Fingerprinting a flow of messages to an anonymous server

Juan A. Elices #1, Fernando Pérez-González *,#2

# *Electrical and Computer Engineering Department, University of New Mexico*
* *Signal Theory and Communications Department, University of Vigo*
1 *jelices@ece.unm.edu,* 2 *fperez@gts.uvigo.es*

***Abstract*—We present an attack to locate hidden servers in anonymous common networks. The attack is based on correlating the flow of messages that arrives to a certain server with the flow that is created by the attacker client. The fingerprint is constructed by sending requests, each request determines one interval. To improve the performance a prediction of the time of arrival is done for each request. We propose an optimal detector to decide whether the flow is fingerprinted, based on the Neyman-Pearson lemma.**

The usefulness of our algorithm is shown for the case of locating a Tor Hidden Service (HS), where we analytically determine the parameters that yield a fixed false positive probability and compute the corresponding detection probability. Finally, we empirically validate our results with a simulator and with a real implementation on the live Tor network. Results show that our algorithm outperforms any other flow watermarking scheme. Our design also yields a small detectability.

I. Introduction

Internet anonymous access has become a necessity for a wide range of users: human rights activists, journalists, military, dissidents, bloggers, citizens in censored countries, etc. This need fostered the development of anonymous communication systems, such as Freenet [1], Tor [2], or I2P [3]. Anonymity is not only an issue for users, but it is also important for servers. The Electronic Frontier Foundation and Reporters Without Borders advise the use of anonymous servers to protect the safety of dissidents who have to overcome censorship.

Unfortunately, anonymous servers are also used for illegal purposes such as distributing child pornography, drug traffic and supporting terrorism. Nowadays tracking the location of theses sites has become a serious concern for law enforcement agencies, as their IPs are not public. For example, the FBI has halted several investigations of sites that traffic in illegal drugs or share child pornography because there is currently no way to trace the origin of their websites [4].

In this paper we propose a flow fingerprint that can be used to identify anonymous servers. For this purpose the attacker client fingerprints the flow of messages and eavesdrops the communication channel at the anonymous server side trying to detect the inserted fingerprint. This adversary model has been shown to be realistic, as some organizations or governments have access to Internet service provider (ISP) data [5]. For that matter, the ISP itself or an autonomous system (AS) can become the adversary.

Fingerprinting is done in the following way: a client sends application requests to the server, disguised as a common user's pattern. The times the request is sent and the response is received are saved. The detector predicts the value of the arrival time of the request to improve the detection performance and then applies Neyman-Pearson lemma. This yields an excellent performance, for example we could identify a HS using the real traffic of Signal Processing Group of the University of Vigo web server (UVigo) with just 8 requests with a detection probability 0.9 given a probability of false positive of 10^{-6}. Our fingerprint does not modify the TCP's intrinsic features, i.e. we do not add any delay to any packet, thus making it impossible to detect using these characteristics as is done by Backlit [6] or MFA [7].

The rest of this paper is organized as follows: Section II reviews previous approaches. In Section III we formally describe our problem and the proposed solution. In Section IV we propose a real application to validate our fingerprint and show how we do the prediction. In Section V we analytically derive the probability of detection and the false positive rate. In Section VI we empirically validate our scheme in a simulator and in a real implementation. We also study the detectability of our algorithm. Section VII summarizes our contribution.

II. Background

A. Attacks against anonymous servers

Øverlier and Syverson [8] proposed an attack to identify Tor Hidden Services. The goal of the attack is that the circuit from the Hidden Service (HS) to the rendezvous point has a compromised node as an entry point. They detect when they have succeeded by a cell counting algorithm.

Elices et al. [9] watermark a flow of requests to leave a fingerprint in an anonymous server log. This attack does not really locate the anonymous server but serves to prove later on that the confiscated machine was the anonymous server. It is easily bypassed if this machine does not keep access logs or stores them with a very coarse time granularity.

B. Watermarking/Fingerprinting algorithms

Houmansadr et al. [10] proposed RAINBOW, a non-blind watermark robust to packet drops and repacketization. They

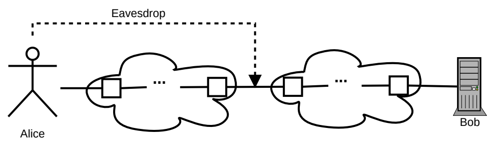


Fig. 1. System Model

record the inter-packet delay previous to embedding the watermark and add a different quantity for embedding a 0 or a 1.

Houmansadr et al. [11] proposed SWIRL, a scalable watermark that is invisible and resilient to packet losses. The watermark is embedded by delaying packets so that they fall into some selected slots.

Pyun et al. [12] proposed a watermark that works better in the cases of flow splitting, chaff packets and repacketization, but with the drawback of being more detectable. The information is embedded in the difference between the number of packets in two contiguous intervals.

All these watermarks can be detected analyzing the intrinsic timing features of TCP flows [6].

III. Description of the Problem

In this section we formally describe the problem and introduce the notation we use in the rest of the paper.

A. Problem

Figure 1 shows the system model. We have a client-server architecture, we call the client Alice and the server Bob. Alice wants to fingerprint her traffic, so she can decide whether the traffic she eavesdrops contains her flow or it does not. We also assume that the network can encrypt the traffic and repacketize it. This means that Alice cannot identify her packets by their size or contents.

Alice sends a fixed number of application messages following a pattern that could belong to a normal client, saving the time when she sends the request and the instant she receives the response. This is the interval considered for each request. We define the ith interval as the period of time $I_i = (o_i, e_i) = (o_i, o_i + T_i), \quad i = 0, \ldots, L-1$, where L is the number of requests that conform our fingerprint, o_i is the instant that the ith request is sent, e_i is the instant that the response is received, and T_i is the interval length. To make intervals independent, Alice only sends a request when there are no other pending requests, i.e. $I_i \cap I_j = \emptyset, \forall i \neq j$.

Alice predicts the value when she expects to see the message that carries her request. We call r_i to the real message time and $\hat{r}_i$ to the prediction. The goal of Alice is to be able to correctly decide if the eavesdropped link contains the fingerprinted flow based on the messages seen on this link.

B. Performance Metrics

To measure the performance of our attack, we define two metrics: the probability of detection (P_D), which represents the probability of deciding that the flow through the eavesdropped link contains the fingerprinted flow when it is actually true; and the probability of false positive (P_F), which represents the probability of incorrectly deciding that the eavesdropped link contains the fingerprinted flow. Formally, we can express this problem via classical hypothesis testing with the following hypotheses:

H_0: The eavesdropped link does not carry the fingerprinted flow.

H_1: The eavesdropped link carries the fingerprinted flow.

Then P_D is the probability of deciding H_1 when H_1 holds, whereas P_F is the probability of deciding H_1 when H_0 holds.

In a practical setting, one fixes a certain value of P_F (that has to be very small if we want to achieve a high reliability) and then measures P_D (which we would like to be as large as possible).

In addition, we want to avoid that Bob or any other party different than Alice notices that the incoming traffic is fingerprinted, i.e., the sequence should have low detectability. As we have mentioned, our algorithm does not delay any packet. Hence the fingerprint can only be detected due to the increment of traffic or if the sequence of requests does not follow a typical pattern. We will measure the detectability with the Kullback-Leibler divergence (KLD) between the requests that the anonymous server receives with and without the fingerprinted flow. We do not consider the second way of detecting the fingerprint, as we have assumed that Alice sends her requests in a normal user pattern.

C. Proposed detector

The proposed detector is based on Neyman-Pearson lemma, as it is the uniformly most powerful test [13]. This means that it has the maximum probability of detection among all possible tests that yield a given P_F. Therefore, our detector is optimal for the proposed model.

This decoder constructs the ratio $\Lambda(\vec{x})$

$$\Lambda(\vec{x}, \vec{P}) = \frac{Pr(\vec{X} = \vec{x} | H_1, \vec{P})}{Pr(\vec{X} = \vec{x} | H_0)} \tag{1}$$

where $\vec{x}$ is the vector of all eavesdropped message times, i.e. $\vec{x}$ includes the time of our message, r_i, and other users' messages. $\vec{P}$ is the vector of predicted message times, i.e. $\vec{P} = \{r_1, \ldots, r_L\}$, and Alice decides the eavesdropped link contains the fingerprint if this ratio is larger than η, a threshold that we fix to achieve a certain probability of false positive.

We assume that messages coming from other users are uniformly distributed inside the interval (see [14]) and independent from the fingerprint message. In this case, the ratio can be simplified to

$$\Lambda(\vec{x}, \vec{P}) = \prod_{i=1}^{L} \frac{f_{error}(r_i - \hat{r}_i)}{\frac{1}{T_i}} \tag{2}$$

where f_{error} is the probability density function (pdf) of the prediction error. However, we find the problem that if Bob receives multiple messages in the interval we cannot identify

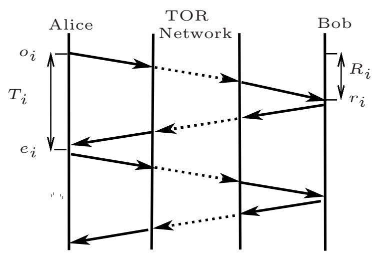


Fig. 2. Cells in a Tor Request

a unique candidate to be the target message, that is, correctly measure the value of r_i.

We consider that the prior probability that any given message is the desired one to be equal for all messages. Hence Alice decides a fingeprint is present if

$$\Lambda(\vec{x}, \vec{P}) = \prod_{i=1}^{L} \sum_{x_j \in I_i} \frac{1}{n_i} f_{error}(x_j - \hat{r}_i) \cdot T_i > \eta. \quad (3)$$

where n_i is the number of messages that fall inside I_i. Note that since $\frac{n_i}{T_i}$ is the rate of requests, the larger this is, the less weight it is given in the decoding.

IV. Application: Detecting a Tor Hidden Service (HS)

In order to show the feasibility and usefulness of our algorithm we apply it to locate a HS. We assume a client, Alice, who fingerprints the flow of messages to a HS, Bob, and eavesdrops the communication channel at Bob's side trying to detect the inserted fingerprint. Figure 2 shows the packet exchange for just one request and the considered interval.

A. Predictor

Our algorithm benefits from a prediction of the time when a cell reaches the HS. The estimation is done based on the information available to Alice, that is, the round-trip delay. Therefore, $\hat{r}_i = f(T_i)$ where f is the prediction function.

1) Measured delays: In order to measure the delays, we create a toy client-server application to measure packet delays in Tor. Traffic is captured and matched to the packets shown in Figure 2. We used 50000 samples that were taken every 10 seconds. As it is customary, we separate these data into two subsets: a training set that includes 80% of the samples, and the remaining 20% as test set.

2) Predictors and performance: We propose two different predictors. The first one is based on polynomial regression and the second one a multilayer perceptron (MLP) [15]. The first model is simpler, but limits the range of available functions to predict, compared to the second one that can approximate any function [15].

We compare in Figure 3 the root-mean-square error (RMSE), as it is a measure of the accuracy of a predictor, for both cases. Results show that a MLP does not give any advantage and increasing the order of the polynomial produces negligible improvements. Hence our predictor is an affine function of T_i, that is, $\hat{R}_i = 0.46 \cdot T_i + 0.02$ with T_i measured in seconds. That is close enough to the expected $\hat{R}_i = 0.5 \cdot T_i$, as the paths of the request and response are essentially identical.

3) Characterizing the prediction error: We saw previously that the decoder needs to model the error function. To this end, we use the errors produced by the affine predictor given above. Errors are normalized dividing by their corresponding T_i.

We select some candidate distributions and estimate their parameters using maximum likelihood estimators (MLE), as this method chooses the value of the parameters that produce a distribution that gives the observed data the greatest probability. Afterwards, we evaluate the goodness-of-fit using different metrics: the KLD, the χ^2 , the Kolmogorov-Smirnov (KS) and the Anderson-Darling (AD). The smaller the value the better is the fit in all of them.

The candidate distributions were selected among the most common continuous distributions that have support at least on the range of our data, i.e $(-1, 1)$. The chosen distributions are Normal, Cauchy, Laplace, Logistic and Gumbel. Their pdfs can be seen in [16].

Table I shows the parameters obtained by MLE. Table II shows the goodness-of-fit test results, where it is shown that Laplace distribution best models our data.

Also we denormalize the intervals, hence we get the distri-

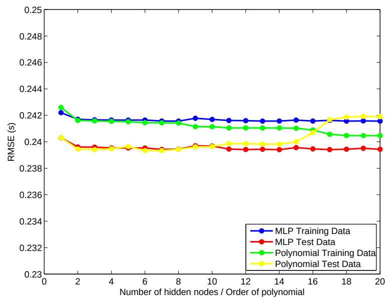


Fig. 3. Performance of the MLP predictor and polynomial predictor

TABLE I
MLE Parameters

Distribution	Parameters
Normal	$\mu = 0.0023$, $\sigma = 0.1474$
Cauchy	$\mu = 0.00048$, $\sigma = 0.0809$
Laplace	$\mu = 0.00014$, $\sigma = 0.1107$
Logistic	$\mu = 0.00013$, $\sigma = 0.0797$
Gumbel	$\mu = -0.0791$, $\sigma = 0.1983$

TABLE II
Goodness of Fit

	Test Filtered Data			
Distribution	**KLD**	χ^2	**KS**	**AD**
Normal	0.1608	801057	0.1005	1090
Cauchy	0.0846	586019	0.0763	197
Laplace	0.0315	592904	0.0310	176
Logistic	0.0888	600276	0.06545	514
Gumbel	0.3043	1540593	0.1561	17761

bution of the error given the measured round-trip, T_i, and we make the simplification $\mu \approx 0$ (cf. Table I). Therefore, we can write $f_{error_i}(\varepsilon_i|T_i) = \frac{1}{2\sigma T_i}\exp\left(-\frac{|\varepsilon_i|}{\sigma T_i}\right)$, where the value of σ is shown in Table I.

V. Analysis and Results

A. Mathematical analysis

In this section we want to calculate the threshold η for a given probability of false positive, and the probability of detection that is achieved with such threshold. First, we statistically model the number of cells per unit of time n_i/T_i and the round-trip time for a cell, T_i. Afterwards, we derive the expressions for η and P_D.

1) Modelling the number of cells per unit of time: We consider four possible models: the usual ones [9], Poisson and Negative Binomial, and their generalizations: Generalized Poisson and Generalized Negative Binomial. Refer to [17] for the formal pdfs.

To validate each model we use seven different World Wide Web server logs from the Internet Traffic Archive and UVigo. We estimate the parameters by MLE [17]. Results (cf. Table III) show that both the Generalized Poisson and the Generalized Negative Binomial can model the number of requests. We choose to use the General Poisson Approximation as it has one degree less of freedom than the General Negative Binomial.

TABLE III
Goodness of Fit for the number of requests

Log	Poisson K-L Div.	NB K-L Div.	Gen. Poisson K-L Div.	Gen. NB K-L Div.
Calgary	0.0121	0.0076	0.0001	0.0001
UVigo	0.0845	0.0504	0.0012	0.0012
Saskatchewan	0.0246	0.0066	0.0014	0.0017
EPA	0.0979	0.0025	0.0003	0.0001
Nasa	0.0882	0.0002	0.0001	0.00005
Clarknet	0.2534	0.0028	0.0009	0.0009
World Cup	1.9701	0.0074	0.0059	0.0056

2) Modelling a cell round trip delay, T_i: A characterization of the pdf of the round-trip time is needed. Our measurements confirm the result by Loesing et al. that the delays can be modeled as a Fréchet distribution [18].

3) Theoretical probablities: Recalling from previous sections, we decide there is a fingerprint if the random variable W defined as

$$W = \prod_{i=1}^{L} \underbrace{\frac{1}{n_i}\sum_{j=1}^{n_i} \underbrace{\exp\left(-\frac{|X_j - \hat{r}_i|}{\sigma T_i}\right)}_{Y_j}}_{Z_i} \qquad (4)$$

is larger than η. Also notice that in (4) we have defined the auxiliary random variables Y_j and Z_i.

W.l.o.g., we consider that X_1 represents the message that can come from our fingerprint or not and X_j, $j = 2, \ldots, n_i$ are messages from any other source, then, for this last kind of message:

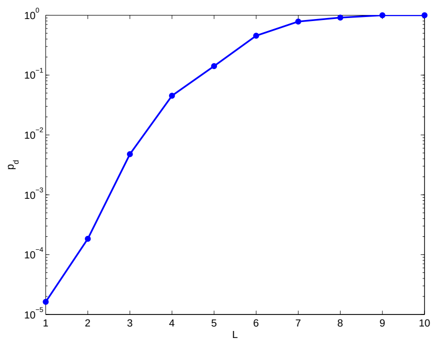


Fig. 4. Probability of watermarking detection for UVigo and a fixed $P_F = 10^{-6}$

$$f_{Y_j|T_i}(y_j|T_i) = \begin{cases} \frac{\sigma}{y_j} & y_j \in (a,b) \\ \frac{2\sigma}{y_j} & y_j \in (b,1) \end{cases}, \; j = 2, \ldots, n_i \qquad (5)$$

where

$$a = \min\left(\exp\left(-\frac{0.46T_i + 0.02}{\sigma T_i}\right), \exp\left(-\frac{0.54T_i - 0.02}{\sigma T_i}\right)\right)$$

$$b = \max\left(\exp\left(-\frac{0.46T_i + 0.02}{\sigma T_i}\right), \exp\left(-\frac{0.54T_i - 0.02}{\sigma T_i}\right)\right).$$

For the case $j = 1$, we have $f_{Y_1|T_i}(y_1|T_i, H_1) \sim \text{Uniform}(0,1)$, and $f_{Y_1|T_i}(y_1|T_i, H_0)$ is identically distributed to $Y_j|T_i, j \neq 1$, whose distribution is shown in (5).

We characterize Z_i as $f_{Z_i}(z_i) = \int_0^\infty \sum_{n_i=0}^\infty f_z(z_i|n_i, t_i)\, p(n_i|t_i) f_{T_i}(t_i) dt_i$, where $p(n_i|t_i)$ and $f_{T_i}(t_i)$ have been characterized in previous sections as Generalized Poisson and Fréchet distributions, respectively. The pdf $f_{Z_i|n_i,T_i}(z_i|n_i, t_i)$ can be computed by convolving the distributions of $Y_j|T_i$, for $j = 1 \ldots, n_i$. Formally,

$$f_z(z_i|n_i, T_i) = \begin{cases} \delta(z_i) & n_i = 0 \\ n_i\left(f_{Y_0|T_i} * \cdots * f_{Y_{n_i}|T_i}\right)(z_i n_i) & n_i > 0 \end{cases}. \qquad (6)$$

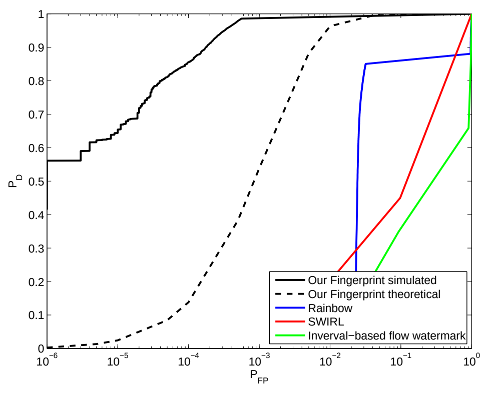
Note that the case $n_i = 0$ is not possible when H_1 holds because the cell from our fingerprint must arrive inside the interval. Also note that the convolution is evaluated at $z_i n_i$, this fact comes from (4) where the sum is multiplied by $1/n_i$.

Lastly, we characterize W, as $f(w) = f_{z_1 \cdot z_2 \cdots z_L}(w)$, where the density of the product of two independent random variables can be obtained as $f_{z_1 \cdot z_2}(z) = \int_z^1 \frac{1}{x} f_{z_1}(x) f_{z_2}(\frac{z}{x}) dx$.

So we calculate the value of the threshold, η, as the $(1 - P_F)$th quantile of $f_{w|H_0}$ and the probability of detection as $P_D = 1 - F_{w|H_1}(\eta)$, where F_w denotes the cumulative distribution function of f_w. As an example, Figure 4 shows the theoretical P_D as a function of L for Uvigo traffic and $P_F = 10^{-6}$.

VI. Results

We have created a simulator to show how close real results are from our predictions, and to compare with other existing



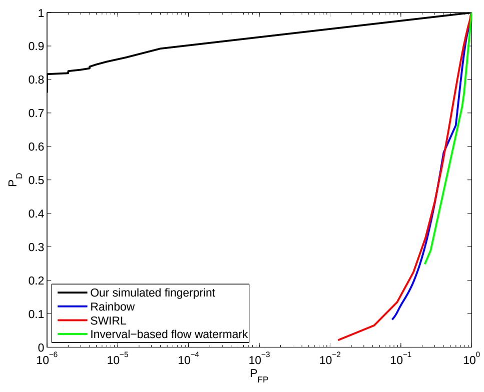


Fig. 5. Simulator Results for (a) NASA web server with 20 requests and (b) UVigo with 3 requests

approaches. Then we implement the fingerprint in the live Tor network against a HS.

A. Simulator

We create a simulator to validate our theorical analysis and to compare with other existing approaches: Rainbow [10], SWIRL [11] and Interval-based [12]. We send a HTTP request every 10 seconds, each request generates two cells (cf. Figure 2). Those flows are fingerprinted with each scheme. To each cell we add a delay equal to a measured one to preserve the time correlation. To model the behavior of other users, we send HTTP requests in the same way as measured in one of the logs.

Each experiment is simulated 10 million times. We run this simulator in two different scenarios. The first case assumes that the other users' requests are the same as for UVigo with 3 HTTP requests. And the second scenario simulates the traffic of the web server at NASA's Kennedy Space Center. This second server is considerably busier than the first one, so we increase the number of requests to 20 HTTP requests.

Results are shown in Figures 5(a) and 5(b). We see that this small number of requests is enough to achieve a very good performance. The performance of our algorithm, measured by the P_D for a given P_F is several orders of magnitude better than other fingerprinting schemes for the same number of messages. We also see that simulation results give a better performance than the analysis. This is due to the fact that in our analysis we assumed no autocorrelation in the log and in the delays, assumption that does not hold in reality.

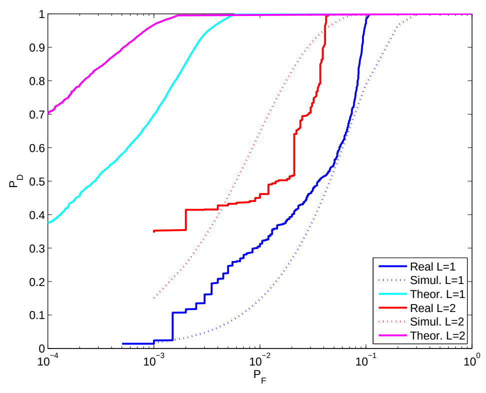


Fig. 6. Real Implementation, Simulation and Theoretical Results for Uvigo web server

B. Real Implementation

Obviously, simulations are not fully realistic. For instance, our simulator assumes that no cells reach the HS other than the two generated by each request, which may lead to overestimating performance. This means that we can filter off all the control cells (i.e. padding, create and destroy [2]) and keep only the relay cells that carry actual information.

The real implementation was done using three computers connected to live Tor: a HS and two clients, one that creates the fingerprint and the other that sends requests according to the log of the simulated machine. We do not perform any kind of filtering, keeping all the controll cells.

In our experiments a request is sent 10 seconds after the response is obtained. The gap between watermarks is fixed to 30 seconds. We create 1000 watermarks for each experiment. The experiment uses UVigo log with $L = 1$ and 2 requests. Results are shown in Figure 6. We see that the lack of filtering reduces the performance of the real implementation compared to the simulator and gives very similar results to the theoretical ones.

C. Detectability

We have mentioned that our scheme is complete undetectable through the TCP's intrinsic features as they are not modified, i.e. we do not add any extra delay to any message. So actual watermark detection algorithms such as Backlit [6] or MFA [7] cannot detect our fingerprint as they use those characteristics. This also makes nearly impossible that an intermediate node detects that the flow is fingerprinted.

We also want to prevent that any other party can detect that it is receiving a fingerprinted flow, i.e. we want a low detectability. There are two ways Bob could detect our watermark: the first, due to the modification of the pdf of the number of received messages, and the second, due to the uncommonness of the messages pattern. We do not consider the second, as we have assumed that Alice's messages follow a normal user's pattern.

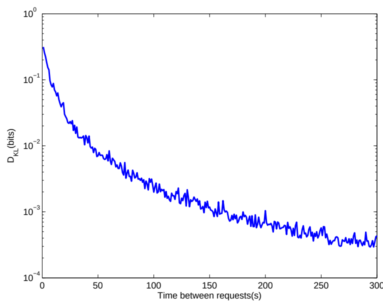


Fig. 7. Detectability using the KLD

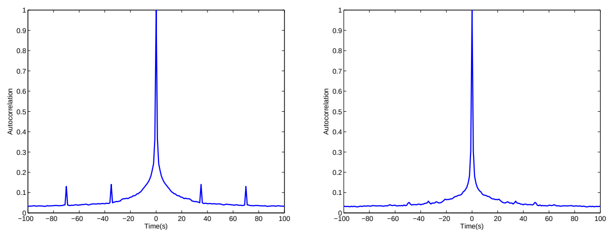


Fig. 8. Autocorrelation of the number of requests per second with time between requests fixed to 30 and uniform between 0 and 60

We measure the detectability using the KLD of the distributions of requests with the watermark flow and in its absence. Note that the detectability decreases with the time between requests, T_{req}, as seen in Figure 7, but we also want that our fingerprinting flow is done in a reasonable time.

Notice that the anonymous server may try to detect the watermark using higher-order statistics. Specifically, an increment of the autocorrelation periodically in $T_{req} + E[T_i]$ indicates the presence of this watermark. This is shown in Figure 8(a), where the parameters are readily obtained. This problem can be easily solved by making the time between requests pseudorandom as seen in Figure 8(b).

VII. Conclusion

This paper proposes a non-blind fingerprint for a flow created by the client. Our proposed scheme outperforms existing methods. This is due to two reasons: first, information generated during the creation of the fingerprint is used in the detection, and second, the use of an optimal decoder.

The fingerprint is constructed by sending requests, each request determines one interval. A prediction of the time of arrival is done for each request. The proposed detector applies Neyman-Pearson lemma to decide whether the flow is watermarked. The performance is studied theoretically and empirically, through both a simulator and a real implementation of the algorithm. Results show that we can create a fingerprint with very few requests: less than 10 for a server with little traffic and of a few tens for a busier web server.

An implementation of the attack against a real HS connected to the Tor network has been carried out, and shows a performance similar to the theoretical results. We also study the detectability of the algorithm, and see that the larger the average time between requests, the less detectable our algorithm is, and that we need the time between intervals to be pseudorandom to avoid detection through autocorrelation.

Acknowledgements

Research supported by the Iberdrola Foundation through the Prince of Asturias Endowed Chair in Information Science and Related Technologies.

References

[1] I. Clarke, O. Sandberg, B. Wiley, and T. Hong, "Freenet: A distributed anonymous information storage and retrieval system," in *Designing Privacy Enhancing Technologies*, ser. Lecture Notes in Computer Science, H. Federrath, Ed. Springer Berlin / Heidelberg, 2001, vol. 2009, pp. 46–66.

[2] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: the second-generation onion router," in *Proceedings of the 13th conference on USENIX Security Symposium - Volume 13*, ser. SSYM'04. Berkeley, CA, USA: USENIX Association, 2004, pp. 21–21.

[3] zzz (pseudonym) and L. Schimmer, "Peer profiling and selection in the I2P anonymous network," in *PET-CON 2009.1.*, TU Dresden, Germany.

[4] C. Farivar, "FBI halted one child porn inquiry because tor got in the way," June 2012. [Online]. Available: http://arstechnica.com/tech-policy/2012/06/fbi-halted-one-child-porn-inquiry-because-tor-got-in-the-way/

[5] S. J. Murdoch and P. Zieliski, "Sampled traffic analysis by internet-exchange-level adversaries," in *In Privacy Enhancing Technologies (PET), LNCS*. Springer, 2007.

[6] X. Luo, P. Zhou, J. Zhang, R. Perdisci, W. Lee, and R. K. C. Chang, "Exposing invisible timing-based traffic watermarks with BACKLIT," in *Proceedings of the 27th Annual Computer Security Applications Conference*, ser. ACSAC '11. New York, NY, USA: ACM, 2011, pp. 197–206.

[7] N. Kiyavash, A. Houmansadr, and N. Borisov, "Multi-flow attacks against network flow watermarking schemes," in *Proceedings of the 17th conference on Security symposium*, ser. SS'08. Berkeley, CA, USA: USENIX Association, 2008, pp. 307–320.

[8] L. Øverlier and P. Syverson, "Locating hidden servers," in *Security and Privacy, 2006 IEEE Symposium on*, may 2006, pp. 15 pp. –114.

[9] J. A. Elices, F. Perez-Gonzalez, and C. Troncoso, "Fingerprinting Tor's hidden service log files using a timing channel," in *Information Forensics and Security (WIFS), 2011 IEEE International Workshop on*, 29 2011-dec. 2 2011, pp. 1 –6.

[10] A. Houmansadr, N. Kiyavash, and N. Borisov, "RAINBOW: A robust and invisible Non-Blind watermark for network flows," in *Network and Distributed Systems Security Symposium*, G. Vigna, Ed. Internet Society, Feb. 2009.

[11] A. Houmansadr and N. Borisov, "Swirl: A scalable watermark to detect correlated network flows," in *NDSS*, 2011.

[12] Y. J. Pyun, Y. Park, D. S. Reeves, X. Wang, and P. Ning, "Interval-based flow watermarking for tracing interactive traffic," *Computer Networks*, vol. 56, no. 5, pp. 1646 – 1665, 2012.

[13] J. Neyman and E. S. Pearson, "On the problem of the most efficient tests of statistical hypotheses," *Philosophical Transactions of the Royal Society of London Series A Containing Papers of a Mathematical or Physical Character*, vol. 231, no. 694-706, pp. 289–337, 1933.

[14] X. Wang, S. Chen, and S. Jajodia, "Network flow watermarking attack on low-latency anonymous communication systems," in *Security and Privacy, 2007. SP '07. IEEE Symposium on*, may 2007, pp. 116 –130.

[15] S. Haykin, *Neural Networks: A Comprehensive Foundation (2nd Edition)*, 2nd ed. Prentice Hall, Jul. 1998.

[16] C. Forbes, M. Evans, N. Hastings, and B. Peacock, *Statistical Distributions*. John Wiley & Sons, 2011.

[17] P. C. Consul and F. Famoye, *Lagrangian Probability Distributions*. Birkhuser Boston, 2006, $10.1007/0 - 8176 - 4477 - 6_2$.

[18] K. Loesing, W. Sandmann, C. Wilms, and G. Wirtz, "Performance measurements and statistics of Tor hidden services," in *The 2008 International Symposium on Applications and the Internet*. Turku, Finland: IEEE, July 2008, pp. 1 – 7.