistinguishability of generated views diminishes specially for stronger adversary knowledges. Consequently, the adversary discards more views and the rate of leakage increases, compared with Figures 10 (b), (e) and Figures 10 (c), (f) for which the fraction $d/D$ are $0.47$ and $0.57$, respectively. In particular, for the worst case of 50% adversary knowledge and when the number of views is less than 50, we can verify that the number of real view candidates for case (1) remains 1 resulting in packet leakage comparable to that of CryptoPAn.

**Effect of the number of views:** As it is illustrated in the figure, increasing the number of views always improves both the number of real view candidates and the packet leakages. All the figures for real view candidates evaluation, show a near linear improvement where the slope of this improvement inversely depends on the adversary’s knowledge. For the packet leakages, we can note that the improvement converges to a small packet leakage rate under a large number of views. This is reasonable, as each packet leakage result is an average of leakages in all the real view candidates.
6 RELATED WORK

In the context of anonymization of network traces, as surveyed in [18], many solutions have been proposed [2, 4, 8, 12, 13]. Generally, these may be classified into different categories, such as enumeration [19], partitioning [21], and prefix-preserving [22, 25]. These methods include removing rows or attributes, suppression, and generalization of rows or attributes [34]. Some of the solutions [8, 31] are designed to address specific attacks and are generally based on the permutation of some fields in the network trace to blur the adversary’s knowledge. Later studies either prove theoretically [5] or validate empirically [14] that those works may be defeated by semantic attacks.

As our proposed anonymization solution fall into the category of prefix-preserving solutions, which aims to improve the utility, we review in more details some of the proposed solutions in this category. First effort to find a prefix preserving anonymization was done by Greg Minshall [48] who developed TCPdpriv which is a table-based approach that generates a function randomly. Fan et al. [3] then developed CryptoPan with a completely cryptographic approach. Several publications [4], [8, 31] have then raised the vulnerability of this scheme against semantic attacks which motivated query based [17] and bucketization based [2] solutions. In the following we review those works in more details.

Among the works that address such semantic attacks, Riboni et al. [2] propose a (k,j)-obfuscation methodology applied to network traces. In this method, a flow is considered obfuscated if it cannot be linked, with greater assurance, to its (source and destination) IPs. First, network flows are divided into either confidential IP attributes or other fields that can be used to attack. Then, groups of k flows having similar fingerprints are first created, then bucketed, based on their fingerprints into groups of size j < k. However, utility remains a challenge in this solution, as the network flows are heavily sanitized, i.e., each flows is blurred inside a bucket of k flows having similar fingerprints. An alternative to the aforementioned solutions, called mediated trace analysis [15, 16], consists in performing the data analysis on the data-owner side and outsourcing analysis reports to researchers requesting the analysis. In this case, data can only be analyzed where it is originally stored, which may not always be practical, and the outsourced report still needs to be sanitized prior to its outsourcing [17]. In contrast to those existing solutions, our approach improves both the privacy and utility at the cost of a higher computational overhead. Table 2 summarizes the most important network trace anonymization schemes, over past twenty years [18] and their main characteristics.

The last step of our solution requires data owner to privately retrieve an audit report of the real view, which can be based on existing private information retrieval (PIR) techniques. A PIR approach usually aims conceal the objective of all queries independent of all previous queries [20, 38]. Since the sequence of accesses is not hidden by PIR while each individual access is hidden, the amortized cost is equal to the worst-case cost [20]. Since the server computes over the entire database for each individual query, it often results in impracticality for large databases. On the other hand, ORAM [39] has verifiably low amortized communication complexity and does not require much computation on the server but rather periodically requires the client to download and reshuffle the data [20]. For our multi-view scheme, we choose ORAM as it is relatively more efficient and secure, and also the client (data owner in our case) has sufficient computational power and storage needed to locally store a small number of blocks (audit reports in our case) in a local stash.

Table 2: Summary of proposed network trace anonymization in literature

| Authors                  | Privacy against semantic attacks | Utility                    |
|--------------------------|---------------------------------|----------------------------|
| Sagedel et al. [47]      | Violated                        | Prefix preserving          |
| McSherry et al. [17]     | Preserved                       | Noisy aggregated results   |
| Pang et al. [8]          | Violate                         | Partial prefix preserving   |
| Riboni et al. [2]        | Preserved                       | Heavily sanitized          |
| Ribeiro et al. [31]      | Violated                        | Partial prefix preserving   |
| Megal et al. [19]        | Violated                        | Aggregated results         |

7 CONCLUSION

In this paper, we have proposed a multi-view anonymization approach mitigating the semantic attacks on CryptoPan while preserving the utility of the trace. This novel approach shifted the trade-off from between privacy and utility to between privacy and computational cost; the later has seen significant decrease with the advance of technology, making our approach a more preferable solution for applications that demand both privacy and utility. Our experimental results showed that our proposed approach significantly reduced the information leakage compared to CryptoPan. For example, for the extreme case of adversary pre-knowledge of 100%, the information leakage of CryptoPan was 100% while under approach it was still less than 10%. Besides addressing various limitations discussed in Appendix 4.3, our future works will adapt the idea to improve existing privacy-preserving solutions in other areas, e.g., we will extend our work to the multi-party problem where...
several data owners are willing to share their traces to mitigate coordinated network reconnaissance by means of distributed (or inter-domain) audit [51].

8 ACKNOWLEDGMENTS
The authors thank the anonymous reviewers, for their valuable comments. We appreciate Momen Oqaily’s support in the implementation. This work is partially supported by the Natural Sciences and Engineering Research Council of Canada and Ericsson Canada under CRD Grant N01823. The research of Yuan Hong is partially supported by the National Science Foundation under Grant No. CNS-1745894 and the WISER ISFG grant.

REFERENCES
[1] Ding, Wen, William Yurcik, and Xiaoxin Yin. "Outsourcing internet security: Economic analysis of incentives for managed security service providers." In International Workshop on Internet and Network Economics, pp. 947-958. Springer Berlin Heidelberg, 2005.
[2] Riboni, Daniele, Antonio Villani, Domenico Vitali, Claudio Bettini, and Luigi V. Mancini. “Obfuscation of sensitive data in network flows.” In INFOCOM, 2012 Proceedings IEEE, pp. 2372-2380. IEEE, 2012.
[3] J. Fan, J. Xu, M. Ammar, and S. Moon. Prefix-preserving IP Address Anonymization: Measurement-based Security Evaluation and a New Cryptography-based Scheme. Computer Networks. 46(3):263-272, October 2004.
[4] Brekne, T. Árnes, A. & Ødebo, A. (2005, May). Anonymization of ip traffic monitoring data. Attacks on two prefix-preserving anonymization schemes and some proposed remedies. In International Workshop on Privacy Enhancing Technologies (pp. 179-196). Springer Berlin Heidelberg.
[5] Brekne, T. & André Árnes. “Circuitwring IP-address pseudoanonymization.” In Communications and Computer Networks, pp. 43-48. 2005.
[6] T.-F. Yen, X. Huang, F. Monrose, and M. K. Reiter. “Playing Devil’s Advocate: Inferring Sensitive Information from Anonymized Network Traces.” In NDSS, vol. 7, pp. 35-47. 2007.
[7] Yurcik, William, and Yifan Li. “Internet security visualization case study: Instrumenting a network for NetFlow security visualization tools.” In 21st Annual Computer Security Applications Conference (ACSAC), 2005.
[8] Coull, Scott E., Michael P. Collins, and Michael K. Reiter. “On Web Browsing Privacy in Anonymized NetFlows.” In USENIX Security, 2007.
[9] Zhang, Q., & Li, X. (2006, January). An IP address anonymization scheme with multiple access levels. In International Conference on International Networking (pp. 795-802). Springer Berlin Heidelberg.
[10] B. Ribeiro, W. Chen, G. Miki, and D. Towsley. Analyzing privacy in enterprise packet trace anonymization, In Proc. NDSS, San Diego, CA, Feb. 2008, pp. 87-100.
[11] Coull, Scott E., Charles V. Wright, Fabian Monrose, Michael P. Collins, and Michael K. Reiter. “Obfuscation of Sensitive Data in Network Flows.” In Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC), 2005.
[12] Coull, S. E., Monrose, F., Reiter, M. K., & Bailey, M. (2009, March). The challenges of effectively anonymizing network data. In Conference For Homeland Security, 2009. CATCH’09. Cybersecurity Applications & Technology (pp. 230-236). IEEE.
[13] Del Piccolo, Valerioti, et al. “A Survey of network isolation solutions for multi-tenant data centers.” IEEE Communications Surveys & Tutorials 18.4 (2016): 2787-2821.
[14] C. Dwork. “Differential privacy: In proceedings of the 33rd International Colloquium on Automata, Languages and Programming [ICALP], ser. Lecture Notes in Computer Science, vol. 4052. Springer-Verlag, 2006.
[15] Dwork. Cynthia. “Differential privacy: A survey of results.” In International Conference on Theory and Applications of Models of Computation, pp. 1-19. Springer, Berlin, Heidelberg, 2006.
[16] Kushilevitz, Eyal, and Rafail Ostrovsky. “Replication is not needed: Single database, computationally-privacy information retrieval.” In Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on, pp. 364-373. IEEE, 1997.
[17] Goldreich, Oded. “Security in outsourcing of association rule mining.” In Proceedings of the 3rd International Conference on Data Engineering, pp. 947-958. Springer Berlin Heidelberg.
[18] Mancini. “Obfuscation of sensitive data in network flows.” In INFOCOM, 2012 Proceedings IEEE, pp. 2372-2380. IEEE, 2012.
[19] Biler, Piotr, and Alfred Witkowski. “Privacy in enterprise packet trace anonymization, In Proc. NDSS, San Diego, CA, Feb. 2008, pp. 87-100.
[20] Coull, Scott E., Charles V. Wright, Fabian Monrose, Michael P. Collins, and Michael K. Reiter. “Playing Devil’s Advocate: Inferring Sensitive Information from Anonymized Network Traces.” In NDSS, vol. 7, pp. 35-47. 2007.
[21] Yurcik, William, and Yifan Li. “Internet security visualization case study: Instrumenting a network for NetFlow security visualization tools.” In 21st Annual Computer Security Applications Conference (ACSAC), 2005.
[22] Coull, S. E., Monrose, F., Reiter, M. K., & Bailey, M. (2009, March). The challenges of effectively anonymizing network data. In Conference For Homeland Security, 2009. CATCH’09. Cybersecurity Applications & Technology (pp. 230-236). IEEE.
[23] Del Piccolo, Valerioti, et al. “A Survey of network isolation solutions for multi-tenant data centers.” IEEE Communications Surveys & Tutorials 18.4 (2016): 2787-2821.
[24] C. Dwork. “Differential privacy: In proceedings of the 33rd International Colloquium on Automata, Languages and Programming [ICALP], ser. Lecture Notes in Computer Science, vol. 4052. Springer-Verlag, 2006.
[25] Dwork. Cynthia. “Differential privacy: A survey of results.” In International Conference on Theory and Applications of Models of Computation, pp. 1-19. Springer, Berlin, Heidelberg, 2006.
[26] Kushilevitz, Eyal, and Rafail Ostrovsky. “Replication is not needed: Single database, computationally-privacy information retrieval.” In Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on, pp. 364-373. IEEE, 1997.
[27] Goldreich, Oded. “Security in outsourcing of association rule mining.” In Proceedings of the 3rd International Conference on Data Engineering, pp. 947-958. Springer Berlin Heidelberg.
[28] Mancini. “Obfuscation of sensitive data in network flows.” In INFOCOM, 2012 Proceedings IEEE, pp. 2372-2380. IEEE, 2012.
Proof. Suppose there exists algorithm A which returns the smallest set of key vectors $V_i$ to reverse the seed trace and obtain the minimum number of real view candidates, given our setup. Also denote by $V$ the set of those key vectors if the adversary follows scheme II. We now show that if $(2d - 2)^D > N$ holds then we have $V = V_i$. First, note that the key indices of different distinct addresses in $L^*_D$ is PRNG$(d, D, 0)$. Therefore, the adversary has to guess $V = C^*_0 - PRNG(d, D, 0)$. However, note that elements of $V$ are in $[-d + 1, d - 1]$ and there will be $(2d - 2)^D$ different combinations for $V$. Thus to minimize this number, the adversary has to use the outsourced parameters which means we have $A(L^*_1, K, V_1, \cdots, V_N, S_d) = V_s$. However, we showed earlier that all these inputs are trivial leakage. Therefore, if $(2d - 2)^D > N$ holds, we have $V = V_i$. \qed

\section{EXPERIMENTS}

In this section, using experiments, we measure the security of the proposed approach against very strong adversaries. In addition, we evaluate the utility of the approach using two real network analyses. Finally, we justify the choice of ORAM in our setup using a comprehensive study on the scalability of ORAM in the literature.

\subsection{Privacy Evaluation against Very Strong Adversaries}

Figure 13 shows the leakage and the real view candidates results for stronger adversaries ($\alpha \in [60, 100]$). Note that in this figures, we only show results for case (2) and (3) as results in case (1) does not show a significant improvement compared with CryptoPAn results because the multi-view approach with fraction of $d/D = 0.154$ cannot defeat the adversary’s knowledge ($\epsilon > 16$).

\subsection{Utility Evaluation Using Real-life Network Analytics}

Figure 12 shows the results of two different network analytics over the original trace (1M records), the real view and one of the fake views generated in our multi-view solution. In the first experiment, we present IP distribution [61] in the trace; reporting the number of distinct addresses within each subnet (IP group). We compare the distribution of distinct IP addresses inside the aforementioned three traces for both temporal distribution; if subnets are indexed based on their time stamps; and cardinality-based distribution result; if subnets are indexed based on their cardinalities. We found that our results (both distributions) generated from the original trace and the real view are identical (see Figure 12(a)). This is reasonable because the real view is a prefix preserving mapping of IPs that keeps the fp-QI attributes intact (preserving both distributions). Moreover, the cardinality based distribution result generated from the fake view is identical to those in the original trace and the real view (see Figure 12(c)). Note that the later is resulted from the indistinguishability of our multi-view solution.

In the second experiment, we present a packet-level analytic [17]. In particular, Figure 12(d,e) shows the empirical cumulative distribution function results for the three traces. Our results clearly show
that the original trace and our scheme results are identical as multi-
view will not have any impact on fingerprinting quasi identifier
attributes.

B.3 Multi-view and the Scalability of ORAM
In practice, we expect analysis reports would have significantly
smaller sizes in comparison to the views, and considering the one
round communication with ORAM (O(logN)-complexity), we be-
lieve the solution would have acceptable scalability. Experiments
using our dataset and existing ORAM implementation (an imple-
mentation [62] of non-recursive Path-ORAM [64] has been made
public) would further confirm this. We generated various set of
analyses reports using snort [63], and we found that for our dataset
the size of audit reports are in the range of KB which is perfect to
be used in fast ORAM protocols, e.g., Path-ORAM. Specifically, for
Path-ORAM, Figure 5 (b) in [62] shows a less than 1MB commu-
nication overhead for the worst-case cost of up to 2^d number of
blocks of size 4KB.

C ALGORITHMS

Input:
\[ \mathcal{L}^*: \text{Original network trace} \]
\[ K_0, K: \text{Cryptographic keys} \]
\[ d: \text{Number of prefix groups} \]
\[ \text{partition}(\mathcal{L}, d): \text{IP partitioning} \]
\[ r: \text{Iteration number of the real view.} \]
\[ V: \text{Random vectors, of size } d \]

Output:
\[ \mathcal{L}^*: \text{Anonymized trace to be outsourced} \]

**Function: anonymize** \((\mathcal{L}, d, K_0, K, r, V)\)

begin
1. \( \mathcal{L} := PP(\mathcal{L}, K_0) \)
2. \( V_0 := \sim r \cdot V \)
3. \( P := \text{partition}(\mathcal{L}, d) \)
4. \( \mathcal{L}^* := \phi \)
5. foreach \( P_j \in P \) do:
6. \( L_i := \text{GetFlows}(\mathcal{L}, P_j) \)
7. \( L_i^* := PP(L_i, V_0(i), K) \)
8. \( \mathcal{L}^* := \mathcal{L}^* \cup L_i^* \)
9. endforeach
10. return \( \mathcal{L}^*, K, V, N \)
end

Algorithm 1: Data owner: Trace anonymization (scheme I)

Following Algorithms are summarized versions of the data owner’s
and the analyst’s roles in our multi-view scheme presented in sec-
tion 4.

**Algorithm 1**:
The data owner’s actions (scheme I).

**Algorithm 2**:
The analyst’s actions (scheme I).

**Algorithm 3**:
The data owner’s actions (scheme II).

**Algorithm 4**:
The analyst’s actions (scheme II).

D COMPLEXITY ANALYSIS
Here, we discuss the overhead analysis, from both the data owner’s
and the data analyst’s side. In particular, table 3 summarizes the
overhead for all the action items in the data owner side. Here, \( C(n) \)
is the computation overhead of CryptoPan and \( D \) is the number of
the distinct IP addresses. Finally, table 4 summarizes the overhead
for all the action items in the data analyst side where \( N.CV(n) \) is
the cost of \( N \) times verifying the compliances (auditing).

### Table 3: Overhead on the data owner side

| Blocks in Multi-view | Computation Overhead | Communication Overhead |
|----------------------|----------------------|------------------------|
| Initial anonymization | \( C(n) \) | – |
| Migration function | \( O(n \log^* n) + \frac{1}{\log^* n} O(n) \) | – |
| Prefix grouping | – | – |
| Index generator | \( N.O(D) \) | \( N.O(D) \) |
| Seed trace | \( \frac{1}{\log^* n} O(n) \) | \( O(n) \) |
| Report retrieval (ORAM) | \( O(n) \) | \( O(n \log^* n) ) \) |

### Table 4: Overhead on the data analyst side

| Blocks in Multi-view | Computation Overhead | Communication Overhead |
|----------------------|----------------------|------------------------|
| N views generation | \( \frac{1}{\log^* n} O(n) \) | – |
| Compliance verification (Analysis) | \( N.CV(n) \) | – |
Figure 12: Distribution of distinct IP addresses in different subnets (IP groups) (out of 1M) (a) for the original trace, the real view and one of the fake views based on the order they appear in the trace (temporal distribution), (b) for the original trace and the real view and (c) for the fake view, based on the cardinalities of the subnets in an ascending order (cardinality-based distribution). Empirical CDF for the packet lengths in (e),(f) the original trace and the real view, and the fake view, respectively.

Figure 13: Percentage of the compromised packets (out of 1M) and number of real view candidates when number of views and the adversary knowledge vary and for case (1) Figures (b),(e) (2) Figures (c),(f) where legends marked by CP denote the CryptoPAn result whereas those marked by MV denote the multi-view results

Figure 14: Comparison between scheme I and scheme II with 137 partitions (prefix groups based on first octet sharing). Figure (a): Percentage of the compromised packets (out of 1M) and Figure (b): Number of real view candidates for 40% adversary knowledge
Input:
\( L \): Original network trace
\( K_0, K \): Cryptographic keys
\( D \): Number of IPs
\( d \): Number of prefix groups
\( \text{Migration}(L, d) \): Migration function
\( \text{partition}(L, D) \): IP partitioning
\( r \): Iteration number of the real view.
\( V_0, V_1, \cdots, V_N \): Vectors of size \( D \) defined by data owner

Output:
\( L^* \): Anonymized trace to be outsourced

Function: \text{anonymize} \((L, d, D, K_0, K, r, V)\)

begin
1-1 \( L := \text{PP}(L, K_0) \)
1-2 \( L := \text{Migration}(L, d) \)
2 \( V_0 := \sim r \cdot V \)
3 \( P := \text{partition}(L, D) \)
4 \( L^* := \emptyset \)
5 \textbf{foreach} \( P_i \in P \) \textbf{do:}
6 \( L_i := \text{GetFlows}(L, P_i) \)
7 \( L_i^* := \text{PP}(L_i, V_0(i), K) \)
8 \( L^* := L^* \cup L_i^* \)
9 \textbf{end}
10 \textbf{return} \( L^*, K, V_1, V_2, \cdots, V_N, N \)
\textbf{end}

Algorithm 3: Data owner: Trace anonymization (scheme II)

Input:
\( L^* \): Seed trace
\( N \): Number of iterations requested by data owner
\( d \): Number of prefix groups
\( D \): Number of partitions
\( \text{partition}(L^*, D) \): IP partitioning
\( K \): Outsourced key
\( V_0, V_1, \cdots, V_N \): Vectors of size \( D \) defined by data owner
\( CV(L^*) \): Compliance verification

Output:
\( \Gamma_i \): Analysis report of \( i^{th} \) view \( L_i^* \), \( i \in \{1, 2, \ldots, N\} \)

Function: \text{analysis} \((L^*, D, \text{partition}(L^*, D), K, N, V_1, V_2, \cdots, V_N)\)

begin
1 \( P := \text{partition}(L^*, D) \)
2 \textbf{for} \( i = 1 : N \) \textbf{do:}
3 \( L_i^* := \emptyset \)
4 \textbf{foreach} \( P_j \in P \) \textbf{do:}
5 \( L_{i,j} := \text{GetFlows}(L_{i-1}, P_j) \)
6 \( L_{i,j}^* := \text{PP}(L_{i,j}, V_i(j), K) \)
7 \( L_i^* := L_i^* \cup L_{i,j}^* \)
8 \textbf{end}
9 \( \Gamma_i := CV(L_i^*) \)
10 \textbf{return} \( \Gamma_i \)
11 \textbf{end}
\textbf{end}

Algorithm 4: Analyst: Network trace analysis (scheme II)