Website Fingerprinting Attack on Anonymity Networks Based on Profile Hidden Markov Model

Zhongliu Zhuo, Yang Zhang, Zhi-li Zhang, *Fellow, IEEE*, Xiaosong Zhang, and Jingzhong Zhang

***Abstract*— Website fingerprinting attacks can reveal the receiver in anonymous networks and cause a potential threat to users' privacy. Previous studies focus more on identifying individual webpages. They also neglect the hyperlink transition information, because it induces extra "noise" to classify the original webpage. However, it is a common scenario that the users surf a website by clicking hyperlinks on the webpage. In this paper, we propose a website modeling method based on profile hidden Markov model (PHMM) which is widely used in bioinformatics for DNA sequencing analysis. Our technique explicitly accounts for possible hyperlink transitions made by users when fingerprinting a target website, and therefore can work in a more realistic environment than existing methods. Using SSH and Shadowsocks, we collect various data sets and conduct extensive evaluations. We also show that our approach could work both in webpage and website identification in a closed world setting. The experimental results demonstrate that our website fingerprinting is more accurate and robust than existing methods.**

***Index Terms*— Anonymous network, website fingerprinting attacks, profile hidden Markov model.**

I. Introduction

PEOPLE increasingly rely on Anonymous Communication (AC) systems to ensure their privacy and anonymity when surfing the Internet nowadays. As for AC system designers, one of the key requirements is to guarantee three types of anonymities: the sender anonymity, the receiver anonymity and the unlinkability of senders and receivers [1]. However, a quick literature survey [2]–[10] reveals that many AC systems, for example, Tor [11], Shadowsocks [12], Anonymizer [13], I2P [14], SSH or VPN tunnel, fail to protect the receiver anonymity. Among the attacks to break the receiver anonymity, website fingerprinting (WFP) attack poses a serious threat to users' privacy. The basic idea of this attack is that an attacker passively monitoring the network traffic (thereafter referred to as a *passive attacker*) could figure out which webpage a user is visiting by exploiting characteristics of packet length, order and timing information and so forth, even if packet payloads are encrypted.

Despite many WFP attack methods existed (as summarized in table I), previous studies focused primarily on fingerprinting individual webpage, instead of a web*site*. In addition, these methods only consider discrete visits to a target webpage, but ignore subsequent visits (e.g., after transitioning to another page on the site by following a hyper-link). In other words, they implicitly assume that the one-time visit to a target webpage is "clean" without "noise."[1] It is also assumed that a passive attacker could easily separate the target traffic from the background traffic and successfully extract the packet sequence from the beginning to the end of a one-time loading of the target webpage. This is clearly a nontrivial task for the attacker [23]. Unfortunately, the "noise" contained in the network traffic may contaminate the feature vector extracted by the previous methods, thus confusing the trained classifier. It is common that users surfing the web often follow links embedded in a webpage. Unlike webpage fingerprinting, web*site* fingerprinting attempts to uncover if a user is visiting a target website, including any page of the site and making transitions among them through hyper-links. In performing the fingerprinting attack, the attacker wants a high degree of confidence in the results [9]. Hyper-link transitions between pages within a website could in fact be leveraged to help improve the accuracy of web*site* fingerprinting.

In this study, we present a web*site* fingerprinting attack technique by exploiting webpage transitional information. The technique is based on Profile Hidden Markov Model (PHMM) [24], which has been widely used for DNA sequencing analysis. Using our technique, an attacker can build a web*site* model taking different webpages from the same site, and use this model to determine if a sequence of target page loads belong to the same website. To the best of our knowledge, we are the first to use the PHMM for fingerprinting an website instead of individual webpages. Our proposed method could handle traditional single webpage fingerprinting as well as hyper-link transition scenario, and it is more accurate and robust than existing WFP methods.

The major contributions of our paper are summarized as follows:

1) We propose a PHMM based WFP attack technique that works in a more realistic scenario than existing webpage fingerprinting methods.

Manuscript received February 26, 2017; revised June 21, 2017 and September 22, 2017; accepted September 29, 2017. Date of publication October 13, 2017; date of current version January 29, 2018. This work was supported by the National Natural Science Foundation of China under Grant 61402080, Grant 61502086, and Grant 61572115. The work of Z. Zhuo was supported by the China Scholarship Council for his study in University of Minnesota, Twin Cities. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Tomas Pevny. *(Corresponding author: Xiaosong Zhang.)*

Z. Zhuo, X. Zhang, and J. Zhang are with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: zhuozhongliu@126.com).

Y. Zhang and Z.-l. Zhang are with the Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2017.2762825

[1]With respect to the first packet sequence, the subsequent webpage's sequence of the same site is defined as a "noise".

TABLE I
KNOWN WEBSITE FINGERPRINTING ATTACK METHODS

	AN	Year	Attack Method	Defense Method	Accuracy[1]	Features	Dataset[2]
1. Liberatore *et al.*[3]	SSH	2006	JC & NB	Packet padding[15]	≈ 75%	Packet size, direction	2000
2. Shi *et al.*[16]	Tor	2009	Cosine similarity	N/A	50%	Packet size, intervals	20
3. Herrmann *et al.*[7]	Tor, JAP, openssh,VPN, stunnel	2009	Multinomial Naive-Bayes classifier	N/A	>90%	Packet size frequency distribution, flow direction	775
4. Panchenko *et al.*[4]	Tor, JAP	2011	SVM	Decoy pages	>50%	Ad-hoc HTTP features	775
5. Cai *et al.*[9]	Tor, SSH	2012	DLSVM,HMM	HTTPOS[17], RP[18]	>80%	Packet size, direction etc.	100
6. Wang *et al.*[6]	Tor	2014	Ameliorated K-NN	Traffic morphing[19], HTTPOS[17], Decoy pages [4],BuFLO[20]	90%	General features Packet size, direction timing, burst, etc.	90/9000
7. Kwon *et al.*[10]	Tor	2015	Cart, C4.5, K-NN	N/A	98%	Tor cell statistics etc.	50/950
8. Gu *et al.*[21]	SSH	2015	Naive Bayes	N/A	75.9%	Packet ordering, size, etc.	50
9. Panchenko *et al.*[5]	Tor	2016	SVM	N/A	90%	cumulative sizes	90/300,000
10. Hayes *et al.*[22]	Tor	2016	RF + K-NN	Traffic morphing[19], HTTPOS[17], Decoy pages [4] BuFLO[20], Tamaraw[15]	91%	Packet statistics, Transmission time, Packet ordering, etc.	55/100,000
11. Jahani *et al.*[8]	Tor	2016	FFT	N/A	95%	Packet size, direction	500
12. Our method	SSH, SS	2017	PHMM	MTU padding, Dummy packet	99%	Packet size, direction	200/2000

JC: Jaccard coefficient **AN**: Anonymous Network, **NB**: Naive Bayes, **DLSVM**: Damerau Levenshtein Support Vector Machine, **RP**: Randomized Pipelining, **HMM**: Hidden Marchov Model, **K-NN**: K Nearest Neighbor, **RF**: Random Forest, **FFT**: Fast Fourier Transform

[1] Reported closed world accuracy and no defense methods were taken.
[2] The dataset column formatted as "closed world / open world website number (if existed)" indicates the number of classes (websites) considered during classification.

2) We develop a novel solution for building the web*site* PHMM, and implement a prototype of the proposed technique.
3) We conduct both closed world and open world experiments by collecting datasets via SSH and Shadowsocks. The evaluation results demonstrate that our technique outperforms existing methods in identifying single target webpage visits as well as fingerprinting target websites where users may browse multiple pages by following hyper-links.
4) Finally, we explore two possible countermeasures to defend our method and assess their effectiveness.

The remainder of this paper is organized as follows. In Section II, we summarize existing WFP attacks and compare them with our proposed technique. In Section III we introduce the basics of an WFP attack and provide an overview of PHMM. In Section IV we describe our proposed web*site* fingerprinting attack technique in details. An implementation of the proposed technique is presented in section V. In Section VI we describe the dataset collection process and present the evaluation results. In Section VII we discuss and explore two possible countermeasures against our proposed WFP attack technique. The paper is concluded in Section VIII.

II. RELATED WORKS

A. Website Fingerprinting

Website Fingerprinting is a hot research topic in recent years [4]–[8], [10], [22], [23], [25]. A taxonomy of existing WFP attacks and our method listed in table I is shown in Fig.1. Website fingerprinting attack was first carried out by Cheng and Avnur [26] in 1998. They exploited the unique size of each downloaded object to identify the page visited by users. Similarly, Hintz [27] used the set of transfer sizes for a given webpage to comprise that pages fingerprint. Sun *et al.* [28] used Jaccard coefficient as similarity measure to find the best match webpage. These prior works provided a first-hand knowledge to this new kind of attack.

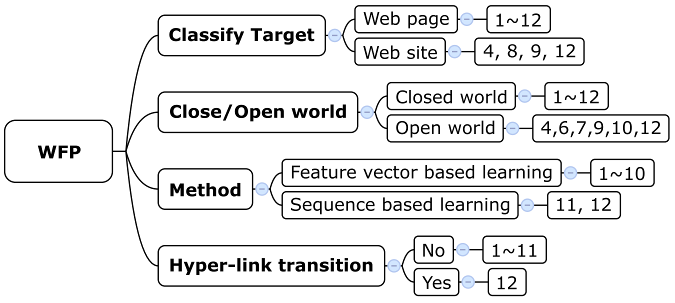


Fig. 1. WFP Taxonomy, the number in the figure corresponds to the paper number in table I. Feature vector based learning methods transform the raw network packet sequence into a feature vector. Sequence based learning methods directly use raw packet sequence.

Bissias *et al.* [29] used the packet size and inter-arrival time distributions to build profiles for individual webpages, they compared the observed traffic to profiles of known sites by using cross correlation score and achieved about a 25% accuracy. After that, Liberatore and Levine [3] only used packets' direction and length, they compared the Jaccards coefficient with the Bayes classifier. On average the Jaccards coefficient performs better than the Bayes classifier. They also evaluated how static packet padding affects their method. Herrmann *et al.* [7] introduced the WFP attack into the anonymous networks such as: JAP [30], Tor [11], OpenSSH, OpenVPN, Stunnel, and Cisco IPsec-VPN. Their multinomial naive bayes classifier outperforms Liberatore's method. Their classifier correctly identifies above 90% of requests on a sample of 775 sites for the single hop systems, but the result on JAP is 20% and only 2.95% on Tor.

Dyer *et al.* [20] presented a comprehensive study on the countermeasures against the WFP attacks and they found low-level traffic analysis countermeasures, such as traffic morphing still cannot effectively protect users' privacy. Cai *et al.* [9] utilized the DLSVM for single webpage and HMM for website fingerprinting. Their attack can determine which webpage a victim is visiting with over an 80% success rate, and over a 90% accuracy when visiting a particular website. In their website model, they created a model with states corresponding to page templates instead of individual pages. They made this assumption in order to simplify their model. Each webpage set with same template corresponds to an HMM state, and state transition probabilities represent the probability that a user would navigate from one page set to another. However, it is a nontrivial work to build the transition probabilities and initial state probabilities of a website, and their study only considered two websites.

Wang *et al*. [6] came up with a weighted KNN classifier for the WFP. Their 3736 dimension features were manually extracted by researchers. Their KNN method can learn the feature's weight and they accomplished a 90% accuracy in a closed world setting. Kwon *et al*. [10] first demonstrated that the WFP can be applied to Tor hidden services. They successfully identified 50 monitored hidden service with 98% accurate rate in the closed world setting.

Gu *et al*. [21] concentrated on the WFP attack in a multi-tab browsing scenario. They first made use of the mahalanobis distance to determine whether the unknown traffic is overlapped. Their Naive Bayes classifier ran on the top 50 ranked websites in Alexa. They showed the possibility to identify two pages respectively, and they could classify the first tab with 75.9% accuracy, and the second tab with 40.5%. Unlike their work, we took the advantage of the first tab and the second tab hidden relationship to improve our accuracy to identify this particular website instead of identifying those webpages separately.

Panchenko *et al*. [5] built a classifier for website fingerprinting: they used 51 different non-index pages to represent a website. Through a 10-fold cross validation they achieved an 85.99% accuracy rate. They further investigated several tactics to improve the success probability for their website fingerprinting. However, in reality their method has the similar assumptions as traditional methods, and they did not consider the subsequent visit or hyper-link transition as we do.

Hayes and Danezis [22] proposed the k-fingerprinting attack. Their method outperforms the current state-of-the-art attacks even against website fingerprinting defenses. They demonstrated the k-fingerprinting is more accurate and faster than other state-of-the-art WFP attacks They also showed the importance of features by using gini coefficient.

Juarez *et al*. [23] gave a realistic testing of recent proposed WFP attacks on Tor network. They showed that previous WFP attacks were based on some important assumptions, for instance, the user browsing behaviors, browser settings and adversary capabilities etc. In a more practical scenario, however, previous attacks were not effective. In our work, we try to relax the previous assumption. To improve the practical feasibility, we consider hyper-link transition situation, instead of assuming users only open one page per visit.

B. Traffic Classification

Since WFP attack basically is a multi-label classification problem, we could get new ideas and inspirations from traffic classification recent advances. He *et al*. [31] proposed an attack to uncover the under traffic types hidden in Tor circuit. Using features like burst volumes and directions, they constructed PHMMs for four types of traffic, *i.e*., P2P, instant message(IM), web browsing and file transfer protocol(FTP). And they achieved a 92% accuracy. Inspired by this study, we employed the PHMM for the WFP attack. Unlike traditional studies, we take the following aspects into consideration: 1) hyper-link transition, 2) web*site* fingerprinting instead of webpage fingerprinting.

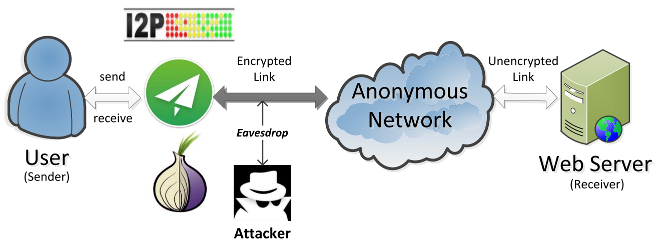


Fig. 2. WFP attack model.

III. Preliminaries

A. Website Fingerprinting Attacks Basics

The ultimate goal of the passive attacker in a WFP attack is to find which the webpage a user is visiting. The basic attack model is shown in figure 2.

Users first select one anonymous network to send their traffic. Then the attacker observes passively on the link, and he cannot modify the transmission (*i.e*. decrypt the packet). The adversary eavesdrops the traffic between the user and the first hop of the route (*i.e*. the customized proxy of Shadowsocks, the entry guard of Tor or the outbound gateway of I2P). Then he will try to match the pattern of those traffic (such as packet sizes, inter packets time, total transmitted length etc.) to a previous trained classifier, the correct result would be the exact site the user just visited. In this way, the receiver anonymity is compromised.

Existing WFP attacks have several assumptions. Juarez *et al.* [23] categories them into three types: user settings, adversary capabilities and the nature of the web. As for user settings, previous attacks assume that a user only browse one page after another, leaving only one tab open, and never save cache to speed next loading. For adversary capabilities, early works assume the adversary can detect the beginning and the end of the webpage loading. This is a strong assumption, in a realistic network environment, it is very hard to separate the target web request correctly from the background traffic [32]. For the nature of the web, traditional WFP attacks do not address the issues such as the websites may have different language versions, browsers with various kinds and versions, website update etc. Conventional works gathered data with Tor browser, and by default the HTTP pipelining is enabled. However, randomizes the pipeline of HTTP requests is ineffective defense evaluated in [6], [9], and [25]. Also, HTTP pipelining is disabled or not implemented in most modern browsers, in our paper we thereby assume users do not enable HTTP pipelining.

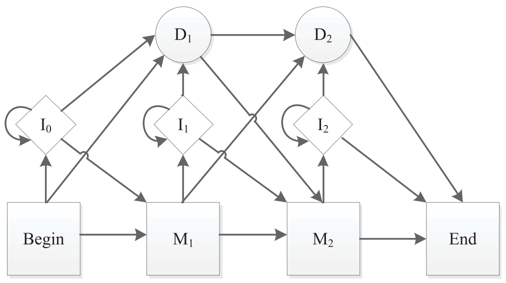


Fig. 3. Example Profile HMM architecture of length 2.

Previous works have two settings to evaluated their methods, namely the closed world and the open world. In the closed world setting, a user only visits a limited k monitored webpages and k is far less than the real number of webpages. In the open-world setting, an adversary is only interested in k targeted pages despite users may access other pages which are beyond adversary's target.

B. Profile HMM

The PHMM is first introduced by Krogh *et al.* [33] and a review of PHMM can be found in [24]. In the past, PHMM was mainly used for biological sequence analyses, such as homological gene sequence searching (e.g. evaluating a new sequence for membership in a family of sequences) and gnomic sequence classification etc.

PHMM is based on the standard HMM theory. It has several advantages over traditional HMM model. Firstly, PHMM exploits positional information contained in the observation sequences, whereas standard HMM ignores it. Secondly, PHMM allows null transitions, which are necessary so that the model can match sequences that include insertions or deletions. The structure of a PHMM is illustrated in figure 3.

In figure 3, the circles are delete states, the diamonds are insert states, and the squares are match states. Insert states give a chance for possible values to be inserted at a given position in a sequence before a match state or a delete state. For instance, given a sequence of symbols, if the sequence has more symbols between the match states M_i and M_j, the path through the model will transit from M_i to one or more insert states until it "consumes" those symbols, then it will transit from the insert state to M_j matching state. Similarly, delete states allows symbols to be removed at a given position. Positions using delete states in the model indicate such places have no corresponding characters in the given sequence. For example, if a symbol sequence has no residues corresponding to match states from M_i to M_j of the model, the path will transit from match state M_{i-1} to a delete state, then it will pass through additional delete states until it finally matches the state M_{j+1}.

IV. Profile HMM for Website Fingerprint Modeling

In this section, we first introduce our high-level idea of this method. Then we discuss why this model could work for the WFP attacks, and what causes the traditional methods to be unreliable. After that, we explain how to apply the PHMM to build a web*site* fingerprint that could work in hyper-link transition browsing situations. Finally, we describe the way to estimate the parameters for the model.

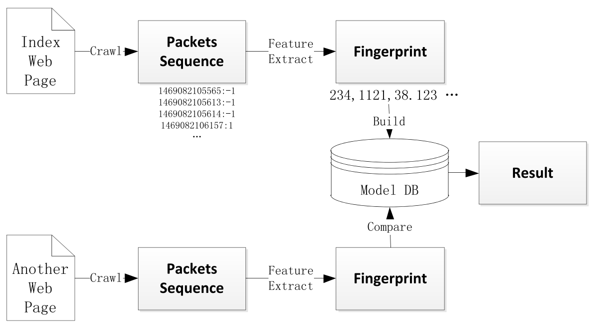


Fig. 4. Traditional website fingerprinting attack process.

A. Intuition

Traditional approaches focus on identifying single webpage, the basic steps are presented in figure 4. Firstly, the authors crawled the webpages for feature extraction. Packet sequence was represented simply as: time stamp and packet direction. Note that in some papers the authors just used packet direction but ignored the packet size due to packet padding techniques in some ANs (such as Tor). Based on the "clean" sequence of each individual visit to a webpage, they extracted its feature vector to represent this page. Finally, when a new page visiting sequence came, the authors compared the unknown features with the known ones to find the best match.

However, traditional methods simply assumed that they could observe the start and the end of the traffic, and another strong assumption is that they could get rid of the background traffic. But in reality this could not be possible. According to the traditional methods, they transformed the "ideal" packet sequence into a self-defined feature vector. If we feed the simulated hyper-link transition browsing (the subsequent visit) traffic into the traditional models, the extended traffic of the second webpage would severely affect the original packet sequence feature, causing a mismatch.

Our proposed method could fix the gap. Intuitively, each webpage contains some special sequence of objects (such as css files, pictures etc.) that will trigger a series of packet sequences uniquely characterize that page, even if they transmitted through an encrypted tunnel — that means, in this paper we assume the packet length is not padded to the same size. Here, we can imagine them as gene sequences. It is known that some genes in an organism would change due to environmental factors, but the essential "functionality" genes can still be the same. When considering the WFP situation, the original packet sequence of a particular website may vary because of noise or be "polluted" by a subsequent visit via a hyper-link, which causes two network flows to intertwine with each other. Although the disorder part couldn't be matched, we argue that there will still be some key elements which act as a "functionality gene" that could help us identify this particular website. Just like biological sequences often only

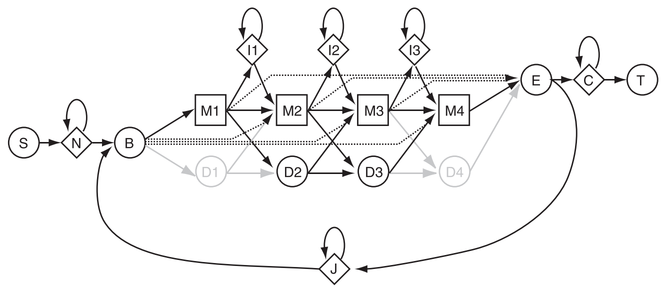


Fig. 5. A profile HMM architecture used by the HMMER called the Plan 7 [35], this model could also be adapted to sequence match in website fingerprinting attack.

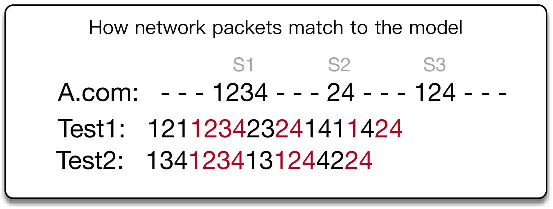

How network packets match to the model
S1 S2 S3
A.com: - - - 1234 - - - 24 - - - 124 - - -
Test1: 121123423241411424
Test2: 134123413124424224

Fig. 6. A toy example to explain our method. The first line shows the "key domains" that the model tries to find, which suggests they have a higher chance to be seen in such position of the sequence. The dash line denotes the packets in this place are not treated as conserved "key domains".

share homologous domains,[2] it is more useful to identify those "key" elements (subsequences) in the target, while treating the remainders of the sequences as non-homologous [34].

B. Homologous Sequence Search in WFP Attack Scenario

Inspired by the idea that to identify the conserved key domains in the protein family, we want to design a model that could also be applied to network packet sequence search in a more realistic WFP scenario.

In general, if a sequence somehow is equivalent to the original model, as described in [33], the path through the model will be in a linear fashion: $Begin \rightarrow M_1 \rightarrow \ldots \rightarrow M_n \rightarrow End$. If the sequence contains a gap in relative to the model, 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. But in real world situations, this *global alignment*[3] is hard to see. Because many mutations exist in the biological sequence, they will add a lot of uncertainties (noises) to the original sequence. Similarly, the real network environment also contains noise. Thus, in biological homologous sequence search, the PHMM is usually a model of several conserved domains, not necessarily a model of the whole target sequence. A network packet sequence is somewhat similar to this, a specific object's network sequence could be buried in a longer target sequence, and there might be more than one conserved domains in the target sequence. In such case, the *local alignment* algorithms are used to find a high-scoring alignment between a subsequence of the target sequence and a part of the query model [35].

The original PHMM in figure 3 can only match one sequence as a whole and then terminate, which cannot satisfy the need in real situations. To match multiple subsequences, additional states are added to the original PHMM. As can be seen in figure 5, apart from previous mentioned states: match state [**M**], delete state [**D**], insert state [**I**], begin state [**B**], end state [**E**]. The extended version of PHMM has more states: start state [**S**], terminal state [**T**], and state [**N**],[**C**],[**J**]. States N,C,J are designed to generate a "random" sequence that is not aligned to the main model.

[2]In evolutionary biology, homologous domains refers to protein or DNA regions that sharing similar sequences.

[3]Meaning the sequence that the model generates usually starts at the first match state and ends at the last match state.

Parameter estimation for the model works as follows. The probability parameters in the main model (M,D,I) are learned from data, which will be introduced in section IV-D. The other states (S,N,C,T,J) combined with entry probabilities from B and exit probabilities to E control the algorithm dependent features of the model. Those parameters are used to control the model for matching various kinds of local or multi-hit alignments. The algorithm dependent parameters are not learned from data, they depend on the alignment style. The self loop transition probability for N, C and J controls the length of unaligned segments in the prefix, suffix and intervening regions in the sequence. This allows the background unaligned packet subsequences to be in the original sequences. If E $\rightarrow$ J transition probability is set to 0, a sequence may just only have one domain. If it is bigger than 0, more than one domain can be found. In our setting, we set N, C, J self-loop transition probability to be $\frac{L}{L+3}$ and transition probability of E $\rightarrow$ J is set to be 0.5, where L is the gene sequence length. Please refer to [34] for more insights on the self-loop transition parameter setting. The general idea is that if the L is longer, the more likely that sequence is favored by the PHMM, causing a bias to compare two scores. Thus Eddy [34] choose this setting to make the score independent of L.

The entry/exit probabilities $B \rightarrow M_i$ and $M_i \rightarrow E$ $(i = 1, 2, \ldots, N)$ are uniform, which are $\frac{2}{N(N+1)}$ and 1 respectively, where N is the node number[4] in the model. A uniform entry/exit probability means that totally $\frac{N(N+1)}{2}$ possible local alignments could match with query sequence with equal probability.

We try to illustrate our approach using an example. Suppose after we trained the model for A.com, we have learned the model parameters for the main model (M,D,I).

Suppose M1, M2, M3, and M4 try to match packet 1, 2, 3 and 4 respectively. The model has learned that A.com has 3 potential subsequences (suppose S1, S2 and S3 come form A's index page and 2 other subpages of A's) shown in figure 6. A possible viterbi path for test case 1 through the model could be: SNNNBM1M2M3M4EJJBM2D3M4EJJJBM1I1M2D3M4ECT. Since the user could browse S3 ahead of S2, in this case the model could still work. A possible viterbi path for test case 2 through the model could be: SNNNBM1M2M3M4EJJBM1M2D3M4EJJBM2D3M4ECT.

[4]A group of three states M,D,I at the same consensus position in the alignment is called a "node", i.e. figure 5 has 4 nodes

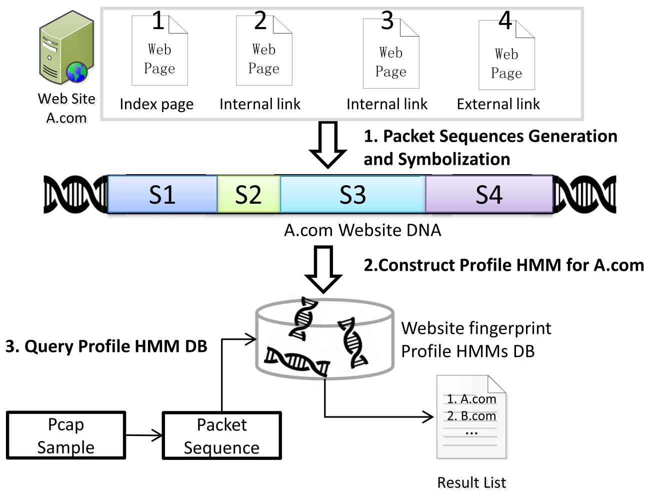


Fig. 7. Profile HMM based website fingerprinting attack.

To summarize why this model works in website fingerprinting attack, we list the following reasons: (1) Additional states of N, C and J enable us to counter the background/noise traffic in the network environment. (2) The transitions of E $\rightarrow$ J and J $\rightarrow$ B allow the matches of different domains to be possible. Thus we could find multiple key elements in the network traffic, regardless of the user webpage visiting sequence order. This is true, because we don't know the user behavior in advance. Most importantly, because of this E $\rightarrow$ J and J $\rightarrow$ B transition, we can build a web*site* fingerprinting model. (3) Also, with insertion and deletion states property, this allows us to have tolerance to match the subsequences. Such as packet dropping (delete state) and retransmission (insert state) etc.

C. Modeling the Website Using Profile HMM

Last subsection introduces the model and why it could be adapted to the WFP scenario. In this subsection, we illustrate how to apply the sequence matching technique from gnomic analyses to build a model for a web*site*.

Figure 7 shows our method's procedures. Note that our method could be adapted to both web*site* fingerprinting and webpage fingerprinting, here we only introduce web*site* fingerprinting. Assuming that the attacker can use a subset of available pages for training of a website, due to enormous amount of existing pages in the website. Suppose A.com contains 4 hyper-links, 2 internal links[5] which go to other sub-domains of A.com, and 1 external link[6] which connects to other domains that might be related to A.com. This hidden relationship, as we notice, could be exploited to find A.com. Like traditional procedures, individual webpages are crawled and filtered to get the corresponding packet sequences. Then for each packet sequence, we need to take another symbolization step (see section V-B). The symbolization can reduce the number of the observable states in HMM and make the models more robust [36]. All the sequences from this target website formed a single "gene" sequence. For training purposes, we could get N training sequences for later MSA (Multi Sequence Alignment) treatment and PHMM building. The reasons for the MSA and how to estimate PHMM parameters will be introduced in section IV-D. Finally, similar to existing methods, we also need to compare the observed sequence with the known ones.

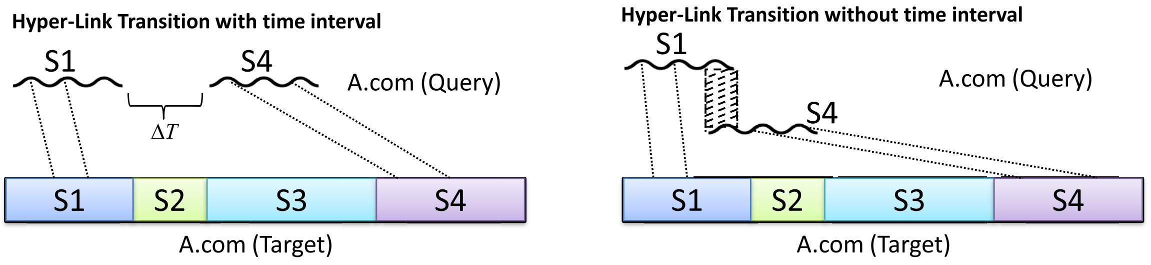
Why could our model work for web*site* fingerprinting attack, and more specifically why it could work for hyper-link transition browsing scenario? As can be seen in figure 8, suppose the model for A.com is the target, whose query sequence is shown above the target model. For hyper-link transition situation, suppose the two continuous visits are S1 and S4 (a user first loads the landing page S1 and then follows the link to S4), and the time gap is ΔT. Suppose attackers do not use the time gap to split browsing traffic, otherwise it will change to per webpage identification and neglect the hyper-link transition information. In the second situation of hyper-link transition browsing, suppose a user loads S4 shortly after S1. The shaded part denotes these two flows intertwine with each other. Note that no matter which the next webpage the user is trying to fetch, it can still match the target model. That is to say, our model can know which website the user is visiting regardless of knowing user's browsing order. This is because we do not match the entire query sequence to the model. Instead, our model tries to match the subsequences, like the dotted regions shown in figure 8. Also, we do not try to match the shaded part in the short time hyper-link transit browsing situation, because the PHMM could not have a consensus at that part.

D. Estimate the Parameters for the Main Model

To construct a PHMM given labeled training data, the parameters for the main model (M/D/I) are determined by the maximum likelihood estimation. The PHMM (λ) is determined by A, E, π, where A denotes the state transition probability matrix, E denotes the emission probability matrix (for match and insert states), and π denotes the initial state probability distribution.

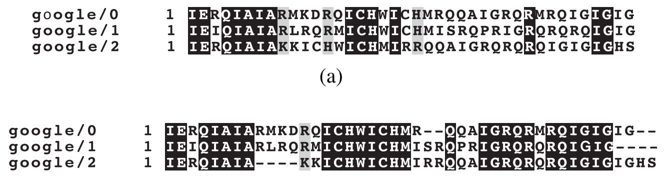
The MSA is commonly used in bioinformatics applications, such as to search evidence that the sequences diverged from a common ancestor [37]. The MSA is the alignment of more than two sequences in such a way that the homologous parts of each of these sequences are aligned to each other in the same columns, to accomplish this, gaps ("–") can be inserted into sequences. Several reasons let us do the MSA before the parameter estimation. To begin with, Bhargava and Kondrak [38] show if a PHMM is constructed with unaligned sequences, it will be easy for the constructing algorithm to stuck around local optima. Moreover, the MSA discovers the common subsequences and puts them in the same column which facilitates the later PHMM parameter estimation. And we later prove that the MSA actually improves the identification accuracy as well. Figure 9 depicts a real example of the MSA where three subsequences extracted from separate visits to google.com. The letters in the figure describe the packet size and direction ordered by arrival time (how

[5]Internal link is one that points to another page on the same domain.
[6]External link is one that points to another page not on the same domain.



Hyper-Link Transition with time interval
S1
S4
ΔT
A.com (Query)
S1
S2
S3
S4
A.com (Target)
Hyper-Link Transition without time interval
S1
S4
A.com (Query)
S1
S2
S3
S4
A.com (Target)

Fig. 8. Our web*site* fingerprinting could model the hyper-link transition browsing situation. The target represents the model we have built for A.com. The wave line (query) denotes observed network packet flow sequences, and the region between two dotted lines represents the possible matched domains with respect to the target model. The shaded part means the two network flows are mixed with each other.



```
google/0     1 IERQIAIARMKDRQICHWICHMRQQAIGRQRMRQIGIGIG
google/1     1 IEIQIAIARLRQRMICHWICHMISRQPRIGRQRQRQIGIG
google/2     1 IERQIAIAKKICHWICHMIRRQQAIGRQRQRQIGIGIGHS
```

(a)

```
google/0     1 IERQIAIARMKDRQICHWICHMR--QQAIGRQRMRQIGIGIG--
google/1     1 IEIQIAIARLRQRMICHWICHMISRQPRIGRQRQRQIGIG----
google/2     1 IERQIAIA----KKICHWICHMIRRQQAIGRQRQRQIGIGIGHS
```

(b)

Fig. 9. MSA visualization of google.com (3 instances), the column of "1" can be ignored. (a) Unaligned subsequences. (b) Aligned subsequences.

TABLE II

AN EXAMPLE OF ALIGNED SUBSEQUENCES AND CORRESPONDING STATES

1	I (M_1)	E (M_2)	-	C (I_2)	Q (M_3)
2	I (M_1)	E (M_2)	-	-	Q (M_3)
3	I (M_1)	E (M_2)	-	-	Q (M_3)
4	V (M_1)	- (D_2)	R (I_2)	-	R (M_3)

to symbolize those letters is presented in section V-B). The columns with box shape shade have the consensus, which means it's highly likely that in next a visit to google.com the same packet (same direction and size) will occur in such position.

After the MSA, the aligned sequences are fed for PHMM parameter estimation. We denote the number of transition times from state k to l as A_{kl}. The probability of transition from state k to state l, is shown in equation (1). Similarly, in equation (2), the probability of state k emitting symbol a is calculated by counting the number of times $E_k(a)$ that emission is used in the alignment.

$$a_{kl} = \frac{A_k l}{\sum_{l'} A_k l'} \tag{1}$$

$$e_k(a) = \frac{E_k(a)}{\sum_{a'} E_k(a')} \tag{2}$$

Table II shows four aligned and extracted subsequences. Before the probabilities are computed, we adopt a simple heuristic rule to determine the state of each column (so we can calculate the initial distribution of states π). If a column has half or more symbols present (instead of a gap), this column is regarded as a match state, otherwise it is labeled as an inserted state. Delete states are used to "jump" the match states. So the "$-$" in column 4 and 5 are not delete states. The emission probabilities from match state M_1 to M_2 is $a_{M_1 M_2} = 3/4$, $a_{M_1 D_2} = 1/4$. And $e_{M_1}(V) = 1/4$, $e_{M_1}(I) = 3/4$.

After constructing the PHMM $\lambda = (A, E, \pi)$, this model is able to predict observed sequences. To match a PHMM, we can either use Viterbi equations to find the most probable alignment π^* of a sequence x and its probability $P(x, \pi^*|\lambda)$ or use forward algorithm to sum over all possible paths to get $P(x|\lambda)$.

V. IMPLEMENTATION

In this section, we come up with two approaches to organize packet sequences for representing a webpage. We also introduce our detailed implementation of the PHMM.

In this paper, we select Shadowsocks [12] to model the web*site* fingerprint. We also implement our proposed method to build the webpage fingerprint for both Shadowsocks and SSH. Like the SSH tunnel, Shadowsocks works as a single hop proxy. Shadowsocks is a cross-platform SOCKS proxy which uses a pre-shared symmetric key to encrypt in-tunnel traffic. It is a trending project on the Github and has been widely used by people to circumvent the Internet censorship. Unlike Tor, which multiplexes multiple connections of the browser into a single TCP connection, Shadowsocks and SSH transfer multiple connections independently, and neither do they have a fixed packet length nor do they merge concurrent messages into a single request packet. To show the feasibility of using PHMM in the WFP attack, we choose these ANs for simplicity reasons.

A. Packet and Flow Sequence Generation

To build a PHMM for a web*site* or a webpage, an attacker first needs to filter the packets. In packet preprocessing step, we filter out SYN, ACK with payload length equals zero for the sake of being large in numbers (similar to MTU packets) as shown in figure 12. Therefore, they are not useful for sequence learning. As for flows, we ignore the flow[7] that contains just 1 packet and the flow with zero length. Based on filtered packets and TCP flows, we consider two possible ways to organize packet sequences to represent the webpage. One way is to organize the sequences by flow length, see Alg.2, and the other is just ordered by packet arriving order, see Alg.1.

The intuitive reason for developing a flow sequence algorithm is as follows. Normally, the client will generate separate requests (flows) to get the web contents of vary length.

[7]The flow is defined as packets sharing the 5-tuple: source IP, source port, destination IP, destination port, transport protocol

Algorithm 1 Packet Sequence Generation Algorithm

```
Input: Filtered TCP packets $P_f$
Input: Gene Max Length L
Output: Symbolized sequence $Seq$
1) $Seq \leftarrow Null$, $Out \leftarrow -1$, $In \leftarrow 1$, $Current \leftarrow 0$
2) for each TCP packet $j \in P_f$ do
3)   if $Current$ <L then
4)     if $j$ is outgoing packet then
5)       $j.length \leftarrow j.length \times Out$
6)     else
7)       $j.length \leftarrow j.length \times In$
8)     end if
9)     $Seq \leftarrow Seq + Symbolize(j.length)$
10)    $Current \leftarrow Current + 1$
11)  end if
12) end for
13) if $Seq.length$ <L then
14)   return AppendtoL("-",$Seq$)
15) else
16)   return $Seq$
17) end if
```

Algorithm 2 Flow Sequence Generation Algorithm

```
Input: Filtered TCP flows $S_f$
Input: Flow Check Length $M$
Input: Gene Max Length L
Output: Symbolized sequence $Seq$
1) $Seq \leftarrow Null$ $Out \leftarrow -1$, $In \leftarrow 1$, $Current \leftarrow 0$
2) $S_f \leftarrow reorderflowsBysize(S_f)$
3) for each TCP flow $s \in S_f$ do
4)   if $Current$ <L then
5)     if $s.size$ <$M$ then
6)       $Seq \leftarrow Seq + PacketSeqGenAlg(s, s.size)$
7)       $Current \leftarrow Current + s.size$
8)     else
9)       $Seq \leftarrow Seq + PacketSeqGenAlg(s, M)$
10)      $Current \leftarrow Current + M$
11)    end if
12)  end if
13) end for
14) if $Seq.length$ <L then
15)   return AppendtoL("-",$Seq$)
16) else
17)   return $Seq$
18) end if
```

Even though the browsers nowadays support the parallel content fetching, making request orders unpredictable. We could possibly achieve a relatively aligned sequence for MSA processing, by sorting those flows by its length, see Alg.2 line 2. (Since the length of each flow corresponds to the web content size).

In algorithm 2 line 5-7, we make use of the packet sequence generation algorithm similar to Alg.1. But we only check first M packets in each flow, if the packet number (flow size) in this flow is less than M, we symbolize the entire flow. Otherwise, only first M packets in this flow are symbolized. If the Seq length is less than L, we append "$-$" to Seq to the same length L (Line 14 in Alg.1 and Line 15 in Alg.2).

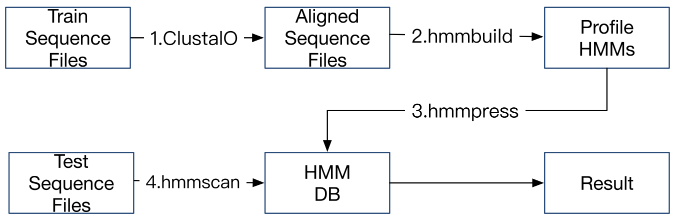


Fig. 10. MSA and Profile HMM building procedures.

B. Symbolization and Profile HMM Building

Symbolization means transforming the packet size and direction information to the alphabet that can be recognized by the PHMM. Since we want to symbolize -1500 bytes to +1500 bytes packets ("+,$-$" shows packet direction), around 3000 unique symbols are needed to represent different packets. But too many symbols render the model complex and not generalized well. On the other hand, if only use very few symbols, like the 20 built-in amino acids symbols in the HMMER [39], the result model could not be accurate. Therefore, we use two alphabets to represent the packet direction size (packet size $\times$ direction). For example, for packet direction size between $-1490 \sim -1500$, all packets in this scope will be represented as AA. Similarly, for packet direction size between $-1480 \sim -1490$, it will be labeled as AB. Therefore, after the symbolization, the output may look like the sequences shown in figure 9. Notice that in a typical web request, we could have a considerable number of packets with size near to Maximum Transmission Unit (MTU). These packets consume a large proportion of sequence length, and worsen the classification result. Also, we find some packets in SSH repeat itself more than other size of packets. As the experiment suggests, it is better to remove or limit those packets in our generated sequences.

In this paper, we use the ClustalO [40] to do the MSA and the HMMER [39] for PHMM building. The ClustalO is faster and the quality of alignments is superior to previous Clustal family versions [40]. The HMMER has a formal implementation of the "Plan 7" profile HMM. It contains a set of tools that create a PHMM of a sequence family which can be later used to query against. When a query comes, the HMMER will apply a series of filters to quickly locate the possible domains. The specific implementation for the MSA and the PHMM building could be seen in figure 10. Both train and test files contain symbolized gene sequences in the Stockholm file format. The HMMER provides several command line programs to build a PHMM database (bmmbuild, hmmpress) and to make queries to profile HMMs (hmmscan). The returned result is a list, with the highest score ranking on the top.

VI. Evaluation

A. Datasets

The proposed approach was evaluated using traces collected in a real network environment. Traces were gathered with

TABLE III
DATASETS USED IN THIS PAPER

	Dataset description	Train Size[1]	Test Size[1]	Purpose
Google	Google search homepage	15×70	15×30	Same domain problem testing
SSH-66	Online open dataset in [3]	66×70	66×30	Webpage fingerprint evaluation and comparison
Alexa-74	Shadowsocks webpage fingerprint	74×17	74×8	Webpage fingerprint evaluation and comparison
Alexa-200	Shadowsocks webpage fingerprint	200×17	200×8	Webpage fingerprint evaluation and comparison
Open world(2000)	Shadowsocks webpage fingerprint	N/A	2000×1	Open world webpage fingerprint evaluation
Alexa-23	Shadowsocks web*site* fingerprint	2400	872	Hyper-link transition without time gap evaluation and comparison
Alexa-30	Shadowsocks web*site* fingerprint	3800	768	Hyper-link transition with time gap evaluation and comparison

[1] The train/test size column are formatted as site number $\times$ instances per site, except Alexa-23 and Alexa-30.

Shadowsocks 2.6.3, and we also employed an online public dataset [3]. The datasets and their corresponding purposes are listed in table III. The automatic site access details are given in Appendix VIII. We also used the default parameter settings from the ClustalO [40] and the HMMER [39] to do the MSA and the PHMM building respectively.

1) Google Dataset: This dataset includes 100 independent accesses to each of 15 Google home pages hosted in different locations around the world.

2) Liberatore's Dataset: It contains a collection of web requests and responses over an encrypted SSH tunnel. The collection spans traces of encrypted connections to 2000 sites. However, the dataset contains many holes, due to various failures in the collection process, thus a considerable amount of pcap files are empty. The average file size per site per instance is about 22.13KB. For consistency reasons, we rebuild a new SSH-66 dataset based on the original dataset (by filtering sites with average instances' size above 100KB). The new dataset does not contain holes and has enough packets for analysis. By saying "enough packets", it means the pcap files for a particular website class should contain the number of filtered packets as closer to the gene length parameter as possible. The total size of this dataset is 1.06GB.

3) Alexa-74 Dataset and Alexa-200 Dataset: We filtered out the same domain in Alexa top 100 sites, such as google.com, google.uk, etc. Thus the Alexa-74 dataset only contains unique domain names. The reason to build a unique domain dataset is discussed in section VI-B1a). For each web URL in Alexa-74, we accessed 25 times, the total size of this dataset is 5.45GB. For our dataset to be statistically significant, we built the Alexa-200 dataset. We randomly selected 200 unique domains from Alexa top 1000. Similarly, we accessed 25 times for each URL in Alexa-200, the total size of this dataset is 15.8GB.

4) Open World 2000 Dataset: To exclude the monitored sites, we randomly selected 2000 different sites from Alexa top 1k to 10K. For each website we only visited once, the total size of this dataset is 5.52GB.

5) Alexa-23 and Alexa-30 Dataset: We randomly selected 30 domains from Alexa top 100. Then we manually removed duplicated websites with the same domain names, and sites with the same web content but in a different domain name such as the blogger.com and the blogspot.com, and websites that cannot be opened or accessed by us (e.g. googleusercontent.com etc.). Then for the remaining 23 domains, we collected their internal and external links. Few websites contain no subsequent visit links seen by our crawler, so we manually visited those websites and collected their URLs by choosing randomly on that page. In total we had 217 unique links, for each link we visited 10 times, yielded 2170 pcap files. Then we added 10 homepage instances for those 23 sites respectively. To simulate a short time hyper-link transit browsing behavior of users, we first loaded the homepage then quickly opened a second random page by following the URLs previous collected for that domain in a 2 seconds delay regardless whether the first page finished loading or not. Finally, we got a 4.39GB training set (including 230 homepage instances) and a 3.08GB testing set.

Based on the Alexa-74 site list, we randomly selected 30 domains from the 74 unique sites to crawl their internal and external links. Considering some sites have more links than others, we set a maximum limit of 10 for internal and 5 for external links. Together, we had 350 unique links. For each link we accessed 10 times, yielded 3500 pcap files. To build a website PHMM, we added 10 homepage instances for those 30 sites respectively. As for testing data, for each site we first loaded its homepage, and then randomly picked a second link in that site (previous collected) to follow within 10 seconds no matter whether the first page loaded complete or not. Because it is reasonable to assume the users usually open a homepage and browse for a while, then click on one of the URLs on that page to visit. In total we had 768 test instances. The total size of the training data is 8.43GB, and 2.79GB for the testing data.

B. Experiment Methodology and Results

We examined real traces of encrypted, proxied HTTP traffic, and our attack's goal is trying to discern the responder to each web request.

1) Test Methods: The training/testing data split is based on time, the first part is used for training, the remaining data are used for testing. We used different gene sequence length L to find out its influence to our method's performance. We also analyzed how the MSA affects our method (aligned or unaligned), as well as the different kinds of sequences (flow seq. or packet seq.) The $M = 10$ in the table means that we inspected first M packets for each flow. The symbols "MTU" and "MTU&Limit"in the table denote that we tried to limit the number of MTU packets, or both the MTU and the most often occurred packet in the sequence respectively. In our experiment, we limited the number of those special packets to 2% length of L. Moreover, according to [5] and [22], we attempted to add 5 another important statistical features in our sequence (denoted as "stat" in the table), they were: the number of incoming packets, the number of outgoing packets, total incoming bytes, total outgoing bytes and total transfered bytes. To integrate these statistical features into the gene sequence, we used the same symbolization method introduced in section V-B. For example, say the total incoming bytes is α. To fit in the range of the symbolization method, we calculated kilobytes by $\alpha/1024$. Then this integer value can be symbolized by using the above method. Note that all the results for the KFP and the KNN were based on 5 times average accuracy value. For the KNN we set the distance learning round to be 5 rounds. Unless otherwise specified, we employed the same training and testing length when comparing our method with existing methods. Also, since both KFP and KNN ignore payload size information (denoted as "no size") because of Tor's packet padding scheme. To make comparison fairer, we added the packet payload size to both algorithms denoted as "with size" in the table.

2) *Test Metrics:* To compare the experimental results, we calculated the overall accuracy defined as follows. The *overall accuracy* determines the general quality of the identification result. It is calculated as the ratio of the number of correctly identified accesses of a web page/site to the total number of the web page/site in the same testing set. For types of errors, we measured the following: True Positive **(TP)** is the number that a monitored page is correctly identified as exactly the same monitored page. False Negative **(FN)** is the number that a monitored page is incorrectly classified as a different monitored page or as a non-monitored page. False positive **(FP)** is defined as the number of a non-monitored page is incorrectly classified as being monitored. True Negative **(TN)** means that a non-monitored page is correctly classified a non-monitored page. True Positive Rate **(TPR)**: The probability that a monitored page is classified as the correct monitored page. False Positive Rate **(FPR)**. The probability that an unmonitored page is incorrectly classified as a monitored page. Bayesian Detection Rate **(BDR)** [2], [23] (shown in equation (3) and (4)) indicates the practical feasibility of the attack or the general probability that the classifier makes a correct prediction.

$$BDR = \frac{TPR \times Pr(M)}{TPR \times Pr(M) + FPR \times Pr(U)} \tag{3}$$

where

$$Pr(M) = \frac{|Monitored|}{|Total Pages|}, \quad Pr(U) = 1 - Pr(M) \tag{4}$$

3) *Experimental Results:*

a) Same domain problem: To test the same domain differentiation problem, we used the Google dataset in our study.

Figure 11 shows the cumulative size visualization technique introduced by Panchenko *et al.* [5], we can clearly see those Google sites are hard to differentiate. We built model for 15 different Google sites using: KNN [6], the state-of-art method K-fingerprinting (KFP) [22], and our PHMM approach. As can be seen in table IV, our PHMM method works slightly better compared to the KFP method. But overall the accuracy is low. To give an accurate assessment of our proposed method, in the following experiments we built several other data sets without duplicate domain names based on the Alexa URLs.

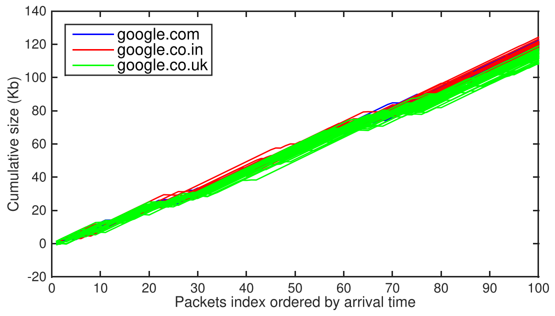


Fig. 11. Google homepage cumulative size graph.

TABLE IV
AVERAGE GOOGLE SITES IDENTIFICATION ACCURACY OVER 10 TRAILS

Method Name	Test Methods	L=50	L=100	L=150
PHMM aligned	pkt seq	58.97	65.13	60.77
Method Name	Test Methods	K=1	K=2	K=3
KFP	pkt seq, with size	**60.84**	60.61	60.22
KNN	pkt seq, with size	54.11	55.64	51.54

TABLE V
WEBPAGE TEST ACCURACY (%) ON DATASET ALEXA-74

Method Name	Test Methods	L=100	L=200	L=300
PHMM aligned	flow seq, $M = 10$	56.12	72.31	80.23
	stat, flow seq, $M = 10$	71.29	74.32	82.25
	ptk seq	90.93	89.42	90.21
	stat, pkt seq	95.95	95.44	95.11
	pkt seq, MTU&Limit	97.46	**97.97**	97.56
PHMM unaligned	pkt seq, MTU&Limit	96.49	97.29	97.31
Method Name	Test Methods	K=1	K=2	K=3
KFP	pkt seq, with size	**97.42**	97.13	96.02
KNN	pkt seq, with size	**96.96**	96.28	95.43

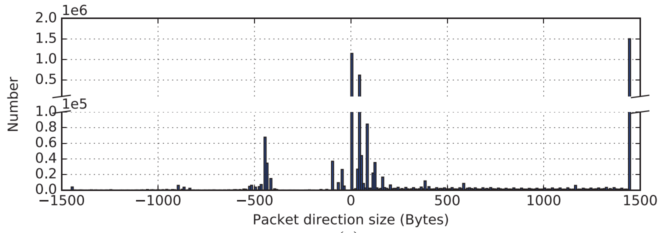
b) Closed world evaluation: The goal to test the webpage fingerprinting attack (under the same assumptions with traditional works) is to show that our web*site* model could be derived to webpage identification and to study how the parameters affect our method. As listed in table V,VI and VII, in some parameter settings our method performs better than existing methods for closed world single page classification. We used the same sequence length for both training and testing in the same column. We noticed a boost after integrating the statistic information into the gene sequence. This is because, those features could help to uniquely represent a single page visit. We can also improve the overall accuracy by limiting the MTU packet number and the often occurred packets (*i.e.* packets with size 600 in case of Shadowsocks, and packets with size 40, 56, 80 bytes in SSH, see figure 12). This is because, those packets constitute a significant amount of the total gene sequence length, thus making each sequence looks similar to each other. From table V, the MSA slightly improves the accuracy, but it is costly to do the MSA when L and the

TABLE VI
WEBPAGE TEST ACCURACY(%) ON DATASET SSH-66

Method Name	Test Methods	L=100	L=200	L=300
PHMM aligned	stat, pkt seq	80.91	84.19	85.96
	stat, pkt seq, MTU&Limit	84.21	**94.14**	92.98
Method Name	Test Methods	K=1	K=2	K=3
KFP	pkt seq, with size	92.12	**92.59**	92.57
KNN	pkt seq, with size	**91.86**	86.82	82.72

TABLE VII
WEBPAGE TEST ACCURACY(%) ON DATASET ALEXA-200

Method Name	Test Methods	L=300	L=400
PHMM aligned	pkt seq, MTU	93.13	94.19
	stat, pkt seq, MTU	96.37	96.49
	pkt seq, MTU&Limit	99.25	**99.31**
	stat, pkt seq, MTU&Limit	99.25	**99.31**
	stat only	57.26	
Method Name	Test Methods	K=1	K=2
KFP	pkt seq, no size	92.93	93.63
	pkt seq, with size	92.98	**93.94**
KNN	pkt seq, no size	92.43	85.81
	pkt seq, with size	**96.06**	92.31


(a)

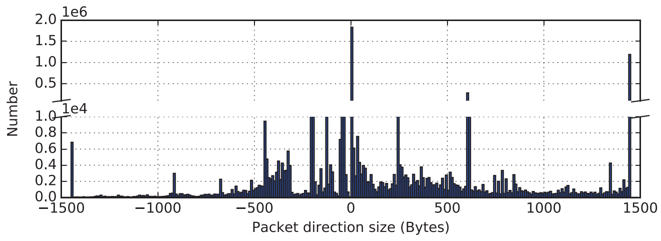

(b)

Fig. 12. SSH and Shadowsocks packet size distribution. (a) Overall SSH packet size distribution. (b) Overall Shadowsocks packet size distribution.

number of training instance are large. Because to find a global optimum for training sequences in the MSA is a NP-complete problem. Also, we can deduce from table V, that the packet sequence (Alg.1) performs better than flow sequence (Alg.2). This is because, we found many flows share a similar pattern: MTU packets are often followed by a small size packet, and they together periodically repeated themselves, thus causing them hard to differentiate with other flows.

c) Open world evaluation: To evaluate how our method performed in an open world scenario, we took 1600 test instances from the Alexa-200 and 2000 instances from the open world dataset. Then we fed them into the PHMM (L=400, pkt seq, MTU). The ideal test results are: (1) our method could discern every closed world (monitored) site from

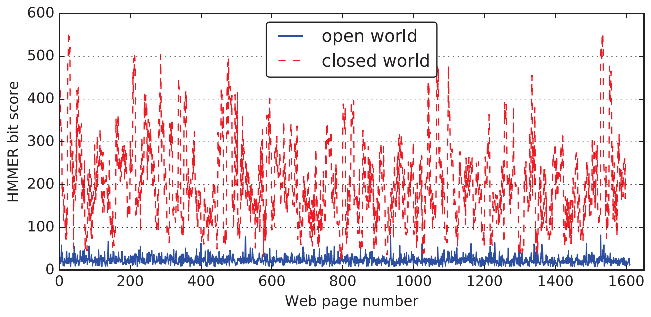


Fig. 13. HMMER bit score for close and open world.

TABLE VIII
OPEN WORLD EVALUATION RESULTS UNDER DIFFERENT THRESHOLDS

	T=0	T=40	T=50	T=60	T=80	T=100
TN/FP	389/1611	1890/110	1968/32	1989/11	1999/1	2000/0
TP	1507	1507	1498	1488	1438	1341
FN-1	93	84	68	50	27	21
FN-2	0	9	34	62	135	238
TPR	94.19%	94.19%	93.63%	93.0%	89.99%	83.81%
FPR	92.55%	5.5%	1.6%	0.55%	0.05%	0%
BDR	44.88%	93.19%	97.91%	99.27%	99.93%	100%

the open world (unmonitored) site. (2) our approach could classify the monitored site to the exactly the same monitored site.

In our implementation, the HMMER [39] uses the bit scores[8] to reflect whether the sequence is a better match to the profile model. As can be seen in figure 13, monitored webpages generally have higher bit scores than unmonitored pages. Notice that in figure 13, we only showed around 1600 unmonitored pages' bit scores, because the remaining 400 unmonitored pages didn't have bit scores since they couldn't match to the profile model.

A natural way to separate the closed world from the open world is to set up a bit score threshold (T). The results for open world evaluation when varying the bit score threshold could be seen in table VIII. We denote "FN-1" as the number that a monitored page is incorrectly classified as a different monitored page and "FN-2" as the number that a monitored page is misclassified as a non-monitored page. FP is high when T=0, this is because most unmonitored sequences could somehow match to a webpage's profile model, however their bit scores are lower than the true webpage's bit score.

d) Webpage transition evaluation: To evaluate the performance differences between our proposed method and traditional method when given a subsequent visit. We took two datasets to evaluate the web*site* fingerprint. The experimental results are listed in table IX and table X. Here, we compared our method with the KFP. As for the KNN, since it requires the number of test instance for each class to be of equal size, then it can use this index to infer the true class label. Therefore, we did not test the KNN in this case, because the number of our test instances for each class is different. Need to pay attention that for both PHMM and KFP method, all the tests

[8]For definition please refer to the HMMER userguide v3.1b2 page 39.

TABLE IX
HYPER-LINK TRANSITION TEST ACCURACY(%) ON DATASET ALEXA-30

Method Name	L=150	L=300	L=400	L=800	L=1200
PHMM aligned, MTU	77.86	74.82	74.93	**79.56**	60.54
PHMM unaligned, MTU	56.78	61.06	67.85	71.75	59.24
KFP K=1	46.35	44.14	49.60	51.95	34.89
KFP K=2	46.48	44.41	49.21	52.86	35.81
KFP K=3	47.26	43.75	49.47	**52.92**	37.89

TABLE X
HYPER-LINK TRANSITION TEST ACCURACY(%) ON DATASET ALEXA-23

Method Name	L=150	L=300	L=400	L=800	L=1200
PHMM aligned, MTU	80.16	80.62	**81.31**	80.29	77.33
PHMM unaligned, MTU	77.38	77.27	76.07	75.06	74.45
KFP K=1	71.48	66.39	70.27	56.19	43.80
KFP K=2	71.78	67.34	70.75	56.65	43.02
KFP K=3	**71.89**	67.20	70.64	54.35	42.56

were carried out in the same parameters setting, except the sequence length L. For PHMM, we set the parameters as using packet sequence, MTU filtered. We did not use statistics in this evaluation, because the statistics can only be used in webpage fingerprinting. If we added the statistics to gene sequence, every test instance will have a different statistics, thus the statistics could not uniquely represent this website. To compare fairly, we added the packet size information as we did above and we set the KFP using the same sequence length L for each test. All KFP results are based on five times average value.

As can be seen in table IX and table X, for the KFP method, as the sequence length becomes longer the accuracy would drop, because the feature vector becomes more unpredictable when a second page shows up. The best accuracy of KFP for using the whole sequence length is only 20.96% in the Alexa-30 and 25.45% in the Alexa-23. For our proposed method, however, when feeding the full packet sequence for testing, the accuracy for Alexa-30 using model L=800 is 88.67%. For the Alexa-23, the accuracy value is 85.21% using model L=400 (and applying MTU&Limit it slightly grows to 85.55%). This is because our test datasets contain two sequentially accessed webpages. And our web*site* model considered the subsequent visits. But existing approaches rely on extracting the features only from the first page, in our case, however, the final feature vector could be affected by the second page, thus deteriorating the identification result.

To demonstrate our PHMM based web*site* model could actually work when a subsequent page shows up, we additionally built two models with the same sequence length L=400 using Alexa-23. The first model trained with the homepage and its subsequent links, we referred to it as the "Website model". The second model only trained with the homepage of each website without the corresponding internal and external links, we called it "Homepage only model". Figure 14 illustrates our web*site* model has the capability to identify the subsequent visit when L grows. Notice that when L is small, the "Homepage only model" performs better. This is because the short testing sequence (query) could match with

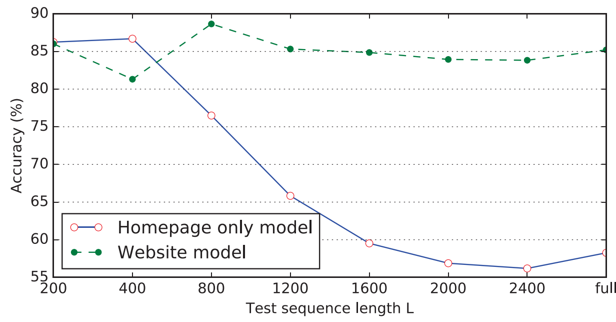


Fig. 14. Comparison of single homepage only model and our model which accounts for subsequent visit on dataset Alexa-23.

other links' sequences in the "Website model". As expected, when the second page starts to load, the "Homepage only model" deteriorates fast. However, our method's accuracy is relatively stable, because compared with the "Homepage only model", our model could match the second webpage's sequence.

Due to the size of each class being different, we need to clarify the true size of each class. Since a false classification for many small classes and a correct classification of a single major class could still yield a high accuracy value. To demonstrate this, figure 15 shows the confusion matrices for two of our cases using heat maps. Note that 6 test instances do not appear in the heat map in Alexa-23, for they fail to match to any website. As we can see from figure 15, our method correctly differentiated most of the sites.

VII. COUNTERMEASURES AND DISCUSSION

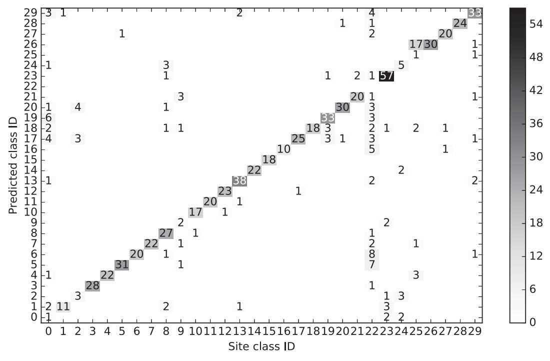
A. Countermeasures

Since our method relies on the packet size information, packet padding seems to be the most direct technique to confuse the classifier. Previous section has demonstrated the MTU packets could worsen the classification result. We can consider the most extreme case: padding all the packets to MTU size, then every sequence will look similar, therefore, making our method hard to identify the target. Due to possible high bandwidth overhead, packet padding technique is not widely deployed in practice, here we show two possible countermeasures to stop our attack.

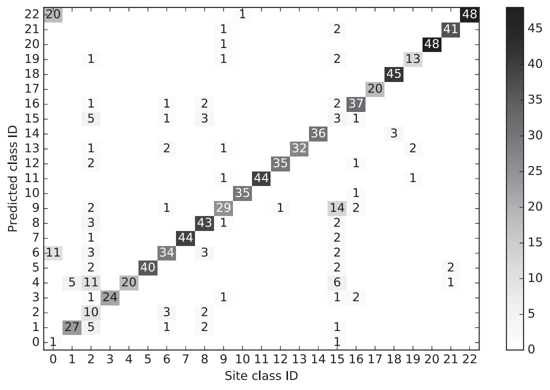
Probabilistic MTU padding: Padding all packets to MTU size is very inefficient, therefore we selectively pad packets to MTU with probability (P). Each packet now has an equal probability (P) to choose whether to pad itself to MTU sized packet. **Probabilistic Dummy packet:** We inserted dummy packets of random direction and size into the testing packet sequence. Each packet has a probability (P) to decide whether to inject a dummy packet in the sequence ahead of it. Figure 16 depicts how performance would degrade in case that the packets would be padded or be injected with dummy packets.

B. Discussion

In our study, we concentrated on the WFP attack on Shadowsocks and SSH. And we addressed web*site* fingerprinting and the hyper-links transition situations based on



(a)



(b)

Fig. 15. PHMM aligned confusion matrices for Alexa-30 and Alexa-23. (a) Confusion matrix for Alexa-30, L=800. (b) Confusion matrix for Alexa-23, L=400.

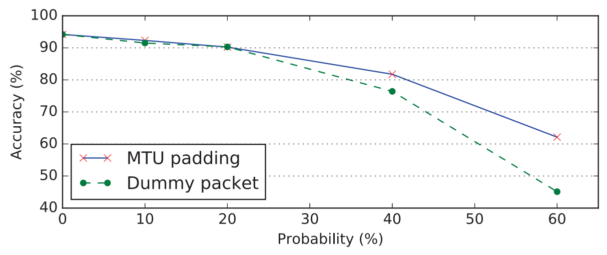


Fig. 16. Accuracy change while varying the probability (P), test using Alexa-200 dataset, L=300 pkt seq and MTU.

those anonymous network traffic. We found that the key steps of our approach are: (1) how to represent the sequence, and (2) how to translate packet size and direction to the symbols that PHMM recognizes. Note that, we only showed two possible ways to generate sequence and one kind of symbolization method. These methods were designed for Shadowsocks and SSH model building. Also, in our paper we used the HMMER [39] to implement our idea, and the alphabet only contained limited 20 amino acids. However, we believe that one can use a special designed alphabet and an advanced symbolization method to overcame these limitations and get our method to work for other protocols, for instance, we could use the CUMUL [5] sequence to represent Tor's traffic. Or one can use the Fourier transform to generate the sequence for further analyses. As for the MSA method used in the paper, it is better to consider the trade-off before aligning a considerable amount of long sequences. Finally, we need to mention that all the results presented in the tables and graphs above are the top ranked results.

Procedure 1 Data Collection Using iMacros

```
1)  for 0 < i < 74 do
2)     for 0 < round < 25 do
3)        Mark start time
4)        Set timeout 60 seconds
5)        Load site i
6)        Mark the end time
7)        Close the tab
8)        Wait for 45 seconds
9)     end for
10) end for
```

To ensure reproducibility and scientific correctness, we published our code and datasets on https://github.com/jason-zhuo/PHMM.

VIII. Conclusions

In this paper, we proposed a web*site* fingerprinting attack based on the PHMM that would deanonymize the receiver. We implemented the proposed method on two anonymous networks: Shadowsocks and SSH. We were the first to apply the sequence analysis techniques from bioinformatics to model the webpage and web*site* fingerprints. We realized two possible forms of sequence representation methods (flow and packet sequence) as well as one symbolization scheme. In total 7 datasets were used to testify our method. Experimental results suggest that our approach has a higher accuracy than existing methods in web*site* classification and outperforms traditional individual webpage identification. Also, we addressed two types of hyper-link transition situations in our experiment. The results demonstrate that our approach could work for web*site* fingerprint attack in a more practical environment, whereas existing methods only concentrate on the single individual webpage fingerprinting and neglect the potential subsequent visit. Finally, two possible countermeasures were evaluated to defend our proposed method.

Appendix
Environment Setting and Dataset Crawling

We collected the dataset in three clean and controlled machines located in the same LAN of our lab. To avoid significant changes or updates of the websites, both crawling programs started roughly in the same day (two for training and one for testing). We used a customized Firefox 50.0, for instance, we set browser.cache.check_doc_frequency=1 so that the browser will ask for a new version of the content every time a page is loaded. The browser neither worked with any working logins, nor did it save login status such as cookies. We also disabled the browser's cache function to prevent browser from loading the cache. And by default the HTTP pipeline in Firefox is disabled (network.http.pipelining=false). We wrote a simple C program with Firefox and tcpdump

Procedure 2 Hyper-Link Transition Testing Dataset Collection

```
1) for $0 < i < 30$ do
2)   for $0 < round < \beta_i$ do
3)     Start tcpdump with options
4)     Load site $i$
5)     Sleep $s$ seconds then load a random link from $i$
6)     Wait for 15 seconds
7)     Kill tcpdump
8)     Kill firefox
9)     Wait for 45 seconds
10)  end for
11) end for
```

command to gather the data. Both programs followed a similar logic, as can be seen in procedure 1.

The steps for collecting the test instances in the hyper-link transition scenario is shown in procedure 2. Where β_i is the total link number for site i. Note that a random link from i doesn't include i itself. Also, in procedure 2 line 4 and 5, the load command is non-blocking, which means as soon as it starts to load site i no matter it finishes or not, it will return immediately. Notice that in procedure 1 line 5, the load command will not return until it finishes loading the page or reaches the timeout. The sleep time s for Alexa-30 is 10 seconds, and 2 seconds for Alexa-23. The tcpdump options were filter rules such as the protocol and the port number etc.

Acknowledgment

The authors would like to thank Professor Zhi-li Zhang for inspiring discussions and the anonymous reviewers for helpful suggestions.

References

[1] A. Pfitzmann and M. Waidner, "Networks without user observability—Design options," in *Workshop on the Theory and Application of Cryptographic Techniques*. Berlin, Germany: Springer, 1985, pp. 245–253. [Online]. Available: https://link.springer.com/chapter/10.1007/3-540-39805-8_29#citeas

[2] J. Hayes and G. Danezis, "k-fingerprinting: A robust scalable website fingerprinting technique," in *Proc. 25th USENIX Secur. Symp. (USENIX)*, Austin, TX, USA, Aug. 2016, pp. 1187–1203. [Online]. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/hayes

[3] M. Liberatore and B. N. Levine, "Inferring the source of encrypted http connections," in *Proc. 13th ACM Conf. Comput. Commun. Secur.*, 2006, pp. 255–263

[4] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website fingerprinting in onion routing based anonymization networks," in *Proc. 10th Annu. ACM Workshop Privacy Electron. Soc.*, 2011, pp. 103–114.

[5] A. Panchenko *et al.*, "Website fingerprinting at Internet scale," in *Proc. 23rd Internet Soc. (ISOC) Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2016.

[6] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting," in *Proc. 23rd USENIX Secur. Symp. (USENIX)*, 2014, pp. 143–157.

[7] D. Herrmann, R. Wendolsky, and H. Federrath, "Website fingerprinting: Attacking popular privacy enhancing technologies with the multinomial Naïve-Bayes classifier," in *Proc. ACM Workshop Cloud Comput. Secur.*, 2009, pp. 31–42.

[8] H. Jahani and S. Jalili, "A novel passive website fingerprinting attack on tor using fast Fourier transform," *Comput. Commun.*, vol. 96, pp. 43–51, Dec. 2016.

[9] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching from a distance: Website fingerprinting attacks and defenses," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2012, pp. 605–616.

[10] A. Kwon, M. AlSabah, D. Lazar, M. Dacier, and S. Devadas, "Circuit fingerprinting attacks: Passive deanonymization of tor hidden services," in *Proc. 24th USENIX Secur. Symp. (USENIX)*, 2015, pp. 287–302.

[11] R. Dingledine, N. Mathewson, P. F. Syverson, "Tor: The second-generation onion router," in *Proc. 13th USENIX Secur. Symp.*, San Diego, CA, USA, Aug. 2004, pp. 303–320. [Online]. Available: http://www.usenix.org/publications/library/proceedings/sec04/tech/dingledine.html

[12] M. Clowwindy and L. Max. (Nov. 2016). *Shadowsocks*. [Online]. Available: http://www.shadowsocks.org/

[13] J. Boyan, "The anonymizer: Protecting user privacy on the Web," *Comput.-Mediated Commun. Mag.*, vol. 4, no. 9, pp. 1–6, Sep.1997.

[14] B. Zantout and R. Haraty, "I2P data communication system," in *Proc. ICN*, Jan. 2011, pp. 401–409.

[15] T. Wang and I. Goldberg, "Comparing website fingerprinting attacks and defenses," Cheriton School Comput. Sci. Univ. Waterloo, Waterloo, ON, Canada, Tech. Rep. 2013-30, 2013. [Online]. Available: http://cacr.uwaterloo.ca/techreports/2013/cacr2013-30.pdf

[16] Y. Shi and K. Matsuura, "Fingerprinting attack on the tor anonymity system," in *Proc. Int. Conf. Inf. Commun. Secur.*, 2009, pp. 425–438.

[17] 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 *Proc. NDSS*, Feb. 2011, pp. 1–20.

[18] M. Perry. (2011). *Experimental Defense for Website Traffic Fingerprinting Tor Project Blog*. [Online]. Available: https://blog.torproject.org/blog/experimental-defense-website-traffic-fingerprinting

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

[20] 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 *Proc. IEEE Symp. Secur. Privacy*, May 2012, pp. 332–346.

[21] X. Gu, M. Yang, and J. Luo, "A novel website fingerprinting attack against multi-tab browsing behavior," in *Proc. IEEE 19th Int. Conf. Comput. Supported Cooperat. Work Design (CSCWD)*, May 2015, pp. 234–239.

[22] J. Hayes and G. Danezis, "k-fingerprinting: A robust scalable website fingerprinting technique," in *Proc. 25th USENIX Secur. Symp., USENIX Secur.*, Austin, TX, USA, Aug. 2016, pp. 1187–1203. [Online]. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/hayes

[23] M. Juarez, S. Afroz, G. Acar, C. Diaz, and R. Greenstadt, "A critical evaluation of website fingerprinting attacks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2014, pp. 263–274.

[24] S. R. Eddy, "Profile hidden Markov models," *Bioinformatics*, vol. 14, no. 9, pp. 755–763, 1998.

[25] T. Wang and I. Goldberg, "Improved website fingerprinting on tor," in *Proc. 12th ACM Workshop Privacy Electron. Soc.*, 2013, pp. 201–212.

[26] H. Cheng and R. Avnur. (1998). *Traffic Analysis of SSL Encrypted Web Browsing*. [Online]. Available: https://citeseer.ist.psu.edu/myciteseer

[27] A. Hintz, "Fingerprinting websites using traffic analysis," in *Proc. Int. Workshop Privacy Enhancing Technol.*, 2002, pp. 171–178.

[28] Q. Sun, D. R. Simon, Y.-M. Wang, W. Russell, V. N. Padmanabhan, and L. Qiu, "Statistical identification of encrypted Web browsing traffic," in *Proc. IEEE Symp. Secur. Privacy*, May 2002, pp. 19–30.

[29] G. D. Bissias, M. Liberatore, D. Jensen, and B. N. Levine, "Privacy vulnerabilities in encrypted http streams," in *Proc. Int. Workshop Privacy Enhancing Technol.*, 2005, pp. 1–11

[30] O. Berthold, H. Federrath, and S. Köpsell, "Web mixes: A system for anonymous and unobservable Internet access," in *Designing Privacy Enhancing Technologies*. Berlin, Germany: Springer, 2001, pp. 115–129. [Online]. Available: https://link.springer.com/chapter/10.1007/3-540-44702-4_7

[31] G. He, M. Yang, J. Luo, and X. Gu, "A novel application classification attack against tor," *Concurrency Comput., Pract. Experim.*, vol. 27, no. 18, pp. 5640–5661, 2015.

[32] S. E. Coull *et al.*, "On Web browsing privacy in anonymized NetFlows," in *Proc. USENIX Secur.*, 2007, pp. 339–352.

[33] A. Krogh, M. Brown, I. S. Mian, K. Sjölander, and D. Haussler, "Hidden Markov models in computational biology: Applications to protein modeling," *J. Molecular Biol.*, vol. 235, no. 5, pp. 1501–1531, 1994.

[34] S. R. Eddy, "A probabilistic model of local sequence alignment that simplifies statistical significance estimation," *PLoS Comput. Biol.*, vol. 4, no. 5, p. e1000069, 2008.

[35] S. Eddy, "HMMER user's guide," Dept. Genet., Washington Univ. School Med., St. Louis, MO, USA, Tech. Rep., vol. 2, no. 1, pp. 1–98, 1992.

[36] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257–286, Feb. 1989.

[37] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison, *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge, U.K.: Cambridge University Press, 1998.

[38] A. Bhargava and G. Kondrak, "Multiple word alignment with profile hidden Markov models," in *Proc. Human Lang. Technol., Annu. Conf. North Amer. Chapter Assoc. Comput. Res. Workshop Doctoral Consortium*, 2009, pp. 43–48.

[39] S. R. Eddy. *HMMER: Biosequence Analysis Using Profile Hidden Markov Models*. Accessed: Nov. 17, 2016. [Online]. Available: http://hmmer.org/

[40] F. Sievers *et al.*, "Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega," *Mol. Syst. Biol.*, vol. 7, no. 1, p. 539, 2011.

Zhongliu Zhuo received the B.S. degree in computer science in 2013. He is currently pursuing the Ph.D. degree with the University of Electronic Science and Technology of China. His research focuses on the anonymous network, machine learning, and network security. He was sponsored by the China Scholarship Council for studying at the University of Minnesota–Twin Cities.

Yang Zhang received the B.S. and M.S. degrees in software engineering and computer science from Southeast University, in 2011 and 2013, respectively. He is currently pursuing the Ph.D. degree with the University of Minnesota–Twin Cities. His research pursuit is to build efficient and secure network systems, and his research interests are network function virtualization and network security.

Zhi-li Zhang (F'12) received the B.S. degree in computer science from Nanjing University, China, in 1986, and the M.S. and Ph.D. degrees in computer science from the University of Massachusetts Amherst in 1992 and 1997, respectively. In 1997, he joined the Computer Science and Engineering Faculty, University of Minnesota, where he is currently a Qwest Chair Professor and a Distinguished McKnight University Professor. His research interests lie broadly in computer communication and networks and Internet technology. He is a member of ACM. He was the co-recipient of four Best Paper Awards and has received a number of other awards. He has Co-Chaired several conferences/workshops including the IEEE INFOCOM06 and served on the TPC for numerous conferences/workshops.

Xiaosong Zhang received the B.S. degree in dynamics engineering from Shanghai Jiaotong University, Shanghai, in 1990, and the M.S. and Ph.D. degrees in computer science from the University of Electronic Science and Technology of China (UESTC) in 2011. He is currently a Professor in computer science with UESTC. His current research involves software reliability, software vulnerability discovering, and network security. He was a recipient of the second award for the National Scientific and Technological Progress. He and his group were committed to research and development of the cutting-knowledge and core technology on cyber-space security.

Jingzhong Zhang was born in 1936. He is currently an Academician of the Chinese Academy of Sciences and a Professor with the University of Electronic Science and Technology of China. His research interests include discrete dynamic systems, mechanized theorem proving, and theory and methods of automated reasoning and their applications to intelligent system. He and his group received the top academic rewards from the Chinese Academy of Sciences in 1995 and the National Association of Science and Technology, China, in 1997, for their contributions to automated geometry theorem proving.