MixFlow: Assessing Mixnets Anonymity with Contrastive Architectures and Semantic Network Information

Reyhane Attarian
imec-COSIC- KU Leuven
Leuven, Belgium
rattaria@esat.kuleuven.be

Esfandiar Mohammadi
University of Lübeck
Lübeck, Germany
esfandiar.mohammadi@uni-luebeck.de

Tao Wang
Simon Fraser University
British Columbia, Canada
taowang@sfu.ca

Emad Heydari Beni
imec-COSIC- KU Leuven
Leuven, Belgium
Emad.HeydariBeni@esat.kuleuven.be

ABSTRACT

In this paper, we propose the MixFlow model, an approach for analyzing the unobservability and unlinkability of instant messaging in Loopix Mix networks. The MixFlow model utilizes contrastive architectures and loss functions inspired by DNA sequence analysis in bioinformatics to identify semantic relationships between entry and exit flows, even after applying significant transformations such as poisson mixing delay and cover traffic. We use the MixFlow model to evaluate the resistance of Loopix Mix networks against a global passive adversary with the ability to control both ends of the network and infer real messages from cover messages. Our experimental results demonstrate that the MixFlow model is highly effective in linking end-to-end flows with a detection rate of over 90%, challenging the common belief that adding poisson mixing delay and cover traffic can obscure the metadata patterns and relationships between communicating parties. Our findings have important implications for existing poisson-mixing techniques and open up new opportunities for analyzing the anonymity and unlinkability of communication protocols.

KEYWORDS

Mixnets; Flow Correlation Attack; Contrastive models

ACM Reference Format:
Reyhane Attarian, Esfandiar Mohammadi, Tao Wang, and Emad Heydari Beni. 2023. MixFlow: Assessing Mixnets Anonymity with Contrastive Architectures and Semantic Network Information. In *Proceedings of ACM Conference (Conference'17)*. ACM, New York, NY, USA, 16 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Mixing networks, such as Loopix [24], have gained attention in recent years as a means of enabling private communication between two parties through the exchange of instant messages. For example, Bahramali et al. [2] have evaluated the vulnerability of various instant messaging apps and suggested using countermeasure techniques, such as cover traffic and poisson mixing delays, to reduce the effectiveness of traffic analysis attacks. These networks, also known as "stop-and-go Mix networks", utilize Poisson mixing strategies in addition to cryptographic operations to provide anonymity and unobservability for the sender and receiver, and to prevent a global passive adversary from linking the entry and exit flows in a flow correlation attack [24]. Flow correlation attacks have been used to compromise anonymous systems and measure the amount of network information leakage against a global passive adversary [5, 24, 27, 32]. These attacks aim to determine the level of uniqueness of entry and exit flows even after significant transformations, such as the addition of cover traffic and random delays in Mixnets. The goal of a flow correlation attack is not to identify a single fingerprint, but rather to understand the similarity between two sets of observations. It is generally believed that drastic flow transformations can effectively disguise network flows and that injecting cover traffic and delays can help anonymize network connections. If a global adversary, who has control over both ends of the network, is unable to learn any identifying characteristics from the transformed network flows, we can conclude that the network is unlinkable and the anonymity goal has been achieved. Further research on developing flow correlation attacks and investigating these attack models on Loopix can help identify more effective defenses for network anonymity.

Despite the potential of Mixnets in protecting anonymity, there is currently no research evaluating the information leakage when short instant messaging is transferred in Mixnets. Previous studies, such as DeepCoffea, FlowTracker, and DeepCorr, primarily focused on deanonymizing traffic on the Tor network and analyzing website traffic traces, rather than evaluating the vulnerability of messaging applications [11, 19, 21]. While FlowTracker [11] and DeepCoffea [21] have been shown to be effective in traffic analysis, they only investigated their attack for flow correlation on website traffic traces. These attack models are efficient only for long flow traffic traces (more than 100 packets per flow), and that shorter flow sequences, especially when mixed with additional poisson mixing noises, are not detectable with their models. As poisson mixing increases, it becomes more difficult to distinguish the true corresponding entry flow from unrelated flows that appear similar to the exit flow, resulting in correlation failure. Moreover, existing flow correlation studies do not address how to leverage information from unrelated flows to create more effective differential representations for attacking. When analyzing Mixnet traffic sequences, it's common to observe discrepancies between the number of sent packets and received packets due to mixing delays, injecting cover traffic, reordering, and network jitters. However, current state-of-the-art attacks rely on packet-level feature sequences for flow correlation, which may not accurately represent the traffic shape as the original one-to-one packet relationship is lost and the same location in the ingress and egress flow sequences may not correspond to the same packet in Mixnet traffic flows [11, 19, 21]. Indeed, there is a gap in the literature when it comes to analyzing correlation attacks on instant messaging in Mixnets. This is because instant messaging traffic flows on Mixnets typically have fewer packets with more traffic perturbation, which limits the applicability of previous correlation attacks [11, 19, 21].

Our Contribution: In this paper, we introduce MixFlow, an optimal flow correlation model for the particular settings of instant messaging communications on Mix networks. MixFlow is a contrastive approach based on flow and label embeddings, along with a Poisson Hidden Markov Model (PHMM) as the feature embedding network, to identify complex, semantic patterns and achieve a predefined similarity goal. Mixflow incorporates two contrasting models that extract semantic network traffic features from two unique perspectives in the embedding space, providing the ability to be used in varying levels of anonymity: The contrastive flow embeddings (ConFEm) model and the Contrastive with Flow and Label embedding (ConCEn) model. These models are adaptable to various external factors such as poisson delays or cover traffic and are not limited to fixed flow lengths, target websites, or clients. We proposed ConFEm, a model that assesses how the flow embeddings and semantic information of uncorrelated exit flows can adjust embedding spaces and find the optimal decision boundary for correlating flows. ConCEN, on the other hand, is a model that demonstrates the effectiveness of using a combination of flow and label embeddings, cluster representations, correlated labels, and a single representative for clusters of uncorrelated flows. The aim is to extract representations and create embeddings based on selected positive and negative exit flows for each entry flow. The following is the highlight of MixFlow's contributions:

- MixFlow is an intuitive approach to explaining flow patterns in terms of hidden structure and serves as a crucial component of traffic correlation analysis.
- MixFlow surpasses prior flow correlation attacks by correlating instant messaging flows. It uncovers semantic network information and traffic sequences, even in the presence of poisson-mixing networks.
- MixFlow presents models that use flow and label embeddings, as well as representative information for uncorrelated flow clusters. The models convert features into an embedding space, resulting in a detection rate above 90% for correlating entry and exit flows in Loopix.
- MixFlow can be utilized as a versatile correlation function to examine the linkability and anonymity of clients in Mixnets. MixFlow's performance is stable across a wide range of conversation lengths, poisson delays, and cover traffic values.
- MixFlow demonstrates the robustness and scalability, even when faced with increased poisson delays and cover traffic, highlighting the need for further development and implementation of traffic analysis countermeasures to protect connections.
- MixFlow is designed using random deterministic projection algorithms and reduces the overall cost of flow correlation attacks.
- MixFlow presents a new approach for assessing the efficacy of poisson-mixing techniques for instant messaging against a global passive adversary for anonymity and unlinkability.

The foundation of our approach lies in the lack of robust traffic obfuscation measures implemented by major poisson-mixing operators, which makes it possible for us to correlate user traffic. It has the potential to inform future efforts to enhance the security of mixing protocols.

2 PRELIMINARIES AND MOTIVATION

MixFlow is an end-to-end contrastive flow correlation attack that aims to identify similar connections at both ends of anonymous networks like Loopix Mixnets. The proposed attack is based on the triplet networks [26] in a supervised contrastive scheme: an embedding network for entry flows and two embedding networks for exit flows, which share weights. These sub-networks are passed through a similarity module, which calculates the distance between embeddings to determine *how similar* they are. Using a predetermined similarity score and a loss function, the similarity distances are evaluated and the model weights are adjusted.

MixFlow is a model that uses representations of flows in embedding space to reduce complexity and the need for pairwise comparisons. This approach is effective in handling increased Mixnets' poisson delays and the cover traffic. Previous contrastive models [11, 21, 30] used sample-wise strategies that compared embeddings and classified elements based on a majority vote, without directly labeling the "correlates to" set. These approaches calculated the distance between the anchor and representative samples of each class using a threshold or by aggregating the distances between samples. In contrast, our approach uses representative-based contrastive strategies, which involve calculating the distance between representative points in the embedding space instead of individual samples. MixFlow models use both flow and label embeddings to simultaneously embed both flows and labels in the same vector space. The embedding vectors of both flows and labels are then used to calculate distances. To improve efficiency, MixFlow replaces correlated and uncorrelated exit flows for each entry flow with a single representative element for both positive and negative samples. This allows each entry flow to be compared to the representatives of both positive and negative flows, rather than comparing each flow individually. By using label embedding in conjunction with flow embedding as templates or coordinates for embedding representations, MixFlow is able to reduce the need for costly pairwise comparisons during training and prediction. This approach allows MixFlow to create more precise references for comparison from a supervised standpoint, leading to more accurate and efficient learning from the data.

2.1 Entry and Exit flows

The proposed framework is based on the assumption that each entry flow, represented by n_i, has a set of associated exit flows, represented by x_k, which are similar to n_i and are identified as $x_k \in Corr(n_i)$. Correlated exit flows are referred to as positive samples, while exit flows that are not correlated to the reference anchor entry flow are called negative samples. The goal of the proposed models is to bring the entry flow n_i closer to positive exit flows in the embedding space, while simultaneously separating n_i from negative exit flows that do not belong to $Corr(n_i)$. In order to achieve this separation, the entry flows n_i are compared with positive samples that are similar to n_i ($x_k \in Corr(n_i)$) as well as with a representative number of negative samples that are not similar to n_i ($x_j \notin Corr(n_i)$). We supposed N flow pairs (n_i, x_i), where n_i is an entry flow with p packets ($n_i \in R^p$) and x_i is an exit flow with p' packets ($x_i \in R^{p'}$) and $p \neq p'$. The flow vectors are created by concatenating the inter-packet delay (IPD) and packet size information for each flow ($[I_i||S_i]$). The IPD vectors (I_i) and packet size vectors (S_i) contain both upstream and downstream packets, with the downstream packets multiplied by -1 to indicate their direction.

2.2 Base Contrastive model (MMFEM)

We consider a Max-Margin over Flow embedding contrastive model (*MMFEm*) with the $Loss = \frac{1}{N}\sum_{i=1}^{N} max(D[\phi(n_i), \phi(x_i^P)] - D[\phi(n_i), \phi(x_i^N)] + 1, 0)$, over the distance between the entry flow n_i and the exit flows x_i^P and x_i^N, as a base architecture for our correlation attack models. The base loss function is based on the max-margin separation strategy to increase the difference between the distance (Euclidean or Cosine similarity distance) between the entry flow n_i and the correlated exit flow x_i^P and the distance between the entry flow n_i and uncorrelated exit flow x_i^N beyond a certain margin. In this model, we consider three separate embedding networks to evaluate the importance of sample pairs and make the feature extraction model converge faster. These networks have different input sizes but the same output size, which represents the embeddings of the respective flows: $(\phi(n_i), \phi(x_i^P), \phi(x_i^N))$. $\phi(a)$ is an embedding transformation that maps the input vectors $a \in R^p$ into a new space with a lower dimension, R^m, where $m < p$. This preserves the representational capacity of the original vectors in the newly mapped vectors and allows the network to discover relationships between the embedded vectors.

2.3 Flow and Label Embeddings

To evaluate the correlation between flow embeddings, we apply a distance function to calculate the distance between the entry flow embeddings and the positive exit flow embeddings ($D[\phi(n_i), \phi(x_i^P)]$), as well as the distance between the entry flow embeddings and the negative exit flow embeddings ($D[\phi(n_i), \phi(x_i^N)]$). These distances are then used as input to the loss function, which calculates the triplet loss and updates the embedding networks to better understand the correlation between the flows. During the prediction phase, we input the entry flow of the test sample and the exit flow we want to test into the model, and we use the trained embedding networks for the entry and exit prototypes to compute their distance. To incorporate label information into the embedding space, MixFlow maps each label to a specific position in the space and uses this position to define the natural representative location for each sample in triplets. In this case, labels are represented using one-hot encoding, which involves creating a zero array of length two (corresponding to the two classes) with a single one placed at the position of the ground-truth label y_i. For each anchor sample with ground-truth label/class y_i, the ground-truth label is represented as L_i^k. Since this is a binary classification problem, the index k_i can take on values of either 0 or 1. The positive label for positive exit flows is defined as $L_{x_i^P} = L_i^k = y_i$) if $L_i^k = y_i$, and the negative label for negative samples is defined as $L_{x_i^N} \neq y_i$). If y_{ij} represents the position of the one-hot-encoded array, then $y_{ij} = 1$ if $j = k_i$ and $y_{ij} = 0$ if $j \neq k_i$.

3 MIXFLOW: CONTRASTIVE FLOW CORRELATION ATTACK

In this work, to implement a contrastive learning framework between the entry and exit flows from different perspectives, we propose two constructive models for performing our flow correlation attack: Contrastive with Flow Embeddings (*ConFEm*) and Contrastive with Flow and Label Embeddings (*ConCEn*). Each proposed model architecture is designed with appropriate loss functions to uncover the functional network information that connects correlated flows.

3.1 Contrastive with Flow Embeddings (ConFEm)

The *ConFEm* model focuses on using a mixture of Max-Margin and Min-Separation strategies to increase the distance, beyond a margin, between the entry flow n_i and negative exit flow x_i^N while reducing the distance, as much as possible, between the entry flow n_i and the positive exit flow x_i^P. Figure 1 shows an overall structure of the *ConFEm* model. *ConFEm* model aims to capture the underlying semantic relations between flow sequences, and the feature representations that are robust and generalizable to unseen data. This Contrastive model is based on the flow embeddings architecture and the distance function $D[\phi(a), \phi(b)]$ that can be the Euclidean or Cosine similarity distance. Finally, we compute the similarity for each testing sample using the trained embedding networks in the prediction phase.

Moreover, uniformity in flow correlation can help to learn separable features, but the excessive pursuit of uniformity can negatively impact the formation of features useful for downstream tasks. This is because the instance discrimination objective, which tries to push all uncorrelated flows apart, ignores the underlying relations between samples. A well-designed contrastive correlation loss should have some tolerance to the closeness of semantically similar flows in order to improve feature qualities and downstream performance. Hence, we define the contrastive loss to meet a uniformity-tolerance dilemma with a temperature τ that can compromise these two properties properly to both learn separable features and be tolerant to semantically similar flows, improving the feature qualities and the downstream performances.

3.1.1 Loss Function and Embedding Parameters: Temperature is used as a hyperparameter to balance the trade-off between uniformity and tolerance in contrastive loss to control the degree of similarity between semantically similar flow sequences. The temperature parameter is a learnable parameter during training, and its value is typically chosen between the range of 0.1 to 2. A lower temperature would result in a higher degree of similarity, and a higher temperature would result in a lower degree of similarity. By adjusting the temperature, the algorithm can be made more or less tolerant of the closeness of semantically similar flows. The good choice of temperature will help to balance the trade-off between uniformity and tolerance in the contrastive loss, thereby improving the feature qualities and downstream performance. Using these parameters, the loss $Loss_{ConFEm}$ for N samples is computed as:

$$Loss_{ConFEm} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{2}[Y_{i,L_j}.D[\phi(n_i), \phi(x_i^P)] \\ + (1 - Y_{i,L_j}).max(1 - D[\phi(n_i), \phi(x_i^N)], 0)] \quad (1)$$

The indicator function Y_{i,L_j} is used to define the positive and negative exit flows for each entry flow, which is used to compute the contrastive loss. The indicator function $Y_{i,L_j} = 1$ if $L_j = y_i$, 0 otherwise. The indicator function for our two-class problem is based on the ground-truth label, and we interpret the distances as a probability loss. In this regard, we applied the *sigmoid* function to transform the distances to a range of values between 0 and 1. This distance computation is now based on a dot product with an additional *sigmoid* function to scale the output in the range of values $[0, 1]$.

3.1.2 Temprature and Indicator Function: The temperature is the scalar hyperparameter typically added to the logits before the sigmoid function and multiplied with the logits to control the strength of the penalties on the hard negative samples. The logits are the output of the model before applying the sigmoid function to convert them into probabilities. The temperature is added to the logits because it modifies the output of the model, it is equivalent to scaling the logits, so it is applied before the sigmoid function. It can also be added to the output of the model after the sigmoid function, but it would have a similar effect as multiplying the logits by temperature. The indicator function Y_{i,L_j} helps to learn more discriminative and representative feature representations and capture the underlying semantic relations between entry and exit flows by encouraging similar flows to be close together in the feature space and dissimilar flows to be far apart to make the feature representations more robust and generalizable to unseen data. This value also impacts the training process by defining the positive and negative flow embeddings, which are used to compute the contrastive loss. This loss is used to train the model to extract feature representations that are more discriminative, representative, and robust to unseen traffic flows. Without the indicator function, the model would not be able to learn discriminative feature representations, that are representative of the underlying data distribution, and learn from the similarity or dissimilarity between flows, which is the main objective of our correlation attack model. In summary, the indicator function is an essential part of the *ConFEm* loss formula, and its absence would greatly decrease the model's performance because the model would not be able to focus on the hardest negative samples. On the other hand, the temperature hyperparameter controls the strength of the penalties on the hard negative samples and the degree of similarity between semantically similar flows, which is used to balance the trade-off between uniformity and tolerance in flow correlation.

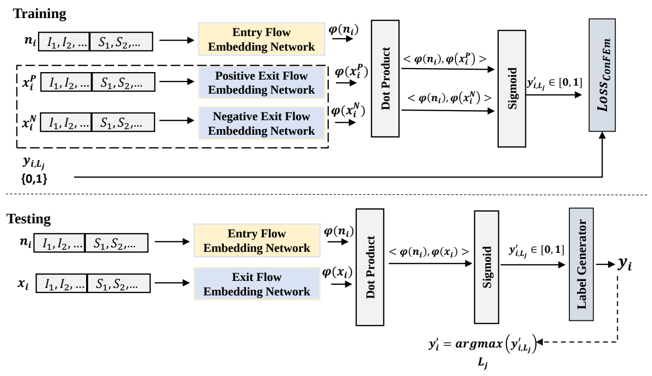
Training
n_i
$I_1, I_2, ...$
$S_1, S_2, ...$
Entry Flow Embedding Network
$\varphi(n_i)$
x_i^P
Positive Exit Flow Embedding Network
$\varphi(x_i^P)$
x_i^N
Negative Exit Flow Embedding Network
$\varphi(x_i^N)$
Dot Product
$< \varphi(n_i), \varphi(x_i^P) >$
$< \varphi(n_i), \varphi(x_i^N) >$
Sigmoid
$y'_{i,L_j} \in [0,1]$
$Loss_{ConFEm}$
y_{i,L_j} {0,1}
Testing
x_i
Exit Flow Embedding Network
$\varphi(x_i)$
$< \varphi(n_i), \varphi(x_i) >$
Label Generator
y_i
$y'_i = \underset{L_j}{argmax}\left(y'_{i,L_j}\right)$

Figure 1: Contrastive over the flow embeddings (ConFEm) with indicator function Y_{i,L_j}.

3.1.3 Extending the Loss to a Binarry Cross-Entropy Loss: Because our attack problem is a two-class classification, we can extend the Contrastive loss $Loss_{ConFEm}$ to a binary Cross-Entropy loss function $Loss_{ConFEmCross}$ as Equation 2. The output value is interpreted as a posterior probability that the entry and exit flow given as inputs correspond to a true pair.

$$Loss_{ConFEmCross} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{2}[Y_{i,L_j}.log(D[\phi(n_i), \phi(x_i^P)]) \\ + (1 - Y_{i,L_j}).log(1 - D[\phi(n_i), \phi(x_i^N)])] \quad (2)$$

The Cross-Entropy $Loss_{ConFEmCross}$ in comparison to $Loss_{ConFEm}$ can better generalize the *ConFEm* model and improve the performance by reducing the distance between embeddings for correlated flow pairs and increase the distance for uncorrelated flow pairs.

3.2 Contrastive with Flow and Label Embeddings (ConCEn)

Increasing the delay and cover traffic in the Loopix network can affect the distance between the embedded flows, causing the distribution of positive and negative exit flows to overlap and making it difficult for the model to distinguish between them. To effectively link two clients, we need to adjust the embedding spaces to ensure that correlated and uncorrelated flows do not overlap. One main problem with previous approaches [11, 19, 21] is the use of majority voting to assign labels to new samples, which involves a large number of pair-wise comparisons. Additionally, previous methods struggle with complexity in selecting representative sets of positive and negative samples and incorporating supervised learning. (*ConCEn*) model addresses these issues by using the labels themselves as the best class representatives. This eliminates the need for a large number of distance comparisons at inference time, as

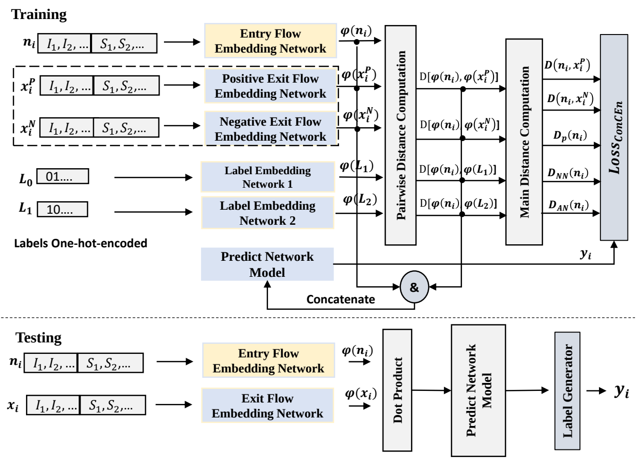
Training
n_i $I_1, I_2, \ldots$ $S_1, S_2, \ldots$
Entry Flow Embedding Network
x_i^P $I_1, I_2, \ldots$ $S_1, S_2, \ldots$
Positive Exit Flow Embedding Network
x_i^N $I_1, I_2, \ldots$ $S_1, S_2, \ldots$
Negative Exit Flow Embedding Network
L_0 01....
Label Embedding Network 1
L_1 10....
Label Embedding Network 2
Labels One-hot-encoded
Predict Network Model
$\varphi(n_i)$
$\varphi(x_i^P)$
$\varphi(x_i^N)$
$\varphi(L_1)$
$\varphi(L_2)$
Pairwise Distance Computation
$D[\varphi(n_i), \varphi(x_i^P)]$
$D[\varphi(n_i), \varphi(x_i^N)]$
$D[\varphi(n_i), \varphi(L_1)]$
$D[\varphi(n_i), \varphi(L_2)]$
Main Distance Computation
$D(n_i, x_i^P)$
$D(n_i, x_i^N)$
$D_p(n_i)$
$D_{NN}(n_i)$
$D_{AN}(n_i)$
$Loss_{ConCEn}$
y_i
&
Concatenate
Testing
n_i $I_1, I_2, \ldots$ $S_1, S_2, \ldots$
Entry Flow Embedding Network
x_i $I_1, I_2, \ldots$ $S_1, S_2, \ldots$
Exit Flow Embedding Network
$\varphi(n_i)$
$\varphi(x_i)$
Dot Product
Predict Network Model
Label Generator
y_i

Figure 2: Model framework for Contrastive with the flow and label embeddings model (*ConCEn*) with Regularization.

the new sample's label is determined by comparing it to the class representative. Furthermore, this approach simplifies the process of selecting representative sets of positive and negative samples and allows for easy integration of supervised learning, which makes our model more efficient and accurate than previous flow correlation attacks. The *ConCEn* model, shown in Figure 2, is designed based on additional useful information such as label embeddings, cluster representations, correlated labels, and a single representative for the cluster of uncorrelated flows.

3.2.1 Training and Prediction Phases: During the training phase of the *ConCEn*, the inputs are the entry flow (n_i), positive and negative exit flows, and L_j where $j \in [1, 2]$ as the probability likelihood of belonging to a corresponding training class. Label probabilities ((L_j)) are also used to train the embeddings, and a separate embedding network is created for label embeddings. The embedding networks for the flows and label embeddings ($\phi(n_i), \phi(x_i^P), \phi(x_i^N), \phi(L_j^*)$) have different input sizes but the same small output dimension. The Euclidean or Cosine distance function is then applied to calculate the distances between the entry and exit flows and label embeddings. These distances are input to the Main Distance Computation function, which computes the distance to the positive label, the distance to the nearest negative exit flow, and the average distance to all the uncorrelated flows. Finally, these distance values are used as inputs to loss functions, and the average loss for all N flow pairs is calculated. During the prediction phase, the inputs to the model are the test entry flow and exit flows. The model output is the distance between the flow embeddings and each label. The smallest distance is then chosen as the distance associated with the predicted label (More detailed information in Appendix A). Using a single forward pass to obtain the shortest distance to the test sample allows for efficient prediction, as demonstrated by experimental results. This approach can significantly reduce prediction times.

3.2.2 Contrastive Losses for ConCEn Model: We proposed three loss functions that can be used with the *ConCEn* contrastive architecture: Max-Margin (*MMConCEn*), Max-Margin with Min-Separation (*MMMS*), and Max-Separation with Min-Separation (*MSMS*). All three of these losses are based on a common architecture that includes four embedding networks. To compute the loss values for these loss functions, three distance functions are applied to the entry n_i, exit x_i, and label L_j^* flows in the embedding space. The first distance function, $D_P(n_i) = D[\phi(n_i), \phi(L_j^P)]$, calculates the distance between the n_i embedding and its positive label embedding (the likelihood of belonging to the positive class). The second function, $D_{NN}(n_i) = min(D[\phi(n_i), \phi(L_j^N)])$, calculates the distance between the n_i entry flow embedding and its nearest negative label embedding L_j^N. Finally, $D_{AN}(n_i) = \frac{1}{C-1} \sum_{j \neq i} D[\phi(n_i), \phi(L_j^N)]$ calculates the average distance between the n_i entry flow and all negative label embeddings. D_{AN} represents the average distance between the entry flow and the negative exit flows and refers to the upper bound of the distance to the centroid of the negative flows. Maximizing this distance helps to increase the distance of any other negatives. The following loss functions for the *ConCEn* model are defined to improve performance when the amount of cover traffic and delays increase. These loss functions calculate the differences between the distances of the entry and exit flows ($D[\phi(n_i), \phi(x_i^P)]$, $D[\phi(n_i), \phi(x_i^N)]$) and the representatives of the uncorrelated label embeddings (D_N, D_{NN}, D_{AN}). These loss functions are specific to the representative Contrastive architecture.

Max-Margin label based losses (MMConCEn): Max-Margin losses aim to make the distance difference greater than a margin that we can compute the loss by combining the distance of both ($D[\phi(n_i), \phi(x_i^P)], D[\phi(n_i), \phi(x_i^N)]$) and D_N, D_{NN}, D_{AN} or extend this loss with configurable weights for each distance term. Hence, the *MMConCEn* loss function is defined based on the distances of the entry and positive exit flows with the Max-Margin of the Nearest Negative label. Moreover, it has been shown through experiments that multiplying weights with the embedding distances can improve the model's generalization and detection rate. The weighted version of the $Loss_{MMConCEn}$ loss function, which is based on the weighted version of the distances for the entry and exit flows and the negative label ($w_i > 0$), is defined as:

$$
\begin{aligned}
Loss_{MMConCEn} = \frac{1}{N}\Big(&\sum_{i=1}^{N} max(w_0 D[\phi(n_i), \phi(x_i^P)] - \\
& w_1 D[\phi(n_i), \phi(x_i^N)] + 1, 0) + \sum_{i=1}^{N} max(w_2 D_P(n_i) - w_3 D_{NN}(n_i) + 1, 0) \\
&+ \sum_{i=1}^{N} max(w_4 D_P(n_i) - w_5 D_{AN}(n_i) + 1, 0)
\end{aligned}
\tag{3}
$$

Max-Margin with Min-Separation losses with label embedding (MMMS): Max-Margin with Min-Separation minimizes the distance between correlated entry and exit flows while maximizing the distance between uncorrelated exit flows and their representative labels. This loss function uses a model that calculates the distance between the flows and their labels. The Contrastive Loss with Max-Margin and Min-Separation is defined as:

$$Loss_{MMMS} = \frac{1}{N}(\sum_{i=1}^{N} D[\phi(n_i), \phi(x_i^P)] + (1 - D[\phi(n_i), \phi(x_i^N)], 0)$$
$$+ \frac{1}{N}(\sum_{i=1}^{N} D_P(n_i) + max(1 - (D_{NN}(n_i), 0))$$
$$+ \frac{1}{N}(\sum_{i=1}^{N} D_P(n_i) + max(1 - D_{AN}(n_i), 0)) \tag{4}$$

Exponential Max-Separation with Min-Separation loss (MSMS): The Max-Separation with Min-Separation loss function uses either exponential or squared exponential loss to measure the difference in distance between the representative vectors of the entry and exit flows. The exponential function (exp) increases quickly as the difference in distances grows and decreases to zero quickly for negative values. This allows the loss function to assign small values of distance between correlated flows and large values for uncorrelated flows, maximizing the separation between positive and negative flows. Moreover, we extend the Max-Separation with Exponential Loss function (*MSMSL*) to include the label embeddings of negative flows in addition to the entry/exit flow embeddings. In label embedding, the probability of belonging to a particular training class, represented by L_i, is calculated for each flow. The squared version of the exponential function (exp^2) enhances the distinction between correlated and uncorrelated flows. Therefore, we extend the $Loss_{MSMS}$ by adding the squared Exponential Loss, and the weights to calculate the optimal distances to the nearest correlated and uncorrelated flow embeddings more accurately (with weights $w_i > 0$). Hence, the distances between the entry flow and the nearest uncorrelated (negative) exit flow using the Max-Separation with Exponential Loss function are defined as:

$$Loss_{MSMS} = \frac{1}{N}\sum_{i=1}^{N}(exp(w_0 D[\phi(n_i), \phi(x_i^P)] - w_1 D[\phi(n_i), \phi(x_i^N)])^2$$
$$+ \frac{1}{N}\sum_{i=1}^{N}(exp[w_2 D_p(n_i) - (w_3 D_{NN}(n_i) + w_4 D_{AN}(n_i))])^2 \tag{5}$$

Contrastive Regularization and Predict Network: To improve the detection of traffic streams even when they are noisy and it is difficult to classify positive and negative flows, we regularized the weighted losses (Appendix C). In regularized contrastive models, the distance between the transformed entry and exit flow embeddings is calculated based on the regularization of the max-margin and max/min-separation, and a combined loss function with cross-entropy loss. The predicted network model is a neural network with a *softmax* nonlinear activation function in the last layer, which takes input flow pairs and label embeddings. The output of this predicted network (y_i) and the distances between embeddings are used as inputs for the compound losses. Regularization reduces the error rate between the expected and predicted labels by keeping the distances between correlated entry and exit flows as close as possible and separating uncorrelated flows.

3.3 Poisson Hidden Markov Model and Feature Embedding Networks

We employed two distinct feature embedding architectures for our flow and label embedding networks. For flow embeddings, we proposed contrastive models based on Poisson Hidden Markov Model (PHMM) feature embedding to identify flows, inspired by the idea of identifying conserved domains in protein families [35]. Our model belongs to the class of contrastive mixture models [17], which are capable of learning semantic information that is crucial for handling complex traffic traces that may contain poisson delays and cover. To model each flow sequence, we use Poisson Hidden Markov Models (PHMMs) [7], similar to how biological homologous genes are modeled. We learn and update the parameters of PHMMs using contrastive losses. In homological gene sequences [7], some of the genes in an organism change due to environmental factors, but their essential *functionality* can still be the same. Similarly, Mixnets alter flow distributions through the use of delay or cover traffic, yet they can still exhibit correlations with each other. By identifying these hidden relationships, we can potentially exploit them to detect correlated flows within traffic patterns. Indeed, there are certain key factors that act as functional "genes" in traffic flows, aiding us in the process of correlating different flows. For the label embedding networks, we utilized a convolution network with one hidden layer and a non-linear activation function (*ReLU*), along with normalization and pooling operations.

3.3.1 PHMM based Feature Embeddings: Addressing Inductive Bias and Improving Generalization.

To eliminate inductive bias and prevent poor generalization of embeddings, we need to address the increasing diversity of data distribution caused by the increasing number of cover traffic packets and delays. Moreover, because each flow sequence has a different flow length and the number of packets, each flow can be clustered in a big deviation for the number of states and transitions. However, instead of padding the input flows or selecting several windows manually (e.g., Deepcoffea window amplification method [21]), in MixFlow, we need to find a constant optimal value for the number of states in the PHMM graph. The optimal number of states is estimated based on the training data and the Expectation-Maximization algorithm [18]. To this end, the K-means algorithm is also adopted to obtain the best value for the number of states by selecting different state clustering and Gaussian mixture components for each observation. Then, based on the obtained results, the maximum likelihood of weights, mean vectors, and covariance matrix parameters are computed for each cluster. Finally, after determining the hidden states of PHMM, the probability of happening of each state is computed with the Forward-Backward algorithm [8].

PHMM-based Contrastive: PHMMs capture differential statistics observed in various regions and extract key elements (subsequences) in the target flow sequences to match multiple subsequences of the main flow. This method combines PHMM and contrastive learning to differentiate between main flow sequences and poisson Mixnets noises by learning a latent variable model based on statistical correlations. PHMM contrastive helps to correlate flows by handling the probable diversity of users and characterizing the customized behavior heterogeneity and multi-behavior patterns. To calculate the similarity score between entry and exit flows in a contrastive PHMM, we use an expectation-maximization algorithm. The expectation step calculates the statistical values based on an input sequence to train the probabilities of PHMMs. This process involves three steps: the forward calculation, the backward calculation, and the updating of parameters. These steps allow us to calculate the probability likelihoods in the PHMM based on all possible combinations of differences between an entry flow and a positive and negative exit flow. The pairwise similarity scores are then computed using forward and backward calculations and optimization methods to update the probabilities in the PHMM graph. Contrastive PHMM allows for more efficient comparison of flow sequences by reducing the number of comparisons needed to tune the embedding parameters and by comparing each flow sequence to a single PHMM that represents multiple sequences. It also allows us to identify the modifications that need to be made to entry flow sequences to detect and correlate changes in the entry and exit flows as they are transferred through the network. For more information on PHMM graphs and estimating their states and transitions, see Appendix B.

4 EXPERIMENTAL SETUP

We evaluate the performance of the proposed models for different cover traffic and poisson delay values in the Mixnet connections. All of the proposed models are designed considering general standards for machine learning models [13, 14] and are specifically tailored for the task of Mixnets traffic analysis. However, these models can also be applied to other network traffic analysis applications depending on the level of noise in the network. The noise level refers to the amount of random delay and the cover traffic introduced by Loopix Mixnets to make the ends of the network unlinkable. Our experimental results show that while the model performance may decrease with increasing levels of noise, delay, or cover traffic, training the models using a combination of flow and label embedding information significantly improves the attack performance. This suggests that by using simple embedding networks and leveraging the available information in the embedding space, we can improve model performance and embedding distances without adding unnecessary complexity that would increase training time.

4.1 Triplet Generator

One of the main challenges in flow correlation attacks is selecting informative positive and negative pairs to improve the embeddings. The goal is to map correlated flows of different connections to the correct cluster while avoiding the generalization of the embeddings due to the memorization of the training data. To achieve this, we use a triplet generator algorithm to select positive and negative exit flows for each anchor flow [21]. In our approach, we use semi-hard negative examples to find more informative pairs for training the embeddings. Semi-hard negative examples are those that are hard enough to contribute to the loss, but easy enough to adjust the parameters and decrease the loss to zero. To generate these examples, we divide the exit flows into two sets in each epoch and use the triplet generator to select positive samples from one set and negative samples from the other. It is important to note that if an exit flow is selected as both a positive and negative sample for creating the embedding networks, it will freeze the triplet loss at a certain value because the same sample is being used interchangeably to both maximize and minimize the distance in the triplet embedding space [21]. By using semi-hard negative examples, we can avoid this issue and train more effective embeddings.

4.2 Performance Metrics

We have used a comprehensive set of metrics to evaluate the performance of the proposed contrastive models. These metrics include accuracy, F1-score, precision, recall, and Matthews Correlation Coefficient (MCC) [16, 25]. The MCC is a correlation coefficient that ranges from -1 to 1, where a value of +1 indicates a strong correlation between the ground truth and predicted results, and a value of -1 indicates complete disagreement. In addition to these classification metrics, we also considered Normalized Mutual Information (NMI) [22] and the Silhouette coefficient to measure the quality of clustering. These clustering metrics provide insight into the classification models from a clustering perspective and measure the transformation of the original features into a low-dimensional embedding space [1]. NMI is a label-based metric that estimates how much uncertainty about class labels is reduced when the correct labels are known.

We have also measured the training and prediction execution times, as well as the number of trainable weights and Floating Point Operations (Flops) for each model. These metrics provide a comparative indicator of complexity and computational load. It is important to note that the absolute timing values will depend on the processor used, and therefore cannot be directly compared across different systems. However, they can still provide useful insight into the computational complexity and performance of the models[1].

4.3 Mixnet Parameters and Anonymity in Extended Conversations

Continuous conversations on Mixnets may compromise anonymity due to the large amount of information shared during the conversation. As more information is exchanged, it becomes easier for an attacker to potentially identify the individuals involved. To address this risk, we have conducted experiments to evaluate the performance of our proposed models in the correlation of instant chat messages transmitted through Mixnets. We have also used these models to determine the optimal Loopix Mixnets parameters that provide the best protection against flow correlation attacks. By selecting parameters that effectively obscure the patterns and characteristics of conversations, we can help to maintain anonymity in the context of extended Mixnets conversations. To identify the most effective Mixnets parameters, we evaluated the models' performance for Mixnet with different anonymity countermeasures. The collected datasets have been generated using poisson delays of $\mu \in \{0, 10, 20, 30, 40, 50\}$ seconds and cover traffic of $\lambda_C \in \{0, 10, 20, 30, 40, 50, 60\}$ packets per 28 minutes (each packet is 500 bytes). The conversation length (payload messages) is set to 20 packets per minute, with the message rate as the mean and half of that as the standard deviation [24]. These ranges have been chosen based on the findings of the Loopix paper [24]. We assume that clients and mix servers continuously generate a flow of real messages (λ_M) with cover traffic (λ_C) injected into the network. The parameter μ represents random poisson delays drawn from an exponential distribution. The average poisson delays ($\frac{1}{\mu}$) is tuned such that decreasing μ increases the average poisson delay [23, 24]. We assume that Mixnet generates messages λ_M from the Poisson distribution to simulate clients with various sending patterns and different message rates, which is more realistic. The Poisson distribution is commonly used in computer networks and telecommunications research to model situations where arrivals come from a large number of independent sources, as is the case in Mixnets with many clients and nodes [2].

4.3.1 Training and Testing Datasets: To ensure that the results of our experiment are realistic and not overly optimistic, it is important to strictly separate the dataset into training, validation, and testing sets prior to generating a learning model [1]. For our experiment, we have collected flow data in a situation where one client is communicating with another client, and we have treated this data as correlated for the training set. We have also assumed that each client is simultaneously communicating with two other clients for the validation and testing data, which consists of flows that are mixed with other conversations in addition to the main connection. This separation of the dataset allows us to accurately assess the performance of the model when it is presented with new, unseen data and helps to prevent temporal snooping, which refers to the use of information from the future to inform the present. By carefully separating the dataset in this way, we can ensure that our experiment and conclusions are reliable and not overly optimistic. In addition, this process allows us to identify useful features, parameters, and learning algorithms, which is essential for obtaining accurate results. For more information on the dataset, refer to Appendix D.

4.3.2 Network Simulation: Due to constraints in our experimental setup, we were unable to collect real Mixnets traffic data to test the proposed models. To compensate for this, we simulated Loopix Mixnets with a stratified topology using the Poisson mixing technique as a continuous-time mixing variant [12, 23, 24]. The Loopix network is implemented on m4.4xlarge instances of EC2 Ubuntu, which are powered by 2.3 GHz machines with 64 GB of RAM memory. We generated 60, 000 flow pairs and extracted metadata such as Inter packet delay (IPD), size, and direction information for the entry and exit flows. To evaluate the performance of the proposed models, we varied cover traffic and poisson delays to estimate the anonymity provided by Loopix under different network countermeasures.

4.3.3 Assumptions: Real-world adversaries may target multiple communications and monitor many clients, but our research simplifies this scenario due to lab limitations and makes the following assumptions:
• We used one-on-one communications for training and one-to-two for testing. Simulating instant text messaging on Loopix, we generated random content matching size and frequency statistics of chat messaging with a volume of 3.85MB and a size range of 1B-4095B.
• We used version 22.10.6 of Signal-cli [28] to write Python programs for sending and receiving generated one-on-one traces in our simulation. We modeled text messaging with a Pareto Type I distribution (scale: 5000ms, shape: 0.93) resulting in a total of 458 hours of communication traces, including 34477 messages for one-to-one and 283329 messages for one-to-two communications [3].
• Our research limitations led us to simulate Signal's text chats to model chat traffic. However, similar traffic patterns are observed in other messaging apps such as WhatsApp and Telegram [3]. Therefore, the results of our research can be applied to these other messaging apps. Further research on evaluating the attack performance on different types of messages like photos, videos, audio, or files is planned but beyond the scope of this current paper.
• To evaluate the impact of different cover traffic and poisson delay values, we assumed no additional network jitter on internet connections and users are always online.

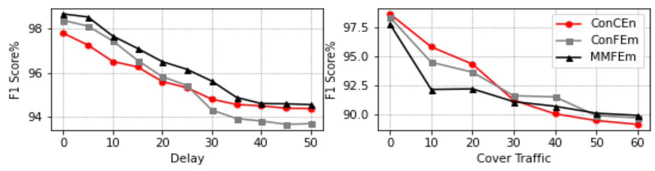

Figure 3: Comparing the *MMFEm*, *ConFEm*, and *ConCEn* models performance for different amounts of delay μ (left plot) and cover traffic λ_C (right plot).

5 EXPERIMENTAL RESULTS

In this section, we evaluate the impact of different Mixnet countermeasures on the performance of MixFlow and compare the effectiveness and efficiency of our proposed models for correlating end-to-end connections.

5.1 Delay Countermeasure

The poisson mixing delay is a crucial factor in improving the anonymity of Loopix Mixnets. To understand the relationship between flow correlation attack performance and different poisson delays, we evaluated the performance of three proposed models by recording traffic flows while varying the delay value from 1 to 50 seconds with cover traffic of $\lambda_C = 0$. Our results showed that the max-margin contrastive model (*MMFEm*) had the best performance when the poisson delay was increased, followed by contrastive models with cross-entropy (*ConCEn*) and contrastive flow embeddings (*ConFEm*). The use of a max-margin loss function in the *MMFEm* model may have contributed to its improved performance, and flow embeddings may be able to capture more complex patterns in the data. The minimum accuracy for all three models was almost reached at $\mu = 0.02$, corresponding to a delay of 50 seconds. This suggests that the $\mu = 0.02$ parameter can be considered one of the best levels of anonymity among the tested delays. Therefore, we chose this delay value as the fixed parameter to evaluate other poisson countermeasures in subsequent experiments.

5.2 Cover Traffic Countermeasure

Cover traffic injection is a technique that adds additional, unrelated traffic to a network to obscure the activity of a particular user or group. In this study, we tested cover traffic as a countermeasure against Loopix Mixnets by using a range of λ_C values from 0 to 60 with $\mu = 0$. The maximum value of $\lambda_C = 60$ is selected based on previous experiments [24]. Figure 3 shows the impact of increasing cover traffic $\lambda_C = [0, 60]$ on the classification performance of our models. Comparing two plots in Figure 3, it is clear that the average performance of the models when only considering cover traffic values is lower than the average performance when poisson delay is injected. This indicates that cover traffic may have a greater impact on anonymity than poisson delay in some scenarios. Additionally, the *MMFEm* and *ConFEm* models experience a larger drop in classification and clustering scores as the cover traffic rate increases. While the accuracy of the *ConCEn* model also decreases with increasing cover traffic, its performance has a smaller decline compared to the other models.

Our findings for cover traffic values from Figure 3 suggest that max-margin contrastive models are more effective at distinguishing traffic mixed with poisson delays. It appears that the *ConFEm* and *ConCEn* contrastive models exhibit better generalization performance when compared to the classification and clustering scores, as they demonstrate an increase in cover traffic. This suggests that more complex embeddings are necessary for accurately correlating flows that are mixed with cover traffic. As a result, the *ConFEm* and *ConCEn* contrastive models exhibit greater stability with increased cover traffic and have the potential to be optimized to minimize loss at higher levels of anonymity.

5.3 Embedding Distance and Complex Data

Settings: To evaluate the performance of the MixFlow contrastive models under varying levels of cover traffic intensity, we conducted experiments with cover traffic range $\lambda_C = [0, 60]$ packets per minute and a fixed poisson delay of $\mu = 0.02$ seconds drawn from an exponential distribution. These specific values were selected based on their ability to significantly impact the model's performance in our experiments.

Results: The results of this evaluation are shown in Figure 4, which compare the models using classification metrics and the distance between embeddings calculated with Cosine and Euclidean distances. These results are the average of five runs of the experiment. Increased cover traffic in Mixnets can negatively affect the generalization ability of the model. However, properly training the embeddings with Cosine similarity distance can improve the model's ability to identify correlated entry and exit flows. When comparing the performance metrics of the proposed models under different network countermeasures, we find that increasing cover traffic and poisson delay to its highest can lead to an increase in the false positive rate of the correlation model. However, *ConFEm* and *ConCEn* contrastive models with Cosine similarity distance show greater robustness compared to the max-margin contrastive model.

5.4 Feature Embedding Networks (FEN)

Settings: To evaluate the impact of using the proposed *PHMM*-based feature embedding network in contrastive flow correlation, we compare PHMM-based models with the same models where we used convolutional neural networks (*CNNs*) for feature embeddings. For CNN flow embeddings, we employed the feature embedding network used in Deepcoffea [21], as we found it to be particularly effective for identifying correlation attacks. The utilized CNN flow embedding network consists of four 1D convolution blocks with 1D convolution layers, a max pooling layer, and a linear activation for the output layer. All three flow embedding networks have the same configurations, and we used a batch normalization layer with a kernel size of 8 after each convolutional layer to capture local feature patterns. For models with added Cross Entropy, the classification network has two hidden layers with ReLU activation and a softmax activation for the output layer. All of the CNN embedding models are neural networks trained with gradient descent, using a batch size of 100 and 100 epochs, with early stopping and a waiting period of 50 epochs. We used the Adam optimizer with its default parameters.

Results: As shown in Figure 5, the use of the *PHMM* feature embedding network leads to improved model performance, particularly when the cover traffic increases. The *PHMM*-based models allow us to identify functional traffic patterns even when there are few packets per flow and the main traffic has been hidden by poisson mixing and cover traffic. The best overall performance results are achieved using the Cosine similarity distance for all models, indicating that this distance is particularly effective at correlating mixing flows and outperforming the Euclidean distance. The *ConCEn* model in Figure 5 has the highest performance and clustering metrics (as measured by the NMI metric). We found that F_1 and MCC are the two metrics that best capture the overall performance and improvement achieved using contrastive models for flow correlation. As cover traffic increases with network delay $\mu = 0.02$, the performance of the contrastive models decreases, but the models based on the *PHMM* feature embedding and Cosine similarity distance still achieve the best overall performance results, even when $\lambda_C = 60$ and $\mu = 0.02$. These results suggest that cover traffic injection can effectively reduce the linkability of entry and exit flows, but a higher amount of cover traffic leads to increased overall transaction delay while flows can still be linked with a well-designed attack model. In conclusion, incorporating feature embedding and the distances between them, along with label embeddings, can improve the performance of contrastive models as a replacement for original flows.

5.5 Contrastive Embedding Losses

The performance of the proposed models is evaluated in Figure 6, using various loss functions and different $\lambda = [0, 60]$ values, with a fixed value of $\mu = 0.02$ and feature embedding network $FEN = PHMM$. We can observe that using the cross-entropy $Loss_{ConFEmCross}$ and $Loss_{MSMS}$ losses significantly improves the generalization of the *ConFEm* and *ConCEm* models based on the Cosine similarity distance function and *PHMM* feature embedding network. The weighted $Loss_{MMConCEn}$ and $Loss_{MSMS}$ for the *ConCEn* model also resulted in a notable increase in the detection rate for all models. The *ConCEn* model with the $Loss_{MSMS}$ achieved an accuracy of 90% compared to other models, even when the cover traffic rate and delay were increased. To further improve the results, we regularized the cross-entropy based $Loss_{MMConCEn}$ and $Loss_{MSMS}$, resulting in the regularized losses $Loss_{RMMConCEn}$ and $Loss_{RMSMS}$. This regularization technique prunes filters by weight and significantly

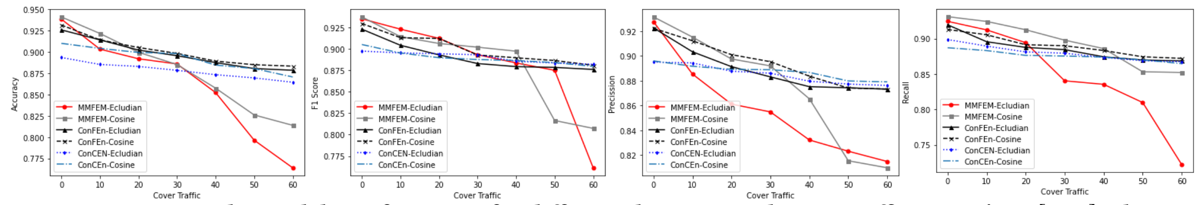

Figure 4: Comparing the models performance for different distances and cover traffic range $\lambda_C = [0, 60]$ where $\mu = 0.02$.

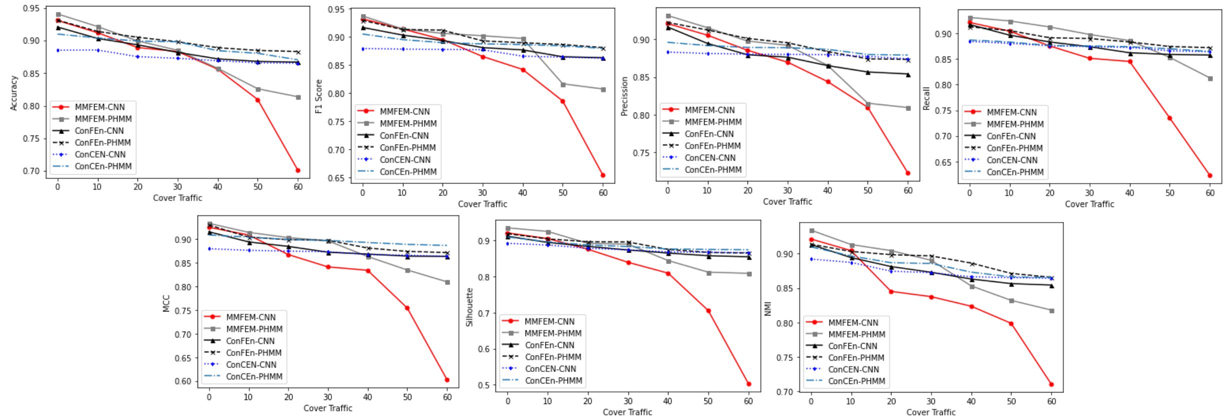

Figure 5: A comparison of performance metrics of the proposed models for different feature embedding networks (FEN) and cover traffic range $\lambda_C = [0, 60]$ where delay $\mu = 0.02$.

improves accuracy. By combining the *ConCEn* model with the regularized weighted losses and the *PHMM* feature embedding model, we have achieved a significant improvement in the detection rate using flow and label embeddings.

5.6 Mixnet Anonymity and the Length of Conversations

The length of the conversations is one of the parameters that impact the Mixflow performance on Loopix. Longer conversations increase the risk that the anonymity of Loopix will be compromised, making it easier for attackers to identify messages and disrupt the system. To determine the optimal conversation length for maximum anonymity, we conducted experiments with six different ranges of conversation lengths using the *ConCEn* model and the $Loss_{RWMSMS}$ metric, setting $\mu = 0.02$ and $\lambda_C = 60$ packets generated by the poisson distribution. We calculated the average number of samples that could be correctly correlated in each case and ran the model 10 times. Our results allowed us to estimate the threshold value for conversation length and anonymity level, which can help us improve Loopix's security and protect against attacks. As shown in Figures 7 and 8, if the length of a conversation exceeds 20 to 30 packets per minute, Loopix Mixnets is unable to provide sufficient anonymity protection even with the highest delay and cover traffic. This is because a longer conversation allows the attacker to access more traffic data, making it easier to correlate flows and reduce anonymity. Therefore, it is important to carefully control the length of conversations in Loopix to maintain a high level of security.

5.7 MixFlow Significantly Outperforms the State-Of-The-Art

Figures 9 and 10 compare the performance of MixFlow models with previous flow correlation algorithms, including DeepCoffea and FlowTracker [11, 19, 21], for various values of poisson delay and cover traffic. While existing flow correlation attacks have made some progress, there are still challenges in analyzing the correlation of instant messaging traffic flows in Mixnets, which have fewer packets compared to website traffic traces and more complex data distribution. To ensure a fair comparison, we optimized the parameters of DeepCoffea, FlowTracker, and Deepcorr for the best performance on our traffic flows. For DeepCorr and FlowTracker, we found the optimal feature dimension (number of packets) to be 100 and the optimal number of training flow pairs. We also analyzed different window partitioning parameters for DeepCoffea and determined the best value to be three windows with 1 second time overlap. When evaluating the performance of previous attacks [11, 19, 21] using F_1 and *MCC* as metrics, we found that MixFlow improves the detection rate by about 90% for $\lambda_C = 60$ and $\mu = 0.02$, while the maximum detection rate of DeepCoffea, FlowTracker, and Deepcorr was less than 81%. We also observed that Deepcoffea's windowing and amplification indicate the smallest impact on improving the detection rate for instant messaging traffic analysis. This may be because the window partitioning method is designed for large historical time series and is classified as a Long-range Dependence (LRD) approach, which is not suitable for predicting instant messaging traffic due to the lack of periodic behavior in the traffic baseline. In contrast, the *MMFEm*, *ConFEm*,

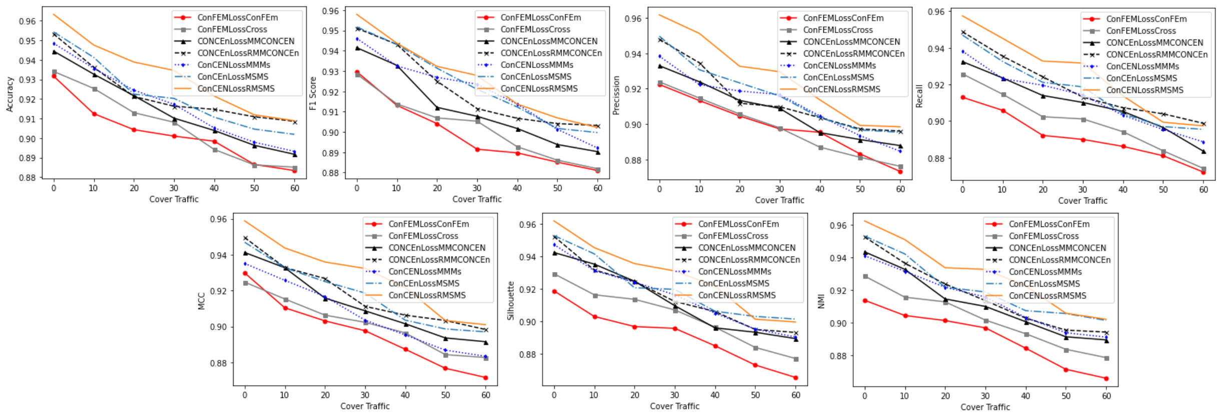

Figure 6: Comparing the *ConCEn* model performance for different losses and cover traffic range $\lambda_C = [0, 60]$ where $\mu = 0.02$.

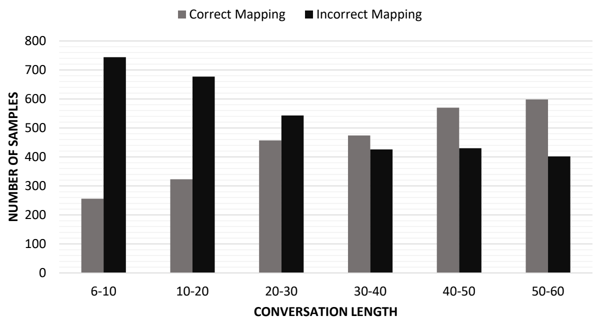

Figure 7: Impact of the conversation length on the detecting flow pairs correctly based on the *ConCEn* model with $Loss_{MSMS}$ where $\lambda_C = 60$ and $\mu = 0.02$.

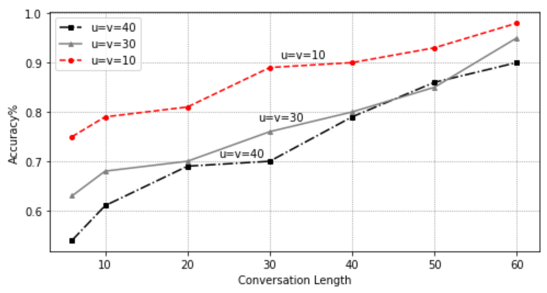

Figure 8: Relation of conversation length and the attack detection rate for different value of mean (u) and variance (v) of the Mixnet poisson distribution. The *ConCEn* model with $Loss_{MSMS}$ is used where $\lambda_C = 60$ and $\mu = 0.02$.

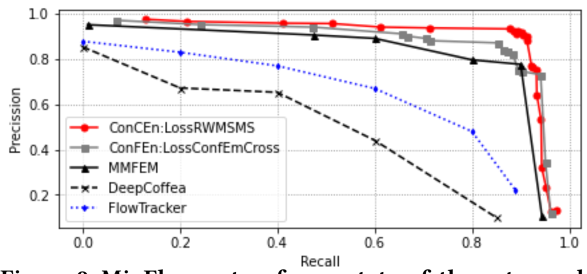

Figure 9: MixFlow outperforms state-of-the-art correlation attacks where poisson delays $\mu = 0.02$ and cover traffic $\lambda = 60$.

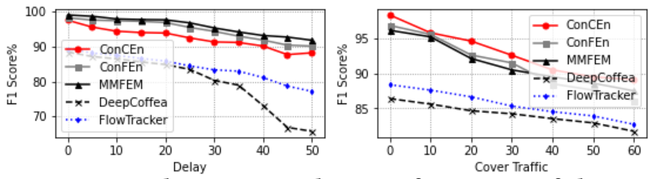

Figure 10: MixFlow consistently outperforms state-of-the-art correlation attacks across a wide range of Mixnets' poisson delays and types of cover traffic.

and *ConCEn* contrastive models in MixFlow showed a significant increase in detection rate and reduction in execution times compared to DeepCoffea.

5.8 MixFlow's Computational Complexity

To evaluate the computational complexity of the proposed models, we considered four metrics: training and testing time, the number of trainable weights, and the number of floating-point operations (Flops) required by each model. In our models, the number of Flops is approximately twice the number of trainable weights, consistent with the relationship between Flops and matrix/vector dimensions. We also found that most of the Flops required by each model are determined by its number of weights, rather than by the operations needed to calculate the loss function. This suggests that the number of weights plays a key role in determining the computational complexity of a model. According to Figure 11, the *ConCEn* model has the highest performance and clustering metrics (as measured by the *NMI* metric), while the *MMFEm* model has the fastest training/prediction time, due to its fewer comparisons compared to the other models. On the other hand, according to Figure 11, the *ConFEm* and *ConCEn* models have large Flops numbers and training/prediction times based on cross-entropy with contrastive regularization. The *MMFEm* contrastive model even with the *PHMM* flow embedding network shows extremely low Flops requirements and prediction times than *ConFEm* and *ConCEn* models due to its architecture, which requires fewer comparisons

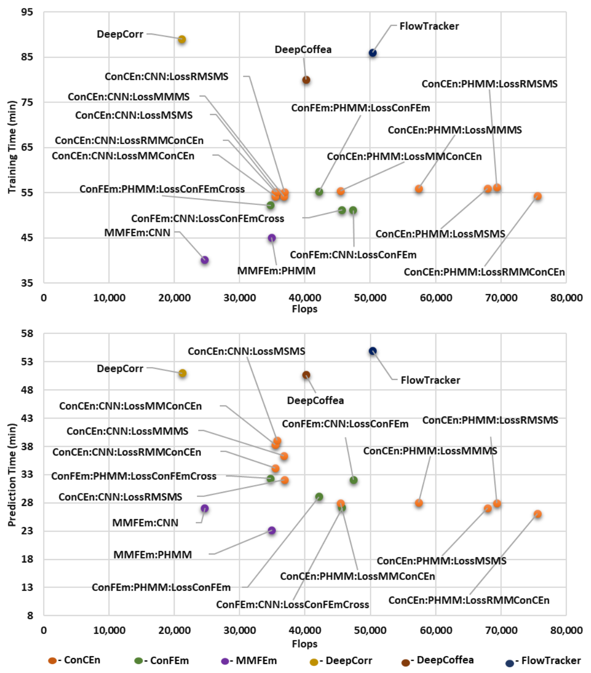

Figure 11: Model complexities for MixFlow models and state-of-the-art attack models in terms of Flops and Training/Prediction Time. Each model is identified by the model name, FEN, and Losses.

to find the closest class to the input sample during prediction. It seems that contrastive models based on PHMM, although requiring more training time, outperform CNN-based contrastive models.

6 DISCUSSION

In this study, we presented two contrastive flow correlation models and demonstrated their ability to accurately identify the entry and exit flows, even after they have been subjected to transformations such as poisson mixing, cover traffic, and delays. Our findings indicate that contrastive models using flow/label embeddings can effectively reduce false positive rates and distinguish between obfuscated, correlated, and uncorrelated flows. Furthermore, we demonstrated that feature embedding networks based on the *PHMM* can accurately extract functional flow information and correlate flows. Our results show that by adjusting contrastive hyperparameters and using flow and label information in the embedding space, we can uniquely correlate anonymized flows in Mixnets, and by developing strong classification models, we can achieve high attack performance.

This suggests that poisson mixing alone may not be sufficient to obscure entry flows, and the message rate of at least $\lambda_M = 20$ packets per minute allows us to achieve the detection rate of over 90% for cover traffic rates of $\lambda_C = 60$ and delay $\mu = 0.02$. Conversation length should also be taken into account as a countermeasure in the design of Mixnets anonymous networks. The results of our experiments show that when the conversation length is significantly lower than the 20 to 30 messages, the probability of correctly correlating flows decreases significantly, where the poisson delay is set to $\mu = 0.02$. While it is challenging to accurately determine the precise parameter values, we do not recommend reducing the rate of cover traffic in relation to real communication traffic as the volume of real traffic increases. Further study of the MixFlow architecture in other Mixing techniques could lead to improved defense strategies against flow correlation attacks. Comparing the effectiveness of various mixing techniques using a more realistic dataset could enable the development of a stronger defense for mixing networks.

7 RELATED WORK

End-to-end flow correlation attacks aim to correlate flows at both ends of the connections in anonymous networks. Researchers have studied the feasibility of these attacks based on the different detection techniques [3, 4, 15, 19, 21, 30, 31]. Y. Zhu et al. [34] studied a passive correlation attack against batching Mixnets and analyzed different transformation models for flow pattern vector extraction. Danezis et al. [5] examined the trade-offs between cover traffic and extra latency in continuous-time Mixnets. The research on end-to-end flow correlation on Tor focuses on how routing dynamics and potential manipulation of the network could allow an adversary to correlate flows [3, 15, 19, 21, 30]. Nasr et al. [20] proposed a compressive traffic analysis for improving flow correlation techniques using compressed sensing, which allows for efficient representation of signals using fewer samples than normally required [10, 33]. DeepCorr [19] is a deep-learning method for correlating website traffic on the Tor network. Sirinam et al. [30] introduced Triplet Fingerprinting, a method for website fingerprinting using compressive representation learning to improve on DeepCorr's performance. Contrastive models, such as those used in Triplet Fingerprinting, map inputs to low-dimensional feature embeddings instead of directly comparing high-dimensional vectors [26]. Triplet Fingerprinting [30] is a method for identifying website fingerprinting attacks by using the cosine similarity of embeddings of website traces to compute the triplet loss. It employs the Deep Fingerprinting neural network model as the feature embedding network and trains the k-nearest neighbors (k-NN) classifier using the acquired website embeddings [29]. Oh et al. [21] proposed DeepCoffea, a flow correlation model based on contrastive learning that aims to minimize the gap between the distances of correlated Tor entry and exit flows. FlowTracker [11], another approach for improving flow correlation attacks on Tor, uses contrastive learning and cumulative representation with stacked autoencoders at the time-window level to optimize distance metrics. Both DeepCoffea [21] and FlowTracker [11] were designed for flow correlation of websites, unlike Triplet Fingerprinting which focuses on website fingerprinting [30]. A key difference between DeepCoffea and Triplet fingerprinting is their data representation [21, 30]. DeepCoffea uses separate embeddings for entry and exit flows on the Tor network, while Triplet fingerprinting uses a single unified embedding for website traces.

8 CONCLUSION AND FUTURE WORK

In this study, we investigate the vulnerability of Loopix Mix networks to flow correlation attacks that aims to identify similarities between the entry and exit flows. We introduce MixFlow as a model of the adversary and define detection rate as a measure of attack performance. Our analysis reveals the underlying principle of flow-correlation attacks and presents the first quantitative analysis of the relationship between Loopix Mixnet parameters such as conversation length, poisson delay, cover traffic, and detection rate. Our findings indicate that flow-correlation attacks can significantly compromise the anonymity of Mixnets by revealing correlations between flows into and out of the network. Our results also suggest that anonymous Mix networks may not adequately obscure metadata to make the entry and exit flows indistinguishable, even when subject to random delays or cover traffic. These results provide valuable insights for the design of anonymous networks that require additional protection against flow-correlation attacks. Our findings suggest areas for future research, such as the attack's effectiveness on more realistic Mixnet traffic patterns and traffic analysis attacks using PHMM graphs and generative adversarial networks.

9 CREDIT AUTHORSHIP CONTRIBUTION STATEMENT

Reyhane Attarian: Conception and design of the study, data collection, implementation, and experiments, Writing the original draft. **Esfandiar Mohammadi:** Review and supervision of the original draft. **Tao Wang:** Supervision in data collection. **Emad Heydari Beni:** Reviewing the paper.

10 ACKNOWLEDGMENTS

We thank Debajyoti Das for valuable discussions, particularly about the Mixnet countermeasures and network settings. We also thank the anonymous reviewers for their helpful feedback that improved this work. Reyhane Attarian is funded by imec-COSIC KU Leuven. This research is partially supported by the Research Council KU Leuven by CyberSecurity Research Flanders. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of any of the funders.

REFERENCES

[1] Daniel Arp, Erwin Quiring, Feargus Pendlebury, Alexander Warnecke, Fabio Pierazzi, Christian Wressnegger, Lorenzo Cavallaro, and Konrad Rieck. 2022. Dos and don'ts of machine learning in computer security. In *Proc. of the USENIX Security Symposium*.

[2] Alireza Bahramali, Ramin Soltani, Amir Houmansadr, Dennis Goeckel, and Don Towsley. 2020. Practical traffic analysis attacks on secure messaging applications. *arXiv preprint arXiv:2005.00508* (2020).

[3] Ardavan Bozorgi, Alireza Bahramali, Fateme Rezaei, Amirhossein Ghafari, Amir Houmansadr, Ramin Soltani, Dennis Goeckel, and Don Towsley. 2022. I Still Know What You Did Last Summer: Inferring Sensitive User Activities on Messaging Applications Through Traffic Analysis. *IEEE Transactions on Dependable and Secure Computing* (2022).

[4] Chen Chen, Daniele E Asoni, Adrian Perrig, David Barrera, George Danezis, and Carmela Troncoso. 2018. TARANET: Traffic-analysis resistant anonymity at the network layer. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 137–152.

[5] George Danezis. 2004. The traffic analysis of continuous-time mixes. In *International Workshop on Privacy Enhancing Technologies*. Springer, 35–50.

[6] Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)* 39, 1 (1977), 1–22.

[7] Sean R. Eddy. 1998. Profile hidden Markov models. *Bioinformatics (Oxford, England)* 14, 9 (1998), 755–763.

[8] John L. Fan. 2001. *Forward-Backward Algorithm*. Springer US, Boston, MA, 97–116. https://doi.org/10.1007/978-1-4615-1525-8_3

[9] Can Firtina, Kamlesh Pillai, Gurpreet S Kalsi, Bharathwaj Suresh, Damla Senol Cali, Jeremie Kim, Taha Shahroodi, Meryem Banu Cavlak, Joel Lindegger, Mohammed Alser, et al. 2022. ApHMM: Accelerating profile hidden markov models for fast and energy-efficient genome analysis. *arXiv preprint arXiv:2207.09765* (2022).

[10] Massimo Fornasier and Holger Rauhut. 2015. Compressive Sensing. *Handbook of mathematical methods in imaging* 1 (2015), 187–229.

[11] Zhong Guan, Chang Liu, Gang Xiong, Zhen Li, and Gaopeng Gou. 2022. Flow-Tracker: Improved flow correlation attacks with denoising and contrastive learning. *Computers & Security* (2022), 103018.

[12] Dogan Kesdogan, Jan Egner, and Roland Büschkes. 1998. Stop-and-go-mixes providing probabilistic anonymity in an open system. In *International Workshop on Information Hiding*. Springer, 83–98.

[13] Dominik Kreuzberger, Niklas Kühl, and Sebastian Hirschl. 2022. Machine Learning Operations (MLOps): Overview, Definition, and Architecture. *arXiv preprint arXiv:2205.02302* (2022).

[14] Indika Kumara, Rowan Arts, Dario Di Nucci, Willem Jan Van Den Heuvel, and Damian Andrew Tamburri. 2022. Requirements and Reference Architecture for MLOps: Insights from Industry. (2022).

[15] Phuc H Le-Khac, Graham Healy, and Alan F Smeaton. 2020. Contrastive representation learning: A framework and review. *IEEE Access* 8 (2020), 193907–193934.

[16] Christopher D Manning. 2008. *Introduction to information retrieval*. Syngress Publishing,.

[17] Geoffrey J McLachlan, Sharon X Lee, and Suren I Rathnayake. 2019. Finite mixture models. *Annual review of statistics and its application* 6 (2019), 355–378.

[18] Todd K Moon. 1996. The expectation-maximization algorithm. *IEEE Signal processing magazine* 13, 6 (1996), 47–60.

[19] Milad Nasr, Alireza Bahramali, and Amir Houmansadr. 2018. Deepcorr: Strong flow correlation attacks on tor using deep learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 1962–1976.

[20] Milad Nasr, Amir Houmansadr, and Arya Mazumdar. 2017. Compressive traffic analysis: A new paradigm for scalable traffic analysis. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2053–2069.

[21] Se Eun Oh, Taiji Yang, Nate Mathews, James K Holland, Mohammad Saidur Rahman, Nicholas Hopper, and Matthew Wright. 2022. DeepCoFFEA: Improved flow correlation attacks on Tor via metric learning and amplification. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1915–1932.

[22] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. 2016. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4004–4012.

[23] Ania M Piotrowska. 2021. Studying the anonymity trilemma with a discrete-event mix network simulator. In *Proceedings of the 20th Workshop on Workshop on Privacy in the Electronic Society*. 39–44.

[24] Ania M Piotrowska, Jamie Hayes, Tariq Elahi, Sebastian Meiser, and George Danezis. 2017. The loopix anonymity system. In *26th USENIX Security Symposium (USENIX Security 17)*. 1199–1216.

[25] David MW Powers. 2020. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061* (2020).

[26] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 815–823.

[27] Vitaly Shmatikov and Ming-Hsiu Wang. 2006. Timing analysis in low-latency mix networks: Attacks and defenses. In *European Symposium on Research in Computer Security*. Springer, 18–33.

[28] Signalcli. 2021. An unofficial commandline and dbus interface for signalapp and libsignal-service-java. https://github.com/AsamK/signal-cli.

[29] Payap Sirinam, Mohsen Imani, Marc Juarez, and Matthew Wright. 2018. Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 1928–1943.

[30] Payap Sirinam, Nate Mathews, Mohammad Saidur Rahman, and Matthew Wright. 2019. Triplet fingerprinting: More practical and portable website fingerprinting with n-shot learning. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 1131–1148.

[31] Yixin Sun, Anne Edmundson, Laurent Vanbever, Oscar Li, Jennifer Rexford, Mung Chiang, and Prateek Mittal. 2015. {RAPTOR}: Routing attacks on privacy in tor. In *24th USENIX Security Symposium (USENIX Security 15)*. 271–286.

[32] Xinyuan Wang, Shiping Chen, and Sushil Jajodia. 2007. Network flow watermarking attack on low-latency anonymous communication systems. In *2007 IEEE Symposium on Security and Privacy (SP'07)*. IEEE, 116–130.

[33] Allen Y Yang, Zihan Zhou, Yi Ma, and S Shankar Sastry. 2010. Towards a robust face recognition system using compressive sensing. In *Eleventh Annual Conference of the International Speech Communication Association*.

[34] Ye Zhu, Xinwen Fu, Bryan Graham, Riccardo Bettati, and Wei Zhao. 2004. On flow correlation attacks and countermeasures in mix networks. In *International Workshop on Privacy Enhancing Technologies*. Springer, 207–225.

[35] Zhongliu Zhuo, Yang Zhang, Zhi-li Zhang, Xiaosong Zhang, and Jingzhong Zhang. 2017. Website fingerprinting attack on anonymity networks based on profile hidden markov model. *IEEE Transactions on Information Forensics and Security* 13, 5 (2017), 1081–1095.

A CONTRASTIVE END TO END FLOW CORRELATION ATTACK

We supposed a global passive adversary who can observe all network traffic between users and providers in the Loopix network (e.g., large intelligence agencies) [24]. Consider a binary classification problem with the data generating distribution P_{XY} being a mixture of timestamp and packet size information. In this problem, the label Y can be either positive $(+1)$ or negative (-1) with an equal probability. As a simple example, in flow correlation, the label is +1 given entry and exit flows have similarity $(X|Y = +1\ N(\mu_1, \sigma^2))$, and -1 if the flows are not correlated $(X|Y = -1\ N(\mu_2, \sigma^2))$. In this regard, the optimal Bayes's classifier can be defined as $f(x) = sign(x - \frac{\mu_1+\mu_2}{2})$ and x is classified as positive if $x > \frac{\mu_1+\mu_2}{2}$. In this regard, we need to estimate the capability of a model to learn the $\frac{\mu_1+\mu_2}{2}$ as a proxy for the model performance.

The proposed models for the approach/separation process in embedding space between similar and dissimilar samples can be categorized into two contrastive models: Sample-wise contrastive models and contrastive ones based on flow and label embeddings. Sample-wise contrastive models employ the distances between flow embeddings without using labels. Indeed, each new element is classified based on the majority class of its K nearest elements, but the distance comparison is not based on the complete population as in the K-Nearest Neighbor model. In this case, a selection of samples from the entry and exit flows is made by random sampling of correlated and uncorrelated flow pairs. However, this solution has an implicit problem because random sampling cannot create a single representative that serves as a prototype for any correlated entry and exit flows. In the distances based on the representative of entry and exit flows in embedding space, the main goal is to reduce the distance between an entry flow (anchor) to the representative of positive exit flows and increase the distance to the representative of negative samples. Moreover, each sample pushes to be as close as possible to correlated label embedding while separating itself from uncorrelated label embeddings.

B PHMM FEATURE EMBEDDING NETWORKS

Like traditional procedures, individual flow connections are generated and preprocessed to get the corresponding packet sequences. An accurate description of traffic shape using proper original features is crucial for flow correlation attacks. In these attacks, due to cover traffic injection, delay, packet loss, reordering, and reassembly, the number of sent packets rarely equals the number of received packets, making it difficult to preserve the original one-to-one packet relationship. As a result, matching entry and exit flow sequences may not correspond to the same packet. Packet attributes, such as size or interval time, also have finite value ranges, making it insufficient to record flow characteristics using packet sequences alone. Aside from typical network noise, Loopix's poisson mixing noises can also generate obfuscation noise by concealing traffic patterns at both ends of the network. When the input flow

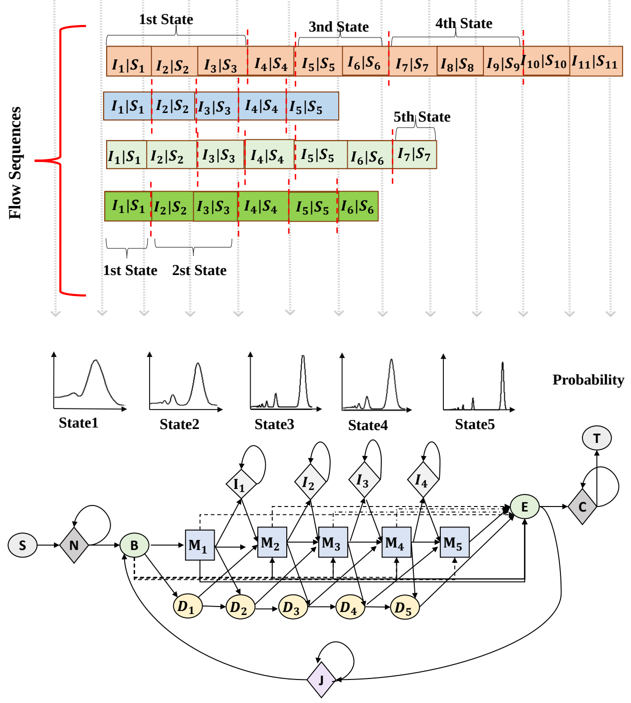

Figure 12: PHMM model for flow sequences. Each flow is divided into five cluster and the parameter values for each state is computed based on each cluster. M_i indicates the matching, I_i insertions, and D_i deletions based on the inter-packet delay and packet size metadata information.

sequences with any length are fed to PHMM, the model tries to learn the key information and capture meaningful signals. In this regard, for each gap relative to the flow sequence, such as a deletion and an insertion, the path will pass through one or more deletion or insertion states before it reaches the end state. The insertion and deletion state properties allow us to have the tolerance to match the subsequences in any network conditions or poisson mixing such as packet dropping, delays, retransmission, or injecting cover traffic packets.

B.1 State Transition Diagram for Contrastive PHMM

Figure 12 shows a schema of a state transition diagram of our PHMM model. In the central chain of PHMM, states are referred to as Match states which means the probability distribution of the main flow sequence. For each state in the main chain, two additional states are considered for variations in exit flows after leaving the mix network: Insert and Delete states. Each flow sequence is supposed to be as a path in a linear fashion $Begin \rightarrow M_1 \rightarrow \ldots \rightarrow M_n \rightarrow End$ that includes match state (M), delete state (D), insert state (I), start state (S), terminal state (T), and states (N, C, J). States N, C, and J generate a random sequence that is not aligned with the main sequence as unpredictable network jitters or delays in Mixnets. Additional states of N, C, and J allow us to counter unpredictable background and dummy noises in the traffic traces and find multiple key elements in the network traffic, regardless of the impact of the delay or dummy packets on the traffic patterns. Insertion states allow for one or more extra packets *inserted* in between two matching states of the chain. The delete state means omitting some matching states (key information) from the sequence. In the context of flow sequences, the *Insert* states can define dummy packets, duplicate packets, and retransmissions, while the Delete states model random delay or packets lost in the network.

In this regard, each PHMM model is defined based on a series of hidden states $Q_i = \{q_1, q_2, ..., q_N\}$ and observation sequences $O_i = \{o_0, o_1, ..., o_L\}$. Each observation o_l is equal to the logarithm of Inter packet delay between the l^{th} packet and $(l+1)^{th}$ packet and packet size S_l of the l^{th} packet ($o_l = \{log(I_l), S_l\}^T$). For each state, two variables of inter-packet delay I_l and packet size S_l are correlated with a mixing coefficient. For each PHMM model $\lambda' = (\epsilon, A, w, m, \phi)$, we define five parameters [7]: ϵ that means the primary probability for hidden stats; A is the state transition matrix and its elements a_{ij} indicate the probability of transition from a state i to state j; Parameter w is the vector of weights as the mixture coefficient that their values are computed based on the Gaussian distribution; m is the mean vector; ϕ defines the covariance matrix of Gaussian distribution.

B.2 Estimating the Number of States in Each Embedding Network

Each flow sequence is quantized on a not linear scale separately, using the k-means algorithm [7]. We trained the model with the different number of states and compared them using penalizes complexity and choosing the best one. After some experimental trials, flow sequences, including the packet timestamps, size, and direction information, are quantized to five values (e.i., five states in the PHMM graph). We employed inter-packet delay and packet size statistics to construct the five-state Markov chain, which we used to model the time series sequences transmitted in a communication stream. In PHMM, states are visiting nodes connected via directed edges called transitions, associated with certain probabilities to identify differences. Each state indicates the presence/absence of specific metadata information at that flow sequence, which is used to learn the parameters of PHMM. Transitions define the correct order and existence of the flow sequence metadata information while allowing insertions and deletions to the flow sequence. We have separated the chain of match states and defined three parallel and interconnected Markov chains (network) of matching states: One network for entry flow and two other networks for positive and negative exit flows.

For training, primary values of the parameters are chosen uniformly to cover the whole range of the observed packet timestamps and size values because the distribution of flow sequences is not solely dependent on the packet's position within the flow. Each state in the PHMM network has different Gaussian weights that equal the covariance matrix of the joint distribution of timestamps and packet size information. Using the Baum-Welch training algorithm [9] all parameters are converged in a few (10) iterations. The probabilities of the PHMM transition states are modified based on the triplet loss function and the input flow sequences. The main goal is to maximize the similarity score of positive exit flows to the entry flow that the PHMM represents. The Expectation-Maximization algorithm [6] along with contrastive triplet loss are used to compute the optimal parameter values and maximize the expected log-likelihood for each observation. Each training data sample is clustered into K states by selecting k observation sub-sequences from the original flow. Each state consists of G Gaussian components, and the probability of the l_{th} observation is computed based on the b^{th} Gaussian component. In the next step, the parameters of each Gaussian component (i.e., m,ϕ, and its weight w_i in state k_i) are updated to find the optimal value by maximization of the Gaussian parameters. At the prediction, the similarity score of the entry and exit flows is computed based on the triplet PHMM models. The test entry and exit flows are assigned to be correlated based on the similarity score of the entry flow's PHMM probability when compared to the PHMM embeddings (for positive and negative exit flows) outputs. Finally, the output of three embeddings is fed to a loss function to make a prediction.

C CONTRASTIVE REGULARIZATION

The cross entropy loss ($Loss_{CrEn}$) is defined as:

$$Loss_{CrEn} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{2} y_i^j log(\breve{y}_i^j) \tag{6}$$

This approach corresponds to the losses based on the categorical Cross Entropy including $Loss_{MMConCEn}$, $Loss_{MMConCEn}$, and $Loss_{MSMS}$. $\breve{y}_i^j \in [0,1]$ is computed using the indicator function in Equation 7 in a supervise problem where $\sum \breve{y}_i^j = 1$, and y_i^j is the real label.

$$y_i^j = \begin{cases} 1 & \text{if } j = k_i \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Finally, the regularized losses ($Loss_{RMMConCEn}$ and $Loss_{RMSMS}$) are equal to the summation of $Loss_{CrEn}$ loss with each of the $Loss_{MMConCEn}$ and $Loss_{MSMS}$ losses. The predicted label $\breve{y}_i^j$ is the output of a Softmax activation function at the last layer of a neural network and is defined as the probability that an exit flow x_i is associated with an entry flow n_i. y_i equals the ground-truth label, and y_i^j corresponds to its one-hot-encoded representation where $y_i^j = 1$ if $j = k_i$ and $arey_i^j = 0$ if $j \neq k_i$ and k_i is the index of the positive class label (i.e., correlated or not).

D DATASET DETAILS

The probability density function of the messages generated in our Loopix simulation closely follows a Poisson distribution. Poisson distribution of traffic traces is a result of the sparse nature of events in typical instant messaging in Loopix mixing communications, as well as the stationary nature of noises in communications, unlike in the case of Tor. In our simulation, each instant messaging chat between clients C_i and C_j was analyzed as a two-way communication flow. It consisted of packets sent from C_i to C_j or received by C_i from C_j. The entry/exit flow was represented by n packets, represented as $f = \{IS_1, IS_2, ..., IS_n\}$. The notation of inter-packet delay (I) and packet size (S), multiplied by packet direction, was used.

An adversary who has obtained entry flow $f(C_i) = \{IS_1^i, IS_2^i, ..., IS_n^i\}$ and exit flow $f(C_j) = \{IS_1^j, IS_2^j, ..., IS_n^j\}$ aims to determine if C_i is communicating with C_j. The adversary's hypotheses can be restated as follows:

- H_0: No communication between C_i and C_j, resulting in independence of $f(C_i)$ and $f(C_j)$. The flow $f(C_j)$ originates from a non-participant in the communication with C_i.
- H_1: If user C_i is communicating with client C_j, then the exit flow $f(C_j)$ will be a noisier version of the communication flow $f(C_i)$.
- H_2: We assumes that client C_i is communicating simultaneously with both clients C_j and C_k. The attacker's objective is to identify the correlation between flows C_i and C_j from a pool of candidate flows that contain unrelated background flow from C_k.

In our research, we only considered the scenario where two clients are communicating simultaneously. However, in real-world situations, with the increasing number of connections per client, it becomes increasingly challenging to differentiate the accurate corresponding entry flow from similar, unrelated ingress flows, resulting in a higher probability of correlation failures.

E ETHICS

Our flow correlation attack was only conducted on virtual Signal instant messaging and communication between simulated clients and did not involve capturing private chat messaging. Our experiments were limited to our simulated clients and did not compromise the privacy of real-world Signal/Loopix members.