Dynamic Traffic Feature Camouflaging via Generative Adversarial Networks

Jie Li, Lu Zhou, Huaxin Li, Lu Yan, Haojin Zhu
Department of Computer Science and Engineering, Shanghai Jiao Tong University

***Abstract*—Traffic analysis attacks, including website fingerprinting and protocol fingerprinting, are widely adopted by Internet censorship to block a specific type of traffic. To mitigate these attacks, some advanced approaches such as traffic morphing and protocol tunneling techniques have been proposed. However, the existing traffic morphing/protocol tunneling techniques suffer from showing a strong traffic pattern or can be uncovered with a low false positive. Further, they mainly rely on learning the pattern for specific traffic, which makes it highly possible to be identified due to a lack of dynamics. In this paper, we propose a dynamic traffic camouflaging technique, coined FlowGAN, to dynamically morph traffic feature as another "normal" network flow to bypass Internet censorship. The core idea of FlowGAN is to automatically learn the features of the "normal" network flow, and dynamically morph the on-going traffic flows based on the learned features by the adoption of the recently proposed Generative Adversarial Networks (GAN) model. To measure the indistinguishability of the target traffic and the morphed traffic, we introduce a novel concept of ϵ-indistinguishability. We evaluate the proposed method on a dataset involving 10,000 real-world flows, and experimental results show that the effectiveness and the efficiency of FlowGAN regarding ϵ-indistinguishability, AUC, and latency. To the best of our knowledge, our work is the first one to adopt GAN for automatic traffic generation and censor circumvention.**

I. Introduction

Internet censorship circumvention is regarded as one of the key applications of anonymous communication techniques [1]. Since the network protocols and connection endpoints adopted by circumvention systems are quite different from those of other Internet services, it is easy for ISP-level censorship to perform the traffic analysis attacks and block them at a low cost without compromising any other network services. The typical traffic analysis attacks include *website fingerprinting* (WF) attacks [2], [3], [4], [5], which aim to infer the web activity of a client (e.g., which web page is visited) even when the client is using the anonymous tool, and *protocol fingerprinting* (PF) attacks [6], which analyze the traffic pattern to measure the similarity between an unknown flow and a specific protocol. In general, traffic analysis attacks are modeled as the classification problem, where the attackers extract features from traffic flows they captured and employ classifiers such as k-Nearest Neighbors (kNN) or deep neural networks (DNN) to fingerprint these flows.

To mitigate traffic analysis attacks, different kinds of defense techniques have been proposed in recent years. A common approach of preventing leaks is to obfuscate the encrypted traffic by changing the statistical features of the traffic, such as the packet size and packet timing information [3], [7], [8]. More advanced approaches include Traffic Morphing, which can optimally morph one class of traffic to look like another class [9], and the protocol tunneling technique, which leverages the popular encrypted online services (e.g., living streaming) to transmit censored content [10]. However, the existing traffic morphing/protocol tunneling techniques suffer from showing a strong traffic pattern or can be uncovered with a low false positive rate [2], [5]. Further, the existing solutions heavily rely on the learning of the traffic pattern of a specific kind of flow, which makes it highly possible to be recognized and blocked by the censors. A good example is CovertCast, which uses Youtube live streaming to transmit the covert channel. However, Youtube has been blocked by more than 10 countries/areas according to [11], which makes the covert channel fail to work either. Therefore, a more dynamic traffic camouflaging technique is highly expected.

In this study, we propose a novel traffic camouflaging technique, coined FlowGAN, which allows a proxy to dynamically morph on-going traffic flows as some other flows. As shown in Fig. 1, by giving a source (or censored) flow and a target (or permitted) flow, the core idea of FlowGAN is to automatically extract the traffic features of the target flow, and morph the source flow to the target flow based on the extracted traffic features. The generated flow should be indistinguishable from the target flow, which can provide the privacy guarantee and censorship circumvention. All of these can be achieved by the adoption of the recently proposed Generative Adversarial Nets (GAN) model. GAN's discriminator tries to classify if the output flow is censored or the permitted, while simultaneously training a generative model to bypass the discriminator's detection. Since GANs learn the pattern that adapts to the data, they can be applied to a multitude of tasks (or multiple target permitted flows) that traditionally would require very different kinds of loss functions. Compared with any previous works, the proposed approach shows its superiority in terms of dynamical traffic camouflaging, which is expected to morph a censored flow to multiple target flows. This is expected to prevent the censors from recognizing and blocking any specific type of traffic. To validate the effectiveness of FlowGan, we have designed a prototype. The evaluation is performed on a dataset involving 10,000 traffic flows to *baidu.com*. We give a formal definition on indistinguishability of two traffic flows and evaluate the effectiveness and efficiency of FlowGan, in terms of a series of metrics including Indistinguishability,

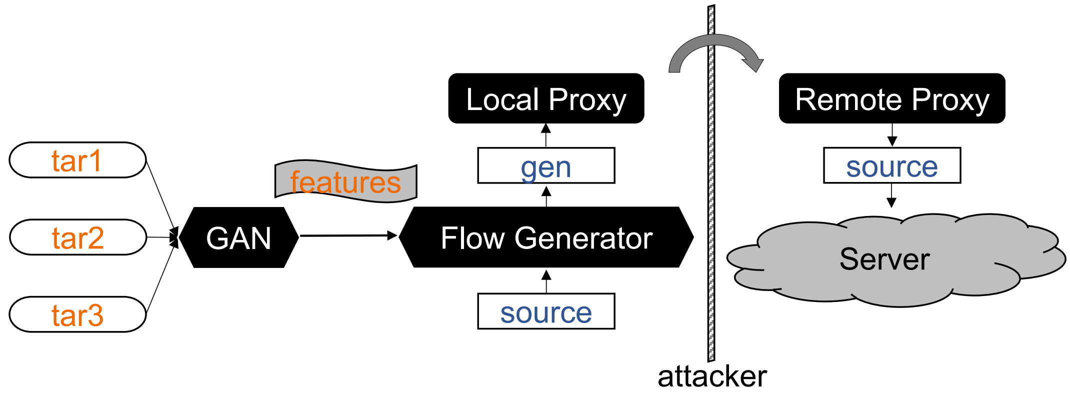


Fig. 1. Overview of FlowGAN

AUC, Latency. The evaluation results well demonstrate that FlowGan outperforms the counterparts and is expected to significantly mitigate existing traffic analysis attacks.

The main contributions of our paper are as follows:

- We proposed a new traffic camouflaging mechanism, FlowGan, to secure anonymous network communications against both website fingerprinting attacks and protocol fingerprinting attacks. FlowGan employs GAN models to generate network flows that are indistinguishable from given traffic flows so that the censor cannot block them.
- We proposed a novel privacy model to evaluate the effectiveness of the proposed work. This new private model measures the indistinguishability between the GAN-generated network flows and the targeted network flows.
- We performed a series of state-of-the-art traffic analysis attacks towards FlowGan based on a real-world dataset. The evaluation results well demonstrated the effectiveness and the efficiency of FlowGAN under a series of state-of-the-art traffic analysis attacks.

To the best of our knowledge, this is the first work which leverages GAN for traffic camouflaging and anonymous communications. The remainder of the paper is structured as follows. Section II surveys the representative and latest related works of traffic analysis attacks and defenses. Section III presents the threat model in our paper. Section IV describes our private model and the evaluation metric. Section V introduces the methodology and implementation of our system. Section VI elaborates experiments and evaluation results. Section VII concludes this paper.

II. Related Works

Traffic analysis can be used to conduct side-channel attack in many scenarios [12], [13], [14], [15]. One of the most common cases is the fingerprinting attack through network traffic analysis, including website fingerprinting and protocol fingerprinting. Recent research works have shown that state-of-art anonymous mechanisms are still vulnerable under these attacks and thus new defense mechanisms are proposed.

A. Traffic Fingerprinting Attacks

Fingerprinting attacks are usually modeled as a classification problem in which attackers extract features from side channel and classify a set of websites/apps [4], [16]. Traffic fingerprinting attacks have been being extensively studied over the past ten years [17]. Dyer et al. [3] provided a first comprehensive study on traffic analysis attacks and defenses in 2012, and their results showed that many of the countermeasures by that time could not resist even simple attacks that leverage general traffic features such as bandwidth and total time.

Cai et al. [18] proposed to compare the similarity of packet traces (e.g., packet orderings) for identifying web pages, and then used Hidden Markov Model (HMM) to fingerprint websites based on the sequence of visited web page under a simulated Tor environment. Wang et al. [5] built a k-Nearest-Neighbor (kNN) classifier using a large set of features. Their classifier has been tested on a large open-world setting and outperformed some previous works. In [19], [20], the authors proposed that side channel data is efficient to uncover enough information. Hayes et al. [21] proposed an attack that employed Random Forest model to produce fingerprints of websites, and launched website fingerprinting attack with a large amount of noisy data as well as Tor environment. In 2016, Panchenko et al. [4] used a SVM classifier to classify the cumulative features (CUMUL) of different flows. Meanwhile, they evaluated fingerprinting at the Internet scale with representative data. However, their experiments showed no existing method can scale for any web page in their datasets.

In the recent two years, approaches that exploited more advanced models, such as deep learning model (supervised model [22], unsupervised model [23]), were studied. Moreover, the fingerprintability [24] was measured and analyzed. Though secure channels and encryption have been widely deployed in network communication, traffic features in side-channel and usage patterns inevitably provide evidence to attackers. Along with larger datasets being evaluated and more powerful attacks being proposed, the implications of traffic fingerprinting attacks are alarming and attract research interests increasingly.

B. Defenses for Fingerprinting Attacks

Various defenses were proposed to counter fingerprinting attacks. Since many of the attacks extracted traffic features to discriminate different websites, a typical category of countermeasures is to conceal the traffic features. Padding and creating dummy traffic are common ways to modify traffic features such as packet size and time interval. For instances, HTTPOS [7] can hide packet length by strategically sending HTTP request, while TrafficMorphing [9], WTF-PAD [25], BuFLO [3], CS-BuFLO [26], and Tamaraw [8] padding traffic packets in different strategies to hide the actual time interval and length. However, many of them have been proved to be ineffective under newly proposed attacks or cause a large overhead [2], [5].

With the urgent requirements for defending fingerprinting attacks, more effective and efficient countermeasures that consider the open-world scenario and overhead were proposed in recent years. To consider the practical environment, Cherubin et al. [27] proposed a server-side defense a lightweight client-side defense implemented as a browser add-on to defend WF attacks at the application level. Walkie-Talkie [2] modified the browser communication from usual full-duplex mode to in half-duplex mode so that the traffic is sent in moldable interleaving bursts and the molding is computationally cheap. Cui et al. [28] used variable realistic network traffic as noise based on the motivation that visiting two websites at once mitigates website fingerprinting attacks. Our solution considers a practical scenario that censors that identify suspicious traffic patterns [29] may block the morphed traffic causing the countermeasure failed. So our goal is to generate traffic that is hard to be fingerprinted and can circumvent Internet censorship.

III. Threat Model

By following the similar threat model as shown in [30], in this study, we consider the attacker model as a state-level adversary, which attempts to classify the censored traffic from normal traffic by leveraging a series of website fingerprint techniques. The censored traffic is expected to take advantage of protocol tunneling techniques to evade the detection of the adversary by using the encrypted applications as the carriers for covert channels. The adversary is not assumed to demand the application providers to decipher and disclose the original content, which enables the easy identification and filtering of the covert data. However, the censorship system can be deployed by domestic ISPs with cooperation with the adversary, which allows it to monitor, store and inspect all traffic flows crossing its borders [30].

The adversary is assumed to perform the various traffic analysis towards some suspicious flow and block the traffic containing the covert channels. In practice, morphing the censored traffic to a specific kind of popular traffic (e.g., livestreaming services such as youtube) is quite risky since this popular traffic may be unavailable in some regions. For example, youtube has been blocked by more than 10 countries including Iran, Afghanistan, Armenia, Brazil, Finland, Germany [11]. Therefore, a more dynamic traffic obfuscation solution which can dynamically transform the censored traffic to other popular applications is highly desirable.

TABLE I
NOTATION

Symbol	Description
Adv	Adversary
Chl	Challenger
Tar	Traffic from morphing target
Gen	Generated traffic
$\mathcal{P}$	Adversary's prior knowledge
$H_{\mathcal{P}}$	Adv's classifier trained with $\mathcal{P}$.
$P_T = \mathcal{P} \cap Tar$	Adv's prior knowledge about T
$P_G = \mathcal{P} \cap Gen$	Adv's prior knowledge about G
$C_1 \subset_R T$	C_1 is a random subset of Tar
$C_0 \subset_R G$	C_0 is a random subset of Gen

Challenger C		**Adversary A**
$Tar\ initialization$		
$Gen \leftarrow GAN(Tar)$		$Hp \leftarrow P$
$S_0 \leftarrow Tar, S_1 \leftarrow Gen$		
$b \leftarrow_{\$} \{0,1\}$		
$c \leftarrow S_b$	$\xrightarrow{c}$	$\hat{b} \leftarrow A(c)$

$$PrivK = 1 \iff \hat{b} = b$$

Fig. 2. Traffic Indistinguishability Game

IV. Traffic Indistinguishability Definition

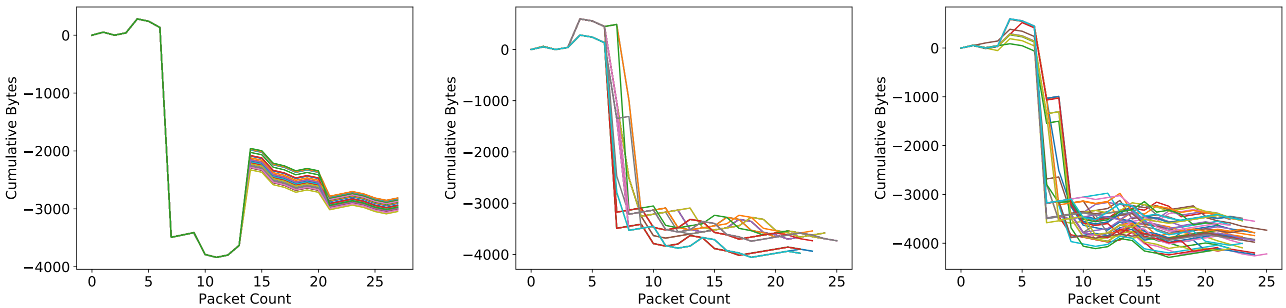
In this section, we formalize traffic indistinguishability under the traffic analysis attack in our problem. We consider the scenario that an attacker attempts to infer if an encrypted traffic flow contains the covert channel via performing the traffic analysis and block a specific traffic flow based on the inference results. Our goal is to generate a traffic flow which is indistinguishable from another "normal" network flow.

A. Notation

The notations in this paper are summarized in Table I. We denote the set of target traffic flows as $Tar = \{T_1, T_2, ..., T_n\}$ and the set of generated traffic flows which are generated based on the target traffic as $Gen = \{G_1, G_2, ..., G_n\}$, where T_i and G_j represent a traffic flow of Tar and Gen, respectively. For an adversary (Adv), we denote his/her prior knowledge as $\mathcal{P} = P_T \cup P_G$ where P_T and P_G are prior knowledge about Tar and Gen, respectively. P_T and P_G are supposed to be known by the adversary and used to train the classifier to classify Tar and Gen.

B. Traffic Indistinguishability Game

We model our problem by a traffic indistinguishability game involving an adversary Adv and a challenger Chl. Traffic indistinguishability game can be described as follows (Fig. 2):



(a) Censored Traffic (b) Uncensored Traffic (c) Generated Traffic

Fig. 3. Flow Visualization

Setup: The challenger Chl initializes the target traffic flows Tar and generates the morphing traffic flows Gen. At the same time, Adv utilizes $\mathcal{P}$ to train a classifier $H_{\mathcal{P}}$ for classifying traffic flows from Tar and Gen.

Challenge: Chl randomly flips a coin b. The challenger sends S_0 if $b = 0$ to the adversary. Otherwise, he will send S_1. Where S_0 and S_1 are the randomly selected traffic flows from Tar and Gen, respectively.

Classification: Adv uses the classifier $H_{\mathcal{P}}$ to judge the traffic flow from Tar or Gen, and output the predicted $\hat{b}$.

The adversary wins the game if and only if $\hat{b} = b$, which means Adv can distinguish traffic flows from Tar and Gen, and we denote it as $PrivK = 1$. Based on the above description, we define ϵ-indistinguishability to represent the indistinguishability of the target traffic Tar and the morphing traffic Gen.

Definition 1. *(ϵ-indistinguishability) Let $PrivK = 1$ iff $\hat{b} = b$. We say the generated traffics is ϵ-indistinguishability if for every adversary A it hold that:*

$$\frac{1}{2} - \frac{\epsilon}{2} \leq Pr[PrivK = 1] \leq \frac{1}{2} + \frac{\epsilon}{2} \quad (1)$$

According to the Definition 1, the smaller the ϵ is, the lower the possibility of a successful classification attack is. When the ϵ equals to a very small value, the classification attack tends to be a random guess, and the traffic can be indistinguishable. Based on these concepts, we further define the indistinguishability under Classification Attack (IND-CA) as Equ. 2 to quantify the traffic indistinguishability.

$$IND - CA = \frac{|Pr[PrivK = 1] - 0.5|}{0.5} \quad (2)$$

IND-CA has the same form as ϵ-indistinguishability, and it is a numerical value which is linear to the indistinguishability within $[0, 1]$. When it is closer to 0, the attack is less effective, and the traffic is more indistinguishable.

V. System Design

A. Motivation

As discussed above, a dynamic traffic feature camouflaging scheme is highly desirable to evade the detection of ISP censors. Different from previous works that mimic the traffic features of a specific popular traffic flow, the proposed system can automatically generate the camouflaged flows, which is indistinguishable from any target flow. To achieve this, we design a novel automatic traffic camouflaging system based on the generative adversarial network (GAN). GAN is originally proposed by [31] and has achieved unparalleled results in scenarios such as image and 3D objects generation. However, less attention has been paid to applying GAN to the area of anonymous communications. To the best of our knowledge, our work is the first one to adopt GAN for automatic traffic generation and censor circumvention.

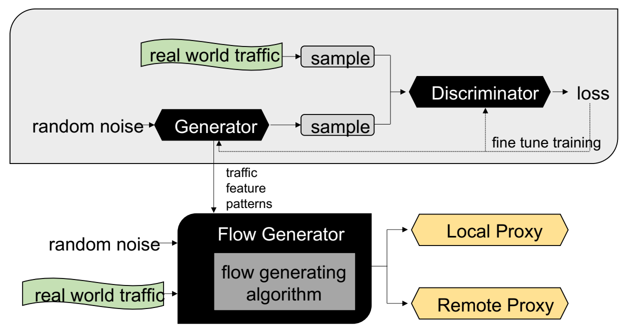


Fig. 4. System Architecture

As shown in Fig. 3(a), we have a censored flow which shows a distinct traffic pattern compared with the permitted flow in Fig. 3(b). Therefore, the adversary can easily classify the censored flow from the permitted ones via traffic analysis techniques. Our goal is to camouflage the censored traffic in Fig. 3(a) to a target flow (e.g., Baidu traffic flow) which is shown in Fig. 3(b). Therefore, we take flows in Fig. 3(b) as training set and generate flows in Fig. 3(c) as the morphed traffic. As shown in Fig. 3(c), the generated flow possesses similar features as the target flows shown in Fig. 3(b). However, they are not simply replicas. In fact, most of the generated flows do not appear in the target flows. This provides randomness to defend the censorship. In order to improve the generation efficiency and technical feasibility, we adopt GAN model to produce important features for a network flow rather than using the entire packet sequence.

The system architecture is shown in Fig. 4. It includes a GAN generator, a network flow generator, a local proxy server, and a remote proxy server. GAN generator is trained to generate important traffic pattern features and forwards them to the flow generator. The flow generator receives these traffic pattern features to generate packet sequence of a network flow. Local and remote proxy servers send packets as the flow generated. In the below, we will present each module of the system as follows.

B. GAN Generator

1) GAN model: GAN is a framework for improving generative models via an adversarial process [31]. GAN simultaneously trains two models, a generative model G to learn the data distribution from training data, and a discriminative model D to distinguish data generated by G from the training data. G and D plays an adversarial game: G generates samples from random noise p_z and aims to fool D that this sample is from training data; D tries to distinguish these samples with a high successful rate. At the end of this game, only one optimal solution exists: the generator G learned the distribution of training data and generate sample indistinguishable from these training data so that discriminator cannot classify generated samples [31].

Formally, the GAN plays a min-max game with loss function $J(G, D)$:

$$\begin{aligned} \min_G \max_D J(G, D) = E_{x \sim pdata}[log\ D(x)] \\ + E_{z \sim pz}[1 - log\ D(G(z))] \end{aligned} \tag{3}$$

The discriminator loss J is computed by cross entropy for binary classifiers:

$$J_D = -E_{x \sim pdata}[log\ D(x)] - E_{z \sim pz}[1 - log\ D(G(z))] \tag{4}$$

The generative loss J_G is theoretically negative of J_D. In order to speed up training, J_G is computed as following:

$$J_G = -E_{z \sim pz} log(D(G(z))) \tag{5}$$

The discriminator and generator are updated by these two loss function respectively through stochastic gradient, until the generator can produce realistic samples that fool the discriminator.

2) Design: In our system, the GAN generator employs a well-tailored GAN model for generating indistinguishable traffic features, as presented in Algorithm 1. Given typical features P_r (e.g., the number of packets, packet length, arrival time interval) extracted from some targeted traffic flows as the training set, the GAN model aims to train a generator that can produce traffic features that are indistinguishable from the targeted traffic.

In this algorithm, both generator G_θ and discriminator D_ω are a dense neural network with random parameters initially. During iterations, the generator is learning to produce more and more realistic traffic feature samples (line 13-15), and the discriminator uses both samples from the training set

Algorithm 1 Traffic Pattern Generation through GANs

Input: Traffic features P_r from the training set, hyper parameters including discriminator iteration factor k, batch size m, regulation parameter λ

Output: A trained traffic feature generator G_θ

```
1:  Initialize a generator G_θ with parameters θ and a discriminator D with parameters ω
2:  while G_θ has not coverage do
3:      // Training the discriminator
4:      for k steps do
5:          x = a batch of m training samples from P_r
6:          z = a batch of m generated samples from random noise
7:          ε = a batch of m uniform random numbers in [0,1]
8:          x̂ = ε × x + (1 − ε) × G_θ(z)
9:          Update the discriminator with RMSProp algorithm by descending the discriminator's loss:
            J_D = E_m[D_ω(G_θ(z)) − D_ω(x)
                  + λ × (‖∇_x̂ D_ω(x̂)‖ − 1)^2]
10:     end for
11:
12:     // Training the generator
13:     z = a batch of m generated samples from random noise
14:     Update the generator with Adam algorithm by descending the generator's loss:
            J_G = −E_m[D_ω(G_θ(z))]
15: end while
```

(i.e., x) and samples from the generator (i.e., $G_\theta(z)$) to get better at distinguishing generated traffic from real traffic (line 3-10). Both generator and discriminator are improved through back propagation, and the gradient information from the discriminator is back propagated to the generator so that the generator knows how to adapt its parameters to produce output data that can fool the discriminator and other traffic analysis attacks. In each iteration, the discriminator is trained by k steps and the generator is trained by one step to avoid overfitting on finite dataset [31].

In order to overcome the challenge that traditional GAN is difficult to train (for example, difficult to achieve Nash equilibrium and difficult to converge during training), we adopt Wasserstein GAN [32] for our GAN generator. To implement the generator and discriminator in the Wasserstein GAN, WGAN [32] is employed as the generator's loss function and WGAN-GP[33] is selected as the discriminator's loss function. As suggested by Arjovsky[33], it is more robust to train the GAN if we remove the sigmoid function from the last layer.

3) Feature Selection: We select six typical traffic data as features to train our GAN model: the total number of outgoing packets N_{out}, the total number of incoming packets N_{in}, the summation of bytes for outgoing packet S_{out}, the summation of bytes for incoming packet S_{in}, the cumulative bytes Cum, and the average interval between packets $AvgInter$. They are useful features for fingerprinting network traffic as discussed in previous works [4], [5]. We use them for training a strong discriminator that can classify traffic from different sources accurately so that the discriminator can help improve the generator model for producing high-quality camouflaged traffic patterns. However, the total cumulative bytes can be calculated by bytes of outgoing packets and bytes of incoming packets. So in our training process, we only select the former five features.

C. Flow Generator

Flow generator produces packet sequences from traffic pattern features provided by GAN generator. In our flow generation algorithm, we use the CUMUL representation [4] of source flows to guide the traffic generation process. For the ease of presentation, we use packet length (byte) as an example. Given a network flow containing packet sequence $P = [p_1, p_2, ..., p_N]$, where p_i's absolute value represent the packet length of i-th packet and $p_i > 0$ indicates an outgoing packet and $p_i < 0$ indicates an incoming packet, the CUMUL representation for this flow is calculated to form a sequence $C = [c_0, c_1, ..., c_N]$, where $c_0 = 0$ and $c_i = c_{i-1} + p_i$ for $i \in \{1, 2, ..., N\}$. Fig. 3 shows an example of the CUMUL representation of three different flows: the censored ones, the uncensored ones and the generated ones. It can be observed that the CUMUL values between censored traffic and uncensored traffic are distinguishable after accumulating their packet length.

In order to produce indistinguishable network flows from a training network traffic set, we derive network flows from the CUMUL that fits the traffic pattern generated by GAN generator. The algorithm is shown in Algorithm 2, where real network flows are selected from target flows to direct the flow generation. In every step, we randomly choose a set of packets that ever existed in the real flow to avoid the unique packet length attack [5], and select the ones whose CUMUL similarity is higher than the threshold to ensure the indistinguishability. We make sure each generated flow has the exact incoming packet count, outgoing packet count, average interval time as the generated features. And at the end, we select the one that has the best similarity with other traffic pattern features from the generated flows as the output. In this way, we produce traffic flows that have similar CUMUL, and are expected to be indistinguishable by traffic analysis attacks.

D. Proxy Servers Design

The proxy servers include a local proxy and a remote proxy, which communicate with each other according to the network flow patterns received from the flow generator. The architecture of proxy servers is shown in Fig. 5.

The local proxy and the remote proxy share the same flow generator with the same random seed. Clients connect to a local proxy and send payloads to it. Then the local server morphs the traffic as the pattern generated by the flow generator. Generated traffic is sent over the censorship to the remote proxy server. The remote proxy is able to restore traffic payloads and send them to the targeted remote servers.

Algorithm 2 Flow Generation Algorithm

Input: traffic pattern features F generated by GAN generator
Output: a flow = $[p_1, p_2, ..., p_N]$

S = first 100 network $flows$ similar to feature F
CUM_S = []
for s **in** S **do**
  CUM_S.append(cumulative representation of s)
end for
$flows$ = []
for $sample = 1$**to**MAX_SAMPLE **do**
  $packet\_count$ = $N_{in} + N_{out}$
  $flow$ = []
  for $packet\_no$ = 1 **to** $packet\_count$ **do**
    loop
      $packet$ = randomly select a packet in S
      $flow$.append($packet$)
      CUM_F= cumulative representation of $flow$
      max_sim =
        $\max(similarity(CUM\_F, cum \in CUM\_S))$
      if $max\_sim > THRESHOLD$ **then**
        break
      else
        $flow$.remove($packet$)
        continue
      end if
    end loop
  end for
  $flows$.append($flow$)
end for
return $flow \in flows$ with best similarity with F

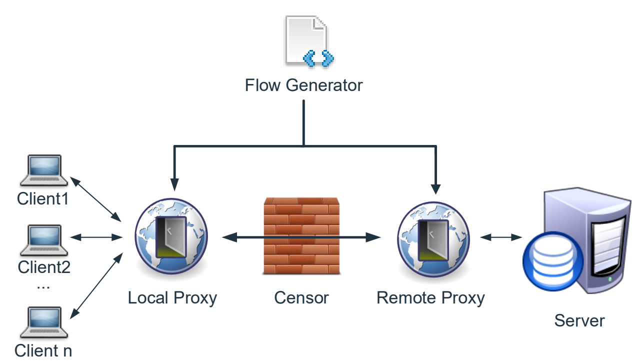


Fig. 5. Proxy Server Architecture

VI. Experiment and Evaluation

A. Data Collection

The network flows used in our experiments are collected at the gateway from a local server in the lab to *baidu.com*, which is the largest Chinese search engine and should hardly be blocked so that it can act as the normal traffic. We use *pypcap* to sniff packets and use *dpkt* to analyze traffic. Since we focus on manipulating network flow, TCP packets are recorded. Each flow is identified with a 4-tuple: (source address, source

TABLE II
GAN NET STRUCTURE

Net	Layer	Number of unit	Activation
Generator	1	20	leaky-relu
	2	10	leaky-relu
	3	5	tanh
Discriminator	1	50	leaky-relu
	2	30	leaky-relu
	3	1	none

TABLE III
IND-CA UNDER DIFFERENT ATTACKS

Attack Method	Total samples	Correct samples	IND-CA	IND-CA w/o defense
Panchenko's [36]	2000	1216	0.216	0.986
Herrmann's[37]	2000	1118	0.118	0.935
VNG++ [3]	2000	1113	0.113	0.813
Bandwidth [3]	2000	1041	0.041	0.782
Time [3]	2000	1000	0.000	0.344
Liberatore's[38]	2000	1085	0.085	0.974
Jaccard's[39]	2000	1002	0.002	0.622
Wright's[9]	2000	1097	0.097	0.974

TABLE IV
AUC AT DIFFERENT EPOCHS

Attack Method	1k epochs	50k epochs	100k epochs
Panchenko's	0.99	0.87	0.61
Herrmann's	0.98	0.80	0.56
VNG++	0.96	0.78	0.59

port, destination address, destination port), and is ensured to be a complete TCP session, i.e., starts with TCP SYN and ends with TCP FIN. Since only metadata are required in the experiment, each network flow is stored with a sequence of packets containing the packet size, packet direction, and packet timestamp. DNS queries are pared to ensure only traffic to **.baidu.com* is collected. In total, more than 10,000 network flows and more than 300,000 packets are collected as our dataset.

B. Data Preprocessing

Given the collected traffic data, we process them into feature vectors that can be the input of the discriminator model and generator model in GANs. To do this, we process traffic packets and calculate statistical features including the number of outgoing packets, number of incoming packets, bytes of outgoing packets, bytes of incoming packets, average interval time between packets, and cumulative bytes for outgoing and incoming packets, for each flow. As a result, each flow is represented by its numerical feature vector, and these feature vectors are the training set for generating dynamic traffic patterns in the next step.

C. GANs Model Training

Data Scaling: Data normalization is a useful technique for better training deep neural networks. In order to improve the performance of the GANs model, we first normalize features, i.e., scaling all the feature values into the range from -1 to 1. By analyzing the distribution of each feature's values, we find a few samples have quite large values while the most are within a small range. So we use log function to scale features into range -1 to 1 with the following equation:

$$f' = 2 \times \log_M(f + 1) - 1 \tag{6}$$

where M is an upper bound of f, f is the origin feature values and f' is scaled feature values to train the GANs model.

Net Structure: Both the generator and the discriminator in the GANs model are three dense layered neural networks (exclude the input layer) implemented with Tensorflow library. we use leaky-relu as activation function for both generator and discriminator to speed up training and increase performance. The detailed network structure is shown in Table II.

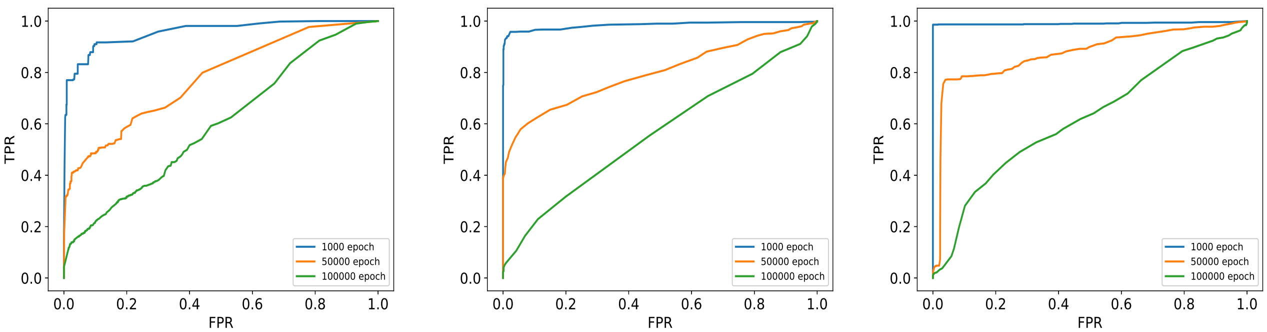
Training: In the experiment, we select 20% data (2,000 flows) as the testing set, and the remaining data as training and validation sets. The parameters in the training are decided by repeated experiments until the GANs can converge well. The discriminator is trained with RMSProp Optimizer [34] with learning rate 10^{-3} and the generator is trained with Adam Optimizer [35] with learning rate 10^{-3} and beta1=0.5. The discriminator's loss function is Wasserstein distance with gradient penalty [33] and generator's loss function is Wasserstein distance [32].

D. Evaluation

To evaluate our defense mechanism, we implement a set of state-of-the-art traffic analysis attacks, including Panchenko's SVM Classifier [36], Herrmann's Bayes Classifier [37], VNG++ Classifier[3], Bandwidth Classifier[3], Time Classifier[3], Liberatore's Bayes Classifier [38], Jaccard's Classifier [39], and Wright's Bayes Classifier[9]. We simulate our proposed defense mechanism against these attacks and evaluate the performance.

1) Performance Metrics: The performance of the FlowGAN is evaluated by both the indistinguishability metric (i.e., IND-CA) defined in Section IV-B and classification metrics in machine learning. Since the attacks are essentially the classification between different traffic, we consider AUC, which measures the entire two-dimensional area underneath a ROC curve. The ROC curve is plotted by True Positive Rate (TPR) and False Positive Rate (FPR), where TPR means rate of a flow is real "normal" flow and the classifier determine the flow is safe, and FPR means rate of a flow is generated flow and the classifier determine the flow is safe in our problem. According to the definition, it can be easily known that AUC ranges from 0 to 1. For a random-guess classifier, the AUC should be 0.5. When TPR is lower and FPR is higher, the AUC is lower, and it indicates the performance of the classification is worse and our countermeasure's defending effect is better.

2) Defense Performance: Table III shows the IND-CA values of the attacks on traffic flows in the testing set with and without our defense. Without the defense from FlowGAN, the IND-CA values are quite high and four attacks achieve



(a) VNG++ (b) Hermann's Classifier (c) Panchenko's Classifier

Fig. 6. ROC Curve at Different Epochs Against Attacks

IND-CA values that are higher than 0.9, which indicates most of these attacks can classify the censored traffic and uncensored traffic. After applying FlowGAN with 100,000 iterations training, we observe significant reductions in IND-CA for all attacks, and the highest IND-CA one is only 0.216. The results show that our system can produce network traffic flows that are indistinguishable from real network flows. With these low IND-CA values, attackers are not able to use these traffic analysis methods to identify and block the generated flows. This indicates our system can be used to protect network flows from censorship.

To extensively evaluate the performance, we also evaluate performance on the flows generated at different iterations (1,000 epoch, 50,000 epoch, 100,000 epoch) in three experiments with the best three attacks methods in Table III (VNG++ classifier, Hermann's Classifier, and Panchenko's Classifier). The ROC curves are plotted in Fig. 6 and the AUC values are calculated in Table IV.

From Fig. 6 and Table IV, we can observe obvious performance improvements with more training iterations. When the GAN model is only trained with 1,000 epochs, the three classifiers can still precisely identify traffic with AUC more than 0.99, because traffic profile has not been fully camouflaged. With more iterations, the performance significantly increases and converges near 0.6 by 100,000 epochs. We can also notice that, in Fig. 6, if a censor needs to ensure true positive rate (TPR) to be more than 90%, the camouflaged traffic has also near 90% probability to pass the censor, which indicates the practical effectiveness of FlowGAN.

3) Overhead: Since GANs features training and flow generation process can be pre-generated offline, the computational overhead can be ignored. The overhead of our mechanism is mainly from the traffic morphing, i.e., compared with the latency/bandwidth of the original flows, the extra latency/bandwidth of generated flows after traffic morphing. However, the overhead is expected and desired, as the packet length and arrival time interval are features that we considered to morph so that the generated traffic can have similar traffic shape as the target traffic. So it depends on how different are the original and the target traffic flows. As a case study, we calculate the

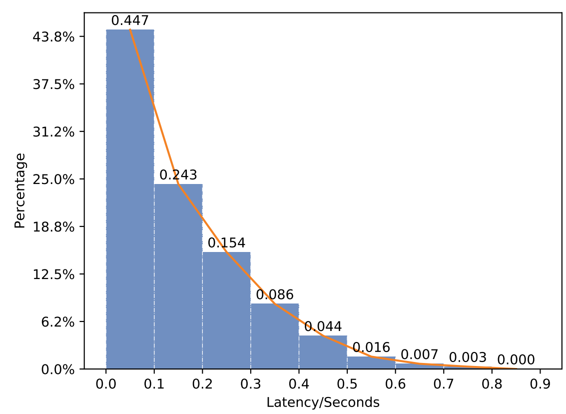


Fig. 7. Latency Distribution

time latency in our experiment that morphs the traffic from *sp0.baidu.com* to *www.baidu.com*. The overall time overhead is evaluated by the average additional time to transfer packages in the origin flow. The result is illustrated in Fig. 7. Most of the latency ($> 96\%$) are less half a second, which indicates our solutions is promising in practical deployment.

VII. Conclusion

In this paper, we present a novel design, FlowGan, to dynamically camouflage a "source" flow to any "target" flow by leveraging generative adversarial networks. FlowGan can be used to thwart the traffic analysis attacks launched by the Internet censorship. Compared with the previous traffic morphing technique which can only morph a source flow to a specific target, FlowGan can dynamically camouflage the flows to any popular application flow. This is expected to prevent the censorship from recognizing and blocking any specific type of traffic. To measure the effectiveness of FlowGan, we introduce a novel anonymity definition, ϵ-indistinguishability, to model the indistinguishability of the target traffic and the morphed traffic. We perform the extensive experimental evaluations based on a real-world dataset collected to baidu.com. The evaluation results including indistinguishability, latency, ROC have well demonstrated its superiority under the state-of-the-art traffic analysis attacks. Our future work includes how to implement the FlowGan and apply it on a large scale anonymous communication system and improve the performance under the real-world network constraints.

Acknowledgment

This work was supported in part by National Science Foundation of China under Grant 61672350 and in part by the China Scholarship Council (201806230109). Haojin Zhu (email: zhu-hj@sjtu.edu.cn) is the corresponding author.

References

[1] J. Victors, M. Li, and X. Fu, "The onion name system," *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 1, pp. 21–41, 2017.

[2] T. Wang and I. Goldberg, "Walkie-talkie: An efficient defense against passive website fingerprinting attacks," in *26th USENIX Security Symposium (USENIX Security 17)*, 2017, pp. 1375–1390.

[3] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail," in *Security and Privacy (SP), 2012 IEEE Symposium on*. IEEE, 2012, pp. 332–346.

[4] A. Panchenko, F. Lanze, J. Pennekamp, T. Engel, A. Zinnen, M. Henze, and K. Wehrle, "Website fingerprinting at internet scale." in *NDSS*, 2016.

[5] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting." in *USENIX Security Symposium*, 2014, pp. 143–157.

[6] M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli, "Traffic classification through simple statistical fingerprinting," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 1, pp. 5–16, 2007.

[7] X. Luo, P. Zhou, E. W. Chan, W. Lee, R. K. Chang, and R. Perdisci, "Httpos: Sealing information leaks with browser-side obfuscation of encrypted flows." in *NDSS*, vol. 11. Citeseer, 2011.

[8] X. Cai, R. Nithyanand, T. Wang, R. Johnson, and I. Goldberg, "A systematic approach to developing and evaluating website fingerprinting defenses," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2014, pp. 227–238.

[9] C. V. Wright, S. E. Coull, and F. Monrose, "Traffic morphing: An efficient defense against statistical traffic analysis." in *NDSS*, vol. 9, 2009.

[10] M. Nasr, H. Zolfaghari, and A. Houmansadr, "The waterfall of liberty: Decoy routing circumvention that resists routing attacks," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 2037–2052.

[11] Censorship of youtube. [Online]. Available: https://en.wikipedia.org/wiki/Censorship_of_YouTube

[12] Y. Liu, R. Zhang, J. Shi, and Y. Zhang, "Traffic inference in anonymous manets," in *Sensor Mesh and Ad Hoc Communications and Networks (SECON), 2010 7th Annual IEEE Communications Society Conference on*. IEEE, 2010, pp. 1–9.

[13] H. Li, Z. Xu, H. Zhu, D. Ma, S. Li, and K. Xing, "Demographics inference through wi-fi network traffic analysis," in *Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on*. IEEE, 2016, pp. 1–9.

[14] Y. Zhu, X. Fu, B. Graham, R. Bettati, and W. Zhao, "Correlation-based traffic analysis attacks on anonymity networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 7, pp. 954–967, 2010.

[15] H. Li, H. Zhu, and D. Ma, "Demographic information inference through meta-data analysis of wi-fi traffic," *IEEE Transactions on Mobile Computing*, vol. 17, no. 5, pp. 1033–1047, 2018.

[16] D. Wang, L. Zhang, Z. Yuan, Y. Xue, and Y. Dong, "Characterizing application behaviors for classifying p2p traffic," in *Computing, Networking and Communications (ICNC), 2014 International Conference on*. IEEE, 2014, pp. 21–25.

[17] H. Li, Q. Chen, H. Zhu, D. Ma, H. Wen, and X. S. Shen, "Privacy leakage via de-anonymization and aggregation in heterogeneous social networks," *IEEE Transactions on Dependable and Secure Computing*, 2017.

[18] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching from a distance: Website fingerprinting attacks and defenses," in *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 2012, pp. 605–616.

[19] L. Zhou, S. Du, H. Zhu, C. Chen, K. Ota, and M. Dong, "Location privacy in usage-based automotive insurance: Attacks and countermeasures," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 1, pp. 196–211, 2019.

[20] H. Li, H. Zhu, S. Du, X. Liang, and X. S. Shen, "Privacy leakage of location sharing in mobile social networks: Attacks and defense," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp. 646–660, 2018.

[21] J. Hayes and G. Danezis, "k-fingerprinting: A robust scalable website fingerprinting technique." in *USENIX Security Symposium*, 2016, pp. 1187–1203.

[22] V. Rimmer, D. Preuveneers, M. Juarez, T. Van Goethem, and W. Joosen, "Automated website fingerprinting through deep learning," in *Network & Distributed System Security Symposium (NDSS)*, 2018.

[23] S. E. Oh, S. Sunkam, and N. Hopper, "Traffic analysis with deep learning," *arXiv preprint arXiv:1711.03656*, 2017.

[24] R. Overdorf, M. Juarez, G. Acar, R. Greenstadt, and C. Diaz, "How unique is your. onion?: An analysis of the fingerprintability of tor onion services," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 2021–2036.

[25] M. Juarez, M. Imani, M. Perry, C. Diaz, and M. Wright, "Toward an efficient website fingerprinting defense," in *European Symposium on Research in Computer Security*. Springer, 2016, pp. 27–46.

[26] X. Cai, R. Nithyanand, and R. Johnson, "Cs-buflo: A congestion sensitive website fingerprinting defense," in *Proceedings of the 13th Workshop on Privacy in the Electronic Society*. ACM, 2014, pp. 121–130.

[27] G. Cherubin, J. Hayes, and M. Juarez, "Website fingerprinting defenses at the application layer," *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 2, pp. 186–203, 2017.

[28] W. Cui, J. Yu, Y. Gong, and E. Chan-Tin, "Realistic cover traffic to mitigate website fingerprinting attacks," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2018.

[29] Z. Ling, J. Luo, K. Wu, W. Yu, and X. Fu, "Torward: Discovery of malicious traffic over tor," in *INFOCOM, 2014 Proceedings IEEE*. Citeseer, 2014, pp. 1402–1410.

[30] D. Barradas, N. Santos, and L. Rodrigues, "Effective detection of multimedia protocol tunneling using machine learning," in *Proceedings of the 27th USENIX Security Symposium*, 2018.

[31] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[32] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International Conference on Machine Learning*, 2017, pp. 214–223.

[33] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in Neural Information Processing Systems*, 2017, pp. 5769–5779.

[34] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[36] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website fingerprinting in onion routing based anonymization networks," in *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society*. ACM, 2011, pp. 103–114.

[37] D. Herrmann, R. Wendolsky, and H. Federrath, "Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier," in *Proceedings of the 2009 ACM workshop on Cloud computing security*. ACM, 2009, pp. 31–42.

[38] M. Liberatore and B. N. Levine, "Inferring the source of encrypted http connections," in *Proceedings of the 13th ACM conference on Computer and communications security*. ACM, 2006, pp. 255–263.

[39] Q. Sun, D. R. Simon, Y.-M. Wang, W. Russell, V. N. Padmanabhan, and L. Qiu, "Statistical identification of encrypted web browsing traffic," in *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*. IEEE, 2002, pp. 19–30.