The Flow Fingerprinting Game

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

# *Electrical and Computer Engineering, University of New Mexico*
1 University of New Mexico, 87131 Albuquerque, NM
1 jelices@ece.unm.edu

* *Signal Theory and Communications Department, University of Vigo*
Campus Lagoas-Marcosende, 36310, Vigo, Spain
2 fperez@gts.uvigo.es

***Abstract*—Linking two network flows that have the same source is essential in intrusion detection or in tracing anonymous connections. To improve the performance of this process, the flow can be modified (fingerprinted) to make it more distinguishable. However, an adversary located in the middle can modify the flow to impair the correlation by delaying the packets or introducing dummy traffic.**

We introduce a game-theoretic framework for this problem, that is used to derive the Nash Equilibrium. As obtaining the optimal adversary delays distribution is intractable, some approximations are done. We study the concrete example where these delays follow a truncated Gaussian distribution. We also compare the optimal strategies with other fingerprinting schemes. The results are useful for understanding the limits of flow correlation based on packet timings under an active attacker.

I. Introduction

Becoming anonymous is a goal for network attackers to avoid prosecution but also a necessity for dissidents, human rights activists, etc. This anonymity can be achieved by passing the traffic through a chain of relays. Network attackers generally use compromised hosts called stepping stones as relays [1], while the rest of the users voluntary hosts that provide this service in low-latency anonymous networks. Interestingly, deanonymizing connections of these two kinds is essentially the same problem [2], which requires matching the egress and ingress flows. In these applications, the traffic is generally encrypted and sometimes divided into identical size packets, making the correlation of packet timings the most suitable solution.

Methods to find correlated flows can be classified in passive analysis and active watermarks. They differ in whether the flow is modified or not. Passive analysis needs slightly longer sequences but it can be less effective when timing patterns are very highly correlated, for instance two HTTP connections to the same web page. Watermarking schemes can avoid this problem but at the expense of being detectable [3], [4].

An adversary (AD), such as a stepping stone or an anonymous network relay, may modify the flow to prevent the correlation by introducing delays to packets or adding dummy packets to the flow. The existence of the AD has been considered in a passive analysis scenario by [5] and [6], where the AD is limited to delaying packets, and [7] analyzes a more complex AD model that, besides delaying packets, can also add and remove packets from the flow.

WIFS'2013, November 18-21, 2013, Guangzhou, China.
ISBN 978-1-4673-5593-3 ©2013 IEEE.

In this paper we study the limits of flow fingerprinting in an adversarial environment. Flow fingerprinting, which as flow watermarking, slightly perturbs the communication patterns, differs with the latter in that the modification is unique to each flow, so that every source sequence can be indistinctively identified. To the best of our knowledge, the only active flow fingerprint method is Fancy [8].

To overcome the loop of proposing an attack and creating an *ad-hoc* solution, we propose a game-theoretic framework and look for the optimum strategies that the players, traffic analyst (TA) and AD, should adopt. A similar game-theoretic framework has been used in other contexts such as Information Hiding [9], Source Identification [10] or in passive traffic analysis [7].

The rest of the paper is organized as follows: in Section II we introduce the notation, together with some basic concepts of game theory. Section III presents a rigorous definition of the flow fingerprinting game. In Section IV we derive the used detector. Section V studies the case when the attack channel is distributed as a truncated Gaussian. Section VI validates the performance and each player decisions using a simulator, and also presents a comparison of our scheme with Fancy in terms of error probability. Conclusions are presented in Section VII.

II. Notation

We use the following notation. Random variables are denoted by capital letters (e.g., X), and their individual realizations by lower case letters (e.g., x). The domains over which random variables are defined are denoted by script letters (e.g., $\mathcal{X}$). Sequences of n random variables are denoted with X^n if they have random nature or by x^n if they are deterministic. X_i or x_i indicate the i?th element of X^n or x^n, respectively. The probability distribution function (pdf) of a random variable X is denoted by $f_X(x),\ x \in \mathcal{X}$. We use the same notation to refer to pdf of sequences, i.e. $f_{X^n}(x^n),\ x^n \in \mathcal{X}^n$. When no confusion is possible, we drop the subscript in order to simplify the notation. We denote with Δ the difference

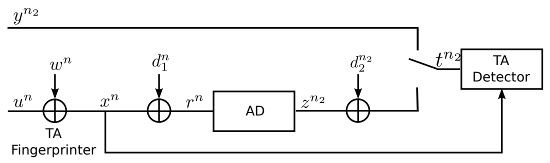


Fig. 1: Model of the FFG

operation of a sequence, i.e $\Delta x^n = \{x_2 - x_1, \ldots, x_n - x_{n-1}\}$ and with $\Delta x_i = x_{i+1} - x_i$ the ith element of this sequence.

A. *Performance Metrics*

To measure performance, we use two metrics: the probability of detection (P_D) and the probability of false positive (P_F). Given two hypotheses: H_0 and H_1, 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.

Typically, performance is graphically represented using the so-called ROC (Receiver Operating Characteristic) curves, which represent P_D vs. P_F. In order to compare different ROCs in a simple way, we use the AUC (area under the ROC curve), a measure that takes a value of 1 in the case of perfect detection and 0.5 in the case of random choice.

B. *Game Theory*

Game theory is the mathematical study of interaction among intelligent rational decision-makers. Formally, a two player game is defined as a quadruple $G(A_1, A_2, u_1, u_2)$, where $A_i = \{a_{i,1}, \ldots a_{i,n_i}\}$ are the actions available to the i player, $u_i : A_1 \times A_2 \mapsto \mathbb{R},\ i = 1, 2$ is the utility function or payoff of the game for player i. An action profile is the double $a \in A_1 \times A_2$. We are interested in zero-sum games, where $u_1(a) + u_2(a) = 0, \forall a \in A_1 \times A_2$, which means that the gain (or loss) of utility of player 1 is exactly balanced by the losses (or gains) of the utility of player 2. In this case, we can simplify the game notation to a triplet $G(A_1, A_2, u)$, where $u = u_1 = -u_2$.

We say that an action profile $(a_{1,i^*}; a_{2,j^*})$ represents a Nash equilibrium (NE) if $u(a_{1,i^*}; a_{2,j^*}) \geq u(a_{1,i}; a_{2,j^*})\quad \forall a_{1,i} \in A_1$ and $u(a_{1,i^*}; a_{2,j^*}) \leq u(a_{1,i^*}; a_{2,j})\quad \forall a_{2,j} \in A_2$, intuitively this means that none of the players can improve his utility by modifying his strategy assuming the other player does not change his own.

Games can be classified in simultaneous games, where both players move unaware of the other player action, and sequential games, where later players have some knowledge about earlier actions. In sequential games, an action profile is a subgame perfect equilibrium (SPE) if it represents a NE of every subgame of the original game. Therefore, a SPE is a refinement of the NE that eliminates non-credible threats.

III. Flow Fingerprinting Game

The Flow Fingerprinting Game (FFG) is represented in Figure 1. In this game, there are two players: the Traffic Analyst (TA) and the Adversary (AD).

The task of the TA is to accept or reject the hypothesis that a flow t^{n_2} is indeed the same flow as a known one, u^n. To improve the efficiency, the TA can modify the flow by embedding a fingerprint w^n. Due to the nature of the problem the modification must be additive, i.e., $x^n = u^n + w^n$. We constrain the fingerprint to delay any packet at most W_C seconds. This flow suffers a network delay of d_1^n before reaching the AD. We denote by r^n the flow received by the AD.

The goal of the AD is to modify the flow, producing z^{n_2}, in such a way that the detector decide that this sequence is not related with u^n. In order to do this, the AD can delay any packet at most A_C seconds and add up to $P_A \cdot n$ dummy packets, hence P_A is the maximum ratio between chaff and real traffic. We denote by a^n the sequence of delays inserted by the AD to each packet, and by c^{n_A} the chaff packets timing sequence, which consists of n_A packets. The output flow of the AD z^{n_2} suffers an additional delay $d_2^{n_2}$ due to the network between the AD and the detector.

We represent by D the delay suffered by a packet in the whole path, i.e. $D = D_1 + D_2$. Note that ΔD is the packet delay variation (PDV), also called jitter.

Let Y^{n_2} represent flows without any relation to x^n, but from the same application, and we assume that $f_{\Delta Y}(\Delta y)$ is known by both players and define the hypotheses:

$$H_0 : t^{n_2} \text{ is not a fingerprinted version of } x^n$$
$$H_1 : t^{n_2} \text{ is a fingerprinted version of } x^n.$$

We define the FFG as follows:

Definition 1. The $FFG(A_{TA}; A_{AD}; u)$ is a simultaneous, zero-sum game played by the TA and the AD, where

- The set of actions the TA can choose from, i.e. A_{TA}, is the duple of possible fingerprint sequences w^n and acceptance regions Λ_1 for which P_F is below a certain threshold η:
$$\begin{aligned} A_{TA} =& \{w^n \times \Lambda_1 : \forall i \in [1, n]\, (0 \leq w_i \leq W_C) \\ & \wedge Pr(Y^m \in \Lambda_1) < \eta\} \end{aligned} \tag{1}$$
- The set of possible attacks that the AD can choose from
$$\begin{aligned} & A_{AD} = \{f(z^{n_2}|r^n) : \exists a^n \left[\forall_i \in [1, n]\ (0 \leq a_i \leq A_C)\right. \\ & \wedge \exists c^{n_A} \left(\frac{n_A}{n} \leq P_A \wedge \forall_j \in [1, n_A]\, (0 \leq c_j \leq r_n + A_C)\right. \\ & \left.\left.\wedge\ z^{n_2} = \text{sort}((r^n + a^n)||c^{n_A})\right)\right]\}, \end{aligned} \tag{2}$$
where $||$ represents the concatenation of sequences, and sort(x^n) is a function that returns a sorted version of the input sequence.
- The utility function is P_D, namely: $u(A_{TA}, A_{AD}) = Pr(T^{n_2} \in \Lambda_1 | H_1)$.

A. *Subgame Perfect Equilibrium*

As the players choose their actions in a given order, then the SPE needs to assume that a given player knows which actions have taken place before his own (otherwise the player could improve its utility given this information). Hence the solution to the game is:

$$u = \max_{w^n} \min_{A_{AD}} \max_{\Lambda_1} u(A_{TA}, A_{AD}). \tag{3}$$

.

As correlating directly timing sequences needs a precise estimation of f_D and in a real implementation this is difficult to obtain, the use of the difference timing sequence, known as inter-packet delays (IPDs), seems more reasonable and has been widely adopted in the literature [8], [5], [7]. The optimal detector, according to Neyman-Pearson Lemma, is the likelihood ratio test:

$$\Lambda_1(t^{n_2}, x^n, \hat{f}(z^{n_2}|r^n)) = \int_{\mathcal{R}^n}\int_{\mathcal{Z}^{n_2}} \frac{f_{\Delta D_2^{n_2}}(\Delta(t^{n_2}-z^{n_2}))}{f_{\Delta Y^{n_2}}(\Delta t^{n_2})} \cdot \hat{f}(z^{n_2}|r^n) f_{\Delta D_1^n}(\Delta(r^n - x^n)) dz^{n_2} dr^n. \tag{4}$$

where $\hat{f}(z^{n_2}|r^n)$ is the assumed distribution of $f(z^{n_2}|r^n)$ by the detector. The test chooses H_1 whenever $\Lambda_1(t^{n_2}, x^n, \hat{f}_{Z^{n_2}|R^n}) \geq \epsilon$, where ϵ is a threshold chosen to achieve $P_F < \eta$. At the SPE, $\hat{f}(z^{n_2}|r^n) = f(z^{n_2}|r^n)$ that gives a utility of

$$u = \max_{w^n} \min_{f(z^{n_2}|r^n)} Pr(\Lambda_1(T^{n_2}, u^n + w^n, f(z^{n_2}|r^n)) > \epsilon). \tag{5}$$

Unfortunately, solving (5) is a computationally intractable problem. In the following sections, we try to approximate the SPE limiting the available actions for each user.

IV. Detector

In this section, we constrain the detector to be implemented in two steps: first, a matching process takes place that outputs two sequences of the same size, and then a likelihood test that needs a one-to-one correspondence between the flows is constructed. Denoting by $m(x^n, t^{n_2})$ the matching function, the action space for TA can be redefined as

$$A_{TA} = \{w^n \times m(x^n, t^{n_2}) \times \Lambda_1 : \forall i \in [1,n]\ [0 \leq w_i \leq W_C \wedge \exists j \in [1, n_2](m(x_i, t^{n_2}) = t_j)] \wedge Pr(m(x^n, y^{n_2}) \in \Lambda_1) < \eta\} \tag{6}$$

A. Matching Process

When dummy packets are added, i.e. $P_A > 0$, there does not exist a one-to-one relation between the flows x^n and t^{n_2}. To deal with this problem, we match each packet of x^n with the most likely from t^{n_2}, later removing those packets of t^{n_2} that have no correspondence on x^n.

We represent the fact that the ith packet from x^n is paired with the jth packet from t^{n_2} by $p(i) = j$. Let $\mathcal{M}$ be the set of all injective functions from $\mathcal{N} = \{1, \ldots, n\}$ to $\mathcal{N}_2 = \{1, \ldots, n_2\}$, i.e $\forall i_1, i_2 \in \mathcal{N},\ p(i_1) = p(i_2) \implies i_1 = i_2$. Then the matching function $m(x^n, t^{n_2})$ is the function from $\mathcal{M}$ that minimizes the mean square error between x^n and a shifted version of t^{n_2} as follows: $m = \arg\min_{\mathcal{M}} \sum_{i=1}^n (t_{p(i)} - x_i - \rho - E(a_i))^2$ where $E(a_i)$ is the expected value for the delay added by AD to the ith packet, recall that $f(a^n|x^n)$ is assumed to be known by the detector, and ρ is a synchronization constant equal to the sample mean of the delays, i.e. $\rho = \frac{1}{n}\sum_{i=1}^n d_i$. In a real implementation, where the sample mean is unknown, ρ can be obtained through an exhaustive search (self-synchronization property).

B. Likelihood Test

Confining the detector to those based on first-order statistics of the IPDs for feasibility reasons, the optimal likelihood ratio test becomes:

$$\Lambda_1(t^n, x^n, f(a^n|\Delta x^n)) = \sum_{i=1}^{n-1} \Bigg(\iint_{\mathcal{A}^2} \frac{f_{\Delta D}(\Delta t_i - \Delta \hat{a}_i)}{f_{\Delta Y}(\Delta w_i)} \cdot f_{A_{i,i+1}|\Delta X^n}(\hat{a}_{i,i+1}|\Delta x^n) d\hat{a}_i d\hat{a}_{i+1} \Bigg), \tag{7}$$

where

$$f(a^n|\Delta x^n) = \int_{\mathcal{R}^n} f(a^n|\Delta r^n) f_{\Delta D_1}(\Delta(r^n - x^n)) dr^n. \tag{8}$$

V. Truncated-Gaussian Attack Channel

As finding the distribution $f(a^n|\Delta r^n)$ that minimizes $Pr(\Lambda_1(t^n, x^n, f(a^n|\Delta r^n)) > \epsilon)$ may not be feasible for the AD, we study the particular case when the AD introduces delays following a truncated Gaussian in the interval $[0, A_C]$, i.e., $a_i \sim N(\mu_i, \sigma^2 | 0 \leq a_i \leq A_C)$. Note that this model includes as limits the deterministic attack $\sigma^2 \to 0$, and a uniform attack $\sigma^2 \gg A_C^2$. In this scenario AD selects the sequence of means μ^n and the variance σ^2, as well as the timing for the dummy packets, hence $A_{AD} = \{\mu^n \times \sigma^2 \times c^{n_A}\}$.

From (3), the SPE actions, denoted with the superscript *, are

$$a_{TA}^* = \arg\max_{w^n} \min_{A_{AD}} Pr(\Lambda_1(m(T^{n_2}, x^n), x^n, f(a^n|x^n)) > \epsilon) \tag{9}$$

$$a_{AD}^* = \arg\min_{A_{AD}} Pr(\Lambda_1(m(T^{n_2}, x^n), x^n, f(a^n|x^n)) > \epsilon) \tag{10}$$

where

$$Pr\Big(\int_{\mathcal{X}^n} \Lambda_1(m(Y^{n+\lfloor P_A \cdot n \rfloor}, x^n), x^n, f(a^n|x^n)) dx^n \leq \epsilon\Big) = \eta. \tag{11}$$

As the AD must decide its action in real time and (10) is computationally expensive, we approximate each decision individually, as explained next

1) Mean sequence (μ^n): a good approximation when ΔD_2 has zero mean and its variance is much smaller than ΔY (as it is the case in practice) is

$$(\mu^n)^* \approx \arg\min_{\mu^n} \sum_{i=1}^{n-1} \log f_{\Delta Y}(\Delta r_i + \Delta \mu_i). \tag{12}$$

Note that under this approximation the AD is maximizing the likelihood of $r^n + a^n$ coming from y^n, i.e., making the sequence as typical as possible.

2) Variance (σ^2): The value of σ^2 presents a trade-off: small values of σ^2 make a^n to be chosen so that the sequence looks more similar to the typical sequence of Y^n but with the disadvantage that the uncertainty of a^n for the detector is smaller. Recall that the detector is assumed to know $f(a^n|\Delta x^n)$.

We calculate the value of σ^2 that minimizes (10) empirically using the simulator and the scenarios presented in next section. A graph of the variation of AUC with σ is depicted in Figure 2.

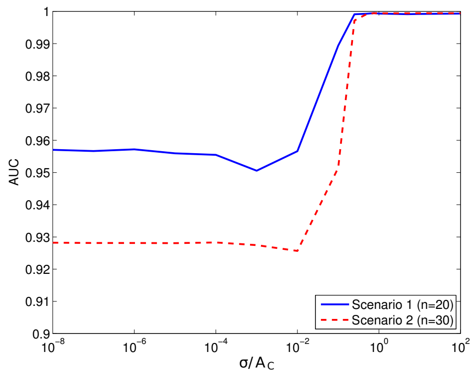


Fig. 2: Dependence of the performance with σ ($A_C = 250$ms, $P_A = 0$, $W_C = 0$ms).

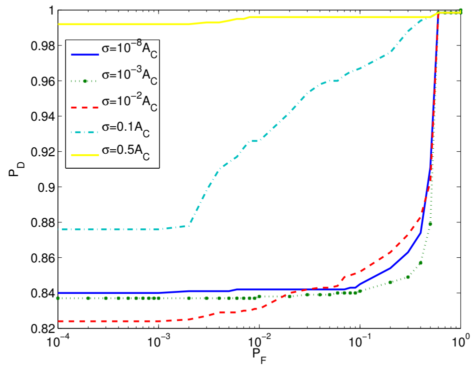


Fig. 3: ROCs for different variances in Scenario A ($n = 20$, $A_C = 250$ms, $P_A = 0$, $W_C = 0$ms).

The constant region on each side corresponds to a deterministic ($\sigma \to 0$) and to a uniform attack ($\sigma \gg A_C$). The minimum performance is obtained in the interval $[10^{-3}, 10^{-2}] \cdot A_C$. This small variance makes the attack virtually deterministic ($\sigma \to 0$), implying that making the sequence more typical is the prevailing factor. Figure 3 let us see better the difference of performance for different values of σ. In the following unless otherwise specified, the AD chooses $\sigma = 10^{-2} \cdot A_C$.

3) Chaff traffic (c^{n_A}): Assuming that $f_{\Delta D}(\Delta d)$ is symmetric around its mean and unimodal (as it is the case in practice), then the matching process selects those packets that give a higher value of $\Lambda_1(t^n, x^n, f(a^n|\Delta x^n))$. Therefore, the AD will choose c^{n_A} so that these dummy packets are removed in the matching process. On the other hand, the AD will need these packets to force the TA to consider longer possible sequences for y^{n+n_A}. According to (11), longer sequences of Y^{n_2} will increase ϵ.

4) Fingerprint (w^n): This is the converse problem as the delays for the AD. The TA wants x^n to be as distinguishable as possible from the typical sequence of y^n. Then,

$$(w^n)^* \approx \arg\max_{w^n} \sum_{i=1}^{n-1} \log f_{\Delta Y}(\Delta u_i + \Delta w_i). \qquad (13)$$

VI. Performance

In this section we present the two scenarios we use in the remaining of the paper and construct a simulator. Afterwards, we compare the performance between the approximated SPE point derived in Sections IV and V and the result when one of the players unilaterally deviates from it. This will show that the utility for this player is reduced.

A. Scenarios and Simulator

We present two scenarios, A and B, that we use in the sequel to evaluate the performance. Scenario A represents a stepping stone that forwards SSH traffic inside the Amazon Web Services network. The TA-Fingerprinter, the AD and the TA-Detector (cf. Figure 1) are EC2 instances located in Virginia, Oregon and California, respectively. We use the IPDs from 8746 replayed SSH connection captures with 64 million packets from [11] and [12]. The simulated delays correspond to Scenario 10 from [13].

Scenario B simulates a web page accessed from the Tor network whose real origin is to be found. In it, the TA-Fingerprinter corresponds to the web server, the AD to the Tor entry relay, and the TA-Detector to the client. We use the IPDs of 113690 replayed HTTP connections that sum around 139 million packets taken from the same repositories. The delays correspond to the measurements of Scenario 11 from [13].

The calculation of Λ_1 in (7) needs an estimation of $f_{\Delta D}$ and $f_{\Delta Y}$. To this end we apply kernel smoothing techniques [14]. As it is customary, we separate the data of each scenario into two subsets: training, to estimate the pdfs, and test, used in the simulator, assigning 50% of the samples to each.

Simulations are carried out in the following way. First, we generate a timing information from the measured IPDs (as explained in the following paragraphs), u^n. Then we introduce the fingerprint w^n according to (13) obtaining x^n. Afterwards, d_1^n is added to each packet of x^n using the measured delays, obtaining r^n. Next, we generate a^n and c^{n_A} according to the procedure discussed in Section V, obtaining z^{n_2}. Subsequently, we introduce another delay d_2^n to generate sequence $t_1^{n_2}$.

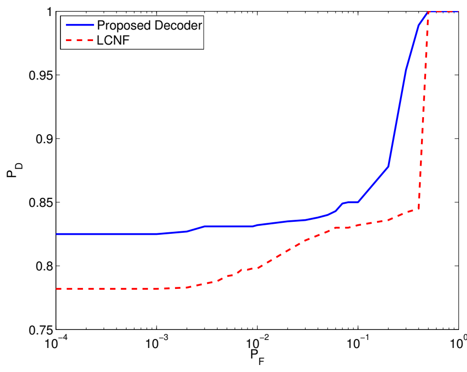
We generate a second timing sequence y^{n_2} with $n_2 = n + \lfloor P_A \cdot n \rfloor$. This sequence has the purpose of evaluating the performance under H_0. Finally, we use the test from (7) to obtain both $\Lambda_1(t_1^{n_2}, x^n, f(a^n|x^n))$ and $\Lambda_1(y^{n_2}, x^n, f(a^n|x^n))$. This experiment is repeated 10^4 times when the simulation has an AD or 10^5 when no AD is present. For different values of ϵ we obtain P_D as the rate of $\Lambda_1(t_1^{n_2}, x^n, f(a^n|x^n)) > \epsilon$, and P_F as the rate of $\Lambda_1(y^{n_2}, x^n, f(a^n|x^n)) > \epsilon$.

Sequences u^n and y_2^n are generated in the following way: we place all the IPDs from the test set on an order-preserving list. The starting point is randomly selected from the list and the generated IPDs are the following values.

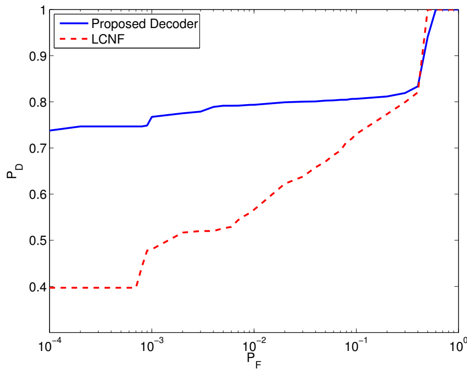
The provided delays are sampled each 50ms. We select one value randomly that we consider time 0 ms; the following values represent the delay at times 50 ms, 100 ms and so on. To obtain the delays at times where we do not have a measurement, we use linear interpolation.

B. Detector comparison

We compare our detector with the one used in [7] that we denote as LCNF (Linking Correlated Network Flows). This

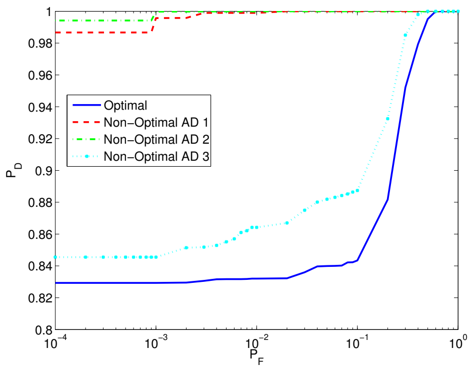


(a) Scenario A ($n = 20$, $A_C = 250$ms, $P_A = 1$, $W_C = 0$ms).

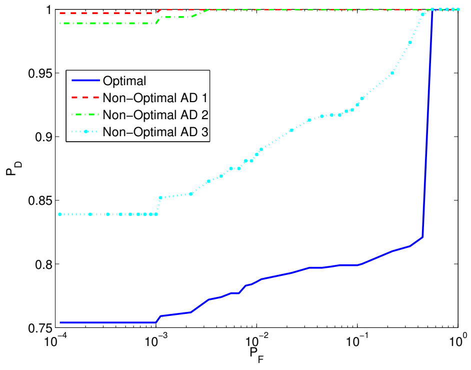


(b) Scenario B ($n = 30$, $A_C = 250$ms, $P_A = 1$, $W_C = 0$ms).

Fig. 4: Comparison of the proposed detector with LNCF.



(a) Scenario A ($n = 20$, $A_C = 250$ms, $P_A = 1$, $W_C = 0$ms).



(b) Scenario B ($n = 30$, $A_C = 250$ms, $P_A = 1$, $W_C = 0$ms).

Fig. 5: Comparison of different AD strategies

detector is claimed to be the optimal among those that estimate a value for a^n and compensate it. Results are depicted in Figures 4a and 4b for Scenarios A and B, respectively. We see that our detector outperforms LCNF in both scenarios. Note that our detector is derived to be optimal among those which use just first-order statistics. Hence, by using higher-order statistics the performance could be improved at the expense of a higher computational cost.

C. AD actions

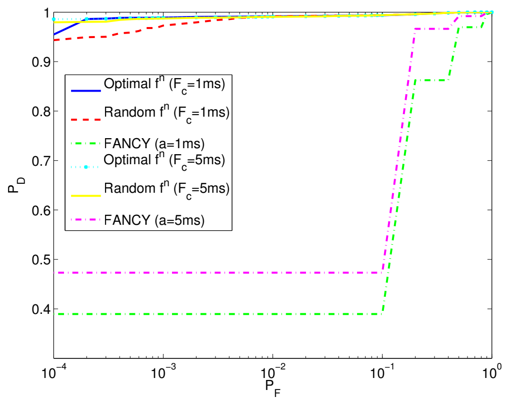
We compare the ROC under an optimal adversary with those corresponding to three non-optimal adversaries: a) the AD selects μ_i randomly according to a uniform distribution between 0 and A_C; b) the AD chooses its delays from a uniform distribution; c) the AD chooses its delays a^n as explained in Section V but the chaff traffic is selected randomly, i.e. c^{n_A} is an i.i.d. sequence uniformly distributed between the timing of the first packet and the last one. Results are depicted in Figures 5a and 5b. The conditions used are $A_C = 250$ms, $P_A = 1$, $W_C = 0$ms, $n = 20$ in Scenario A, and $n = 30$ in Scenario B. We see that the delay distribution has a great effect on both scenarios but the dummy packets have a more significant influence on Scenario B. In any case, notice that the AD attack derived in Section V impairs the flow correlation in a much more severe way than the suboptimal strategies.

D. Fingerprint actions

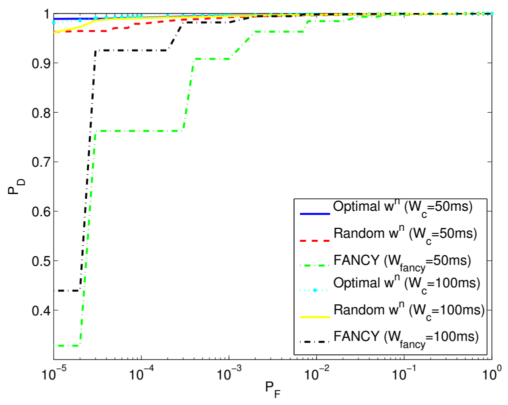
We compare the optimal fingerprint with two other TA-Fingerprinter strategies: a) the delays are chosen from a uniform distribution between $[0, W_C]$; b) the Fancy algorithm, that embeds its fingerprint as follows $\Delta x = \Delta u + W_{fancy} \cdot w^{n-1}$, where each $w_i \in \pm 1$, for all $i = 1, \ldots, n-1$, and W_{fancy} is Fancy's fingerprint amplitude. We compare these strategies under the same maximum IPD variation, therefore $W_{fancy} = W_C$. We study the performance under two conditions: 1) no AD is present ($A_C = 0$ms, $P_A = 0$); 2) the AD is present. In the no AD situation we use $n = 5$, $W_C = 1$ and 5ms in Scenario A, and $n = 20$, $W_C = 50$ and 100ms in Scenario B. Results are depicted in Figures 6a and 6b respectively. The Fancy detector outputs a sequence of bits which is error-corrected; for this reason, the ROC shows a stepwise behavior. We can see the significant difference between Fancy and the other two mechanisms, that is due to the optimality of the detector. The difference between choosing w^n optimally or randomly exists but is not so notable. In the AD environment ($A_C = 250$ms and $P_A = 1$), shown in 7a and 7b for Scenarios A and B, and again the optimal fingerprint improves the performance of a uniform fingerprint, but this time the difference is more noticeable in the plots than in no AD situation. Fancy's ROC is not depicted under present AD conditions as it is not designed to withstand an active AD.

VII. Conclusion

We have analyzed the flow fingerprinting game, that consists in deciding if two flows are linked or not, allowing a slight perturbation at the fingerprinter and having an adversary in the middle who tries to impair the correlation. Using this framework, we obtain the optimal detector that uses first-order

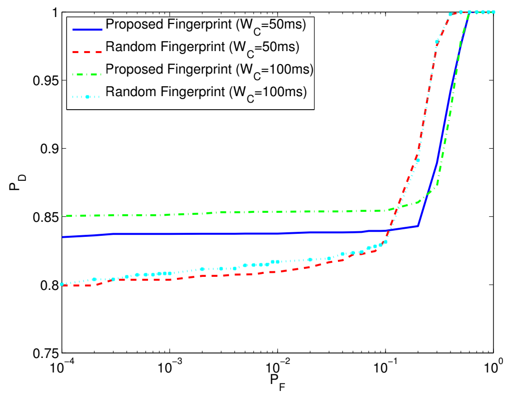


(a) Scenario A ($n = 5$).

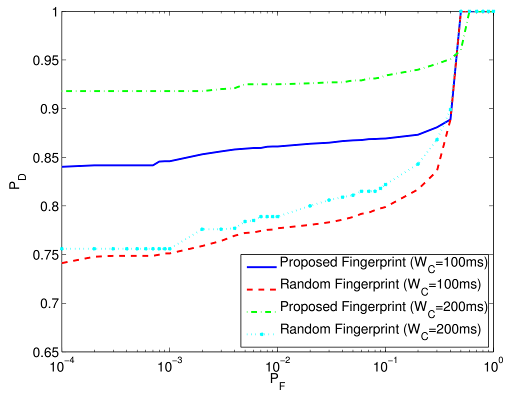


(b) Scenario B ($n = 20$).

Fig. 6: Performance with no AD ($A_C = 0$ms and $P_A = 0$).



(a) Scenario A ($n = 20$).



(b) Scenario B ($n = 30$).

Fig. 7: Performance with AD ($A_C = 250$ms and $P_A = 1$).

statistics. Then we study the case where the adversaries' delays come from a truncated Gaussian, concluding that the adversary has to act nearly deterministically even if the detector knows the distribution. Finally, we validate the optimality of the user actions using a simulator. This simulator is also used to show that the proposed scheme outperforms the state-of-the-art in flow fingerprinting.

Acknowledgment

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

References

[1] S. Staniford-Chen and L. Heberlein, "Holding intruders accountable on the internet," in *Security and Privacy, 1995. Proceedings., 1995 IEEE Symposium on*, may 1995, pp. 39 –49.

[2] J. A. Elices and F. Perez-Gonzalez, "Fingerprinting a flow of messages to an anonymous server," in *Information Forensics and Security (WIFS), 2012 IEEE International Workshop on*. IEEE, 2012, pp. 97–102.

[3] 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*. ACM, 2011, pp. 197–206.

[4] Z. Lin and N. Hopper, "New attacks on timing-based network flow watermarks," in *USENIX Security Symposium*. Bellevue,WA: USENIX Association, Aug. 2012.

[5] D. L. Donoho, A. G. Flesia, U. Shankar, V. Paxson, J. Coit, and S. Staniford, "Multiscale stepping-stone detection: detecting pairs of jittered interactive streams by exploiting maximum tolerable delay," in *Proceedings of the 5th international conference on Recent advances in intrusion detection*, ser. RAID'02. Berlin, Heidelberg: Springer-Verlag, 2002, pp. 17–35.

[6] A. Blum, D. Song, and S. Venkataraman, "Detection of interactive stepping stones: Algorithms and confidence bounds," in *Recent Advances in Intrusion Detection*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2004, vol. 3224, pp. 258–277.

[7] J. A. Elices and F. Pérez-González, "Linking correlated network flows through packet timing: a game-theoretic approach," 2013, submitted available: http://arxiv.org/abs/1307.3136.

[8] A. Houmansadr and N. Borisov, "The need for flow fingerprints to link correlated network flows," in *Privacy Enhancing Technologies*. Springer, 2013, pp. 205–224.

[9] P. Moulin and J. O'Sullivan, "Information-theoretic analysis of information hiding," *Information Theory, IEEE Transactions on*, vol. 49, no. 3, pp. 563–593, 2003.

[10] M. Barni and B. Tondi, "The source identification game: An information-theoretic perspective," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 3, pp. 450–463, 2013.

[11] D. Kotz, T. Henderson, I. Abyzov, and J. Yeo, "CRAWDAD trace set dartmouth/campus/tcpdump (v. 2004-11-09)," http://crawdad.cs.dartmouth.edu/dartmouth/campus/tcpdump, Nov. 2004.

[12] R. R. R. Barbosa, R. Sadre, A. Pras, and R. van de Meent, "Simpleweb/university of twente traffic traces data repository," Centre for Telematics and Information Technology University of Twente, Enschede, Technical Report, 2010.

[13] J. A. Elices and F. Pérez-González, "Measures to model delays on internet," http://www.unm.edu/~elices/captures.html, Jan. 2013.

[14] A. W. Bowman and A. Azzalini, *Applied Smoothing Techniques for Data Analysis: The Kernel Approach with S-Plus Illustrations (Oxford Statistical Science Series)*. Oxford University Press, USA, Nov. 1997.