Preserving Both Privacy and Utility in Network Trace Anonymization

Meisam Mohammady
Concordia University
m_ohamma@encs.concordia.ca

Lingyu Wang
Concordia University
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
Concordia University
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 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, CryptoPAn, Semantic Attacks

ACM Reference Format:
Meisam Mohammady, Lingyu Wang, Yuan Hong, Habib Louafi, Makan Pourzandi, and Mourad Debbabi. 2018. Preserving Both Privacy and Utility in Network Trace Anonymization . In *2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18), October 15–19, 2018, Toronto, ON, Canada.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3243734.3243809

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].

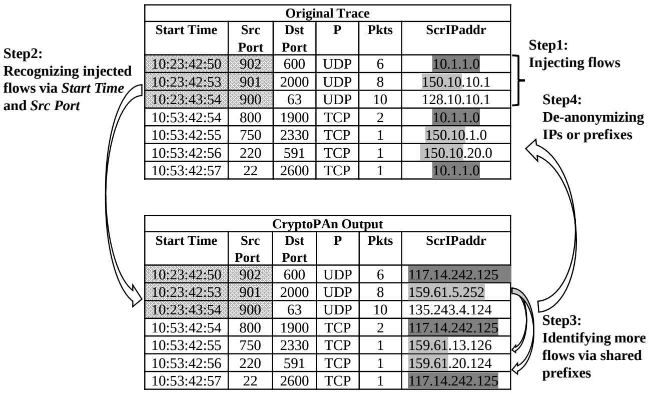
In cases where data owners are convinced to share their network traces, the traces are typically subjected to some anonymization techniques. The anonymization of network traces has attracted significant attention (a more detailed review of related works will be given in section 6). For instance, *CryptoPAn* 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*

CCS '18, October 15–19, 2018, Toronto, ON, Canada

ACM ISBN 978-1-4503-5693-0/18/10…$15.00
https://doi.org/10.1145/3243734.3243809

example, without loss of generality, we only focus on source IPs. Inside each table, similar prefixes are highlighted through similar shading.



Original Trace					
Start Time	Src Port	Dst Port	P	Pkts	ScrIPaddr
10:23:42:50	902	600	UDP	6	10.1.1.0
10:23:42:53	901	2000	UDP	8	150.10.10.1
10:23:43:54	900	63	UDP	10	128.10.10.1
10:53:42:54	800	1900	TCP	2	10.1.1.0
10:53:42:55	750	2330	TCP	1	150.10.1.0
10:53:42:56	220	591	TCP	1	150.10.20.0
10:53:42:57	22	2600	TCP	1	10.1.1.0

CryptoPAn Output					
Start Time	Src Port	Dst Port	P	Pkts	ScrIPaddr
10:23:42:50	902	600	UDP	6	117.14.242.125
10:23:42:53	901	2000	UDP	8	159.61.5.252
10:23:43:54	900	63	UDP	10	135.243.4.124
10:53:42:54	800	1900	TCP	2	117.14.242.125
10:53:42:55	750	2330	TCP	1	159.61.13.126
10:53:42:56	220	591	TCP	1	159.61.20.124
10:53:42:57	22	2600	TCP	1	117.14.242.125

Step2: Recognizing injected flows via *Start Time* and *Src Port*
Step1: Injecting flows
Step4: De-anonymizing IPs or prefixes
Step3: Identifying more flows via shared prefixes

Figure 1: An example of injection attack

(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 flows must have prefix* 150.10*, and the IPs of the fourth and last flows must be* 10.1.1.0*.*

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 (see section 6).

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 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, briefly review CryptoPAn, provide a high level overview of our multi-view approach, and finally define our privacy property.

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 $A^{\text{IP}} = \{\text{IP}_{src}, \text{IP}_{dst}\}$, 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 Γ. To ensure privacy, instead of sending $\mathcal{L}$, an anonymization function $\mathcal{T}$ is applied to obtain an anonymized version $\mathcal{L}^*$. 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 $\mathcal{L}^*$, 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 $\mathcal{L}^*$. 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 $\mathcal{L}^*$. 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 α ($\leq d$) groups (e.g., the demilitarized zone (DMZ) or other subnets to which the adversary has access). Accordingly, we say that the adversary has $\mathcal{S}_\alpha$ 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 $a = a_1a_2....a_{32}$ and $b = b_1b_2....b_{32}$, and a one-to-one function $F(.) : \{0,1\}^{32} \to \{0,1\}^{32}$, we say that*

- *a and b share a k-bit prefix ($0 \leq k \leq 32$), if and only if $a_1a_2....a_k = b_1b_2....b_k$, 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 $a = a_1a_2 \cdots a_{32}$ and $F(a) = a'_1a'_2 \ldots a'_{32}$, the prefix-preserving anonymization function F must necessarily satisfy the canonical form [3], as follows.

$$a'_i = a_i \oplus f_{i-1}(a_1a_2 \cdots a_{i-1}),\ i = 1, 2, \ldots, 32 \tag{1}$$

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 $L\Big(R\big(P(a_1a_2 \ldots a_{i-1}), K\big)\Big)$ 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).

Step 3: 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 ϵ-indisinguishablity property as follows.

Definition 2.2. ***ϵ-Indisinguishable Views:*** *A multi-view solution satisfies* ϵ-Indistinguishability *against an $\mathcal{S}_\alpha$ adversary if and*

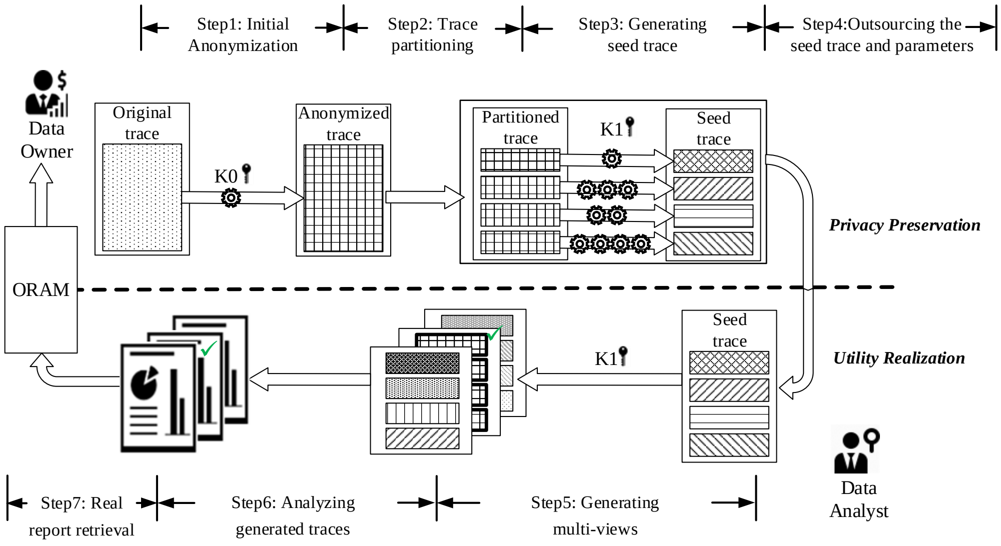

Step1: Initial Anonymization
Step2: Trace partitioning
Step3: Generating seed trace
Step4:Outsourcing the seed trace and parameters
Data Owner
Original trace
K0
Anonymized trace
Partitioned trace
K1
Seed trace
Privacy Preservation
ORAM
Utility Realization
Seed trace
K1
Data Analyst
Step7: Real report retrieval
Step6: Analyzing generated traces
Step5: Generating multi-views

Figure 2: An overview of the multi-view approach

only if (probabilities below are from the adversary's point of view)

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

In Defintion 2.2, a smaller ϵ 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 to the adversary (from now on, we call these views the *real view candidates*). In practice, the value of ϵ 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 $PP^j(a, K)$ ($j > 1$) 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, $PP^i(a, K)$ and $PP^i(b, K)$ will always result in two IP addresses that also share a k-bit prefix (i.e., PP^i is prefix-preserving). More generally, the same also holds for applying PP under a sequence of indices and keys (for both IPs), e.g., $PP^i(PP^j(a, K_0), K_1)$ and $PP^i(PP^j(b, K_0), K_1)$ will also share k-bit prefix. Finally, for a set of IP addresses $\mathcal{S}$, iterative PP using a single key K satisfies the following associative property:

$$\forall \mathcal{S}, K, \ \ \text{and} \ \ i, j \in \mathbb{Z}\ (integers): \ \ PP^i\big(PP^j(\mathcal{S}, K), K\big)$$
$$= PP^j\big(PP^i(\mathcal{S}, K), K\big) = PP^{(i+j)}\big(\mathcal{S}, K\big) \tag{3}$$

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

$$\begin{aligned} &RPP(b, K) = c = c_1 c_2 \cdots c_{32} \\ &where\ c_i = b_i \oplus f_{i-1}(c_1 \cdots c_{i-1}) \end{aligned} \tag{4}$$

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 number of iterations (Step 3). In this subsection, we discuss this concept.

Given $\mathcal{S}$ as a set of n IP addresses, we may divide $\mathcal{S}$ into partitions in various ways, e.g., forming equal-sized partitions after sorting $\mathcal{S}$

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 $\mathcal{S}$ divided as a set of m partitions $\{P_1, P_2, \cdots, P_m\}$, we define a *key vector* $V = \begin{bmatrix} v_1 & v_2 & \cdots & v_m \end{bmatrix}$ where each v_i is a positive integer indicating the number of times PP should be applied to P_i, namely, the *key index* of P_i. Given a cryptographic key K, we can then define the *partition-based* prefix preserving anonymization of $\mathcal{S}$ as $PP(\mathcal{S}, V, K) = \big[PP^{v_1}(P_1, K),\ PP^{v_2}(P_2, K), \ldots, PP^{v_m}(P_m, K)\big]$.

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).

$$PP[PP(\mathcal{S}, V_1, K), V_2, K] = PP(\mathcal{S}, (V_1 + V_2), K) \tag{5}$$

3.3 IP Migration: 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.

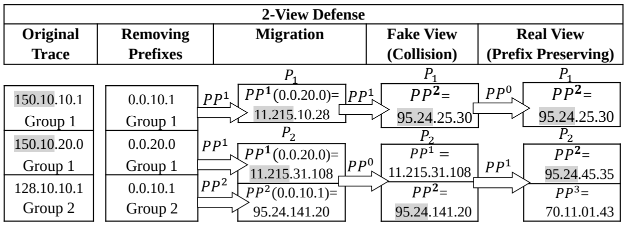
CryptoPAn (Collision Resistant)

3-View Defense			
Original Trace	**Fake View 1**	**Fake View 2**	**Real View**
P_1	$PP^4(P_1)$	$PP^2(P_1)$	$PP\ (P_1)$
150.10.10.1	144.5.116.249	50.19.13.26	159.61.5.252
128.10.10.1	39.250.139.225	83.180.10.3	135.243.4.124
P_2	$PP^3(P_2)$	$PP^5(P_2)$	$PP(P_2)$
150.10.20.0	17.8.78.28	159.61.20.124	159.61.20.124

Figure 3: An example showing only the real view contains shared prefixes (can be identified by adversaries)

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.



2-View Defense				
Original Trace	**Removing Prefixes**	**Migration**	**Fake View (Collision)**	**Real View (Prefix Preserving)**
		P_1	P_1	P_1
150.10.10.1 Group 1	0.0.10.1 Group 1	PP^1(0.0.20.0)= 11.215.10.28	PP^2= 95.24.25.30	PP^2= 95.24.25.30
		P_2	P_2	P_2
150.10.20.0 Group 1	0.0.20.0 Group 1	PP^1(0.0.20.0)= 11.215.31.108	PP^1 = 11.215.31.108	PP^2= 95.24.45.35
128.10.10.1 Group 2	0.0.10.1 Group 2	PP^2(0.0.10.1)= 95.24.141.20	PP^2= 95.24.141.20	PP^3= 70.11.01.43

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 xoring 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 $\mathcal{S}$ be a set of IP addresses consisting of d groups of IPs $S_1, S_2, \cdots, S_d$ with distinct prefixes $s_1, s_2, \cdots, s_d$ respectively, and K be a random CryptoPAn key. Migration function $M : \mathcal{S} \times C$(set of positive integers) $\longrightarrow \mathcal{S}^*$ is defined as*

$$\begin{aligned} \mathcal{S}^* &= M(\mathcal{S}) = \{S_i^* | \forall i \in \{1, 2, \cdots, d\}\} \\ \text{where } S_i^* &= \{PP^{c_i}(s_i \oplus a_j, K), \forall a_j \in S_i\} \end{aligned} \tag{6}$$

where $C = PRNG(d, d) = \{c_1, c_2, \cdots, c_d\}$ is the set of d non-repeating random key indices generated between $[1, d]$ using a cryptographically secure pseudo 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. In our first scheme, 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 $\mathcal{L}_0^*$ together with some parameters sent to the analyst for generating different views. These actions are summarized in Algorithm 1, and detailed as below.

(1) Applying CryptoPAn using K_0: First, the data owner generates two independent keys, namely K_0 (key used 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 $\mathcal{L}_0$=$PP(\mathcal{L}, K_0)$. 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 initially anonymized trace.

(2) Trace partitioning based on IP-value: The initially anonymized trace is partitioned based on IP values. Specifically, let $\mathcal{S}$ be the set of IP addresses in $\mathcal{L}_0$ consisting of d groups of IPs $S_1, S_2, \cdots, S_d$ with distinct prefixes $s_1, s_2, \cdots, s_d$, respectively; we divide $\mathcal{L}_0$ 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.

(3) 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.

- *Generating a random key vector:* The data owner generates a random vector V of size d using a cryptographically secure pseudo random number generator $PRNG(d\ , d)$ (which generates a set of d non-repeating random numbers between $[1, d]$). This vector V and the key K will later be used by the analyst to generate different views from the seed trace. For example, in Figure 5, for the two partitions, $V = \begin{bmatrix}1 & 2\end{bmatrix}$ is generated. Finally, the data owner chooses the total number of views N to be generated later by the analyst, based on his/her requirement about privacy and computational overhead, since a larger N will mean more computation by both the data owner and analyst but also more privacy (more real view candidates will be generated which we will further study this through experiments later).
- *Generating a seed trace key vector and a seed trace:* The data owner picks a random number $r \in [1, N]$ and then computes $V_0 = -r \cdot V$ as the key vector of seed trace. Next, the data owner generates the seed trace as $\mathcal{L}_0^* = PP(\mathcal{L}_0, V_0, K)$. This ensures, after the analysts applies exactly r iterations of V on the seed trace, he/she would get $\mathcal{L}_0$ back (while not being aware of this fact since he/she does not know r). For example, in Figure 5, $r = 3$ and $V_0 = \begin{bmatrix}-3 & -6\end{bmatrix}$. We can easily verify that, if the analyst applies the indices in V on the seed trace three times, the outcome will be exactly $\mathcal{L}_0$ (the real view). This can be more formally stated as follows (the r^{th} view $\mathcal{L}_r^*$ is actually the real view).

$$\begin{aligned}\mathcal{L}_r^* &= PP(\mathcal{L}_0^*, r \cdot V, K), \text{ using (5)}\\ &= PP(\mathcal{L}_0, V_0 + r \cdot V, K), \text{ using (5)}\\ &= PP(\mathcal{L}_0, -r \cdot V + r \cdot V, K) = \mathcal{L}_0\end{aligned}$$

(4) Outsourcing: Finally, the data owner outsources $\mathcal{L}_0^*$, V, N and K to the analyst.

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 E and formalized below.

$$\begin{aligned}&\mathcal{L}_0^*, \text{ is the seed view}\\ &\mathcal{L}_i^* = PP(\mathcal{L}_{i-1}^*, V, K),\ i \in \{1, \ldots, N\}\end{aligned} \qquad (7)$$

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 $\Gamma_1, \Gamma_2, \cdots, \Gamma_N$.

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 Γ_r. To minimize communication overhead, instead of requesting all the analysis reports Γ_i of the generated views, the data owner can fetch only the one that is related to the real view Γ_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 ϵ-indistinguishable if and only if

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

$$e^{-\epsilon} \leq \frac{Pr(\text{view } i \text{ may be the real view})}{Pr(\text{view } r \text{ may be the real view})} \leq e^{\epsilon}$$

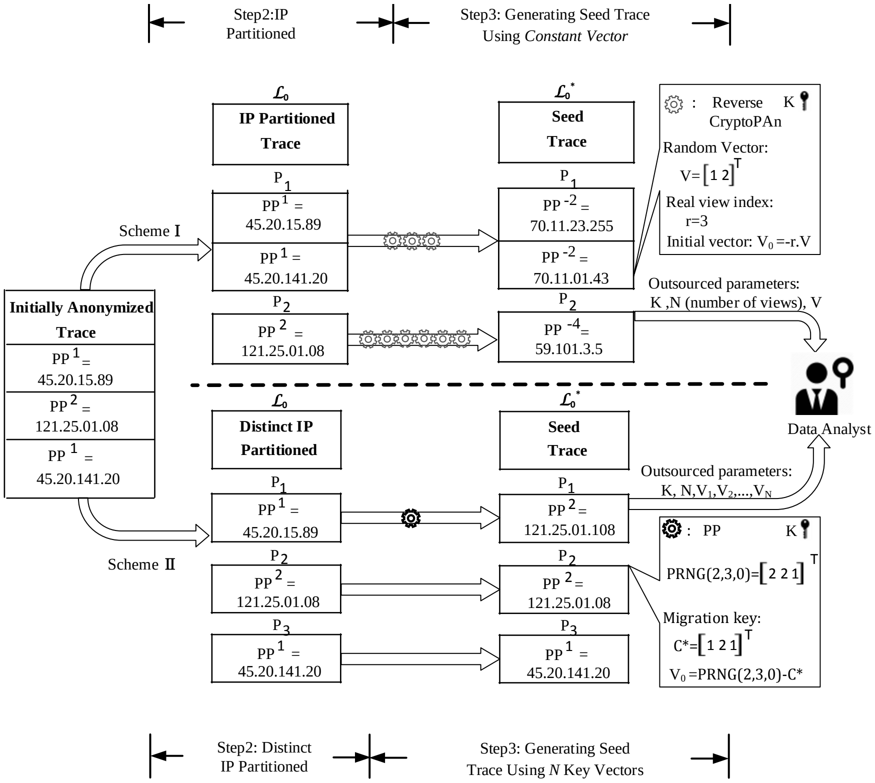


Figure 5: An example of a trace which undergoes multi-view schemes I, II

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 $\mathcal{S}_\alpha$ knowledge. Moreover, we note that generated views differ only in their IP values (fp-QI attributes are similar for all the views). Hence, the adversary's decision can only be based on the published set of IPs in each view through comparing shared prefixes among those IP addresses which he/she already know ($\mathcal{S}_\alpha$). Accordingly, in the following, we define a function to represent all the prefix relations for a set of IPs.

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

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

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

$$\mathcal{R}(\mathcal{S}) = \{Q(a_i,a_j)|\ \forall a_i, a_j \in \mathcal{S}, i,j \in \{1,2,\cdots,n\}\} \qquad (8)$$

Note that PIS remains unchanged when CryptoPAn is applied on $\mathcal{S}$, i.e., $\mathcal{R}(PP(\mathcal{S},K)) = \mathcal{R}(\mathcal{S})$. 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 indicator sets out of them. Accordingly, we denote by $\mathcal{R}_{\alpha,i}$ the PIS constructed by the adversary in view i.

Definition 4.2. *Let* $\mathcal{R}_\alpha$ *be the PIS for the adversary's knowledge, and* $\mathcal{R}_{\alpha,i}$*,* $i \in \{1,\cdots,N\}$ *be the PIS constructed by the adversary in view i. A multi-view solution then generates* ϵ-indistinguishable *views against an* $\mathcal{S}_\alpha$ *adversary if and only if*

$$\forall i \in \{1,2,\cdots,N\}, e^{-\epsilon} \le \frac{Pr(\mathcal{R}_{\alpha,i} = \mathcal{R}_\alpha)}{Pr(\mathcal{R}_{\alpha,r} = \mathcal{R}_\alpha)} \le e^{\epsilon} \qquad (9)$$

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

$$\forall i \in \{1,2,\cdots,N\}, Pr(\mathcal{R}_{\alpha,i} = \mathcal{R}_\alpha) \ge e^{-\epsilon} \qquad (10)$$

Proof. $Pr(\mathcal{R}_{\alpha,r} = \mathcal{R}_\alpha) = 1$ as view r is the prefix preserving output. Moreover, $\forall \epsilon \ge 0$ we have $e^{\epsilon} \ge 1$. □

From the above, we only need to show $\mathcal{R}_{\alpha,i} = \mathcal{R}_\alpha$ (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 ($1 \le i \le d$), any two IP addresses a and b inside P_i, and for any $m,n \le N$, we have $\mathcal{R}_m(a,b) = \mathcal{R}_n(a,b)$ 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 $m,n \le N$, we have $\mathcal{R}_m(a,b) = \mathcal{R}_n(a,b) = 0$ since they do not share any prefixes. □

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 required 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 next present 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, 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. 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.

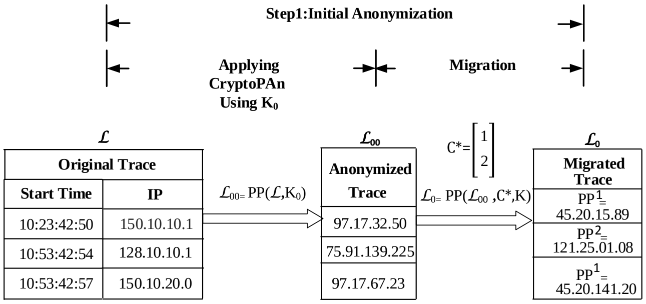


Figure 6: The updated initial anonymization (Step 1 in Figure 2) for enforcing migration

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 150.10.10.1,128.10.10.1, 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 first scheme, although the key V_0 will be generated completely differently, as detailed below.

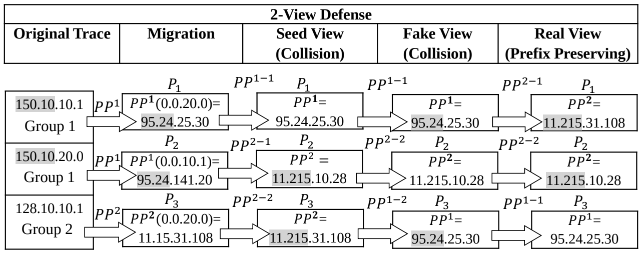


Figure 7: An 3 views generation example under scheme II

Let $\mathcal{S}^* = \{S_1^*, S_2^*, \cdots, S_d^*\}$, be the set of IP addresses after the migration step. Suppose $\mathcal{S}^*$ 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.

$$V_i = PRNG(d, D, i) - PRNG(d, D, i - 1), \quad (11)$$
$$\forall i \neq r \in [1, 2 \cdots, N]$$

and

$$V_0 = PRNG(d, D, 0) - C^* \quad (12)$$
$$V_r = C^* - PRNG(d, D, r - 1)$$

Example 4.1. *In Figure 7, the migration and random vectors are* $C^* = [1\ 1\ 2]$, $PRNG(2, 3, 0) = [1\ 2\ 2]$, $PRNG(2, 3, 1) = [1\ 2\ 1]$, *and* $PRNG(2, 3, 2) = [2\ 2\ 1]$, *respectively. The corresponding key vectors will be* $V_0 = [0\ 1\ 0]$, $V_1 = [0\ 0\ -1]$ *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 $\mathcal{S}_i^* = \{S_1^i, S_2^i, \cdots, S_d^i\}$ 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 $\mathcal{S}_i^*$ consisting of d prefix groups. Thus, it is straightforward to verify that the r^{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

ϵ). Specifically, to verify the indistinguishability of the scheme, we calculate $Pr(\mathcal{R}_\alpha = \mathcal{R}_{\alpha,i})$ 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:

$$N_{total} = \frac{D!}{|S_1|!|S_2|! \cdots |S_d|!} \tag{13}$$

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 $\mathcal{R}_{\alpha,i} = \mathcal{R}_\alpha$ is:

$$N_{\text{real view candidates}} = \frac{\alpha!\,(D-\alpha)!\,\sum_{i=1}^{\binom{d}{\alpha}} \left(\Pi_{i=1}^{\alpha} |S_{a_i}|\right)}{|S_1|!|S_2|! \cdots |S_d|!} \tag{14}$$

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

$$\forall i \leq N: \ Pr(\mathcal{R}_{\alpha,i} = \mathcal{R}_\alpha) = \frac{N_{\text{real view candidates}}}{N_{total}} =$$

$$\mathcal{A} = \frac{\alpha!\,\sum_{i=1}^{\binom{d}{\alpha}} \left(\Pi_{i=1}^{\alpha} |S_{a_i}|\right)}{\Pi_{i=0}^{\alpha-1}(D-i)} \geq e^{-\epsilon} \tag{15}$$

Thus, to ensure the ϵ-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 ϵ of the generated views in scheme* II *is lower-bounded by*

$$\ln\left[\frac{D^\alpha}{d^\alpha} \cdot \Pi_{i=0}^{\alpha-1} \frac{(d-i)}{(D-i)}\right] \tag{16}$$

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

$$M_k = \frac{\sum\limits_{1 \leq i_1 \leq i_2 \leq \cdots \leq i_k \leq n} b_{i_1} b_{i_1} \cdots b_{i_k}}{\binom{n}{k}} \tag{17}$$

Per Maclaurin's inequality [29]: $M_1 \geq \sqrt[2]{M_2} \geq \cdots \geq \sqrt[n]{M_n}$, where $M_1 = \sum\limits_{i=1}^{n} b_i/n$, we have

$$\mathcal{A} = \frac{\alpha!\binom{d}{\alpha} M_\alpha}{\Pi_{i=0}^{\alpha-1}(D-i)} \leq \frac{\Pi_{i=0}^{\alpha-1}(d-i)(M_1)^\alpha}{\Pi_{i=0}^{\alpha-1}(D-i)}$$

and since $M_1 = \sum\limits_{i=1}^{n} |S_i|/n = D/d$, we have $A \leq \frac{D^\alpha}{d^\alpha} \cdot \Pi_{i=0}^{\alpha-1} \frac{(d-i)}{(D-i)}$. □

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 ϵ or increasing the chance of identifying the real view. Moreover, as it is illustrated in the figure, when fraction d/D grows, ϵ tends to converge to very small values. Hence, to decrease ϵ, 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.

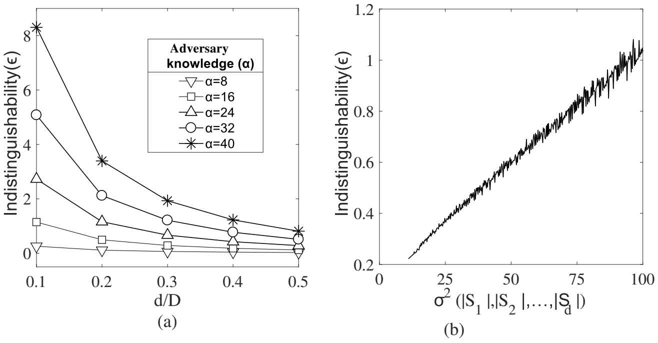


Figure 8: (a) Bound for ϵ as adversary's knowledge varies. (b) The exact value of ϵ in equation 15 for $\alpha = 8$, $d/D = 0.1$ and as variance of cardinalities varies

Finally, Figure 8(b) shows how variance of the cardinalities affects the indistinguishability for a set of fixed parameters d, D, α. In fact, when the cardinalities of the prefix groups are close (small σ), $\mathcal{A}$ 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 protocol in semi-honest model under secure multiparty computation (SMC) theory [43], [44].

Lemma 4.5. *Scheme* II *only reveals the CryptoPan Key K and the seed trace $\mathcal{L}_0^*$ 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: number of views to be generated, (2) K: the outsourced key, (3) $\mathcal{L}_0^*$: the seed trace, and (4) $V_1, V_2, \cdots, V_N$: 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 $V_1, V_2, \cdots, V_N$ are in $[-d, d]$ where d is the number of groups. First, given integer $0 < N \in N^*$, the probability that N is simulated in the domain would be $Pr[Simulator = N] = \frac{1}{N^*}$. 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 entires in $V_1, V_2, \cdots, V_N$ 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 $\mathcal{L}_0^*$ in semi-honest model. □

Note that the outsourced key can be considered as a public key and leakage in $\mathcal{L}_0^*$ is the leakage in output of the protocol which was studied earlier. Finally, we study the *setup leakage* and show that the adversary cannot exploit outsourced parameters to increase ϵ (i.e., decrease the number of real view candidates) by building his/her own key vector.

Lemma 4.6. *(Proof in Appendix B.2) For an $\mathcal{S}_\alpha$ 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

Application to EDB: We believe the multi-view solution would be applicable to other related areas. For instance, processing on encrypted databases (EDB) has a rich literature including searchable symmetric encryption (SSE) [52], [53], fully-homomorphic encryption (FHE) [54], oblivious RAMs (ORAM) [44], functional encryption [55], and property preserving encryption (PPE) [56], [57]. All these approaches achieve different trade-offs between protection (security), utility (query expressiveness), and computational efficiency [59]. Extending and applying the multi-view approach in those areas could lead to interesting future directions.

Comparing the Two Schemes: As discussed earlier, 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) demonstrates 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.

Choosing the Number of Views N: The number of views N is an important parameter 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 affordable computational overhead. Specifically, as implied by Equation 10 and demonstrated in our experimental results in section 5, the number of real view candidates is approximately $e^{-\epsilon} \cdot N$. The data owner can first estimate the adversary's background knowledge α (number of prefixes known to the adversary) and then calculate ϵ either using Equation 15 or (approximately) using Equation 16. As shown in Figures 8(a) and 9(b), a bigger α results in weaker indistinguishability and demands a larger number of views. An alternative solution is to increase the number of prefix groups (D) by sacrificing some prefix relations among IPs, e.g., grouping them with the first 3 octets.

Utility: The main advantage of the multi-view approach is preserving both privacy and utility. In particular, we have shown that the data owner can receive an analysis report based on the real view (Γ_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 $\mathcal{L}_r$ (migration output) to find the mapped IP addresses corresponding to each original address, and apply necessary security policies to the IP addresses that are reported to violate some policies in Γ_r.

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.

5 EXPERIMENTS

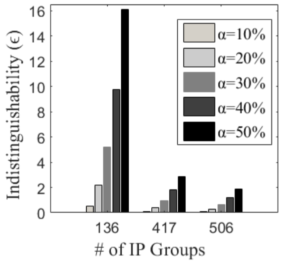
This section experimentally evaluates our technique with real data.

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 $Timestamp$, $IPaddress$, and $PacketSize$ in our experiments, and the meta-data are summarized in Figure 9(a).

Attribute	Value
Type Header	Header
Start date	August 1st, 2016
End date	October 21st, 2016
Size	70 GB
Format	pcap and TXT
Distinct IP addresses	883
Number of packets	1M

(a)



(b)

Figure 9: (a) Metadata of the traces (b) ϵ for different number of prefix groups and adversary knowledge

In order to measure the security of the proposed approach, we implement the frequency analysis attack [4, 59]. This attack can compromise individual addresses protected by existing prefix-preserving anonymization in multi-linear time [4]. 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 [59]. 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]: measure F_i is defined as the total number of addresses that has at least i most significant bits known, where $i \in \{1, 2, \cdots, 32\}$.

To model the adversarial knowledge, we define a set of prefixes known to the adversary ranging from 10% up to 100% of all the

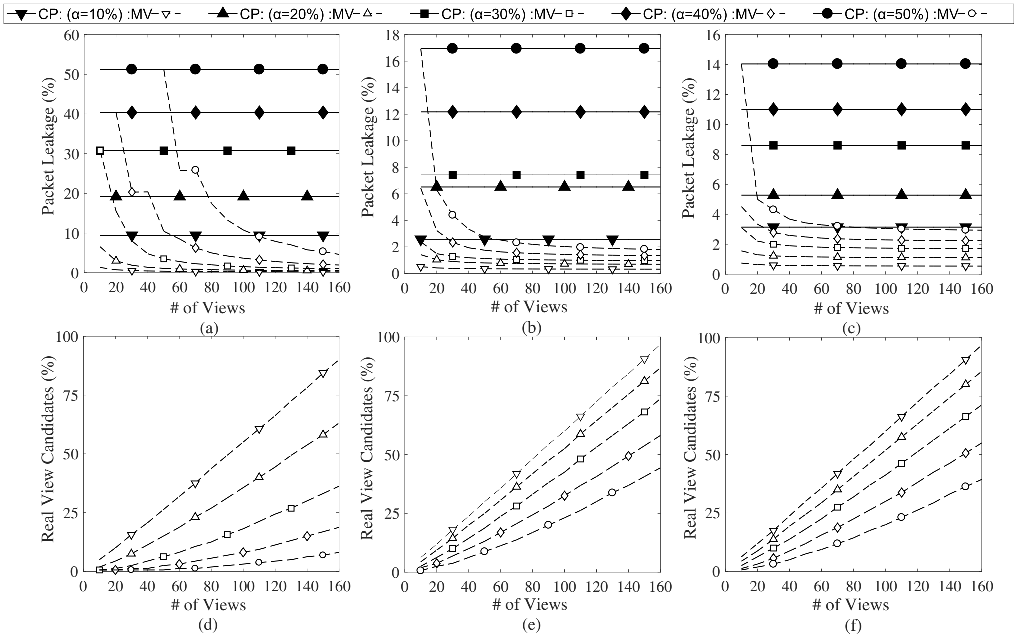


Figure 10: Percentage of the compromised packets (out of 1M) and number of real view candidates when number of views and the adversary knowledge vary and for the three different cases (1) Figures (a),(d), (2) Figures (b),(e), and (3) Figures (c),(f) where legends marked by *CP* denote the CryptoPAn result whereas those marked by *MV* denote the multi-view results

prefixes in the trace. This knowledge is stored in a two dimensional vector that includes α 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 indexes, 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 8 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 discussed in Appendix C.2 and C.3, respectively.

We conduct all experiments on a PC running Windows with an Intel Core i7-6700 3.40 GHz CPU, 4 GB Memory, and 500 GB HDD.

5.2 Results

5.2.1 Information Leakage Analysis. First, the numerical results of the indistinguishability parameter ϵ 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). The results show that ϵ decreases (more privacy) as the number of prefix groups increases, and it increases as the adversarial knowledge increases.

We next validate those numerical results through experiments in Figure 10. Specifically, we first analyze the performance of scheme II before comparing the two schemes in Appendix C. 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 have knowledge of no more than 50% of the prefix groups (Figure 12 in Appendix C.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) show the results of packet leakage and number of real view candidates when $d = 136$, respectively. As the numerical results in Figure 8 anticipates, since the fraction $d/D = 0.154$ is relatively low, the indistinguishability of generated views diminishes specially for stronger adversarial 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 $d/D = 0.47$ and 0.57, respectively. In the worst case of 50% adversarial knowledge and the number of views is below 50, we can verify that the number of real view candidates for case (1) remains 1 resulting in an equal packet leakage as that of CryptoPAn.

Effect of the number of views: As 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 adversarial 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. However, since each of the fake views leaks a certain amount of information, increasing the number of views beyond a certain value will no longer affect the end result. In other words, the packet leakage converges to the average of leakages in the (fake) real view candidates. Finally, the results show that our proposed scheme can more efficiently improve privacy by (1) increasing the fraction d/D (*number of prefix groups/number of distinct addresses*) or (2) increasing the number of views. The first option may affect utility (since inter-group prefix relations will be removed), while the second option is more aligned with our objective of trading off privacy with computation.

5.2.2 Computational Overhead Evaluation. We evaluate the computational overhead incurred by our approach. Figure 11 shows the time required by our scheme in each grouping cases, when the number of views varies for a trace including one million packets. We observe that, when the number of views increases, the computational overhead increases near linearly. However, each case shows a different slope depending on the number of groups. This is reasonable as our second scheme generates key vectors with a larger number of elements for more groups, which leads to applying CryptoPAn for more iterations (see the complexity analysis in Appendix D). Finally, linking this figure to the information leakage results shown in Figure 10 demonstrates the trade-off between privacy and computational overhead.

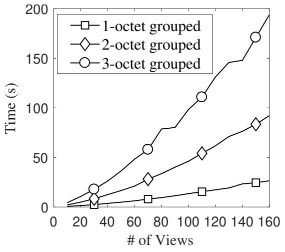


Figure 11: Runtime for different prefix grouping cases

6 RELATED WORK

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

As our proposed technique falls into the category of prefix-preserving solutions, which aims to improve the utility, we review more works 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 randomly generates a function. Fan et al. [3] then developed CryptoPAn with a completely cryptographic approach. Several works [4, 8, 31] have then raised its vulnerability against semantic attacks which motivated query based [17] and bucketization based [2] solutions.

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 utility loss. 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. A summary of the most important network trace anonymization schemes [18] and their main features are given in Appendix A.2.

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 [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]. Then, we choose ORAM since 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). Appendix C.3 presents more details on the scalability of using ORAM in our approach.

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 latter has been significantly decreased, making our approach a more preferable solution for applications that demand both privacy and utility. Our experimental results showed that our proposed approach also drastically reduced the information leakage compared to CryptoPAn.

8 ACKNOWLEDGEMENTS

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 and by PROMPT Quebec. It is also 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, 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(2):263-272, October 2004.

[4] Brekne, T., Årnes, A., and Øslebø, A. Anonymization of ip traffic monitoring data: Attacks on two prefix-preserving anonymization schemes and some proposed remedies. In PETS, pp. 179-196. Springer, 2005.

[5] Brekne, Tønnes, and André Årnes. "Circumventing IP-address pseudonymization." In Communications and Computer Networks, pp. 43-48. 2005.

[6] T.-F. Yen, X. Huang, F. Monrose, and M. K. Reiter, "Browser fingerprinting from coarse traffic summaries: Techniques and implications," in Proc. of Detection of Intrusions and Malware and Vulnerability Assessment, vol. 5587. Springer, 2009, pp. 157-175.

[7] M. Burkhart, D. Brauckhoff, M. May, and E. Boschi, "The risk-utility tradeoff for IP address truncation." In Proceedings of the 1st ACM workshop on Network data anonymization, 2008, pp. 23-30.

[8] Pang, R., Allman, M., Paxson, V. and Lee, J. "The devil and packet trace anonymization. ACM SIGCOMM Computer Communication Review." 36(1), pp.29-38, 2006.

[9] Coull, Scott E., Michael P. Collins, Charles V. Wright, Fabian Monrose, and Michael K. Reiter. "On Web Browsing Privacy in Anonymized NetFlows." In USENIX Security. 2007.

[10] Wong, Wai Kit, David W. Cheung, Edward Hung, Ben Kao, and Nikos Mamoulis. "Security in outsourcing of association rule mining." In Proceedings of the 33rd international conference on Very large data bases, pp. 111-122. 2007.

[11] Tai, C. H., Yu, P. S., and Chen, M. S. k-Support anonymity based on pseudo taxonomy for outsourcing of frequent itemset mining. In Proceedings of KDD, pp. 473-482. ACM, 2010.

[12] Slagell, Adam J., Kiran Lakkaraju, and Katherine Luo. "FLAIM: A Multi-level Anonymization Framework for Computer and Network Logs." In LISA, vol. 6, pp. 3-8. 2006.

[13] Foukarakis, Michael, Demetres Antoniades, and Michalis Polychronakis. "Deep packet anonymization." In Proceedings of the Second European Workshop on System Security, pp. 16-21. ACM, 2009.

[14] M. Burkhart, D. Schatzmann, B. Trammell, E. Boschi, and B. Plattner, The role of network trace anonymization under attack, Computer Communication Review, vol. 40, no. 1, pp. 5-11, 2010.

[15] Mogul, Jeffrey C., and Martin Arlitt. "Sc2d: an alternative to trace anonymization." In Proceedings of the 2006 SIGCOMM workshop on Mining network data, pp. 323-328. ACM, 2006.

[16] Mittal, Prateek, Vern Paxson, Robin Sommer, and Mark Winterrowd. "Securing Mediated Trace Access Using Black-box Permutation Analysis." In HotNets. 2009.

[17] McSherry, Frank, and Ratul Mahajan. "Differentially-private network trace analysis." In ACM SIGCOMM Computer Communication Review, vol. 40, no. 4, pp. 123-134. ACM, 2010.

[18] K. Mivule and B. Anderson, "A study of usability-aware network trace anonymization," 2015 Science and Information Conference (SAI), London, 2015, pp. 1293-1304.

[19] T. Farah, and L. Trajkovic, "Anonym: A tool for anonymization of the Internet traffic." In IEEE 2013 International Conference on Cybernetics, 2013, pp. 261-266.

[20] Mayberry, Travis, Erik-Oliver Blass, and Agnes Hui Chan. "Efficient Private File Retrieval by Combining ORAM and PIR." In NDSS. 2014.

[21] A.J. Slagell, K. Lakkaraju, and K. Luo, "FLAIM: A Multi-level Anonymization Framework for Computer and Network Logs." In LISA, vol. 6, 2006, pp. 3-8.

[22] J. Xu, J. Fan, M.H. Ammar, and Sue B. Moon, "Prefix-preserving ip address anonymization: Measurement-based security evaluation and a new cryptography-based scheme.", In 10th IEEE International Conference on Network Protocols, 2002, pp. 280-289.

[23] Wang, Xiao Shaun, Yan Huang, TH Hubert Chan, Abhi Shelat, and Elaine Shi. "SCORAM: oblivious RAM for secure computation." In CCS, pp. 191-202. ACM, 2014.

[24] W. Yurcik, C. Woolam, G. Hellings, L. Khan, B. Thuraisingham, "Measuring anonymization privacy/analysis tradeoffs inherent to sharing network data", IEEE Network Operations and Management Symposium, 2008, pp.991-994.

[25] Gattani, Shantanu, and Thomas E. Daniels. "Reference models for network data anonymization." In Proceedings of the 1st ACM workshop on Network data anonymization, pp. 41-48. ACM, 2008.

[26] Zhang, Jianqing, Nikita Borisov, and William Yurcik. "Outsourcing security analysis with anonymized logs." In Securecomm Workshops, pp. 1-9. IEEE, 2006.

[27] Saroiu, Stefan, P. Krishna Gummadi, and Steven D. Gribble. "Measurement study of peer-to-peer file sharing systems." In Electronic Imaging 2002, pp. 156-170. International Society for Optics and Photonics, 2001.

[28] Chor, Benny, et al. "Private information retrieval." Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on. IEEE, 1995.

[29] Biler, Piotr, and Alfred Witkowski. "Problems in mathematical analysis." (1990).

[30] Zhang, Q., & Li, X. (2006, January). An IP address anonymization scheme with multiple access levels. In International Conference on Information Networking (pp. 793-802). Springer Berlin Heidelberg.

[31] B. Ribeiro, W. Chen, G. Miklau, and D. Towsley, Analyzing privacy in enterprise packet trace anonymization, in Proc. NDSS, San Diego, CA, Feb. 2008, pp. 87-100

[32] 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.

[33] 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.

[34] 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.

[35] Del Piccolo, Valentin, et al. "A Survey of network isolation solutions for multi-tenant data centers." IEEE Commun. Surveys & Tutorials 18.4 (2016): 2787-2821.

[36] C. Dwork, "Differential privacy." In ICALP, ser. Vol. 4052. Springer-Verlag, 2006.

[37] 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, 2008.

[38] Kushilevitz, Eyal, and Rafail Ostrovsky. "Replication is not needed: Single database, computationally-private information retrieval." In Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on, pp. 364-373. IEEE, 1997.

[39] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious RAMs. J. ACM 43(3) 431-473, 1996.

[40] C. Dwork, F. McSherry, K. Nissim, and A. Smith, “Calibrating noise to sensitivity in private data analysis,” in Proceedings of the Third Theory of Cryptography Conference, 2006, pp. 265–284.

[41] Le Ny, Jerome, and Meisam Mohammady. "Differentially private MIMO filtering for event streams." IEEE Tran. on Automatic Control 63, no. 1 (2018): 145-157.

[42] Le Ny, Jerome, and Meisam Mohammady. "Differentially private MIMO filtering for event streams and spatio-temporal monitoring." In Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on, pp. 2148-2153. IEEE, 2014.

[43] Yao, Andrew Chi-Chih. "How to generate and exchange secrets." In Foundations of Computer Science, 1986., 27th Annual Symposium on, pp. 162-167. IEEE, 1986.

[44] Goldreich, Oded. "Secure multi-party computation." Manuscript. Preliminary version (1998): 86-97.

[45] Cormen, Thomas H., et al. "Data structures for disjoint sets." Introduction to Algorithms (2001): 498-524

[46] Sedgewick, Robert. "Implementing quicksort programs." Communications of the ACM 21.10 (1978): 847-857.

[47] Slagell, Adam, Jun Wang, and William Yurcik. "Network log anonymization: Application of crypto-pan to cisco netflows." Proceedings of the Workshop on Secure Knowledge Management 2004. 2004.

[48] Minshall G. TCPdpriv command manual. 1996. http://ita. ee. lbl. gov/html/contrib/tcpdpriv. 0. txt. 1996.

[49] Paul, Ruma R., Victor C. Valgenti, and Min Sik Kim. "Real-time Netshuffle: Graph distortion for on-line anonymization." In Network Protocols (ICNP), 2011 19th IEEE International Conference on, pp. 133-134. IEEE, 2011.

[50] Aggarwal, Gagan, Tomás Feder, Krishnaram Kenthapadi, Samir Khuller, Rina Panigrahy, Dilys Thomas, and An Zhu. "Achieving anonymity via clustering." In Proceedings of SIGMOD-SIGACT-SIGART, pp. 153-162. ACM, 2006.

[51] Boldyreva, Alexandra, Nathan Chenette, Younho Lee, and Adam O'neill. "Order-preserving symmetric encryption." In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 224-241. Springer, Berlin, Heidelberg, 2009.

[52] Curtmola, Reza, Juan Garay, Seny Kamara, and Rafail Ostrovsky. "Searchable symmetric encryption: improved definitions and efficient constructions." Journal of Computer Security 19, no. 5 (2011): 895-934.

[53] Song, Dawn Xiaoding, David Wagner, and Adrian Perrig. "Practical techniques for searches on encrypted data." In Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on, pp. 44-55. IEEE, 2000.

[54] Craig Gentry. 2009. Fully homomorphic encryption using ideal lattices. In Proceedings of the 41st ACM symposium on Theory of computing (STOC '09). 169-178.

[55] Boneh, Dan, Amit Sahai, and Brent Waters. "Functional encryption: Definitions and challenges." In Theory of Cryptography Conference, pp. 253-273. Springer, Berlin, Heidelberg, 2011.

[56] Bellare, Mihir, Alexandra Boldyreva, and Adam O'Neill. "Deterministic and efficiently searchable encryption." In Annual International Cryptology Conference, pp. 535-552. Springer, Berlin, Heidelberg, 2007.

[57] Boldyreva, Alexandra, Nathan Chenette, Younho Lee, and Adam O'neill. "Order-preserving symmetric encryption." In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 224-241. Springer, Berlin, Heidelberg, 2009.

[58] Islam, Mohammad Saiful, Mehmet Kuzu, and Murat Kantarcioglu. "Access Pattern disclosure on Searchable Encryption: Ramification, Attack and Mitigation." In Ndss, vol. 20, p. 12. 2012.
[59] Naveed, Muhammad, Seny Kamara, and Charles V. Wright. "Inference attacks on property-preserving encrypted databases." In Proceedings of the 22nd ACM CCS, pp. 644-655. ACM, 2015.
[60] Hautakorpi, Jani, and Gonzalo Camarillo Gonzalez. "IP Address Distribution in Middleboxes." U.S. Patent Application No. 12/518,452.
[61] Chang, Zhao, Dong Xie, and Feifei Li. "Oblivious ram: a dissection and experimental evaluation." Proceedings of the VLDB Endowment 9, no. 12 (2016): 1113-1124.
[62] Caswell, Brian, and Jay Beale. Snort 2.1 intrusion detection. Elsevier, 2004.
[63] Stefanov, E., Van Dijk, M., Shi, E., Fletcher, C., Ren, L., Yu, X. and Devadas, S., 2013, November. Path ORAM: an extremely simple oblivious RAM protocol. In Proceedings of the 20th CCS, pp. 299-310. ACM, 2013.

A TABLES

A.1 Notation Table

Table 1: The Notation Table.

Symbol	Definition	Symbol	Definition
$\mathcal{L}$	Original network trace	$\mathcal{L}^*$	Anonymized trace
A^{IP}	IP attributes: source and destination IP	*fp-QI*	Fingerprint quasi identifier
r_i	Record number i	n	Number of records in $\mathcal{L}$
α	Number of IP prefixes known by the attacker	$\mathcal{S}_\alpha$	The set of addresses known by attacker
$\mathcal{S}_0^*$	The set of IP addresses in the seed view	$\mathcal{S}_i^*$	The set of IP addresses in view i
PP	CryptoPAn function	RPP	Reverse of CryptoPAn
P_i	Partition i	m	Number of partitions in $\mathcal{L}$
r	Index of real view	K_0, K_1	Private key and outsourced key

A.2 Network Trace Anonymization Schemes

Table 2: Network trace anonymization Schemes

Authors	Privacy against semantic attacks	Utility
Slagell 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
Mogul et al. [15]	Violated	Aggregated results

B PROOFS

B.1 Proof of Theorem 3.1

Proof. 3.1 We must show that $c = a$. To do so, we use induction:

$$c_1 = b_1 \oplus f_0(\lambda), \;\; b_1 = a_1 \oplus f_0(\lambda)$$
$$\Rightarrow c_1 = a_1 \oplus f_0(\lambda) \oplus f_0(\lambda) = a_1$$

where λ is empty string and $f_0(\lambda)$ is a constant bit in $\{0, 1\}$ that depends only on padding function and cryptographic key K. Assume $\forall j < k; \;\; c_j = a_j$, thus:

$$c_k = b_k \oplus f_{k-1}(c_1 \cdots c_{k-1})$$
$$\text{and: } \forall j < k; \quad c_j = a_j, \; b_k = a_k \oplus f_{k-1}(a_1 \cdots a_{k-1})$$
$$\Rightarrow c_k = a_k \oplus f_{k-1}(a_1 \cdots a_{k-1}) \oplus f_{k-1}(a_1 \cdots a_{k-1}) = a_k$$

Thus, this completes the proof. □

B.2 Proof of Lemma 4.6

Proof. Suppose there exists algorithm A which returns the smallest set of key vectors V_s 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 $V = V_s$. First, note that the key indices of different distinct addresses in $\mathcal{L}_0^*$ is $PRNG(d, D, 0)$. Therefore, the adversary has to guess $V = C^* - PRNG(d, D, 0)$. 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 $\text{A}(\mathcal{L}_0^*, K, V_1, \cdots, V_N, \mathcal{S}_\alpha) = V_s$. However, we showed earlier that all these inputs are trivial leakage. Therefore, if $(2d - 2)^D > N$ holds, we have $V = V_s$. □

C ADDITIONAL EXPERIMENTS

We also utilize experiments to 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.

C.1 Privacy against Very Strong Adversaries

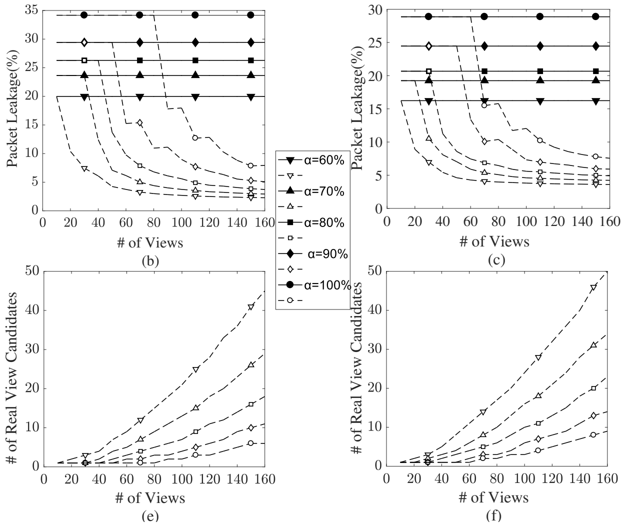


Figure 12: Percentage of the compromised packets (out of 1M) and number of real view candidates as number of views and the adversarial knowledge vary and for case (1) Figures (b),(e), and (2) Figures (c),(f) where *CP* denotes the CryptoPAn result and *MV* denotes the multi-view results

Figure 12 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 ($\epsilon > 16$).

C.2 Utility Evaluation Using Real-life Network Trace Analyses

Figure 13 shows the results of two different network analyses 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 [60] 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, as seen in Figure 13(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 13(c)).

In the second experiment, we present a packet-level network analysis [17]. In particular, Figure 13(d,e) shows the results of *empirical cumulative distribution function (CDF)* for the three traces. Our results clearly show that the original trace and our scheme results are identical since multi-view will not have any impact on fingerprinting quasi-identifier attributes.

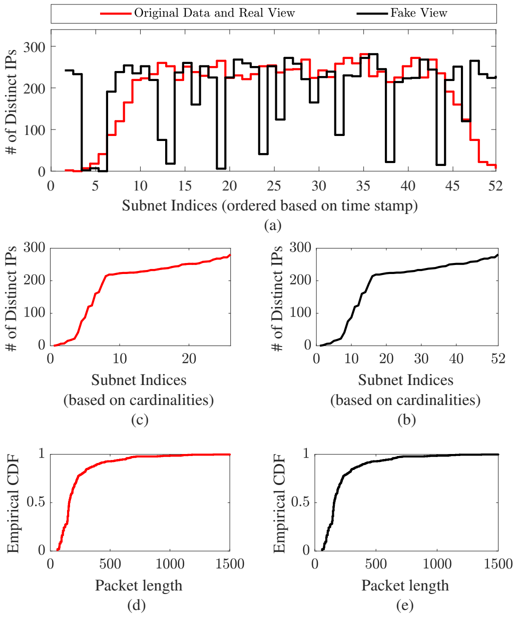


Figure 13: Distribution of distinct IP addresses in different subnets (a-c) and empirical CDF for the packet lengths (d,e)

C.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 believe the solution would have acceptable scalability. Experiments using our dataset and existing ORAM implementation (an implementation [61] of non-recursive Path-ORAM [63] has been made public) have further confirmed this. We generated various sets of analyses reports using *snort* [62]. Our large-scale experimental dataset only results in *Kilobytes*-level audit reports, which can be perfectly used in fast ORAM protocols, e.g., Path-ORAM. Specifically, for Path-ORAM, Figure 5 (b) in [61] shows a less than 1MB communication overhead for the worst-case cost of up to 2^{24} number of blocks of size 4KB.

C.4 Comparison between Two Schemes

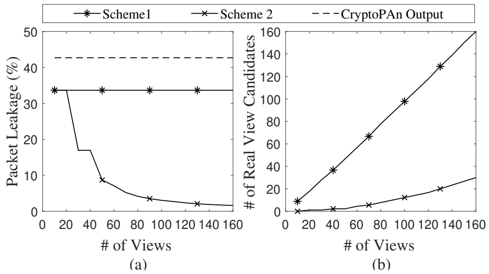


Figure 14: Comparison between scheme I and II with 137 partitions (prefix groups based on first octet sharing).

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 $N \cdot CV(n)$ 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	$O(nlog^n) + \frac{\sum_{i=1}^{d} C(i)}{d} C(n)$	—
Prefix grouping	—	—
Index generator	$N \cdot O(D)$	$N \cdot O(D)$
Seed trace	$\frac{\sum_{i=1}^{D} V_0(i)}{D} C(n)$	$O(n)$
Report retrieval (ORAM)	—	$O(log^N)\omega(1)$

Table 4: Overhead on the data analyst side

Blocks in Multi-view	Computation Overhead	Communication Overhead
Seed view	—	$O(n)$
N views generation	$\frac{\sum_{i=1}^{N} \sum_{j=1}^{D} V_i(j)}{D} C(n)$	—
Compliance verification (Analysis)	$N \cdot CV(n)$	—

E ALGORITHMS

Following Algorithms are summarized versions of the data owner's and the analyst's roles in our multi-view schemes presented in section 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).

```
Input:
  $\mathcal{L}$: Original network trace
  $K_0, K$: Cryptographic keys
  $d$: Number of prefix groups
  $partition(\mathcal{L}, d)$: IP partitioning
  $r$: Iteration number of the real view
  $V$: Random vectors, of size $d$
Output:
  $\mathcal{L}^*$: Anonymized trace to be outsourced
Function: anonymize $(\mathcal{L}, d, K_0, K, r, V)$
begin
    $\mathcal{L} := PP(\mathcal{L}, K_0)$
    $V_0 := -r \cdot V$
    $P := partition(\mathcal{L}, d)$
    $\mathcal{L}^* := \phi$
    foreach  $P_i \in P$  do:
        $\mathcal{L}_i :=$ GetFlows$(\mathcal{L}, P_i)$
        $\mathcal{L}_i^* := PP(\mathcal{L}_i, V_0(i), K)$
        $\mathcal{L}^* := \mathcal{L}^* \cup \mathcal{L}_i^*$
    end
    return $\mathcal{L}^*, K, V, N$
end
```

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

```
Input:
  $\mathcal{L}^*$: Seed trace
  $N$: Number of iterations requested by data owner
  $d$: Number of prefix groups
  $partition(\mathcal{L}^*, d)$: IP partitioning
  $K$: Outsourced key
  $V$: Vector of size $d$ defined by data owner
  $CV(\mathcal{L}^*)$: Compliance verification
Output:
  $\Gamma_i$: Analysis report of $i^{th}$ view $\mathcal{L}_i^*$, $i \in \{1, 2, \ldots, , N\}$
Function: analysis $(\mathcal{L}^*, d, partition(\mathcal{L}^*, d), K, N, V)$
begin
    $P := partition(\mathcal{L}^*, d)$
    for $i = 1 : N$  do:
        $\mathcal{L}_i^* := \phi$
        foreach  $P_j \in P$  do:
            $\mathcal{L}_{i,j} :=$ GetFlows$(\mathcal{L}^*{}_{i-1}, P_j)$
            $\mathcal{L}_{i,j}^* := PP(\mathcal{L}_{i,j}, V(j), K)$
            $\mathcal{L}_i^* := \mathcal{L}_i^* \cup \mathcal{L}_{i,j}^*$
        end
        $\Gamma_i := CV(\mathcal{L}_i^*)$
        return $\Gamma_i$
    end
end
```

Algorithm 2: Analyst: Network trace analysis (scheme I)

```
Input:
  $\mathcal{L}, K_0, K, d, r$: refer to scheme I
  $D$: Number of IPs
  $Migration(\mathcal{L}, d)$: Migration function
  $partition(\mathcal{L}, D)$: Distinct IP partitioning
  $V_0, V_1, \cdots, V_N$: Vectors of size $D$ defined by data owner
Output:
  $\mathcal{L}^*$: Anonymized trace to be outsourced
Function: anonymize $(\mathcal{L}, d, D, K_0, K, r, V)$
begin
    $\mathcal{L} := PP(\mathcal{L}, K_0)$
    $\mathcal{L} := Migration(\mathcal{L}, d)$
    $P := partition(\mathcal{L}, D)$
    $\mathcal{L}^* := \phi$
    foreach  $P_i \in P$  do:
        $\mathcal{L}_i :=$ GetFlows$(\mathcal{L}, P_i)$
        $\mathcal{L}_i^* := PP(\mathcal{L}_i, V_0(i), K)$
        $\mathcal{L}^* := \mathcal{L}^* \cup \mathcal{L}_i^*$
    end
    return $\mathcal{L}^*, K, V_1, V_2, \cdots, V_N, N$
end
```

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

```
Input:
  $\mathcal{L}^*, N, d, K, CV(\mathcal{L}^*)$: refer to scheme I
  $partition(\mathcal{L}^*, D)$: IP partitioning
  $V_0, V_1, \cdots, V_N$: Vectors of size $D$ defined by data owner
Output:
  $\Gamma_i$: Analysis report of $i^{th}$ view $\mathcal{L}_i^*$, $i \in \{1, 2, \ldots, , N\}$
Function: analysis
  $(\mathcal{L}^*, D, partition(\mathcal{L}^*, D), K, N, V_1, V_2, \cdots, V_N)$
begin
    $P := partition(\mathcal{L}^*, D)$
    for $i = 1 : N$  do:
        $\mathcal{L}_i^* := \phi$
        foreach  $P_j \in P$  do:
            $\mathcal{L}_{i,j} :=$ GetFlows$(\mathcal{L}^*{}_{i-1}, P_j)$
            $\mathcal{L}_{i,j}^* := PP(\mathcal{L}_{i,j}, V_i(j), K)$
            $\mathcal{L}_i^* := \mathcal{L}_i^* \cup \mathcal{L}_{i,j}^*$
        end
        $\Gamma_i := CV(\mathcal{L}_i^*)$
        return $\Gamma_i$
    end
end
```

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