ResTor: A Pre-Processing Model for Removing the Noise Pattern in Flow Correlation

Zhong Guan*†, Gang Xiong*†, Zhen Li*†, and Gaopeng Gou*†✉
*Institute of Information Engineering, Chinese Academy of Sciences
†School of Cyber Security, University of Chinese Academy of Sciences
Beijing, China
{guanzhong, xionggang, lizhen, gougaopeng}@iie.ac.cn

***Abstract*—Flow correlation is a common approach to break the anonymity of anonymous communication. However, unpredictable network noise caused by multiple factors in open Internet raises the bar for existing correlation methods. Traditional methods comparing statistical distance of data flows and deep learning methods such as convolutional neural network behave worse because network noise changes traffic shape. In this paper, we design a pre-processing model called ResTor to perform the noise reduction before actually correlating entering and exiting flows. ResTor treats the byte accumulation sequences smoothed at fixed intervals as fitting targets, and takes advantage of the stacked auto-encoder architecture to remove noise in two phases. Experiment results show that the exiting Tor flows processed by ResTor are closer to their corresponding entering flows, thus the correlation task can be finished effectively even using traditional correlation ways: cosine distance and other statistical metrics assisted by ResTor achieve less computational overhead and higher correlation accuracy on Tor compared to the state-of-the-art method of DeepCorr, especially when traffic is obfuscated.**

***Index Terms*—anonymous communication, flow correlation, denoising model**

I. Introduction

Anonymous networks provide users with strong privacy. For instance, the Second-Generation Onion Router (Tor) uses circuits composed of three random nodes around the world to forward traffic of users [1]. Each node only knows the IP addresses of its previous and next nodes. Therefore, the sources of communication data are difficult to determine, let alone the real implementers of network behaviors. Apart from identity protection, the anonymity can also be exploited to engage in criminal activities such as illegal trades hidden in the darknet market [2]. Although identifying this kind of communication relationships has great significance on cyber security, direct content-based analysis is impossible due to traffic encryption. Moreover, tracking crimes hop by hop worldwide needs sufficient resource controls which is obviously hard to achieve. Fortunately, with the help of flow correlation techniques when necessary, flows of the same malicious session can be associated using content-independent features only at two sides of the network.

✉ Corresponding author. E-mail address: gougaopeng@iie.ac.cn

978-1-7281-8086-1/20/$31.00 ©2020 IEEE

Flow correlation can be described as the problem of linking the related entering and exiting flows of the anonymous network. Existing correlation methods include traditional methods [3]–[8] and learning-based methods [9]. Traditional methods use statistical metrics such as cosine distance, pearson coefficient, mutual information, etc. to measure the similarity of data flows. Each flow is abstracted as a vector characterizing the flow shape (e.g., the sequence of packet directions, packet sizes, packet intervals, byte bursts), and then input into the metrics-calculating formula to estimate the similarity. If similarity exceeds certain threshold, the flow pair is considered to be related. What the statistical metrics capture are just shallow features, which determine the poor robustness of this method. A little noise caused by network status (e.g., congestion, dynamic bandwidth allocation, devices failure) can greatly affect vector representations, thus the correlation accuracy is degraded dramatically. Learning-based methods apply learning models like neural networks which have stronger nonlinear fitting ability to correlate flows, hence allow a better resistance to the network status noise. However, noise influence becomes more serious in anonymous networks especially in Tor, such as packets misalignment and circuits variation.

a) Packets Misalignment: The number of packets sent by the source host is almost never equal to the number of packets received by its destination. Firstly, stepping stones of Tor may alter packet counts through dropping packets, retransmission or repacketization [10]. In addition, obfuscation plug-ins used by Tor like Obfs4 [11] can also truncate packets for the purpose of obfuscating traffic. As shown in Fig. 1, packets misalignment randomly destroys the one-to-one mapping relationship between packet sequences of entering and exiting flows. Therefore, what the learning models try to correlate are probably different parts of two data flows.

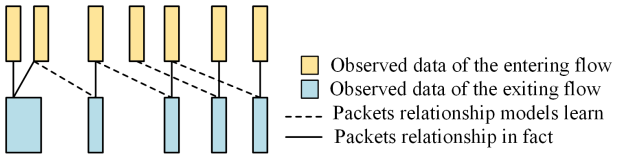
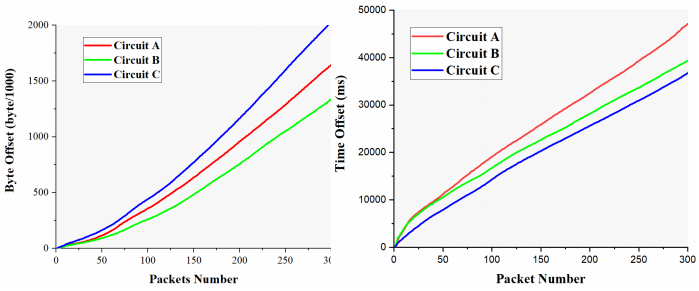


Fig. 1. Packet Misalignment

b) Circuits Variation: Tor circuits change dynamically to reduce the risk of being tracked. Since the routing nodes scatter all over the world, the new circuit is likely to traverse completely different areas [12]. We quantitatively measure the average deviation of exiting flows from entering flows in packet sizes and intervals under three Tor circuits. As shown in Fig. 2(a) and Fig. 2(b), different circuits have different patterns of network status noise on the flow shape, and the more packets are transmitted, the more obvious the deviation is. Consequently, the same entering flow may correspond to various statistical distributions at the exit of network due to circuits variation.



(a) Packet sizes deviation (b) Packet intervals deviation

Fig. 2. Deviation from original entering flows

Not all learning models are easily converged to a satisfactory state, especially when packets misalignment and circuits variation bring a large ambiguity which seriously misleads the model training. In order to solve this problem, we present a pre-processing model for flow correlation to learn and remove the noise pattern in exiting flows of Tor or other anonymous networks. The de-noising model is called ResTor, meaning the exiting flows can be restored to their original shape. Specifically, the contributions of this paper are listed below:

- ResTor is the first de-noising model applied in flow correlation area. It uses smoothed byte accumulation to give a robust description on the shape of anonymous data flows, and the stacked auto-encoder architecture of ResTor efficiently removes the ordinary status noise and obfuscation noise separately.
- For non-obfuscated traffic, ResTor help traditional methods behave far better and achieve as good results as DeepCorr (i.e., the best method so far) with a shorter flow observation. For traffic obfuscated by Obfs4, the true positive rate keeps over 80% which greatly exceeds DeepCorr.
- The used feature and architecture are proved to play their expected role through validation tests on certain datasets. All results are obtained on the same dataset of DeepCorr.

The rest of this paper is organized as follows: Section II reviews typical works of flow correlation and noise reduction, and we introduce the proposed method in Section III including feature transformation and model architecture. Experiment implementation, results analysis and validation tests are detailed in Section IV. Conclusion is given in Section V.

II. Related Work

A. Flow Correlation

Most of the flow correlation methods are based on statistical metrics, the difference between them is the content-independent features used to describe the flow shape and how to calculate the similarity based on these features. Zhang et al. [3] model the idle periods in transmission as sequences of ON/OFF flags and compare their similarity. Nasr et al. [4] compare the cosine distance of packet interval sequences after compressing them as in the signal processing area. Zhu et al. [7] treat flows as random variables and employ the mutual information as metrics which reflects the dependence between two variables [6]. RAPTOR [8] introduces the asymmetric traffic analysis based on spearman coefficient with the traffic observation on AS-level.

Considering that statistical metrics are unreliable, the deep learning method is put forward in DeepCorr [9] for the first time. DeepCorr arranges the packet sizes and intervals of two flows as a matrix for convolution processing, and finally gives a related probability. However, the true positive rate of DeepCorr decreases from over 90% to 50% when confronted with the obfuscation noise. Machine learning models like random forest also succeed in determining whether flows' Markov probability vectors are related or not, just relying on a few packets in each flow [13]. Nevertheless, the dataset of [13] is captured by simple repeating scripts instead of observation in the open environment. Thus the similarity of related flows in this dataset is actually too higher than it should be in reality.

B. Noise Reduction

Researches about noise reduction mainly focus on the field of image de-noising. Jain et al. [14] find that the convolutional neural network (CNN) provides superior performance to the wavelet and Markov de-noising methods under certain conditions. Burger et al. achieve better de-noising results using multi-layer perceptron (MLP) and they believe that MLP is more adaptive to random noise because it is able to approximate nearly all functions, while CNN is limited by the assumptions on image patches [15].

As a special MLP structure which has symmetric parameters, auto-encoders also show excellent ability in noise reduction [16]–[18]. The optimization target of auto-encoders is making the output as close to the input as possible, thus the hidden layer of a well trained auto-encoder can be seen as an alternative representation of input. In order to keep the model away from degenerating into a simple equation like X=X=X, noise is usually added to the input to help auto-encoders learn a more robust representation [19]. Inspired by existing de-noising researches and the demand for noiseless data in flow correlation, we apply the de-noising idea to pre-process anonymous network traffic, hoping improve correlation results.

III. Proposed Method

In this section we perform the noise reduction on byte accumulation sequences using stacked auto-encoder architecture. The framework is shown as Fig. 3. Original packet

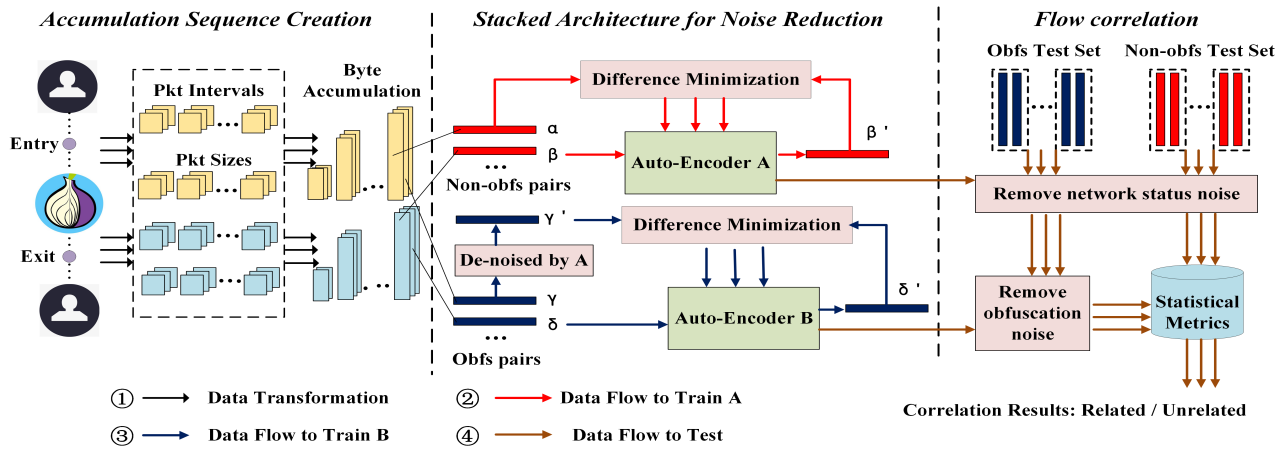


Fig. 3. Framework of pre-processing model for noise reduction

sizes and intervals used by most current correlation works are transformed as accumulation sequences to characterize the flow shape. Then the stacked architecture removes the noise after learning its pattern through mapping sequences of entering flows to corresponding exiting ones. Note that ResTor is a pre-processing model and actual correlation is still carried out using statistical metrics.

A. Accumulation Sequence Creation

Packet sizes and intervals are most intuitive but changeable features for flow correlation. In order to enhance the stability, algorithm 1 puts them together to form the byte accumulation sequence. The algorithm calculates total transmitted bytes so far at regular time windows with fixed size, just like making an interpolation or smooth transition toward the sum of bytes over time. The window size represents the correlation granularity: smaller window size can describe the byte growth more finely but it is also more susceptible to noise influence, which will bring a lower true positive as well as false positive rate. Conversely, larger window size has opposite effect.

Once we specify the window size and sequence length, the duration of flow observation needed for correlation is determined as their product, regardless of how many packets are in the flow or what order the packets follow. This is one of the reasons why the byte accumulation is more robust. Experiments in Section IV will empirically confirm that the byte accumulation is more stable under the anonymous communication, while other features mentioned in related work suffer from the changing environment a lot.

Algorithm 1: Byte accumulation sequence creation

Input:
(a) seq_length: the length of byte accumulation sequence
(b) pkt_i: the packet interval sequence
(c) pkt_s: the packet size sequence
(d) win_size: the size of time window for interpolation

```
last_time ← 0
index ← 0
seq [1 .. seq_length]
for i ← 1 to length of pkt_i(pkt_s) do
    pkt_i [i] ← pkt_i [i] + pkt_i [i - 1]
    pkt_s [i] ← pkt_s [i] + pkt_s [i - 1]
for i ← 0 to seq_length do
    if last_time <pkt_i [index] then
        seq [i] ← pkt_s [index]
    else
        while last_time ≥ pkt_i [index] do
            if index <length of pkt_i - 1 then
                index ← index + 1
            else break
            end if
        seq [i] ← pkt_s [index]
    end if
    last_time ← last_time + win_size
return seq
```

B. Stacked Architecture

Although the accumulation sequence relieves noise influence, the sum of bytes may also change in transmission, and the change will be more dramatic if packets are padded by obfuscation plug-ins. Thus we still need a learning model to fit the noise instead of comparing accumulation sequences directly. Considering that the obfuscation and the transmission are two successive processes, ResTor adopts the stacked auto-encoder to learn and remove the noise pattern in two phases. Specifically, training the model requires both obfuscated and non-obfuscated traffic. Assuming that we have these two types of traffic in the form of flow pairs, and each flow pair is composed of the accumulation sequences of two related flows. Firstly, (α, β) and other non-obfuscated pairs (denoted as *NO*) are used to train auto-encoder A. Exiting flow β is seen as the input with noise, and entering flow α is taken as the fitting target. Through minimizing the difference between output and fitting target (1), auto-encoder A stores the mapping relationship between two flows in the same non-obfuscated pair. As a result, the process of an exiting flow's passing

through well-trained auto-encoder A is exactly the process of removing the noise in it.

$$w_a = argmin_\theta \sum_{(\alpha,\beta)\in NO} \left(\alpha - De\left(En\left(\beta,\theta\right),\theta^T\right)\right)^2 \quad (1)$$

As the second part, auto-encoder B is trained by obfuscated pairs (denoted as O) like ($\boldsymbol{\gamma}$, $\boldsymbol{\delta}$). Exiting flow $\boldsymbol{\delta}$ passes through auto-encoder A (2) before used to train auto-encoder B (3), thus module B can only learn the pattern of obfuscation noise. That's why it is called a stacked architecture. Now the whole training process is completed.

$$\delta^{'} = De\left(En\left(\delta, w_a\right), w_a^T\right) \quad (2)$$

$$w_b = argmin_\theta \sum_{(\gamma,\delta)\in O} \left(\gamma - De\left(En\left(\delta^{'},\theta\right),\theta^T\right)\right)^2 \quad (3)$$

On the testing step, exiting flows of non-obfuscated test pairs are de-noised by auto-encoder A only, while exiting flows of obfuscated test pairs need to be processed by auto-encoders A and B in succession. Then new pairs we get after de-noising operation can be judged as related or unrelated pairs by calculating the similarity with statistical metrics.

IV. Experiment Implementation

A. Dataset and Evaluation Criteria

Experiments are performed on the dataset of DeepCorr which is captured under the real network environment. We count upstream flows and downstream flows separately to double the dataset scale. The size of expanded dataset is displayed as Table I in the number of flow pairs. We randomly allocate 99 exiting flows to each entering flow to build unrelated flow pairs, in order to simulate a scenario where the target session generates only 1% of total flows. Abbreviations in the last column are given for the convenience of following introduction to various experiments. The first place of an abbreviation denotes that the set is a train set or test set. The second place indicates whether the set is obfuscated by Obfs4. Obfs4 is now the most widely used plug-in of Tor due to developed elliptic curve encryption and obfuscation methods. The last place distinguishes whether flows in the set are transmitted through different Tor circuits.

B. Experiment Settings

Experiment settings about model design details are given as Table II. To optimize the noise reduction effect, we conduct several variable-controlled experiments to determine parameter values or value ranges of ResTor, including the time window size, the accumulation sequence length, the neuron number in the hidden layer and the decision threshold on related or not. Fig. 4 displays the correlation results of cosine distance method assisted by ResTor with different window sizes. Results obtained on T.N.D and P.N.D show that the correlation accuracy is positively related to sequence length. The accuracy with sequences shorter than 200 elements has a decline because transmission modes show almost no significant difference between each flow at the early period. Besides, the true positive as well as false positive rate all rise when time window size increases as discussed above. Longer sequence length and more neurons in the hidden layer bring better performance but computational cost also grows. Thus we make a balance between accuracy and resource consuming when selecting parameters.

TABLE II
Parameter Settings

To be determined		Details
Time window size		0.05s
Accumulation sequence length		(100, 500)
Neuron number in hidden layer		input number * 2
Decision threshold	Euclidean	2.0
	Cosine	0.9
	MutualInfo	2.5
Activation function		Relu
Loss function	Euclidean	Sum-squared error
	Cosine	Cosine distance
	MutualInfo	Cross entropy

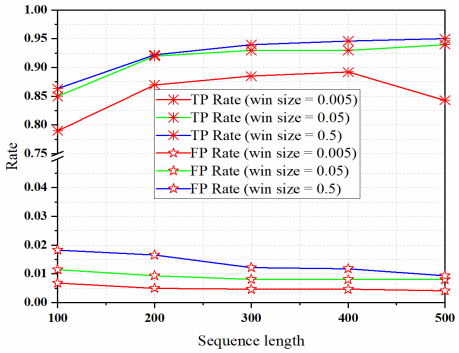


Fig. 4. Correlation results with different time window size

Besides, we use three traditional methods to examine the noise reduction effect of ResTor: euclidean distance, cosine similarity and mutual information. Different metrics have their own best threshold on determining whether the flow pair is related or not, and the loss function of ResTor should correspond to the used metric. For example, if we choose the euclidean distance to correlate flow pairs de-noised by ResTor, the sum-squared error should become the loss function when training ResTor. Similarly, cosine similarity and mutual information correspond to cosine distance and cross entropy, respectively. Next we will use these settings to compare ResTor with simply traditional ways and deep learning ways.

C. ResTor *vs* Statistical Metric Methods

Experiments are conducted on non-obfuscated dataset, in which T.N.D is used to train a single auto-encoder, and P.N.D is for testing. Firstly we directly calculate the similarity of

TABLE I
COMPOSITION OF DATASET

Train/Predict	Flow Pair Type	Traffic Type	Circuit Type	Flow Pairs Num	Abbreviation
Train Set	Related Flow Pairs (Positive Samples)	Non-obfuscated Traffic	Different Circuit	22835	T.N.D
			Single Circuit	22081	T.N.S
		Obfuscated Traffic	Different Circuit	5964	T.O.D
	Unrelated Flow Pairs (Negative Samples)	Number of Related Flow Pairs * 99			
Predict Set	Related Flow Pairs	Non-obfuscated Traffic	Different Circuit	2537	P.N.D
			Single Circuit	2453	P.N.S
		Obfuscated Traffic	Different Circuit	662	P.O.D
	Unrelated Flow Pairs	Number of Related Flow Pairs * 99			

flow pairs measured by euclidean distance, cosine similarity and mutual information to obtain the results of simple metric methods. Then ResTor is trained using the loss function matched with each metric. The well-trained model transforms P.N.D to a new noiseless dataset and we calculate the similarity of flow pairs in it again to obtain the results of ResTor. As shown in Fig. 5, performance of metric methods with ResTor evidently improves compared to using metric methods only.

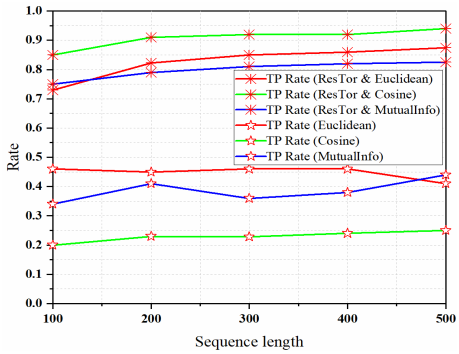


Fig. 5. Results of ResTor and metric methods

D. ResTor vs DeepCorr

We compare ResTor and DeepCorr on non-obfuscated traffic in Fig. 6 and obfuscated traffic in Fig. 7. Horizontal lines indicate the results of DeepCorr when observing 300 packets in each flow, and broken lines are results of ResTor combined with the cosine distance method, which performs best with the help of ResTor.

For non-obfuscated traffic, ResTor and DeepCorr all achieve extremely high true positive and low false positive rate, outperforming simple metric methods by a big margin. However, ResTor reaches the same good results earlier than DeepCorr at the point where accumulation sequence has about 200 elements. As shown in Table III, since the time window size we determined is 0.05s, ResTor actually takes 10 seconds of observation in each flow when correlating flow pairs. While for DeepCorr, 300 packets of observation needs the time ranging from 11.4 to 67.4 seconds according to the dataset. Shorter observation means more and even short-lived data

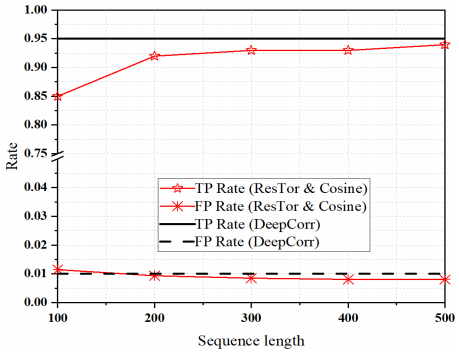


Fig. 6. Results of ResTor and DeepCorr on non-obfuscated dataset

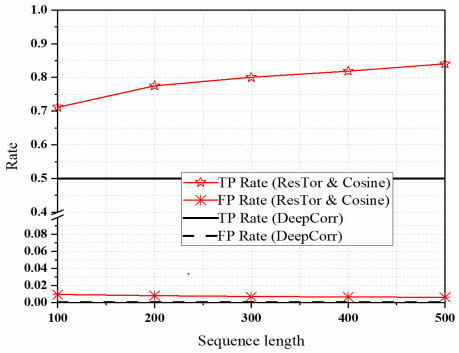


Fig. 7. Results of ResTor and DeepCorr on obfuscated dataset

flows can be exploited to make correlation. Besides, low complexity of ResTor supports a faster correlation which only costs less than half of DeepCorr's time, thus closer to a real-time detection. **For obfuscated traffic,** false positive rate of two models are all kept at a low level, but ResTor improves the true positive rate by about 30 percents.

E. Validation Tests

Validation on feature selection. We confirm the effect of accumulation sequences by comparing its correlation results to other typical content-independent features on T.N.D and T.N.S, including transmission direction [3], packet intervals and packet sizes [4], [10]. Table IV shows that true positive rate has almost no reduction on different circuits when ResTor

TABLE III
COMPARISON ON TIME COST

Model		Observation	Correlation
DeepCorr		(11.4s, 67.4s)	2ms
ResTor	Euclidean	10s	≈0.7ms
	Cosine	10s	≈0.9ms
	MutualInfo	10s	≈1.5ms

takes accumulation sequences as input, which proves its better resistance against circuits variation.

TABLE IV
VALIDATION RESULTS ON STACKED ARCHITECTURE AND MLP

	Single Circuit(TP)	Different Circuit(TP)
Accu Seq	0.97	0.94
Pkt Interval	0.73	0.51
Pkt Size	0.69	0.42
Direction	0.51	0.44

Validation on stacked architecture. Similarly, we compare ResTor to a plain MLP with same layers number on T.O.D and T.N.D. Performance of MLP degrades obviously on obfuscated traffic like DeepCorr in Table V, hence ResTor can not be replaced by directly training a simple fully connected network end to end. On the contrary, two-phase training on stacked architecture is appropriate for removing the network status noise and obfuscation noise which have different patterns.

Flow correlation techniques hardly cause privacy risks even when abused because correlation models just analyse content-independent features, and only observers at the ISP level can acquire entering and exiting traffic of anonymous networks at the same time. Besides, locating specific malicious acts only needs single-point observation which just reveals whether users act maliciously instead of recording their long-term detailed behavior. Users can also replace low-latency anonymity protocols with traditional mix-net, just in case. Anonymity protocols based on mix-net and its variants [20] can resist correlation attacks at the cost of high communication delay.

TABLE V
VALIDATION RESULTS ON STACKED ARCHITECTURE AND MLP

	ResTor(TP)	MLP(TP)	ResTor(FP)	MLP(FP)
Non-obfs	0.94	0.8	0.008	0.03
Obfs	0.87	0.66	0.006	0.04

CONCLUSION

In this paper we propose a flow correlation method based on a pre-processing model called ResTor. It outperforms the statistical metrics and the deep learning models, particularly when the flow shape is disturbed by network noise. Experiment results and further validation tests support that the smoothed byte accumulation sequence is more stable than other features in describing the flow shape of anonymous communication, and the stacked auto-encoder architecture is suitable for two-phase noise reduction in obfuscated traffic.

ACKNOWLEDGMENT

This work is supported by The National Natural Science Foundation of China (No.U1636217) and The National Key Research and Development Program of China (No.2016QY05X1000, No.2018YFB1800200 and No.2020YFE0200500).

REFERENCES

[1] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: the secondgeneration onion router," in *Proceedings of the 13th USENIX Security Symposium*, 2004.
[2] N. Christin, "Traveling the Silk Road: A measurement analysis of a large anonymous online marketplace," in *Archives of Neurology*, vol. 2, no. 3, pp. 213–224, 2012.
[3] Y. Zhang and V. Paxson, "Detecting stepping stones," in *Proceedings of the 9th USENIX Security Symposium*, 2000.
[4] M. Nasr, A. Houmansadr, and A.Mazumdar, "Compressive Traffic Analysis: A New Paradigm for Scalable Traffic Analysis" in *ACM Conference on Computer & Communications Security*, 2017.
[5] V. Shmatikov and M. Wang, "Timing analysis in low-latency mix networks: attacks and defenses," in *European Symposium on Research in Computer Security*, 2006.
[6] T. Chothia and A. Guha, "A Statistical Test for Information Leaks Using Continuous Mutual Information," in *Computer Security Foundations Symposium*, 2011.
[7] Y. Zhu, X. Fu, B. Graham, R. Bettati, and W. Zhao, "On Flow Correlation Attacks and Countermeasures in Mix Networks," in *International Workshop on Privacy Enhancing Technologies*, 2004.
[8] Y. Sun, A. Edmundson, L. Vanbever *et al.*, "RAPTOR: routing attacks on privacy in Tor," in *Proceedings of the 24th USENIX Security Symposium*, 2015.
[9] M. Nasr, A. Bahramali, and A. Houmansadr, "DeepCorr: strong flow correlation attacks on Tor using deep learning," in *ACM Conference on Computer & Communications Security*, 2018.
[10] Y. J. Pyun, Y. H. Park, X. Wang, D. S. Reeves, and P. Ning, "Tracing Traffic through Intermediate Hosts that Repacketize Flows," in *26th IEEE International Conference on Computer Communications*, 2007.
[11] Obfs4 (The obfourscator), Available: https://raw.githubusercontent.com/Yawning/obfs4/master/doc/obfs4-spec.txt
[12] K. Kohls, K. Jansen, D. Rupprecht, T. Holz, and C. Popper, "On the Challenges of Geographical Avoidance for Tor," in *Network and Distributed Systems Security Symposium*, 2019.
[13] Z. Guan, G. Gou, Y. Guan, and B. Wang, "An Empirical Analysis of Plugin-Based Tor Traffic over SSH Tunnel," in *IEEE Military Communications Conference*, 2019.
[14] V. Jain and H. S. Seung, "Natural Image Denoising with Convolutional Networks," in *Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems*, 2008.
[15] H. C. Burger, C. J. Schuler, and S. Harmeling, "Image denoising: Can plain Neural Networks compete with BM3D," in *Computer Vision and Pattern Recognition*, 2012.
[16] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. Manzagol, "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion," in *Journal of Machine Learning Research*, vol. 11, no. 12, pp. 3371–3408, 2010.
[17] J. Xie, L. Xu, and E. Chen, "Image Denoising and Inpainting with Deep Neural Networks," in *Advances in neural information processing systems*, 2012.
[18] X. Mao, C. Shen, and Y. Yang, "Image Restoration Using Very Deep Convolutional Encoder-Decoder Networks with Symmetric Skip Connections," in *30th Conference on Neural Information Processing Systems*, 2016.
[19] P. Vincent, H. Larochelle, and Y. Bengio, "Extracting and composing robust features with denoising autoencoders," in *Machine Learning, Proceedings of the Twenty-Fifth International Conference*, 2008.
[20] P. Kotzanikolaou, G. Chatzisofroniou, and M. Burmester, "Broadcast anonymous routing (BAR): scalable real-time anonymous communication," in *International Journal of Information Security*, 2016.