Ideal Packet Length Masking against Traffic Classification

Alfonso Iacovazzi, Andrea Baiocchi
DIET Department - University of Roma "Sapienza", Via Eudossiana 18, 00184 Roma, Italy
e-mail: {iacovazzi,baiocchi}@infocom.uniroma1.it

I. INTRODUCTION

Traffic flow classification has been attracting an increasing interest. Among the exploited flow features, a key role is played by the sequence of packet lengths. We aim at understanding if and how complex it is to obfuscate this information, referred to as packet length masking. Masking can be obtained by means of padding and fragmenting. We define formally what the ideal target of masking is, and then define the masking problem as a statistical optimization problem, aiming at minimizing the required overhead. We find the optimal solution of the masking problem in case of two application types. An explicit efficient algorithm is given to compute the optimal masking sequence. Numerical results are provided, based on measured traffic traces of HTTP, POP3, SMTP, FTP-control and VoIP traffic. One of the most striking findings is that fragmenting does not achieve any significantly better performance than simple padding does as far as overhead-obfuscation trade-off is concerned.

II. REFERENCE SCENARIO

We consider a packet network where we can identify two end points (hosts) running a given application protocol within a secure communication channel. We assume in the following that the information stream between the two hosts is ciphered and authenticated (message authentication), at least in the network section that is deemed to be insecure.

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. In the following we define a flow as the ordered sequence of packets observed at a given network section and carrying the same value of a number of selector fields. In IP networking context, typical fields that identify an application layer flow are: source IP address (SA), destination IP address (DA), source port (SP), destination port (DP) and protocol type (PT). A direction is associated to a flow, going from the initiating host (the one that sends the first packet) to the responding host. In case TCP is chosen to support the application level protocol, a flow coincides with packets belonging to a TCP connection. In case of UDP, we need to add a time constraint, i.e. two consecutive packets carrying the same value of a number of selector fields.

To construct the transformation \( \phi \) we define ordered couples \( (X_i, Y_i) \), with \( X_i \in \Omega_i \) and \( Y_i \in \Omega_i \), for \( i = 1, \ldots, n \), and knows any algorithm the sender may have used to pad, fragment and encipher the packets of the original flow. The aim of the adversary is to guess the application the original flow belongs to. This is summarized by an algorithm named \( \text{T}A(Y) \).

III. MIXING TWO APPLICATIONS: OPTIMAL SOLUTION

Let us consider two application, \( A_0 \) and \( A_1 \) and flows made up of \( m \) packets from each application. Assume the sample space (with non null probability) of application \( A_i \) be \( \Omega_i \subset [1, \ell]^m \), with \( \omega_i = |\Omega_i| \), for \( i = 0, 1 \).

To construct the transformation \( \phi(\cdot) \) we define ordered couples \( (X_i^{(0)}, X_i^{(1)}) \), with \( X_i^{(0)} \in \Omega_0 \) and \( X_i^{(1)} \in \Omega_1 \), for \( h = 1, \ldots, \omega_0 \) and \( k = 1, \ldots, \omega_1 \). For each couple \((X_i^{(0)}, X_i^{(1)}) \) we find the optimum flow pattern \( Y_{h,k} \) made up of \( n_{h,k} \) packet lengths that the flows in the couple can be mapped to by means of padding and fragmentation. Optimum here refers to minimization...
of the overhead required to convert each of the two flows of the couple into the masked flow \( y_{h,k} \). If we start out from flow \( x_h^{(0)} \), the overhead is \( |y_{h,k}| - |x_h^{(0)}| + (n_{h,k} - m - 1)H \); if instead the input flow is \( x_k^{(1)} \), the resulting overhead is \( |y_{h,k}| - |x_k^{(1)}| + (n_{h,k} - m - 1)H \), where \( |\cdot| \) is the sum of entries of \( \alpha \). Since the packet lengths of the input flows are given, we can take as optimization target the function \( E_{h,k} = |y_{h,k}| + n_{h,k}H \), that is simply the overall length of the output masked flow \( y_{h,k} \).

Perfect masking is obtained by requiring that coupling probabilities are chosen so that

\[
P(Y = y_{h,k} | X = x_h^{(0)}) \cdot p_0(x_h^{(0)}) = P(Y = y_{h,k} | X = x_k^{(1)}) \cdot p_1(x_k^{(1)}),
\]

(1)

This requirement guarantees that the adversary has no clue to what application has emitted the original flow from the observation of the masked flow \( y_{h,k} \). Formally, thanks to eq. (1), the a posteriori probability that \( x_h^{(0)} \) be the original flow given that \( y_{h,k} \) is observed is:

\[
P(X = x_h^{(0)} | Y = y_{h,k}) = \frac{P(Y = y_{h,k} | X = x_h^{(0)})p_0(x_h^{(0)})}{P(Y = y_{h,k})} = \frac{P(Y = y_{h,k} | X = x_h^{(0)})p_0(x_h^{(0)})}{P(Y = y_{h,k} | X = x_h^{(0)}) + P(Y = y_{h,k} | X = x_k^{(1)})p_1(x_k^{(1)})} = \frac{1}{2}
\]

where the last equalities are due to the construction of the pairings between flow of application 0 and 1.

Then the problem is to determine \( c_{h,k} \) so as to minimize the output flow overhead, or, what is the same, the output flow length:

\[
z = \sum_{h=1}^{\omega_0} \sum_{k=1}^{\omega_1} c_{h,k} \cdot E_{h,k}
\]

(2)

subject to constraints:

\[
0 \leq c_{h,k} \leq \min \left\{ p_0(x_h^{(0)}), p_1(x_k^{(1)}) \right\}, \quad \forall (h,k)
\]

\[
\sum_{k=1}^{\omega} c_{h,k} = p_0(x_h^{(0)}), \quad h = 1, \ldots, \omega_0
\]

\[
\sum_{h=1}^{\omega} c_{h,k} = p_1(x_k^{(1)}), \quad k = 1, \ldots, \omega_1
\]

We can relate the above optimization problem to the well-known Transportation Problem [18], 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 solved by Hungarian algorithm [19].

The quantities \( E_{h,k} \) represent the minimum amount of overhead required to mix the \( h \)-th flow of application \( A_0 \) with the \( k \)-th one of application \( A_1 \). If we consider just padding and then no fragmentation, \( E_{h,k} \) can be easily computed through the relation:

\[
E_{h,k} = \sum_{r=1}^{m} |x_h^{(0)}(r) - x_k^{(1)}(r)|
\]

If fragmentation is used as well, computing the weights \( E_{h,k} \) is more complex, but it can be done simply by exhaustive search among all flow patterns \( y_{h,k} \) derived via padding and fragmentation from \( x_h^{(0)} \) and \( x_k^{(1)} \), provided \( m \) is small, e.g. less than 10.

We can summarize the ideal masking algorithm for two applications in the following steps:

1) take as input a flow \( \varphi_{h^*} \in A_0 \) (or: \( \psi_{k^*} \in A_1 \));
2) draw a random index in the set \([1, \omega_1] \) of value \( k^* \) with probability \( \frac{c_{h^*,k^*}}{p_0(\varphi_{h^*})} \) (or: in the set \([1, \omega_0] \) of value \( h^* \) with probability \( \frac{c_{h^*,k^*}}{p_1(\psi_{k^*})} \));
3) transform the input flow \( \varphi_{h^*} \) (or: \( \psi_{k^*} \)) into the output masked flow \( y_{h^*,k^*} \).

We have implemented this algorithm by filling the table of output flows \( y_{h,k} \) and solving the above stated optimization problem to find the values of \( c_{h,k} \). A discussion of the significance and limits of this algorithm is given in Section VI.
IV. Traffic Data Set Description

Performance of the algorithms developed in this work is analyzed by using the same datasets collected and commented in [5]. The traffic is generated by considering clients and servers placed in University lab and industrial lab LANs, as well as private domestic locations. All connections are through public Internet. Servers and clients use different operating systems, so as to collect a representative traffic sample. Collected flows belong to different application protocols: HTTP, SSH, POP3, FTP (control session, FTP-c), VoIP. For each of the considered application protocols 2000 flow are collected. Flows have been obtained by considering TCP connection, removing SYN and SYNACK initial segments and collecting the ensuing m packets. For each flow we define a feature vector, containing the ordered sequence of feature tuples taken from flow packets. For this work, the feature vector consists of only m packet lengths. Packet lengths for the used traffic dataset range from 52 bytes up to 1500 bytes (sizes refer to IP packets). Payloads range from 1 byte up to $\ell = 1448$ bytes.

V. Numerical Results

We have compared the optimum masking algorithm for two application given in Section III in two different ways. In the first case, we apply both fragmentation and padding when constructing optimum output flow $y_{h,k}$ paired with input flows $x_{h}^{(0)}$ and $x_{k}^{(1)}$. In the second case, only padding is allowed, so that $y_{h,k}$ is made up m packets, whose lengths are just $y_{h,k}(r) = \max\{x_{h}^{(0)}(r), x_{k}^{(1)}(r)\}$, $r = 1, \ldots, m$. We consider also a more practical approach to remove all information leaked by the packet length values, namely fixed length masking. This amounts simply in fragmenting and padding all incoming packets into packets of fixed length $S$. 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 each of them need.

We have considered various couples of applications drawn from the HTTP, SSH, POP3, FTP-c and VoIP. In the table I we compare the average overhead required by the different masking algorithms. Fixed-size packets is reported in the third column; padding only optimum masking is given in the second column; joint padding and fragmentation optimum masking is given in the first column.

<table>
<thead>
<tr>
<th>Applications Pair</th>
<th>Fragmentation AND Padding</th>
<th>Padding Only</th>
<th>Fragmentation with fixed length</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP - SSH</td>
<td>0.3191</td>
<td>0.3605</td>
<td>0.4591</td>
</tr>
<tr>
<td>HTTP - FTP-c</td>
<td>0.4088</td>
<td>0.4126</td>
<td>0.5328</td>
</tr>
<tr>
<td>HTTP - POP3</td>
<td>0.4351</td>
<td>0.4392</td>
<td>0.5629</td>
</tr>
<tr>
<td>HTTP - VoIP</td>
<td>0.3864</td>
<td>0.4126</td>
<td>0.5406</td>
</tr>
<tr>
<td>SSH - FTP-c</td>
<td>0.3008</td>
<td>0.3162</td>
<td>0.5378</td>
</tr>
<tr>
<td>SSH - POP3</td>
<td>0.3495</td>
<td>0.3715</td>
<td>0.5700</td>
</tr>
<tr>
<td>SSH - VoIP</td>
<td>0.2353</td>
<td>0.3546</td>
<td>0.5049</td>
</tr>
<tr>
<td>FTP-c - POP3</td>
<td>0.2094</td>
<td>0.2336</td>
<td>0.5302</td>
</tr>
<tr>
<td>FTP-c - VoIP</td>
<td>0.2303</td>
<td>0.2752</td>
<td>0.4891</td>
</tr>
<tr>
<td>POP3 - VoIP</td>
<td>0.2477</td>
<td>0.2477</td>
<td>0.4892</td>
</tr>
<tr>
<td>HTTP over SSH - SFTP</td>
<td>0.2090</td>
<td>0.2248</td>
<td>0.5205</td>
</tr>
</tbody>
</table>

TABLE I

AVERAGE AMOUNT OF OVERHEAD INTRODUCED BY DIFFERENT PACKET LENGTH MASKING ALGORITHMS FOR VARIOUS COUPLE OF APPLICATION FLOWS.

By observing the results, we notice that the amount of overhead can vary significantly depending on the applications we mix. In addition, we can see that masking with fixed-size packets leads to introduction of high amount of overhead, while algorithm with only padding does not cause a significant increase in overhead compared to the generalized masking in which we can fragment. In a lot of cases, adding fragmentation improve marginally the achieved fraction of overhead. 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.

VI. Discussion and Concluding Remarks

A number of issues are left open by the approach and analysis presented above.

First, the aim of the study is to assess what the optimum performance in terms of overhead could be under the constraint of perfect masking. This is done in case of two application protocol types. The approach taken assumes the entire sequence of packet lengths of a flow used in the masking process can be observed to take the masking decision. Actual application usually develop through a bidirectional message exchange, made of requests and replies. As a matter of example, HTTP has a typical pattern of a request message from the client browser and a burst of messages sent back by the server, with the objects of the requested page. POP3 opens with a pattern of request-reply messages. Common to all such examples is the fact that if messages are not released on one side to be delivered to the other side, the protocol does not carry on and no new messages...
are produced. So, it is not usually the case that the first \( m \) messages of an application type are available at the transformation point. A practical masking algorithm should operate message by message. By doing so, correlation among packet lengths cannot fully be accounted for, so that some information about packet lengths can leak even after masking. Our approach is therefore useful to characterize the full concealment case and evaluate its cost in terms of overhead.

A second point is the impact of other packet features besides packet lengths. Timing of subsequent packet emissions and direction of packets (from client to server or vice versa) are important features as well. Masking should be performed separately by each party in a bidirectional communication. Even if packets are tunneled within a secure channel using encryption, packet direction can be identified by external envelope addresses (secure channel tunnel endpoints). To mask direction would actually require also acting deeply on network routing, which anyway brings us away from our end-to-end paradigm. As for timing of packets in a given direction, fragmenting is completely ineffective if packets are dealt with one at a time as they arrive at the transformation box. As a matter of example, if a sequence of messages arrives at the transformation box at time \( t_i \), \( i = 1, \ldots, m \), each of them is fragmented into a number of pieces that are sent out immediately, the outcome could be \( m \) bursts of chunks, possibly well separated in time from one another. Then, the adversary can easily guess that a burst of \( n_i \) chunks sent out back to back in the channel starting from time \( t_i \) and separated from the subsequent and preceding bursts by time spans much larger than inter-chunk times comes from a single original message whose length is about the sum of length of payloads of the chunks in the burst. To conceal this information, chunks coming from a single message should be scattered over time and fake chunks should be created and sent out during idle times. Alternatively, a whole set of \( m \) application messages could be stored and then processed all together by the transformation box. This could result in an intolerable delay as \( m \) increases, e.g., in case of VoIP, or be impossible, as in case of request-reply application traffic patterns.

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 the \( m \)-dimensional joint probability density functions for the \( M \) applications. When the number of packets you want to mask increase, the amount of data required becomes very high.

A final technical issue is the extension of our approach to more than two application protocol types. Numerical algorithms for solving the optimization problem of Section III are known only up to \( M = 3 \).

**REFERENCES**


