Internet traffic privacy enhancement with masking: optimization and trade-offs

Alfonso Iacovazzi, and Andrea Baiocchi, *Member, IEEE*

Abstract—An increasing number of recent experimental works have demonstrated that the supposedly secure channels in the Internet are prone to privacy breaking under many respects, due to packet traffic features leaking information on the user activity and traffic content. We aim at understanding if and how complex it is to obfuscate the information leaked by packet traffic features, namely packet lengths, directions, times: we call this technique traffic masking. We define a security model that points out what the ideal target of masking is, and then define the optimized traffic masking algorithm that removes any leaking (full masking). Further, we investigate the trade-off between traffic privacy protection and masking cost, namely required amount of overhead and realization complexity/feasibility. Numerical results are based on measured Internet traffic traces. Major findings are that: i) optimized full masking achieves similar overhead values with padding only and in case fragmentation is allowed; ii) if practical realizability is accounted for, optimized statistical masking attains only moderately better overhead than simple fixed pattern masking does, while still leaking correlation information that can be exploited by the adversary.

Index Terms—Privacy, Traffic masking, Traffic flow classification, padding, fragmentation, Internet traffic.

1 Introduction

A Number of works over the last few years have given extensive experimental evidence that even within a secure channel, a packetized flow leaks information to an adversary through observation of features of traffic flows, e.g., the ordered sequence of packet lengths, packet inter-arrival times, packet directions. Based on these information that is available to an adversary even when the flow is carried within a secure channel (e.g. SSL/TLS or SSH connections), it has been shown that a number of approaches yields feasible algorithms to identify the type of service or application protocol run among a given set of alternatives (*traffic classification*; see [1][2][3][4][5] for surveys). Other privacy breaking attacks based on analysis of packet flow features have been demonstrated, e.g. to profile web access [6], to infer language of phone calls [7] or even conversation transcripts [8]. An extensive account of related literature is given in Appendix A.

This communication privacy break is a positive proof that ciphering does not conceal all relevant information of a packetized application flow; hence we aim at investigating *protection* of privacy against traffic analysis. Besides being a privacy issue, traffic analysis tools can be useful to network administrators and operators for enforcement of security policies and traffic filtering, or to support quality of service mechanisms. A key point for the robustness of those "legitimate" uses of traffic analysis tools is to check how much effort is needed to fool them. Those reasons motivate us to investigate how traffic feature leakage can be concealed to "adversaries" exploiting traffic analysis. We term this *traffic masking*.

During the years various countermeasures have been developed[1]. Wright et al. [10] make use of convex optimization techniques to modify the source packet lengths distribution in order to look like a target distribution (*morphing*), with minimum overhead. No explicit solution is provided and protocol complexity of morphing a given flow into a different one is not accounted for. Shui Yu et al. [11], [12] implement a new strategy of packet padding aiming at offering perfect anonymity on web browsing. The proposed solution allows to reduce overhead by exploiting as padding web pages that the user is expected to download in the future, according to a prediction algorithm. Luo et al. [13] propose to modify statistical features of a traffic flow by using a set of transformations both at the application and at the transport layer. In a recent work [9] K.P. Dyer et al. show that it is still possible to classify traffic flows after masking. They consider nine masking countermeasures applied to web pages, and show that they can still identify which webpage is being downloaded with standard algorithms, achieving accuracy as high as 98%.

As for contributions of this work, we define formally the problem of privacy against traffic classification, thus finding what the ideal traffic masking should do. Then, we define the achievable performance bounds for a masking algorithm, by defining an optimization problem to find an ideal masking algorithm that minimizes over-

• *A. Iacovazzi and A. Baiocchi are with the University of Roma Sapienza, Department of Information Engineering, Electronics and Telecommunications, Roma, Italy.*
E-mail: alfonso.iacovazzi@diet.uniroma1.it, andrea.baiocchi@uniroma1.it

Manuscript received October 1, 2012; revised February 13, 2013.

1. Some countermeasures against traffic analysis are already implemented by the main secure channel protocols, such as SSH, TLS and IPSec. Their main purpose is to conceal messages lengths by choosing a random amount of padding. It can be readily verified that random padding do not really achieve any effective masking [9].

Digital Object Indentifier 10.1109/TPDS.2013.72

head cost. Interestingly, we find that, with ideal masking, message fragmentation gives a marginal benefit as to overhead with respect to much simpler approaches where only message padding is used. We also define practical masking algorithms that allow trade-offs of leakage against implementation complexity. Trade-offs are illustrated with many numerical examples based on real traffic traces. It turns out that simple deterministic traffic masking can be the only practical solution for a full privacy protection, albeit it entails a bigger overhead than more sophisticated statistical masking.

In the rest of this paper we state the traffic privacy model framework in Section 2, then give ideal (Section 3) and practical (Sections 4) algorithms for traffic masking. Numerical results are provided in Section 5 and conclusions are drawn in Section 6.

2 Model of Traffic Flow Masking

Let us consider two end points (hosts) in a packet network, running a given application. The information flow between the two hosts is ciphered. In spite of using a secure channel for communication, still there is information leaking to an adversary observing the information flow between the two hosts, as evident from works cited in Appendix A. In the rest of this Section we set notation for the description of features of a generic application flow and define the adversary model and the corresponding concept of masking for perfect privacy against the given adversary model.

2.1 A general model of an application flow

Any application traffic flow between an initiator entity A and the responder entity B (e.g., client and server for the given flow, respectively) can be cast into a sequence of $N \geq 1$ message bursts[2]. Each burst consists of one or more messages in one direction (A→B or B→A). Bursts in the two opposite directions alternate, starting from the initial burst sent by the initiator A to the responder B.

A full description of the flow is obtained with:

- the vector $\mathbf{K} = [K_1, \ldots, K_N]$ of the numbers of messages in each burst;
- message lengths in each burst, denoted as $\mathbf{L}_i = [L_i(1), \ldots, L_i(K_i)]$, for $i = 1, \ldots, N$;
- message epochs[3], denoted as $\mathbf{T}_i = [T_i(1), \ldots, T_i(K_i)]$, for $i = 1, \ldots, N$; we use also *message gap times*, defined as $\Delta t_i(r) = T_i(r+1) - T_i(r)$, for $r = 1, \ldots, K_i$, with $T_i(K_i + 1) \equiv T_{i+1}(1)$.

Figure 1 depicts an example of two flows, where A and B sides are represented by vertical lines and time increases downward. In the example flows shown in Figure 1 the flow labeled (a) has $N = 4$, with $\mathbf{K} = [1, 3, 1, 1]$. For flow (b) it is $N = 6$ and $\mathbf{K} = [1, 1, 1, 1, 1, 1]$.

2. We use the generic term "message", since the model is applicable to general traffic flows, from web pages to IP layer flows.

3. In practice, time epochs refer to time-stamps associated to packets collected by the adversary at the traffic capture point in the network.

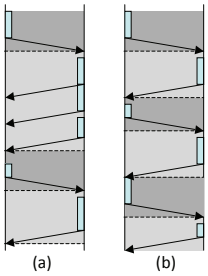

Fig. 1. Examples of message exchanges of two application flows: (a) one way data transfer, like http; (b) alternate messages, like most signaling and control protocols.

A more compact description is obtained by considering only *burst sizes* $B_i = \sum_{r=1}^{K_i} L_i(r)$ and *burst epochs* $\Theta_i = T_i(1)$, for $i = 1, \ldots, N$; hence a feature vector $\tilde{\mathbf{X}} = [\mathbf{B}, \boldsymbol{\Theta}]$. In other words, the flow description is simplified to the sequence of (alternating) burst lengths $B_1, B_2, \ldots, B_N$ and the burst start epochs (time stamps of the first packet of each burst), $\Theta_1, \Theta_2, \ldots, \Theta_N$.

2.2 Privacy against flow classification

Let the feature vector associated to a flow be $\mathbf{X}$. According to our definition, a full description of the flow entails $\mathbf{X} = [\mathbf{K}, \mathbf{L}, \mathbf{T}]$. In general, the feature vector can be a subset of the full description, according to adversary aims and resources, e.g., for real time classification only the first m messages of a flow could be considered, with m typically ranging from one up to several units. In case of web pages classification, more aggregate features could possibly be used by the adversary, e.g., the amount of bytes sent per burst or even the overall amount of bytes sent with the flow in either direction.

Each entry of $\mathbf{X}$ is modeled as a positive, discrete random variable[4] with a finite support $[1, \ldots, \ell]$ for some suitable constant ℓ. Let Ω denote the state space of $\mathbf{X}$.

We define a privacy model, where each flow can belong to one of M classes. An adversary can observe flow features by means of network traffic analysis and aims at identifying the class each flow belongs to. The example we focus on is *flow classification*, where the classes are applications, but results of this Section hold in general for classification problems.

Let us consider M applications, denoted by a label $\mathcal{A} \in \{1, \ldots, M\}$. We let $p_j(\mathbf{x}) = \mathcal{P}(\mathbf{X} = \mathbf{x} | \mathcal{A} = j)$, $\mathbf{x} \in \Omega, j = 1, \ldots, M$, be the *probability distribution function* (pdf) of the feature vector, conditional on the flow belonging to the j-th application; we further denote with $P_j = \mathcal{P}(\mathcal{A} = j)$ the a priori probability that a flow belongs to application j.

In general, the traffic masking operation includes introducing dummy flows, to modify the a priori probabilities P_j into new values Q_j, and transforming each flow

4. While $\mathbf{K}$ and $\mathbf{L}$ are natively discrete, the time values $\mathbf{T}$ can be made discrete by specifying a time quantum, e.g., the time measurement resolution or a small multiple thereof.

sent through the network so that the output flow features are given by $\mathbf{Y} = \phi(\mathbf{X}; \mathcal{A})$, thus altering the original feature pdf. The flow transformation implies message padding, fragmenting, insertion of dummy messages, message delaying. This transformation and the relevant modification of feature values can depend in general on the application the input flow belongs to, which must be known at the masking device. This is made explicit by highlighting the parameter $\mathcal{A}$ in the mapping $\phi(\cdot; \mathcal{A})$. This mapping induces a probability measure on $\mathbf{Y}$, denoted by $q_j(\mathbf{y}) = \mathcal{P}(\mathbf{Y} = \mathbf{y}|\mathcal{A} = j)$. We assume that the state space of $\mathbf{Y}$ is the same as that of $\mathbf{X}$, namely Ω.

As matter of example, if $\mathbf{X}$ reduces only to message lengths, a possible transformation is padding, i.e., $\mathbf{Y} = \mathbf{X} + \mathbf{U}$, $\mathbf{U}$ being a non negative vector, whose characteristics depend on the application $\mathbf{X}$ belongs to.

We assume an *eavesdropping* adversary, aiming at flow classification. The adversary can observe ciphered and masked flows (including dummy flows) and can detect the feature vector $\mathbf{y}$ for each observed flow. In other words, the adversary can collect samples $\mathbf{y}$ of the random variables $\mathbf{Y}$. Moreover, the adversary knows ciphering and masking algorithms used in the secure channel, and is given knowledge of the conditional pdfs of the masked flow features[5] $q_j(\mathbf{y})$. The aim of the adversary is to guess the application the original flow belongs to. This is summarized by an algorithm named $TA(\mathbf{Y}) : \Omega \to \{1, \ldots, M\}$ yielding the application label for the observed masked flow feature vector.

An overall scheme of the masking plus enciphering at sending side is shown in Figure 2. The reverse operations (deciphering and de-masking) take place at receiver side. Dummy flows are added to modify the a priori pdf P_j into Q_j, for $j = 1, \ldots, M$. Ideally, the Q_j's should be uniform ($Q_j = 1/M$). Coupled with perfect masking this brings the adversary success probability of correctly classifying observed flows to its theoretical minimum $1/M$. If dummy flows are suppressed to save overhead, the a priori application pdf can be exploited by the adversary, if known. Clearly, if one class is overwhelmingly more probable the the others, simple guessing gives the adversary a high success probability, i.e., the endpoint traffic is highly predictable.

Given the adversary model above, by *full masking* we mean removing any information leakage that could be exploited by the adversary to classify observed flows. With full masking, the success probability of the adversary is minimized to $1/M$. Conversely, *partial masking* lets some information leak by establishing a trade-off between privacy and feasibility/complexity/overhead/delay of masking. We also define *ideal masking* an algorithm that makes use of the knowledge of the whole feature vector of a flow to mask every message of it, whereas *practical masking* refers to the subset of algorithms that can run in real time, by processing messages of the masked flow as they arrive, independently at the two endpoints. A practical masking algorithm can use features of the first k messages to decide on masking of the $(k+1)$-th one.

5. As a matter of example, the adversary has a database containing a set of masked flows for each application, with metadata assessing the application that originated those flows; such a database can be used to train a classification algorithm; this is similar to a known/chosen plaintext eavesdropper model, depending on the way the database is constructed.

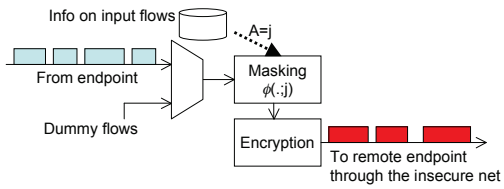

Fig. 2. Traffic flow masking block scheme at sending side: dummy flows are added to modify the a priori pdf of generating applications.

In the Appendix B it is proved that full masking entails finding a transformation ϕ of the original flow features $\mathbf{X}$, so that the output features $\mathbf{Y}$ have a same pdf irrespective of the input application, i.e., $Q_i = 1/M$ and $q_1(\mathbf{y}) = \cdots = q_M(\mathbf{y})$, $i = 1, \ldots, M$. In Section 3 we state an optimization problem to find the full masking algorithm that minimizes the average overhead within the set of ideal algorithms. In Section 4 we restrict our attention to practical algorithms.

3 Optimal Full Masking

We focus on the case of two applications ($M = 2$) and state an optimization problem that yields a constructive solution for a full masking algorithm $\phi(\cdot; \mathcal{A})$ that achieves minimum overhead in the set of ideal masking algorithms. This optimal full masking algorithm serves as a term of comparison for practical masking, while it is unfeasible to realize, both because of computational complexity and since in principle it requires the entire flow to be available to the masking device to decide upon each message transformation.

Let us assume the sample space (with non null probability) of application i be Ω_i and let $\omega_i = |\Omega_i|$ be the cardinality of Ω_i, for $i = 1, 2$. Outcomes of Ω_i are denoted $\mathbf{x}_r^{(i)}$, $r = 1, \ldots, \omega_i$ ($i = 1, 2$). For typical application flows, most feature values have null or negligible probability so that $\omega_i \ll |\Omega|$. As found in Section 2, full masking entails that the output flow features have the same pdf irrespective of the application that feeds the input of the masking device. To construct the masking algorithm $\phi(\cdot; \mathcal{A})$ we take all ordered couples $(\mathbf{x}_h^{(1)}, \mathbf{x}_k^{(2)})$, with $\mathbf{x}_h^{(1)} \in \Omega_1$ and $\mathbf{x}_k^{(2)} \in \Omega_2$, for $h = 1, \ldots, \omega_1$ and $k = 1, \ldots, \omega_2$. For each couple $(\mathbf{x}_h^{(1)}, \mathbf{x}_k^{(2)})$ we find the optimum masked flow with feature vector $\mathbf{y}_{h,k}$ that the flows in the couple can be mapped to by means of padding (including insertion of dummy messages), fragmentation and delaying. Optimum here refers to minimization of the overhead required to convert each

of the two flows of the couple into the masked flow $\mathbf{y}_{h,k}$. Full masking is obtained by requiring that $\mathcal{P}(\mathbf{Y} = \mathbf{y}_{h,k}|\mathcal{A} = 1) = \mathcal{P}(\mathbf{Y} = \mathbf{y}_{h,k}|\mathcal{A} = 2) \equiv c_{h,k}$ for all h and k. Then, we have

$$\mathcal{P}(\mathbf{Y} = \mathbf{y}_{h,k}) = \mathcal{P}(\mathcal{A} = \mathcal{A}_1)\mathcal{P}(\mathbf{Y} = \mathbf{y}_{h,k}|\mathcal{A} = \mathcal{A}_1) + \mathcal{P}(\mathcal{A} = \mathcal{A}_2)\mathcal{P}(\mathbf{Y} = \mathbf{y}_{h,k}|\mathcal{A} = \mathcal{A}_2) = c_{h,k}. \quad (1)$$

Optimal full masking is obtained by finding the values of $c_{h,k}$ that optimize the average cost of masking, $z' = \mathrm{E}[D(\mathbf{Y}) - D(\mathbf{X})]$. Here $D(\cdot)$ represents a "cost" measure associated to the flow features. Since the input flow are given, we can reduce the target function to $z = \mathrm{E}[D(\mathbf{Y})] = \sum_{h,k} c_{h,k} D_{h,k}$, where $D_{h,k} \equiv D(\mathbf{y}_{h,k})$ and $c_{h,k} = \mathcal{P}(\mathbf{Y} = \mathbf{y}_{h,k})$. Let $\mathbf{y}_{h,k} = [\kappa_{h,k}, \lambda_{h,k}, \tau_{h,k}]$. Then, if we aim at optimizing byte overhead, we have $D_{h,k} = |\lambda_{h,k}|$; if we are interested in minimizing time overhead (delay), we take $D_{h,k} = \max\{\tau_{h,k}\}$.

Summing up, given the costs $D_{h,k}$, the probabilities $c_{h,k}$ are found by solving the following linear optimization problem (*global optimization*):

$$z = \sum_{h=1}^{\omega_1} \sum_{k=1}^{\omega_2} c_{h,k} \cdot D_{h,k} \quad (2)$$

subject to constraints $0 \leq c_{h,k} \leq 1, \forall(h,k)$, $\sum_{k=1}^{\omega_2} c_{h,k} = p_{1,h}$ for $h = 1, \ldots, \omega_1$ and $\sum_{h=1}^{\omega_1} c_{h,k} = p_{2,k}$ for $k = 1, \ldots, \omega_2$, where $p_{1,h} = \mathcal{P}(\mathbf{X} = \mathbf{x}_h^{(1)})$, $h = 1, \ldots, \omega_1$ and $p_{2,k} = \mathcal{P}(\mathbf{X} = \mathbf{x}_k^{(2)})$, $k = 1, \ldots, \omega_2$.

We can relate the above optimization problem to the well-known Transportation Problem [14], with the only difference that in our case we have the quantities to "transport" expressed as fractions of the total amount. The problem at hand is easily solved by the Simplex Method. No efficient solution are known for $M > 3$.

The problem is now reduced to find a constructive way to define the flow $\mathbf{y}_{h,k}$ that minimizes $D_{h,k}$ given the two input flows $\mathbf{x}_h^{(1)}$ and $\mathbf{x}_k^{(2)}$ for all values of (h,k) (*local optimization*). We have set up an exhaustive search to solve the local optimization that turns out to be feasible in case the feature vector is in the order of a dozen components. We have considered special cases that reduce the computational complexity of the full ideal masking: i) the feature vector reduced to message lengths only, $\mathbf{X} = [\mathbf{L}]$; ii) the burst feature vector $\mathbf{X} = [\mathbf{B}, \mathbf{\Theta}]$[6]. This gives us sufficient material to use statistical full masking as a comparison benchmark to understand basic trade-offs (see discussion in Section 5).

We summarize the optimal full masking as follows:

1) take a flow φ belonging to $\mathcal{A}_1$, with features $\mathbf{x}_h^{(1)}$ (or: flow ψ belonging to $\mathcal{A}_2$ with features $\mathbf{x}_k^{(2)}$);
2) draw a random index in the set $[1, \omega_2]$ of value k with probability $c_{h,k} / \sum_{k=1}^{\omega_2} c_{h,k}$ (or: in the set $[1, \omega_1]$ of value h with probability $c_{h,k} / \sum_{h=1}^{\omega_1} c_{h,k}$);
3) transform the input flow φ (or: ψ) into the output masked flow ξ with features $\mathbf{y}_{h,k}$.

The probabilities $c_{h,k}$ are obtained by solving the global optimization stated in eq. (2). The output feature vector $\mathbf{y}_{h,k}$ is found by solving the local optimization for any couple of input flows $(\mathbf{x}_h^{(1)}, \mathbf{x}_k^{(2)})$.

6. Once the overall quantity of bytes and the start time of each burst have been modified for masking, the detailed message pattern can be defined according to a fixed scheme within each burst.

4 Practical and Partial Masking

A key limitation of ideal masking algorithms is the requirement that the entire flow be available to decide on masking, before sending any message out. This cannot work for transactional, interactive applications, where a message burst is produced by the application based on previously received bursts from the remote entity.

In this Section we aim at *practical* masking algorithm, i.e., applicable to a message/burst only based on features of previous messages/bursts. First, we consider statistical masking all messages of the input traffic (subsection 4.1), then we introduce an algorithm to mask input traffic only partially, so as to be able to trade-off between overhead and degree of masking (subsection 4.2). Neither case leads to full masking, since at least correlations among features of different bursts cannot be fully masked. In both cases, we define masking so as to minimize overhead. A simple deterministic full, practical algorithm is defined in subsection 4.3.

4.1 Practical masking of the whole flow

The key idea is to apply the optimized full masking of Section 3 *burst by burst*, so that the decision on the masking flow can be taken at each endpoint as the traffic flow runs. Hence, the feature sub-vector of message lengths and gap time for each burst separately. The complexity of the ideal, full masking *within* a burst is limited, since typical bursts comprise one or few messages. The overall flow masking is no more full, since correlations across bursts are not taken care of.

In general, padding, fragmentation and dummy messages can be used. If only padding is used, we define the Burst-by-Burst Padding Only (BbBPO) masking algorithm and minimum byte and time overhead is obtained as follows for each burst. Given feature sub-vectors $\mathbf{x}_b^{(i)} = [\mathbf{l}_b^{(i)}, \mathbf{t}_b^{(i)}]$ of application i ($i = 1, 2$), the shortest of the two sub-vectors is padded out with zeros. Then, the output feature sub-vector is $\mathbf{y}_b = [\lambda_b, \tau_b]$ with $\lambda_b = \max\{\mathbf{l}_b^{(1)}, \mathbf{l}_b^{(2)}\}$ and $\tau_b = \max\{\mathbf{t}_b^{(1)}, \mathbf{t}_b^{(2)}\}$. Figure 3 shows an example of message length masking with padding only for bursts of two applications. Dark shadowed portions of messages are padding bytes.

A different practical masking algorithm can be defined by resorting to *additive masking*. Let $V^{(i)}$ denote a random variable representing a generic feature of a flow of application i ($i = 1, \ldots, M$). We define $F_V^{(i)}(v) \equiv \mathcal{P}(V^{(i)} \leq v)$. With additive masking the masked feature $\tilde{V}$ is built as $\tilde{V} = V^{(i)} + U^{(i)}$, with $U^{(i)}$ a suitable *non-negative* random variable such that the pdf of $\tilde{V}$

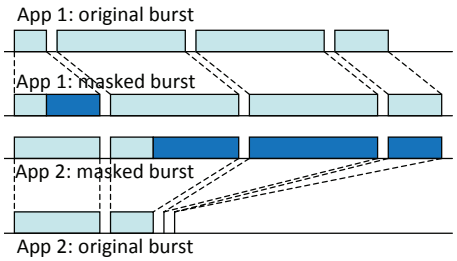

Fig. 3. Examples of message length masking of bursts of two applications.

is independent of i ($1 \leq i \leq M$). In [15] it is shown that minimum overhead additive masking is obtained by choosing $F_{\tilde{V}}(v) = \min\{F_V^{(1)}(v), \ldots, F_V^{(M)}(v)\}$. This choice minimizes $\mathrm{E}[\tilde{V}]$, hence $\mathrm{E}[U^{(i)}]$ for a given value of $\mathrm{E}[V^{(i)}]$, $1 \leq i \leq M$. Once $F_{\tilde{V}}(v)$ is computed, the pdf of the $U^{(i)}$'s can be calculated. Given a sample v of $V^{(i)}$, the masking quantity is sampled from the pdf of $U^{(i)}$, say u, and the masked feature value is $v+u$, with $u \geq 0$.

The Burst-by-Burst Statistical Additive (BbBSA) masking paradigm can be applied to a mix of M applications, whose flows are described by the feature vector $\tilde{\mathbf{X}} = [\mathbf{B}, \mathbf{\Theta}]$, introduced in Subsec. 2.1. As for burst number we let $\tilde{N} = N^{(i)} + U_N^{(i)}$, $i = 1, \ldots, M$, so a flows with $N^{(i)} = n$ is padded out with $U_N^{(i)} = u_N$ bursts, whose lengths and epochs are chosen according to the pdfs of the corresponding features. Let $\tilde{n} = n + u_N$ be the resulting number of bursts, with sizes $\mathbf{B}^{(i)} = [B_1^{(i)}, \ldots, B_{\tilde{n}}^{(i)}]$ and gap times $\mathbf{\Theta}^{(i)} = [\Theta_2^{(i)} - \Theta_1^{(i)}, \ldots, \Theta_{\tilde{n}}^{(i)} - \Theta_{\tilde{n}-1}^{(i)}]$ for the i-th application. Then, each burst size and gap time is masked as $\tilde{B}_j = B_j^{(i)} + U_{B,j}^{(i)}$ and $\tilde{\Theta}_j = \Theta^{(i)} + U_{\Theta,j}^{(i)}$, $j = 1, \ldots, \tilde{n}$. Once the sizes of bursts of the masked flow are determined, message length and timing can be defined according to a fixed, pre-defined scheme, independent of the input application flow.

Additive masking as presented above is a practical algorithm, since it can be applied by each endpoint separately, burst by burst, as they arrive at the endpoint. Knowledge of burst size and gap times statistics is required, but the pdfs of the number of bytes to add to bursts and their delays can be pre-computed, so that the masking device operations can be reduced to table lookup. The device data base comprises M tables for each feature to be masked (e.g. burst lengths). For a given feature V, the table of the i-th application has a number of rows and columns equal to the number of possible outcomes of the feature to be concealed. The entry (u, v) of the i-th table contains the conditional probability $\mathcal{P}(U_V^{(i)} = u | V^{(i)} = v)$, where U_V is the additive masking applied to V.

4.2 Partial masking

Complete flow masking can imply massive overhead and/or significant delay, as shown by quantitative examples in Section 5. Next we introduce a practical, partial masking algorithm that defines a trade-off between leaked information and amount of overhead and delay. We focus on message length masking for $M = 2$, as an example. Everything applies verbatim to other features as long as we use additive masking.

Let us focus on a specific message position within the flow sequence. Assume we know the pdfs $a_k = P(L^{(1)} = k)$ and $b_k = P(L^{(2)} = k)$, for $k = 1, \ldots, \ell$. Instead of $L^{(1)} + U_L^{(1)} \sim L^{(2)} + U_L^{(2)} \sim \tilde{L}$, partial masking requires that $c_k^{(i)} \equiv \mathcal{P}(L^{(i)} + U_L^{(i)} = k)$ $(i = 1, 2)$ satisfy the constraint:

$$\sum_{k=1}^{\ell} \min\{c_k^{(1)}, c_k^{(2)}\} \geq q \tag{3}$$

where q is a similarity measure, $q \in [0, 1]$. Complete flow masking is recovered for $q = 1$, that forces $c_k^{(1)} = c_k^{(2)}$, $\forall k$.

Our purpose is to find two pdfs $\{c_k^{(1)}\}_{1 \leq k \leq \ell}$ and $\{c_k^{(2)}\}_{1 \leq k \leq \ell}$ that satisfy eq. (3) and minimize the average overhead $\mathrm{E}[OH] \equiv Q_1 \mathrm{E}[U_L^{(1)}] + Q_2 \mathrm{E}[U_L^{(2)}]$. To solve this problem we have developed algorithm 1. In that algorithm, the function *PadAlg* outputs the pdf $\{\gamma_k\}_{1 \leq k \leq \ell}$ such that $\sum_{i=1}^{k} \gamma_i = \min\left\{\sum_{i=1}^{k} \alpha_i, \sum_{i=1}^{k} \beta_i\right\}$ $(1 \leq k \leq \ell)$, for two given pdfs $\{\alpha_k\}_{1 \leq k \leq \ell}$ and $\{\beta_k\}_{1 \leq k \leq \ell}$.

Algorithm 1 Partial_Masking(q,$\mathbf{c}^{(1)}$,$\mathbf{c}^{(2)}$)

1: $\alpha = 1 - q$
2: $\{a'_i\}_{i=1,\ldots,\ell} = \{a_i\}_{i=1,\ldots,\ell}$
3: $\{b'_i\}_{i=1,\ldots,\ell} = \{b_i\}_{i=1,\ldots,\ell}$
4: **for** $d \leftarrow \ell - 1$ to 0 **do**
5: **for** $i \leftarrow 1$ to ℓ **do**
6: **for** $j \leftarrow \{i + d, i - d\}$ **do**
7: **if** $1 \leq j \leq \ell$ **then**
8: $m = \min\{a'_i, b'_j, \alpha\}$
9: $[a'_i \ b'_j \ \alpha] = [a'_i \ b'_j \ \alpha] - m$
10: **end if**
11: **end for**
12: **end for**
13: **end for**
14: **assert** $\sum_{k=1}^{\ell} c'_k = \sum_{k=1}^{\ell} b'_k = \sum_{k=1}^{\ell} a'_k = 1 - \alpha$
15: $\{c'_i\} = PadAlg(\{a'_i\}, \{b'_i\})$
16: $\{c_i^{(1)}\} = \{a_i - a'_i + c'_i\}$
17: $\{c_i^{(2)}\} = \{b_i - b'_i + c'_i\}$

The algorithm takes as input the two pdfs, $\{a_k\}_{1 \leq k \leq \ell}$ and $\{b_k\}_{1 \leq k \leq \ell}$, of the message lengths of the two applications we want to mix. Each of them is split in two components according to $a_k = a'_k + a''_k$ and $b_k = b'_k + b''_k$ for $k = 1, \ldots, \ell$. The two components are such that $\sum_{k=1}^{\ell} a'_k = \sum_{k=1}^{\ell} b'_k = q$. Let a new message of application 1 with length k arrive. With probability a''_k/a_k it is passed onto the enciphering algorithm as it is (no masking), while with probability a'_k/a_k its length is modified according to additive masking so that the conditional output length pdf is $\{c'_k\}/q = PadAlg(\{a'_k\}/q, \{b'_k\}/q)$.

This partial masking yields an output pdf of message lengths equal to $\{c_k^{(1)}\}$ given in line 16 of algorithm

1, as we show in the following. If $u_j^{(1)}(h)$ denotes the conditional probability that padding is $h \geq 0$ bytes, given that the input message belongs to application 1 and has length j, then $c'_k/q = \sum_{j=1}^{k} u_j^{(1)}(k-j)a'_j/q$. Then, the probability of a masked message of length k at the output of the *partial* masking algorithm is $c_k^{(1)} = a_k(a''_k/a_k) + \sum_{j=1}^{k} a_j(a'_j/a_j)u_j^{(1)}(k-j) = a''_k + c'_k = a_k - a'_k + c'_k$, that is just line 16 of algorithm 1. Entirely analogous argument applies to application 2.

The choice of the density portions $\{a'_n\}$ and $\{b'_n\}$ in the rows 4-15 of the algorithm 1 is made so that the amount $|\sum_{k=1}^{\ell} a'_k \cdot k - \sum_{h=1}^{\ell} b'_h \cdot h|$ is minimized. In fact the algorithm gradually constructs them in an iterative way, by excluding the most distant density portions of the two original pdfs, until reaching a set of probabilities with weight q. The components $\{a'_n\}_{1\leq n\leq \ell}$ and $\{b'_n\}_{1\leq n\leq \ell}$ are used to feed *PadAlg*, whose output is the pdf $\{c'_n\}_{1\leq n\leq \ell}$. The resulting pdfs are given in lines 16 and 17 of algorithm 1. It can be checked that these are proper pdfs and that they satisfy eq. (3).

4.3 Fixed pattern masking

Full or practical flow masking algorithms in previous Sections are anyway statistical in nature. They imply knowledge of pdfs of the features to be masked. As a counterpart, they promise some kind of optimization of the introduced overhead and delay. A much simpler and supposedly far from optimal approach is *fixed pattern* masking. In general, it means that the input flow, whatever its originating application, be forced to be framed into a pre-defined pattern with features $\mathbf{y}_0 = [\mathbf{K}_0, \mathbf{L}_0, \mathbf{T}_0]$. Enforcement of these features is obtained practically as follows. Upon emission of a given burst, the sending application entity shapes the message(s) making up the burst according to the desired fixed pattern, by using fragmenting, padding and delaying. If at any given time, an output message must be issued according to the fixed masking pattern, while there are no input bytes to be sent, a dummy message is emitted.

Imposing such a fixed pattern to input flows generated by whatever application is certainly possible and it is guaranteed to raze any possible information useful to the classification adversary. Moreover, it can be applied message by message. So fixed pattern masking is a practical, full masking algorithm. The choice of the values of the fixed pattern features $\mathbf{y}_0$ influences the resulting overhead and delay. In general, the choice of $\mathbf{y}_0$ leading to minimal overhead is a multi-dimensional optimization problem, given the overall mix of traffic that it is expected at the masking device. Special cases of the fixed pattern, that simplify its implementation, are obtained by, e.g., setting a common value for features of all bursts in a same direction.

5 Numerical Results

Numerical examples are based on datasets built on IP traffic traces, described in the Appendix C. Since we refer to an IP networking environment, in this Section we replace the generic term "message" with packet. After introducing performance metrics in subsec. 5.1, results are grouped according to the aspect we investigate. In subsec. 5.2 we assume a real time adversary, whose classifier is based on the first few messages of each flow. The aim is to understand the theoretical lower bound of overhead under the full masking constraint and check possible benefits of fragmentation as opposed to padding-only masking. This lower bound is compared with overhead yielded by practical, partial masking, as the trade-off parameter q is varied. Then, in subsec. 5.3 we investigate practical masking.

5.1 Performance Metrics

Efficiency can be measured by the average amount of overhead introduced by masking:

$$\mathrm{E}[OH] = \frac{\sum_{i=1}^{M}\sum_{h=1}^{\omega_i} p_{i,h}\left[|\phi(\mathbf{x}_h^{(i)};i)| - |\mathbf{x}_h^{(i)}|\right]}{\sum_{i=1}^{M}\sum_{h=1}^{\omega_i} p_{i,h}|\phi(\mathbf{x}_h^{(i)};i)|} \quad (4)$$

where $|\mathbf{u}|$ equals the sum of the components of $\mathbf{u}$, $p_{i,h} = \mathcal{P}(\mathbf{X} = \mathbf{x}_h^{(i)}|\mathcal{A} = i)$ can be estimated from the collected flow for each application and applications have been assumed equiprobable ($P_i = 1/M$). By taking $\mathbf{x}_h^{(i)}$ as the packet lengths we get the *byte overhead*; when times are used, we get the *time overhead*.

Information leakage against flow classification is measured as follows. According to Section 2, the adversary defines a classifier TA yielding the application deemed to have generated the observed (masked) flow. Let $\eta_{i,j} = \mathcal{P}(TA = j|\mathcal{A} = i)$ for $i, j = 1, \ldots, M$ and let $\mathbf{H} = [\eta_{i,j}]$ be the flow classifier confusion matrix. Diagonal elements represent success probabilities of the classifier, while off-diagonal elements are mis-classification probabilities. A flow classifier corresponds to a discrete information channel that maps input flows ($\mathcal{A}$ "symbols") into classification decisions (TA "symbols") and is therefore described by the matrix $\mathbf{H}$. By assuming the a priori pdf for application be uniform, i.e. $Q_j = 1/M, j = 1, \ldots, M$, the average mutual information of this "information channel" can be computed as:

$$I(\mathcal{A}; TA) = \log_2 M + \frac{1}{M}\sum_{i=1}^{M}\sum_{j=1}^{M} \eta_{i,j}\log_2\left(\frac{\eta_{i,j}}{\eta_j}\right) \quad (5)$$

where $\eta_j = \sum_{r=1}^{M} \eta_{r,j}$. In the following we consider the normalized mean mutual information $\hat{I} = \hat{I}(\mathcal{A}; TA) = I(\mathcal{A}; TA)/\log_2 M$, that is the fraction of the mean mutual information of an ideal classifier that a real classifier attains. With this definition, $\hat{I}(\mathcal{A}; TA) = 0$ in case of full masking, while $\hat{I}(\mathcal{A}; TA) = 1$ for the ideal classifier. We define also the probability P_{succ} that the adversary correctly classifies the flow application, that is

$$P_{succ} = \frac{1}{M}\sum_{j=1}^{M} \eta_{j,j} \quad (6)$$

Computation of both last metrics requires a matrix $\mathbf{H}$, hence an instance of flow classifier which exploits the masked flow features. We use four classification algorithm (Naive Bayes, Logistic Model Trees, Random Tree and K-means) offered in the WEKA implementation, which is a machine learning workbench distributed under the GNU General Public License [16]. Each machine learning was trained by feeding it with *masked* flows, so that it can learn to recognize any information that possibly leaks from the masked flow features, that is to say *after* the application of the masking algorithm.

5.2 Masking against a real time adversary

In this Section we restrict ourselves to an adversary exploiting lengths of the first m packets of each flow, hence $\mathbf{X} = [L_1, \ldots, L_m]$. The results are obtained with $m = 5$, which is consistent with a real time flow classification target. Albeit restricted, this scenario is enough to highlight interesting issues on full and partial masking.

In Table 1 we compare the average byte overhead required by the different masking algorithms. The numbers represent the fraction of the output bytes due to masking overhead, evaluated according to eq. (4).

Applications Pair	Optimum full masking (FAP)	Optimum full masking (PO)	Fixed length masking
HTTP - SSH	0.3191	0.3605	0.4591
HTTP - FTP-c	0.4088	0.4126	0.5328
HTTP - POP3	0.4351	0.4392	0.5629
HTTP - VoIP	0.3864	0.4126	0.5406
HTTP - emule	0.4111	0.4219	0.5477
SSH - FTP-c	0.3008	0.3162	0.5378
SSH - POP3	0.3495	0.3715	0.5700
SSH - VoIP	0.2353	0.3546	0.5049
SSH - emule	0.3141	0.3673	0.5070
FTP-c - POP3	0.2094	0.2336	0.5302
FTP-c - VoIP	0.2303	0.2752	0.4891
FTP-c - emule	0.2035	0.2392	0.4486
POP3 - VoIP	0.2477	0.2477	0.4892
POP3 - emule	0.1753	0.1803	0.4160
VoIP - emule	0.1878	0.1902	0.3520
HoS - SFTP	0.2090	0.2248	0.5205

TABLE 1
$\mathrm{E}[OH]$ values with different packet length masking algorithms for various couples of applications (FAP=Fragmentation And Padding; PO=Padding Only).

Optimum full masking as defined in Section 3 comes in two different ways. In the first case, we apply both fragmentation and padding when constructing optimum output flow $\mathbf{y}_{h,k}$ paired with input flows $\mathbf{x}_h^{(1)}$ and $\mathbf{x}_k^{(2)}$ (local optimization). In the second case, only padding and dummy packets are allowed. All considered approaches lead to full masking of traffic flows as regards the packet length information. The metric to compare different approaches is therefore the average amount of overhead. We include also fixed packet length masking, that is the special case of fixed pattern masking for the case at hand. In Table 1 unrestricted optimum full masking is given in the first column, padding only optimum full masking in the second column, while fixed length masking is reported in the third column.

The length L_0 used by the fixed length masking algorithm in Table 1 is chosen so as to minimize the average overhead for each masked traffic mix. Figure 4a shows the average overhead as a function of L_0 for five different couples of applications that are mixed (HTTP-FTP-c, HTTP-emule, VoIP-FTP-c, SSH-POP3, POP3-emule). There is an optimum choice of L_0, since for very small values the overhead due to fragmentation (additional IP packet headers) is dominant, whereas for large values of L_0 padding is dominant. In any case, the average amount of overhead is quite large, never less than about 0.5, i.e. about 50% of all bytes sent out by the traffic flow masking device are overhead bytes.

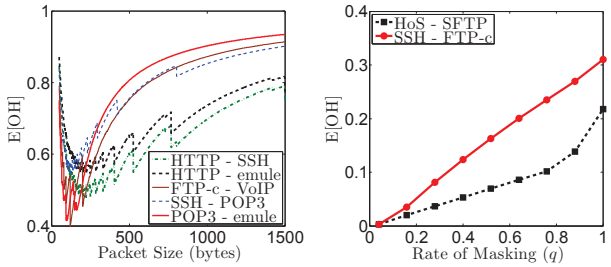

Fig. 4. Overhead OH metric as defined in eq. (4): (a) OH vs. L_0 for five different couples of applications; (b) OH vs. the masking rate q for two scenarios.

Results in Table 1 point out that the amount of overhead can vary significantly, depending on the applications mix.Fixed length masking roughly doubles the required overhead with respect to optimal masking, thus showing that the optimization of full masking could bring significant efficiency gains. An interesting outcome of the results in Table 1 is that optimum full masking, constrained to use padding only, does not cause a significant increase in overhead as compared to the unrestricted optimum full masking, where we can fragment packets. In a lot of cases, adding fragmentation improves overhead marginally. This is a strong argument advocating the use of padding only, although this is not intuitive at first. Main reason is that full masking requires not only masking the length of each packets of the flow but also the amount of bytes of the entire flow.

As for partial masking, numerical results are obtained in the same real time adversary context as above, by applying the partial masking algorithm packet-by-packet. The graphs in Figs. 4b - 5b show the obtained results. Figures 7a and 5a refer to Scenario 1 (SSH and FTP-c flows), while Figs. 7b and 5b refer to Scenario 2 (HoS and SFTP flows, into SSH tunnels). We plot as a function of the masking rate q the average overhead $\mathrm{E}[OH]$ in Figure 4b and P_{succ} in Figures 7a and 7b. Figures 5a and 5b plot the trade-off between the information leakage measured

by $\hat{I}(\mathcal{A};TA)$ and the average overhead. Four different classification algorithm have been considered.

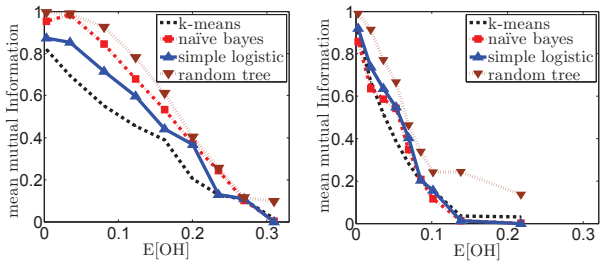

Fig. 5. Trade-off between normalized mutual information and overhead for two scenarios: 5a SSH and FTP-c; 5b HTTP over SSH and SFTP.

The interesting indication of these results is that the amount of overhead cannot be reduced significantly with respect to full masking, if strict leakage requirements are set (q close to 1), yet substantial reduction of overhead with respect to full masking can be attained, if a success probability P_{succ} of about 0.7-0.6 is acceptable (a trivial classifier can attain $P_{succ} = 0.5$). In that case, we can fix $q = 0.8$, that leads to $\mathrm{E}[OH]$ about halving with respect to full masking (scenario 2). A smaller gain is obtained in case of scenario 1. As for the trade-off between $\hat{I}(\mathcal{A};TA)$ and $\mathrm{E}[OH]$, the smaller the average mutual information leaked to the adversary, the larger the overhead required. The trade-off appears to be much more favorable in case of SSH tunneled application flows (scenario 2). Other tests with different application protocols (e.g. VoIP) have been carried out, yielding similar behavior.

5.3 Practical masking on extended flows

In this section we report about performance of practical algorithms described in subsection 4.1. First, we compare optimum full masking, Burst-by-Burst Statistical Additive (BbBSA) masking, Burst-by-Burst Padding Only (BbBPO) practical masking, and fixed pattern masking. In the last case, the same burst size is chosen and independently optimized for each flow direction. We have considered padding only in case of BbBPO since it gives rise to much simpler implementation than in case fragmentation is used and results in subsection 5.2 point out that the overhead penalty is marginal.

Results for the byte overhead, calculated according eq. (4), are shown in Table 2 in case of compact feature vector $[\mathbf{B}, \mathbf{\Theta}]$ (burst sizes and epochs) and by considering the first 12 bursts for each flow. Remarkably, BbBSA masking incurs a minor penalty as for overhead with respect to optimized full masking. It can be considered as a sort of lower bound of the practical algorithms. Notice that it is possible that $\mathrm{E}[OH]$ for BbBSA be less than for full masking, since BbBSA is a practical masking, hence it is not *full* masking, i.e., it does not fulfill the requirement of removing *any* leakage. Much more overhead is demanded by BbBPO masking and especially by the fixed pattern masking, where a same size of all bursts has been set, with the best choice for each application mix.

Application Pair	Opt. full masking	BbBSA Masking	BbBPO Masking	Fixed Burst Size
HTTP - SSH	0.3663	0.3704	0.4428	0.5059
HTTP - FTP-c	0.4104	0.4105	0.4358	0.5844
HTTP - POP3	0.4233	0.4324	0.4479	0.5951
HTTP - VoIP	0.4080	0.3933	0.4255	0.5571
HTTP - emule	0.4302	0.4157	0.4347	0.5917
SSH - FTP-c	0.3022	0.3564	0.3799	0.6082
SSH - POP3	0.3489	0.3808	0.3890	0.6285
SSH - VoIP	0.2936	0.3100	0.3824	0.5657
SSH - emule	0.3384	0.3476	0.4012	0.6237
FTP-c - POP3	0.1869	0.2439	0.2710	0.5263
FTP-c - VoIP	0.2231	0.2384	0.3174	0.4872
FTP-c - emule	0.2310	0.2030	0.3162	0.5168
POP3 - VoIP	0.2700	0.2793	0.3421	0.4987
POP3 - emule	0.2276	0.2208	0.2927	0.5355
VoIP - emule	0.2257	0.2144	0.3212	0.4372
HoS - SFTP	0.2752	0.2823	0.3229	0.5187

TABLE 2
$\mathrm{E}[OH]$ values with practical and fixed burst size masking (BbBSA = Burst-by-Burst Statistical Additive; BbBPO = Burst-by-Burst Padding Only).

Notice that only optimum full masking and fixed burst masking remove any leakage of traffic flow features. Whereas the former is only a theoretical benchmark and cannot be realized, the latter is quite easily implementable and does not even require statistical data. The price to pay for this simplicity is about doubling the overhead with respect to the optimum and also with respect to the BbBSA masking. The appeal of the statistical *practical* masking whose results are reported in the two central columns of Table 2 depends on their capability of reducing leakage enough to make classification essentially fail. Since correlations are not taken care of when masking is decided burst by burst, we expect some information is leaked by statistical practical algorithms. Figure 6 plots the mutual information $\hat{I}(\mathcal{A};TA)$ leaked by BbBPO masking for some application mixes as a function of the number of bursts N_B used by the adversary for classification. The classification algorithm is Random Tree in the WEKA implementation.

It is apparent that the first few bursts do not yield a significant amount of information that can be exploited by the adversary. As the number of inspected bursts grows in the order of ten or more, a major leakage is found. Correspondingly probabilities of successful classification range between 0.87 and 0.92 for $N_B \gg 10$.

These findings are confirmed by results in Table 3, that refer to BbBPO masking applied to flows by considering up to 50 bursts. The first column shows the values of the average overhead; time overhead in the second column is calculated in the same way by replacing packet lengths with inter-packets gap times. On the third column, the average packet delay introduced by the masking device is displayed. The four columns of the Table 4 show

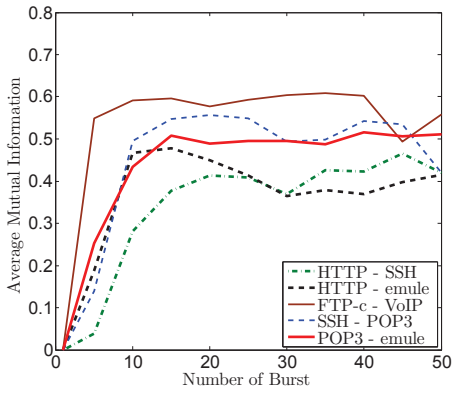

Fig. 6. Average mutual information leaked by BbBPO masking as a function of the number of burst N_B used in the classification for various application mixes.

the values of P_{succ} obtained with the considered flow classification algorithms, implemented in WEKA [16].

Application Pair	Byte Overhead	Time Overhead	Average Packet Delay (ms)
HTTP - SSH	0.4363	0.4702	63.310
HTTP - FTP-c	0.4713	0.4239	32.756
HTTP - POP3	0.4408	0.4515	23.879
HTTP - VoIP	0.4523	0.4258	27.313
HTTP - emule	0.4554	0.4674	66.936
SSH - FTP-c	0.4210	0.4750	80.345
SSH - POP3	0.4274	0.4814	68.027
SSH - VoIP	0.4198	0.4692	72.255
SSH - emule	0.4410	0.4632	70.402
FTP-c - POP3	0.3992	0.4314	35.263
FTP-c - VoIP	0.3983	0.4156	38.617
FTP-c - emule	0.4233	0.4181	54.700
POP3 - VoIP	0.4322	0.4529	41.988
POP3 - emule	0.4261	0.4757	82.109
VoIP - emule	0.4052	0.4329	33.656
HoS - SFTP	0.3856	0.4728	50.133

TABLE 3
Average byte and time overheads and average delay introduced by BbBPO masking algorithm.

The results point out that, when we mask all flow features, burst structure, packet lengths and timing information, we get a significant worsening of the traffic load on the network compared with the optimal masking for packet lengths only. The overhead for most cases is more than 40% of the whole output traffic, and in one case it peaks at 47%. Average delay ranges from few tens of ms up to about 80 ms; this is not an issue for most applications, but it can become critical for VoIP.

In spite of the massive overhead introduced, the key result is that there is still enough leakage for the adversary to be able to classify flow with rather high accuracy, even though a suitable classifier must be chosen. Since intra-burst masking is full (and even optimal), the leakage can be ascribed to correlations among features of different bursts, that are not removed by additive masking. The comparison of the practical masking in

Application Pair	Naïve Bayes	Logistic	Random Forest	K-means
HTTP - SSH	0.6576	0.5303	0.8611	0.6905
HTTP - FTP-c	0.6863	0.5235	0.8779	0.7225
HTTP - POP3	0.5472	0.5094	0.8887	0.7080
HTTP - VoIP	0.5245	0.5112	0.8520	0.6324
HTTP - emule	0.5766	0.6034	0.8738	0.5969
SSH - FTP-c	0.5350	0.5900	0.9290	0.5825
SSH - POP3	0.5735	0.7725	0.9297	0.7180
SSH - VoIP	0.6019	0.5128	0.9102	0.6854
SSH - emule	0.5650	0.6233	0.8672	0.6491
FTP-c - POP3	0.6715	0.6445	0.8914	0.6595
FTP-c - VoIP	0.6223	0.6355	0.8922	0.5922
FTP-c - emule	0.5328	0.5916	0.8790	0.6218
POP3 - VoIP	0.5560	0.6326	0.9022	0.6976
POP3 - emule	0.5158	0.5481	0.8937	0.6557
VoIP - emule	0.5497	0.5097	0.8582	0.6855
HoS - SFTP	0.5490	0.6606	0.8649	0.5712

TABLE 4
P_{succ} for the four classification algorithms considered.

Figures 7a and 7b with $q = 1$ with results in Table 4 show that additive masking can be effective if the adversary is limited to observation of the features of a small number of flow packets, as required to attain real time classification, but it fails if the adversary can take the time to observe the flow at length.

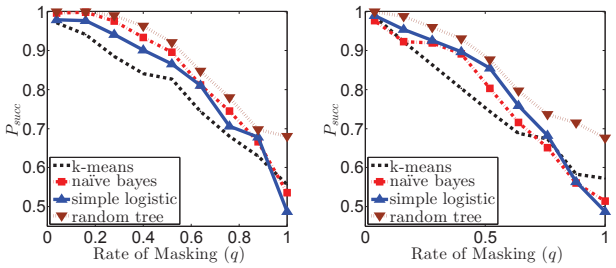

Fig. 7. Probability of a successful classification as a function of the masking rate q for two scenarios: 7a SSH and FTP-c; 7b HTTP over SSH and SFTP.

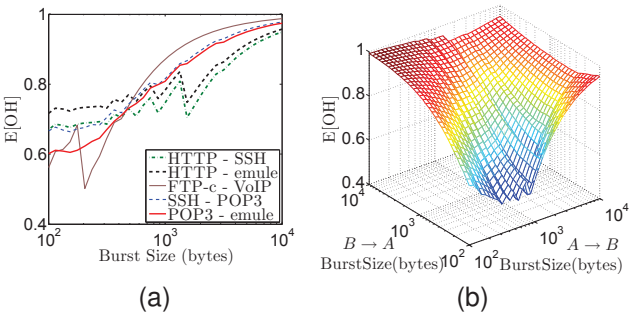

Fig. 8. Fixed pattern masking: (a) average overhead of various traffic mixes as a function of the fixed burst size; (b) average overhead for the mix SSH-VoIP as a function of the burst sizes in the two directions.

Figure 8a shows the average overhead as a function of burst size (equal for all bursts, in both directions)

for various application mixes. Figure 8b plot $\mathrm{E}[OH]$ as a function of the two independent values of burst sizes that can be set in the two opposite directions. The graphs highlight that there is a big margin of efficiency gain by properly selecting burst sizes, especially if two dimensional optimization is used, by allowing different value of the burst sizes in the two directions. Even after optimization, overhead values are quite high, but interestingly they are not terribly larger than those of locally optimized additive masking. On the other hand, with fixed pattern masking leakage is stopped completely and implementation complexity is definitely lower than in the much more sophisticated additive masking, that uses statistical information to optimize overhead.

6 Discussion and concluding remarks

The main objective of this paper is to assess the performance in terms of overhead and complexity of a traffic flow masking device against a classification adversary. We characterize the optimal masking algorithm under the constraint of perfect masking. To overcome implementation difficulties arising in case conversational application are masked, we have relaxed to practical masking algorithms, working burst by burst (or message by message) or even masking only a fraction of the overall flow. Practical masking lets correlation over features of different bursts leak, yet it offers practical realizability of the masking device and reduced overhead, though leakage might be critical in case of adversaries that are not time constrained, as shown by the numerical results.

Numerical results point out that a basic distinction must be done between *real time* adversary and *off line* adversary, observing features of an extended segment of the flow. In the former case, even if some information useful to the adversary leaks when relaxing to practical masking from full masking, still it appears that classification is essentially impaired (success probabilities of best algorithms we could find are below 0.6 in case of two applications). This is tied to the limited number of features used by the adversary, compelled by the need to decide on the class of the observed masked flow in real time. Things turn out to be completely different if the adversary can take time to classify the flow and observe its features over several bursts, in the order of tens. In that case practical masking, though canceling any information useful for classification inside each burst, cannot remove entirely correlations across features of different bursts[7]. So, statistical, practical masking, even though it minimized overhead burst by burst, still introduces a considerable amount of overhead while failing to protect privacy against an off line adversary. Another question concerns the amount of data requested in advance by the masking device, in order to carry out an accurate process of obfuscation. In fact, it requires knowledge of estimates of the probability density functions for all used features of the considered M applications. As the duration of a flow grows up, the amount of data required becomes very high and quite hard to estimate reliably.

Our numerical investigation gives merit to simpler masking approaches, that give up to global or local optimization leveraging on statistical masking and resort instead to rigid, fixed pattern masking. While increasing overhead with respect to statistical masking, as expected since no optimization is attempted with fixed masking, yet that approach removes any information that could be exploited by the classification adversary, preserves implementation simplicity, and overhead price is not terribly greater than that entailed by optimized solutions (within a factor of 2 from the full, ideal optimized masking), at least in cases we have experimented.

A different and potentially promising approach we are currently pursuing is aggregate masking, i.e., application of feature masking at IP flow level to the aggregate flow carried over an IP tunnel, e.g., an IPSec connection.

7. This is similar to mono-alphabetic ciphering, when attacked by exploiting language redundancy, hence correlation.

References

[1] H. Kim, K. C. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, "Internet traffic classification demystified: myths, caveats, and the best practices," in *CoNEXT*, 2008, p. 11.

[2] T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *Communications Surveys Tutorials, IEEE*, vol. 10, no. 4, pp. 56 –76, 2008.

[3] M. Mellia, A. Pescapè, and L. Salgarelli, "Traffic classification and its applications to modern networks," *Computer Networks*, vol. 53, no. 6, pp. 759–760, 2009.

[4] A. C. Callado, C. A. Kamienski, G. Szabo, B. P. Gero, J. Kelner, S. F. L. Fernandes, and D. F. H. Sadok, "A Survey on Internet Traffic Identification," *IEEE Communications Surveys and Tutorials*, vol. 11, no. 3, pp. 37–52, 2009.

[5] A. Dainotti, A. Pescape, and K. Claffy, "Issues and future directions in traffic classification," *Network, IEEE*, vol. 26, no. 1, pp. 35 –40, january-february 2012.

[6] S. Chen, R. Wang, X. F. Wang, and K. Zhang, "Side-Channel Leaks in Web Applications: A Reality Today, a Challenge Tomorrow," in *IEEE Symposium on Security and Privacy*, 2010.

[7] C. V. Wright, L. Ballard, F. Monrose, and G. M. Masson, "Language Identification of Encrypted VoIP Traffic: Alejandra y Roberto or Alice and Bob?" in *In Proceedings of the 16th Annual USENIX Security Symposium*, Boston, MA, August 2007.

[8] A. M. White, A. R. Matthews *et al.*, "Phonotactic Reconstruction of Encrypted VoIP Conversations: Hookt on Fon-iks," in *IEEE Symposium on Security and Privacy*, 2011.

[9] K. P. Dyer, S. E. Coull *et al.*, "Peek-a-Boo, I Still See You: Why Efficient Traffic Analysis Countermeasures Fail," in *IEEE Symposium on Security and Privacy*, 2012.

[10] C. V. Wright, S. E. Coull, and F. Monrose, "Traffic Morphing: An Efficient Defense Against Statistical Traffic Analysis," in *NDSS*, 2009.

[11] S. Yu, T. Thapngam, H. I. Tse, and J. Wang, "Anonymous web browsing through predicted pages," in *IEEE Globecom Workshops*, 2010, pp. 1581–1585.

[12] S. Yu, G. Zhao, W. Dou, and S. James, "Predicted Packet Padding for Anonymous Web Browsing Against Traffic Analysis Attacks," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 4, pp. 1381–1393, 2012.

[13] X. Luo, P. Zhou, E. W. W. Chan *et al.*, "HTTPOS: Sealing Information Leaks with Browser-side Obfuscation of Encrypted Flows," in *NDSS*, 2011.

[14] F. Hitchcock, "The distribution of a product from several sources to numerous localities," *J. Math. Phys.*, vol. 20, pp. 224–230, 1941.

[15] A. Iacovazzi and A. Baiocchi, "Optimum Packet Length Masking," in *International Teletraffic Congress*, 2010.

[16] "Weka 3: Data Mining Software in Java," http://www.cs.waikato.ac.nz/ml/weka/.

Alfonso Iacovazzi received his MSc Degree in Telecommunication Engineering from *Sapienza* University of Rome, Italy, in 2008, and and his PhD degree in Information and Communications Engineering from the same University, in 2013. Since March 2013 he is a Postdoctoral Research Fellow at DIET Dept, Rome, Italy. He is part of the Networking Group. His main research interests are about communications security and privacy, traffic analysis and monitoring, traffic anonymization, cryptography (mathematical aspects and applications)

Andrea baiocchi received his Laurea degree in Electronics Engineering in 1987 and his PhD degree in Information and Communications Engineering in 1992, both from the University of Roma “La Sapienza”. Since January 2005 he is a Full Professor in the Department of Information Engineering, Electronics and Telecommunications of the University of Roma “Sapienza”. The main scientific contributions of Andrea Baiocchi are on traffic modeling and traffic control in ATM and TCP/IP networks, queueing theory, radio resource management, MAC protocols, traffic analysis for flow classification. His current research interests focus on wireless access protocols, specifically for VANET applications, and on protection of traffic privacy against classification adversaries. Andrea’s research activities have been carried out also in the framework of many national (CNR, MIUR) and international (European Union, ESA) projects, also taking coordination and responsibility roles. Andrea has published more than a hundred papers on international journals and conference proceedings, he has participated in the Technical Program Committees of more than forty international conferences; he also served in the editorial board of the telecommunications technical journal published by Telecom Italia for ten years.