BIT-TRAPS: Building Information-Theoretic Traffic Privacy into Packet Streams

Suhas Mathur and Wade Trappe, WINLAB, Rutgers University
{suhas, trappe}@winlab.rutgers.edu

Abstract—Sniffing encrypted data packets traveling across networks can often be useful in inferring non-trivial information about their contents because of the manner in which the transmission of such packets is handled by lower layers in the communications protocol stack. In this paper, we formally study the side-channel formed by variable packet sizes, and explore obfuscation approaches to prevent information leakage while jointly considering the practical cost of obfuscation. We show that randomized algorithms for obfuscation perform best and can be studied as well-known information-theoretic constructs, such as discrete channels with and without memory. We envision a separate layer called a Bit-Trap, that employs buffering and bit-padding as orthogonal methods for obfuscating such side channels. For streams of packets, we introduce the use of mutual-information rate as an appropriate metric for the level of obfuscation that captures non-linear relationships between original and modified streams. Using buffering-delay and average bit-padding as the respective costs, a Bit-Trap formulates a constrained optimization problem with bounds on the average costs, to implement the best possible obfuscation policy. We find that combining small amounts of delay and padding together can create much more obfuscation than either approach alone, and that a simple convex tradeoff exists between buffering delay and padding for a given level of obfuscation.

I. INTRODUCTION

The successful adoption of personal communication technologies is leading to an increase in the amount of sensitive or private information being exchanged. Sensitive information, such as a personal phone call or financial transactions, cross our networks everyday and it is very easy for adversaries on the network to monitor this traffic, especially when a wireless link is involved. Although a first line of defense to ensuring confidentiality involves encrypting the packets prior to transmission and forwarding over a network, recent evidence has shown that higher layer encryption alone is insufficient to providing the strict guarantees of privacy that most users expect for their transactions. In numerous different application scenarios, it has been shown that basic traffic analysis involving the statistical analysis of packet sizes and inter-arrival times can identify significant levels of contextual information in spite of the underlying session being properly encrypted.

Adversaries armed with knowledge of the application and the underlying network functions, yet no knowledge of the cryptographic material associated with security primitives, have been able to identify a surprising amount of information, including the language and specific phrases spoken in encrypted voice-over-IP calls [1], [2], or the identity of a video clip [3].

In this paper, we cast the problem of protecting the contextual privacy associated with encrypted packet streams in an information-theoretic setting, where the objective is to obfuscate the meaning of these packet streams by randomly padding packets or changing their sizes by buffering them. Starting from an assumption that packets can be broken into smaller chunks (for the sake of discussion, we’ll assume the minimal indivisible unit is a bit), padded, and buffered, we seek to minimize the mutual information [4] between an incoming packet stream and an outgoing packet stream, while maintaining desirable constraints (e.g. delay and average bits spent on padding). We assume that there is some fixed cost (in bits per packet) for obfuscating packet sizes, perhaps in the form of protocol bits that tells the destination how to reassemble the original packets. We do not deal with the details of this mechanism, so as to focus on the tradeoffs between the level of obfuscation achieved and the amount of resources required. We envision Bit-Trap as a layer as depicted in Figure 1 that sanitizes packetized traffic being handed down to it, so as to remove the existence of a leaking side-channel as much as possible, given a set of user-specified constraints on the available resources that Bit-Trap can spend. This paper takes steps in this direction by quantifying the relationship between resources and the amount of leakage possible.

Through our analysis, we uncover several useful insights that can guide practical traffic analysis countermeasures:

• In a single packet model, random padding of packets is the best way to achieve privacy against traffic analysis, and given a fixed average padding budget, an optimal method for random padding can be quickly found via a convex program. We also prove that the space of padded packets need not be different from the space of true packet sizes and that the trade-off between the achieved obfuscation and the amount of padding allowed is a convex rate-distortion type relationship.

• For streams of packets, adding a small amount of
buffering-delay allows us to distort packet sizes without adding padding bits. An optimal obfuscation strategy is shown to exist when buffering-delay and padding are used in concert.

We begin with a review of related literature in Section II to set the context for our work. In Section IV we analyze the single packet case, which proves most instructive, and then move onto the general case of a stream of packets in Section V. We conclude the paper in Section VI.

II. RELATED WORK

The subject of traffic privacy has a rich history and prior literature. These can be broadly classified into attacks and defenses dealing with: (i) making inferences about the source and/or origin of communications in a network, and (ii) gleaning contextual information about the information content itself. It is the latter with which we are concerned in this paper. Work in this model has focused much more heavily on attacks than on defenses. Attacks have ranged from inferring the spoken language and specific phrases in encrypted VoIP streams [1], [2] from the sizes of packets in a stream, to inferring contextual information from key strokes in encrypted SSH sessions [5] and the identity of video clips [3] from the variable timing of packets on a network connection. One reason for the greater attention given to attacks is that these works most often suggest a supposedly ‘simple’ guarding measure: regularize the quantity that leaks out information. In the above examples, this would translate to making all packets the same length and making intern-packet durations the same for all packets. While expending sufficient resources on the problem (extra bits, delay, etc.) can eliminate a known side-channel, the amount of extra resources that might be needed to remove such side-channels may be non-trivial. In this paper we formally study the problem of side-channel leaking out information, in the context of variable packet sizes. In particular, we show that there often exist optimum solutions when resources to be expended are limited. Finally, we note that our work has some connections with the study of timing-capacity of queues [6], in which information is encoded in the intervals between packets arriving at a queue. However, variable packet sizes and inter packet duration are separate sources of randomness - we specifically used a model with one packet per slot (including zero-sized ‘packets’) so as to focus on the information contained in the packet sizes and not their timing.

III. NOTATION

We will denote the sizes of packets found in a network or a packet stream by $A = \{A_1, \ldots, A_M\}$, where $A$ is the set of all packet sizes, and $A_1 < A_2 < \cdots A_M$ are the possible sizes of the packets. We will denote the probabilities of occurrence of the $M$ possible packet sizes by the probability mass function $P = \{p_1, \ldots, p_M\}$, where $\sum_i p_i = 1$. The packet sizes, after being modified by an obfuscator are denoted by $D = \{D_1, \ldots, D_N\}$, where we will assume with loss of generality that $D_1 < D_2 < \cdots D_N$, where $N$ need not equal $M$. The letters $A$ and $D$ are chosen to denote arrivals and departures respective, where the arrivals and departures are with respect to an obfuscator, which may add extra bits and/or delay to the packets. We will use the letters $A$ and $D$ to denote the random variable associated with the true size of a packet and modified size of a packet respectively. We use the notation $y^+$ to denote $\max(y, 0)$.

We will use the terms obfuscator, obfuscation channel and obfuscation system interchangeably, to refer to the operator that we are interested in designing, responsible for preventing the leakage of information through the packet-size side channel.

With slight abuse of notation, in the section on packet streams, we will use $A_i$ and $D_i$ to denote the true and modified size of the $i^{th}$ packet in a packet stream respectively, and $A^n$ & $D^n$ to denote the sequences $A_1, A_2, \ldots, A_n$ and $D_1, D_2, \ldots, D_n$. Lower case letters $a$ and $d$ denote the realization of a true packet size and modified packet size respectively. Likewise, $a_i$ and $d_i$ denote a specific realization of the true size and modified size of the $i^{th}$ packet in a sequence of packets respectively. We will use $\{G\}$ to represent the random process formed by successive realizations of a random variable $G$.

IV. SINGLE PACKETS

Consider a sensor network which monitors several types of events. As each type of event is triggered or observed by a node in the network, it records or generates data corresponding to that event and send it in the form of a messages towards the sink. Let the set of all possible events in the network be denoted by $E = \{E_1, \ldots, E_M\}$ with probabilities of occurrence $P = \{p_1, \ldots, p_M\}$ and message sizes $A = \{A_1, \ldots, A_M\}$. Consider an adversary observing traffic at an opportune point in the network. The adversary can attempt to infer which event has occurred simply by looking at the size of the message it intercepts over the air. We need a formal metric to quantify the amount of uncertainty in the attackers inference, given the observations available to her. If an adversary observes a packet of size $s$, the amount of uncertainty in the adversary’s inference about the event in $E$ given her observation of $s$ is captured by Shannon’s entropy function:

$$H(E|s) = - \sum p(E_i|s) \log p(E_i|s)$$

A network designer’s goal would be to maximize this conditional entropy by altering the sizes of packets transmitted after each event. Let these altered packet sizes be reflected as $D = \{D_1, \ldots, D_N\}$. The problem of providing traffic privacy then becomes: Maximize $\sum_{j=1}^n H(E_j|s)$ while minimizing the expected communication overhead due to the privacy protection methods. There are two broad strategies to improve the privacy in this situation:

- **Constant packet size:** The key idea is to attempt to make all packets in the network have the same size. It requires padding all packet to the size of the largest packet.
- **Randomized packet sizes:** This approach relies on randomizing the size of every single packet to create an uncertainty about the type of packet and underlying event.

The suggested solution for preventing analysis of packet sizes in much of the prior work highlighting attacks, is padding...
packets to all have the same size. This strategy, though, is only satisfactory when the variation in sizes across packets is small. However, when the variation is large, padding packets can significantly increase the usage of bandwidth. Furthermore, for scenarios in which nodes have limited energy, such as wireless sensor networks, this increase in traffic can lead to an unacceptable increase in energy consumption. In general, we may obfuscate the size of a packet by making it larger (padding bits), or by splitting it into multiple packets. This latter approach is not beneficial for sporadic communication, when the network only infrequently transmits a single packet, since an adversary can simply add up packet sizes and subtract known header sizes to arrive at the true packet size. Hence, we believe that padding packets is the most general and promising recourse for preventing traffic analysis when dealing with single packets or a collection of a few packets rather than a long stream.

To begin, we will investigate whether it is possible to achieve obfuscation without having to pad all packet to the same size. Suppose that we have $M$ different types of packets traversing a network with packet sizes given by $A$ above, and a priori probabilities of occurrence given by $P$ above. Further, suppose we have an average padding budget of $B$ bits. Our task is to pad the packets randomly so as to achieve the greatest obfuscation of the true packet lengths subject to our padding budget. Let $A \in A$ be a random variable that denotes the true packet size and $D \in D$ denote the size of the packet after it has been padded. Then $D = A + Z$, where $Z \in \mathbb{Z}^+$ denotes the number of padding bits added. All packet sizes, $A$ and $D$ are integers. The following result declare that there exists a uniquely optimal obfuscation strategy:

**Theorem 1:** Maximizing obfuscation using randomized packet padding of single packets with a bound on the given average padding budget is a convex optimization problem.

**Proof:** We assume that a maximum packet size of $D_{\text{max}}$ is allowable in the network (where $D_{\text{max}} \geq A_m$) and that the set $D$ of packets that can be observed after padding are $D = \{A_1, A_1 + 1, A_1 + 2, \ldots, D_{\text{max}}\}$, with $D_k$ being the $k$th element of this ordered array. Let $P_{ij}$ denote the probability that a packet of size $A_i \in A$ is padded to a packet of size $D_j$ and $P$ denote the $M \times |D|$ matrix of these probabilities, where $|D| = D_{\text{max}} - A_1 + 1$. Our objective may be stated to be simply

$$\max_{P(D|A)} H(A|D)$$

(1)

where the variables of optimization are the the contents of the transition probability matrix (henceforth, TPM) $P$ and $H(A|D)$ denotes the average uncertainty about $A$ upon having observed $D$, averaged across all packets. Note that $P_{ij} = 0$ whenever $A_i > D_j$. The constraints of the problem can be formally stated as follows:

$$\text{Cost} = E[Z] = \sum_{i=1}^{M} \sum_{j=1}^{|D|} P_{ij}(D_j - A_i)p_i \leq B$$

(2)

$$\sum_j P_{ij} = 1 \quad \text{for all} \quad i = 1, \ldots, M$$

(3)

$$P_{ij} = 0 \quad \text{for all} \quad i, j \text{ such that } D_j < A_i.$$  

(4)

This can be recognized as a type of discrete memoryless channel (DMC) with input alphabet $A$ and output alphabet $D$. The DMC probabilistically adds bits to the input packets, transforming $A$ to $D$ and is represented by $P$, whose elements $P_{ij}$ are the probabilities that a packet of size $A_i$ is converted to a packet of size $D_j$. We wish to determine the worst possible DMC so that an adversary observing its output learns as little as possible about the input, given an average constraint on the additive noise, $E[Z]$.

Observe that the a priori packet probabilities is fixed and that $P_{ij}$ are the variables. Hence, maximizing $H(A|D)$ is equivalent to minimizing the mutual information $I(A; D)$. This quantity is convex in the transition probabilities $P_{ij}$ for a discrete memoryless channel, when the input distribution is fixed (Theorem 2.7.4 of [4]). Therefore, we have a convex objective function (1), with constraints (2)-(4) that are linear in $P_{ij}$. Thus, this is a convex optimization problem with a unique solution that can be computed in time that is polynomial in the number of variables using known convex programming algorithms (e.g. gradient search, etc).

The optimization variables are the elements $P_{ij}$ of the matrix $P$. We note that while the size of the input alphabet $|A|$ is fixed, the size of the output alphabet $|D|$ is up to us as the system designer.

Algorithms for solving a convex optimization problem have running times that are polynomial in the number of variables. Although we have shown that the problem of finding the optimal random padding of packets is convex, the size of the set of variables, namely $P$, can be very large if we allow for all integer values between $A_1$ and $D_{\text{max}}$ in the output alphabet $D$. We now show that in any optimal solution, the output alphabet need not be larger than the input alphabet. This result is significant because it shows that the problem is not only convex but practically solvable since it can be simplified to have only a fairly limited number of variables.

**Theorem 2:** The output alphabet $D$ in the optimal allocation can be the same as the input alphabet $A$.

**Proof:** The proof is based on the concavity of the entropy function and the structure of our cost function. Let $D_k$ be any letter in $D$ with non zero probability of occurrence such that $D_k \notin A$ and $A_1 < A_i < D_k < A_{i+1}$, where $A_1, A_i$ and $A_{i+1}$ are letters in the input alphabet $A$ and have corresponding same-size letters in the output alphabet $D_1 = A_1, D_i = A_i$ and $D_{i+1} = A_{i+1}$. Our objective is to

---

Fig. 2. The discrete memoryless channel (DMC) [4] representing randomized packet padding. Each input packet is mapped to a packet of equal or greater size in accordance with a transition probability matrix $p(D|A)$.
maximize the equivocation

\[ H(A|D) = \sum_{d \in D} H(A|D = d)p(D = d) \]  

(5)

We will prove the above theorem by showing that the equivocation can only increase when any letter of the type \( D_k \notin A \) is eliminated, without increasing the average cost. To begin, assume that the input distribution \( p(A) \) is fixed and an optimal allocation of conditional probabilities \( p(D|A) \) has been found for the chosen output alphabet containing \( D_k \). Consider the effect of eliminating the letter \( D_k \) from the output alphabet by diverting all the conditional probabilities \( p(D = D_k|A) \) for all \( A < D_i \) to the output letter \( D_i \) which is the next letter in \( D \) with packet-size smaller than \( D_k \) (i.e., if we initially allowed all integer sizes in \( D \), then \( D_i = D_k - 1 \)). That is

\[ p'(D_i|A_m) = p(D_i|A_m) + p(D_k|A_m) \]  

(6)

where the prime denotes the new conditional probabilities. This action can only decrease the average cost

\[ \sum_{a \in A} \sum_{d \in D} p(A = a)p(D = d|A = a)(d - a) \]

. Since this action only affects the terms in (5) pertaining to \( D_i \) and \( D_k \), it is sufficient to prove that

\[ H'(A|D = D_i) \cdot p'(D = D_i) \geq H(A|D = D_i) \cdot p(D = D_i) + H(A|D = D_k) \cdot p(D = D_k) \]

where the primes denote the new value of conditional entropy and probability. However, since we have \( p(D = d) = \sum_{a \in A} p(a)p(D = d|A = a) \), we have \( p'(D = D_i) = p(D = D_i) + p(D = D_k) \). Dividing the inequality (7) by \( p'(D = D_i) \) and denoting \( \alpha = \frac{p(D = D_i)}{p'(D = D_i)} \), (7) can be re-written as

\[ H'(A|D = D_i) \geq \alpha H(A|D = D_i) + (1 - \alpha)H(A|D = D_k) \]  

(8)

The last inequality is the definition of a concave function. All that remains is to show that \( p'(A|D = D_i) \) for the entropy term on the left hand side is an \((\alpha, 1-\alpha)\) linear combination of the conditional distributions \( p(A|D = D_i) \) and \( p(A|D = D_k) \) corresponding to the two terms on the right hand side. This can be shown using Bayes’ rule:

\[ p'(A = A_j|D = D_i) = \frac{p(A = A_j) \cdot p(D = D_i|A = A_j)}{\sum_m p(D = D_i|A = A_m) \cdot p(A = A_m)} \]

\[ = p(A = A_j) \cdot \left( \frac{p(D = D_i|A = A_j)}{p(D = D_i)} \cdot \alpha + \frac{p(D = D_i|A = A_k)}{p(D = D_k)} \cdot (1 - \alpha) \right) \]  

(10)

which proves that the new conditional distribution of \( p'(A|D = D_i) \) after redirecting all the conditional probabilities from \( D_k \) to \( D_i \) is a \((\alpha, 1-\alpha)\) linear combination of the distributions \( p(A|D = D_i) \) and \( p(A|D = D_k) \). Since entropy is a concave function of the conditional probabilities, this proves (7). Since this procedure can be carried out for any such letter \( D_k \notin A \) without increasing the average cost, the result follows.\(^1\)

The above result is of practical significance because it allows us to use a matrix \( P \) of size \( M \times M \), thereby limiting the number of variables to \( \frac{M(M+1)}{2} \).

Corollary 1: The smallest possible mutual information is achieved if the output alphabet has size \( |D| = 1 \).

This follows from the fact that if \( |D| = 1 \), then output would have an entropy of \( H(D) = 0 \) and the mutual information, which is non-negative, is upper bounded by the output entropy. It also follows from the theorem above because the same argument as the one in the proof above can be repeatedly applied to each letter of the output alphabet, this time by transferring all conditional probabilities arriving at a given output letter to the next larger letter. However, this increases the padding overhead. In particular, if the constraint \( E[Z] \leq B \) is not violated, the optimal solution is to pad all packets to the size of the largest packet.

The problem of determining the smallest mutual information \( R \) that the obfuscation channel can allow for a given padding budget \( B \) can be thought of as finding the rate distortion function \( R(B) \) with the allowable total average-padding \( B \) as the distortion measure and \( R(B) \) being the smaller mutual information achievable with distortion of \( B \) or less. The rate distortion function is convex and goes to zero at a value of distortion \( B_{\text{max}} \) that is large enough for all packets to be padded to the size of the largest packet. The convexity of \( R(B) \) can be proved as follows. Consider two conditional probability distributions, \( p_1(D|A) \) and \( p_2(D|A) \). For a given distribution of packet sizes, let these conditional distributions lead to distortions and rates of \( B_1, R_1 \) and \( B_2, R_2 \) respectively. Consider a conditional distribution \( p \alpha[D|A] = p_1(D|A) + (1 - \alpha)p_2(D|A) \). This must lead to a distortion of \( B_\alpha = \alpha B_1(y|x) + (1 - \alpha)B_2(y|x) \), since the distortion \( B = E[Z] \) is a linear function of the conditional distribution. Since mutual information is a convex function of the conditional distribution, \( R_\alpha \leq \alpha R_1 + (1 - \alpha)R_2 \), which implies that the \( R(B) \) curve is convex. Further, in a padding-only obfuscation channel, the only condition under which the mutual information can be brought to zero is if all packets are padded to the size of the largest packet. Otherwise, the

\(^1\)Note that any packet of the smallest packet size \( A_1 \in A \) must either be padded to the next larger packet \( A_2 \in A \) or be left unchanged, because padding to a length between \( a_1 \) and \( a_2 \) does not buy any equivocation but incurs a non-zero cost. Therefore the above proof is trivially true for any \( D_k \) in the interval \( A_1 < D_k < A_2 \).
sequence of random variables $A_i$ notation with respect to the previous section, we will denote time slotted stream of data packets (one packet per slot) and packets together to form larger packets. We model a discrete as well as splitting packets into smaller packets and fusing information by employing the variability in packet sizes. To defend against such traffic analysis, we will examine padding of the padding budget, all packets can be padded to the size of the largest, providing perfect obfuscation.

occurrence of any packet at the output conveys the information that the input packet was of equal or smaller size.

V. PACKET STREAMS

We now study a stream of variable sized packets. We will find that the insights we have developed using the single packet model will prove to be useful in analyzing streams. A number of recent attacks on encrypted data streams [1], [2] have used machine classification techniques to infer useful information by employing the variability in packet sizes. To defend against such traffic analysis, we will examine padding as well as splitting packets into smaller packets and fusing packets together to form larger packets. We model a discrete time slotted stream of data packets (one packet per slot) and the manipulations carried out on it using an information-theoretic discrete channel as before. With slight abuse of notation with respect to the previous section, we will denote the stream to be obfuscated as a time ordered sequence $A^N$ of random variables $A_i$, with $i = 1, \ldots, N$ representing time slots and $A_i$ representing the size of the packet in the $i^{th}$ slot, such that $A_i \in A$. Similarly, the output $D^N$ is a sequence of random variables $D_i$, $i = 1, \ldots, N$, such that $D_i \in D$ for all $i$ and $A, D \in \mathcal{X}^+$. We will use the notation $G$ to refer to the sequence of random variables $G_1, \ldots, G_N$, and $\{G\}$ to mean the random process formed by successive realizations of $G$.

As before, we will use the mutual information as a metric to quantify how dissimilar the modified stream of packets $\{D\}$ is, compared to the original stream $\{A\}$. However, we will no longer be able to use the single-letter mutual information metric that we have used in Section IV; instead, we will develop a mutual information metric that is appropriate for streams, treating each stream as a stationary, discrete-time stochastic process.

Mutual information provides a general upper bound on the performance of any classification algorithm, and that it is one of the few metrics that can measure non-linear relationships between sequences (though there are many measures that quantify only linear relationships [7]). This is important for us as the redistribution of bits across packets can easily create a non-linear relationship between the input and output of an obfuscator with memory. This is the reason why single-letter mutual information is not appropriate for measuring the dependence between streams.

When the input stream $A^n$ has a finite memory (i.e., there exists some $k$ such that for all $i$, $A_i$ and $A_{i+k}$ are independent for all practical purposes), a data stream can be treated as a sequence of independent vector inputs to a discrete memoryless vector channel in a manner similar to that of Section IV. Obfuscating information in the packet-size side channel could then be accomplished by the following steps:

1) Collect $k$ packets arriving sequentially and treat the sequence $\{A_1, \ldots, A_k\}$ as a single vector input to a vector discrete memoryless channel.

2) Map each input vector of length $k$ to a vector of packets of lengths $\{D_1, \ldots, D_l\}$, where $l$ may be different from $k$, such that $\sum_{i=1}^k A_i = \sum_{i=1}^l D_i$ (conservation of total information bits).

Since the above approach employs a discrete memoryless channel, it can be considered to be a straightforward extension of the method of Section IV using vector inputs instead of scalars. However, it is not practical for two reasons. First, it requires the obfuscating vector-DMC to collect $k$ packets from the stream before deciding the sequence of output packets, thereby introducing a fixed large delay in the stream, which may not be suitable for delay-sensitive applications. Secondly, it requires the computation of a large transition probability matrix to characterize the vector-DMC that is of size $|A|^k \times |D|^l$.

An alternative approach that seeks to avoid the problems with a vector-DMC is that of a discrete channel with memory. Here, the obfuscating channel is defined in general by the conditional probability distribution $p(D_n|A^n, D^{n-1})$ wherein the output of the channel is explicitly allowed to depend upon past inputs and outputs. Since the input to the channel does not have a chance to observe the output, we say that the channel is used without feedback, i.e. therefore, $p(A_n|A^{n-1}, D^{n-1}) = p(A_n|A^{n-1})$. Our objective can be now be stated as

$$\min_{p(D_n|A^n, D^{n-1})} I(\{A\}; \{D\})$$

where $I(\{A\}; \{D\})$ is the mutual information rate between the discrete time processes $\{A\}$ and $\{D\}$, defined as

$$I(\{A\}; \{D\}) \triangleq \lim_{n \to \infty} \frac{1}{n} I(A^n; D^n)$$

$$= \lim_{n \to \infty} \frac{1}{n} \left( H(A^n) - H(A^n|D^n) \right)$$

$$= \lim_{n \to \infty} \left( H(A_n|A^{n-1}) - H(A_n|A^{n-1}, D^n) \right)$$

The idea of splitting and fusing packets can be generalized by considering a buffer that allows us to store the remaining bits of a packet that has been split (thereby allowing $D_n < A_n$) as a first-in-first-out queue. The queue is modeled using an internal state variable whose value $Y_n$ at time instant $n$
denotes the number of bits contained in the buffer just before the \(n\)th arrival, \(A_n\). Thus, delay can be used as a resource for obfuscating the stream \(\{A\}\). It can now be seen why mutual information rate is an appropriate metric - the non-linearity introduced by the queue cannot be captured by a metric that only measures linear relationships. Note that in general, a packet stream (non-necessarily discrete-time) may contain information in the packet sizes as well as in the inter-packet timing. This is true for discrete-time streams as well. For example, variable periods of silence between spoken words (zero-sized packets) can convey information about the language being spoken. We accommodate this phenomenon in our model by including zero-sized packets in the input and output alphabets \(A\) and \(D\), respectively.

We can augment the use of a buffer with randomized padding to create further randomization between input and output streams. While the padding resource is limited by the average number of padding bits available for random padding, the delay resource is limited by the (average) delay that is introduced into the stream. Therefore, we now have two separate constraints that have to do with average delay and average padding bits:

\[
E[Y] \leq Q \quad (13)
\]
\[
E[Z] \leq B \quad (14)
\]

where \(Q\) denotes the largest mean delay that can be tolerated by the stream, and \(B\) denotes the bit-padding budget as in Section IV.

In the following subsections, we will develop, step-by-step, a framework for studying the obfuscation of streams of variable sized packets as constrained optimization problems, using padding bits and delay as separate resources.

A. Obfuscation by Padding only

In this model, packets are altered only by adding padding-bits, as in Section IV. Successive packets are allowed to have correlated sizes. Since the output of the channel \(D_n\) at time \(n\) can, in general be arbitrarily dependent on the set of all past inputs \(A^n\) and past outputs \(D^{n-1}\), we may write our objective in its most general form as:

\[
\min_{p(D_n|A^n, D^{n-1})} I(\{A\}; \{D\})
\]

We will refer to the variable \(p(D_n|A^n, D^{n-1})\) as the obfuscation channel. How should we design this channel? Notice that in communications engineering we are never faced with the problem of designing a channel with memory to minimize mutual information. Indeed, this is just the opposite of the usual goal of designing a coding scheme for a given channel with memory (e.g. a fading channel). As a first step we will consider what the output alphabet \(D\) should be, when padding a stream of packets with possibly correlated sizes.

**Theorem 3:** The output alphabet \(D\) in the obfuscation of a stream of packet sizes by padding only can be the same as the input alphabet of packet sizes \(A\).

**Proof:** The proof is similar to the proof of Theorem 2 and therefore we will only provide a sketch here. The proof is trivially true for an iid stream \(\{A\}\) because such a stream can be treated as independent repeated occurrences of a single packet, and we have already proved the result for a single packet. Let us consider the case of a correlated stream \(\{A\}\). Instead of considering a single letter characterization of the obfuscation channel, \(p(D|A)\) as depicted in Figure 2 and described in Section IV, one may treat the obfuscation channel as if it were operating on a sequence of packets \(A^n\) at once, and one may therefore characterize the channel as \(p(D^n|A^n)\), with the understanding that there is no delay introduced and \(p(D^n|A^n) = 0\) if \(D_i < A_i\) for any \(i = 1, \ldots, n\). With this characterization, we can now treat the obfuscation channel as a DMC with vector inputs and outputs. We wish to prove that the output of this DMC need only consist of vectors whose elements belong to \(A\).

Consider an output of this DMC \(D^n_{(k)}\), where the superscript \(N\) denotes the length of the vector and the subscript \(k\) simply denotes the index of this vector in a lexicographic ordering of the output vectors. Let \(D^n_{(k)}\) contain elements that do not belong to \(A\). Let \(D^n_{(i)}\) denote the output vector in which each element belongs to \(A\) and is such that each element of \(D^n_{(k)}\) that is not in \(A\) is replaced by the largest element in \(A\) smaller than it. With respect to the proof of theorem 2, \(D^n_{(k)}\) plays the role of \(D_k\) and \(D^n_{(i)}\) plays the role of \(D_i\). Similarly, \(p(D^n = D^n_{(k)}) \cdot p(D^n = D^n_{(i)})\) and \(p(D^n = D^n_{(i)})\) play roles analogous to \(p(D = D_k), p(D = D_i)\) and \(p(D = D_i)\) respectively. With this equivalence, it can be shown, using the set of arguments identical to the ones in the proof of theorem 2, that

\[
H' \left[ A^n | D^n^{(i)} \right] \cdot p \left[ D^n = D^n^{(i)} \right] 
\geq
\]
\[
H \left[ A^n | D^n^{(i)} \right] \cdot p \left[ D^n = D^n^{(i)} \right] 
\]
\[
H \left[ A^n | D^n^{(k)} \right] \cdot p \left[ D^n = D^n^{(k)} \right]
\]

which in turn, using an argument based on the concavity of the entropy function, identical to the one used in the proof of Theorem 2 proves that the output need only consists of elements from the input alphabet \(A\).

Can the construction of the padding-only obfuscator be simplified from the cumbersome form \(p(D_n|A^n, D^{n-1})\)? Yes, it can be reduced to the form \(p(D_n|A^n)\); that is, the output \(D_n\) at time \(n\) need not depend upon the previous outputs given the previous inputs. This can further be reduced to the form

![Fig. 6. The general model for a discrete channel with memory in which packets can be split up and combined. The memory of the channel is manifest in the form of a buffer \(Y_n\) that hold bits that have not been sent out yet. In addition, a random amount of bits \(Z_n\) can also be used for padding each departing packet.](image-url)
Before producing a memory of $A^n$ or is Markovian.

Lemma 1: The following is a Markov Chain:

$$A_n \leftrightarrow A^{n-1} \leftrightarrow D^{n-1}$$

Proof: This can be proved by a simple constructive proof that utilizes the vector-DMC characterization above. Let us assume that $A^{n-1}$ is the input to a vector-DMC and $D^{n-1}$ is the corresponding output. The DMC is therefore characterized by the conditional distribution $p(D^{n-1}|A^{n-1})$. The key observation is that even though in reality our obfuscation channel does not observe the entire vector $A^{n-1}$ before producing $D^{n-1}$, we can still construct the conditional distribution $p(D^{n-1}|A^{n-1})$ because each $D_i$ only depends upon past inputs $A_i$ and outputs $D^{i-1}$. Similarly, consider another DMC which takes $A^n$ as input and produces $A^n$ as output. We now have two DMCs, both taking $A^{n-1}$ as input and producing $D^{n-1}$ and $A^n$ as their respective outputs. From this construction, it follows that $A_n \leftrightarrow A^{n-1} \leftrightarrow D^{n-1}$ is a Markov Chain.

Let us now focus on the objective function:

$$\min_{p(D_n|A^n, D^{n-1})} I(\{A\}; \{D\})$$

$$= \min_{p(D_n|A^n, D^{n-1})} H(\{A\}) - H(\{A\} | \{D\})$$

$$= H(\{A\}) - \max_{p(D_n|A^n, D^{n-1})} \lim_{n \to \infty} H(\{A\} | A^{n-1}, D^n)$$

$$= H(\{A\}) - \max_{p(D_n|A^n, D^{n-1})} \lim_{n \to \infty} H(\{A\} | A^{n-1}, D^n)$$

$$= \min_{p(D_n|A^n, D^{n-1})} \lim_{n \to \infty} I(A_n; D_n | A^{n-1})$$

$$= \min_{p(D_n|A^n, D^{n-1})} \lim_{n \to \infty} H(D_n | A^{n-1}) - H(D_n | A^n)$$

Why (16) follows from Lemma 1 and (19) follows from (18) because the argument of (18) does not require $D_n$ to depend upon $D^{n-1}$ but only upon $A^n$. We have therefore shown that the general obfuscator $p(D_n|A^n, D^{n-1})$ can be simplified to $p(D_n|A^n)$. The obfuscator can be further simplified if it is known that the input stream $\{A\}$ has a known finite memory or is Markovian.

Corollary 2: If the input stream is Markovian such that $p(A_n|A^{n-1}) = p(A_n|A^{n-1}_{n-k})$ then the obfuscator can be simplified from $p(D_n|A^n)$ to $p(D_n|A^{n-1}_{n-k})$.

Proof: If the input stream is $k$th order Markovian, then $A_n \leftrightarrow A^{n-k} \leftrightarrow A^{n-k-1}_t$ is a Markov chain. Therefore, we have

$$\min_{p(D_n|A^n)} \lim_{n \to \infty} H(A_n | A^{n-1}) - H(A_n | A^{n-1}, D^n)$$

$$= \min_{p(D_n|A^n)} \lim_{n \to \infty} H(A_n | A^{n-1}_{n-k}) - H(A_n | A^{n-1}_{n-k}, D^{n-k}_n)$$

$$= \min_{p(D_n|A^n)} \lim_{n \to \infty} I(A_n; D_n | A^{n-1}_{n-k})$$

$$= \min_{p(D_n|A^n)} \lim_{n \to \infty} H(D_n | A^{n-1}_{n-k}) - H(D_n | A^{n-1}_{n-k})$$

where (20) follows because the argument to be maximized in the last equation does not require $D_n$ to depend upon inputs prior to $A^{n-k}_n$; therefore the obfuscator can be simplified to $p(D_n|A^{n-1}_{n-k})$.

Note that the simplified objective function

$$\max_{p(D_n|A^n)} H(A_n | A^{n-1}, D^n)$$

is convex in the variable of optimization $p(D_n|A^n)$. This can be seen as follows. $H(A_n | A^{n-1}, D_n)$ is concave in the distribution $p(A^{n-1}, D_n | A_n)$, which in turn can be expressed as $p(D_n|A^n) \times p(A^{n-1}|A_n)$, implying that $p(A^{n-1}, D_n | A_n)$ is affine in $p(D_n|A^n)$. Therefore, $H(A_n | A^{n-1}, D_n)$ must be concave in $p(D_n|A^n)$. Further, if we use a memoryless channel, i.e., we use the simplified variable $p(D_n|A_n)$ instead of the more cumbersome $p(D_n|A^n)$, the objective function continues to be a convex function of $p(D_n|A_n)$, although the optimal value may be smaller. This can be seen as follows. $H(A_n | A^{n-1}, D_n)$ is concave in the distribution $p(A^{n-1}, D_n | A_n)$. Now, if we use a memoryless channel $p(D_n|A_n)$, then $D_n$ is independent of $A^{n-1}$ given $A_n$.

Finally, it must be noted that the rate at which information leaks out through the obfuscation channel - in bits per packet - is lower for a correlated stream $\{A\}$ than for an iid stream, as one would intuitively expect. This is true even if one picks the possibly suboptimal padding rule $p(D_n|A_n)$ – that is, a rule where the output is conditionally independent of previous packet sizes, given the present packet size. This can be shown as follows, starting with the objective function in (17):

$$\min_{p(D_n|A^n)} I(A_n; D_n | A^{n-1})$$

$$\leq \inf_{p(D_n|A_n)} I(A_n; D_n | A^{n-1})$$

$$= \min_{p(D_n|A_n)} I(A_n; D_n | A^{n-1})$$

$$\leq \min_{p(D_n|A_n)} I(A_n; D_n)$$

where (21) follows from the fact that using an obfuscator $p(D_n|A_n)$ where the output of the obfuscator is conditionally independent of past inputs given the most recent input $D_n$.
can only worsen the optimal mutual information achieved by the more general obfuscator \( p(D_n|A^n) \). The equality in (22) follows from the fact that \( \inf \) can be replaced by \( \min \) because replacing \( p(D_n|A^n) \) by \( p(D_n|A_n) \) as the variables of optimization preserves the convexity of the variable set and hence \( I(\{A\}; \{D\}) \) remains convex in the new variable. The inequality in (23) follows from the fact that when the obfuscator is \( p(D_n|A_n) \) rather than \( p(D_n|A^n) \), then

\[
D_n \leftrightarrow A_n \leftrightarrow A^{n-1}
\]

is a Markov Chain and this implies that

\[
I(A_n; D_n) \geq I(A_n; D_n|A^{n-1})
\]

(see for e.g. [4], Section 2.8).

**B. Obfuscation by buffering only**

In this subsection, we will develop a mechanism to obfuscate a stream of packets using buffering only. We will find that this problem translates to the analysis of a queue as an information-theoretic channel, which proves to be a hard problem. However, the insights we develop in this section serve us greatly in understanding how an obfuscation system that combines buffering and padding will work.

When using buffering alone, the obfuscator is a queue of bits. In the \( i^{th} \) slot, a packet of size \( A_i \) arrives at the queue and a packet of size \( D_i \) departs from the head of the queue. There are no padding bits added to the stream as a whole. After the \( (n-1)^{th} \) departure \( D_{n-1} \), the buffer contains the set of bits \( Y_n = \sum_{j=1}^{n-1} (A_j - D_j) \) that have arrived at the queue but have not departed yet. We use \( X_n = Y_n + A_n \) to denote the buffer contents just after the \( n^{th} \) arrival \( A_n \) (see Figure 5). Let us begin by stating the objective in its most general form:

\[
\min_{p(D_n|A^n, D^{n-1})} I(\{A\}; \{D\})
\]

Without any bound on the delay, the above optimization problem might cause an infinite delay (for eg. corresponding to buffering all packets and then releasing them in arbitrary sizes). In practice we cannot tolerate large delays and therefore need to model this as a constraint on some measure of delay. Some possible measures of delay we might consider are:

- Average queue length
- Maximum delay per bit
- Buffer overflow probability

The intuition is that the larger the queuing delay that we are willing to tolerate (say, average delay \( \leq Q \)), the more unrelated the output of the obfuscation channel can be to the sequence of arrivals at the queue. If the queue is long enough, the output of the channel \( D_n \) at time \( n \) can be chosen to be completely independent of the sequence of arrivals \( A^n \). However, given, say a bounded mean queue length, the queue would reduce in size sometimes and may even be empty at times, invariable forcing \( D_n \) to have some dependence on the size of the queue and hence on the arrival process. In such a situation it is not possible to choose the output of the obfuscation channel independent of the queue size. We are interested in studying the trade-off between the mutual information between \{A\} and \{D\} and an appropriate measure of delay.

It is difficult to attack the problem in its complete generality, for all possible types of channels with memory and all types of correlation in \{A\}. Therefore, using the intuition given above, we will restrict our attention to the class of queues for which \( p(D_n|D^{n-1}, A^n) = p(D_n|A^n) \). That is, the class in which the departure \( D_n \) in any slot depends only on the size of the queue \( X_n = Y_n + A_n \) at that instant, i.e. after the arrival \( A_n \). This will allow the construction and analysis of simple queue control strategies which can make decisions based only on the present state of the buffer. The great advantage of a simplification to \( p(D_n|X_n) \) is the vastly reduced search space for the control variables \( p(D_n|D^{n-1}, A^n) \). A number of interesting models can be formed using this class of queues, comprising both stochastic and deterministic control policies:

1. \( D_i = \min(X_i, S_i) \) where \( S \) is a random variable, independent of \{A\} with a fixed distribution \( p(S) \), chosen by us.
2. \( D_i = \min(X_i, c) \) for some constant \( c \).
3. \( D_i = c \) if \( X_i \geq C \) and 0 otherwise, for a constant \( c \).
4. \( D_i = c \) if \( X_i \geq c \) and \( X_i \) otherwise, for a constant \( c \).

Model 1 above is a random queue control policy, while the other are deterministic queue controllers (i.e. \( D \) is deterministic given \( X \)). We will use model 1 above in the sequel. Model 2 is a specific case of Model 1, when the distribution \( p(D) \) converges to a delta function. Further, we will stick to considering average delay\(^2\) as a measure of the queuing delay. Throughout, we will consider a discrete random variable \( S \), with pmf \( p(S = i) \), \( i \in S \) to be the variables of our problem, assuming that a support \( S \) for \( S \) has been fixed. Finally, we will assume that the input process \{A\} has a first order Markov correlation, i.e \( p(A_n|A^{n-1}) = p(A_n|A_{n-1}) \). The justification for this comes from a result in [8] Chapter 6, which explains that a source with arbitrary correlation structure can be approximated by an \( M^{th} \)-order Markov chain with arbitrary precision. For analytical ease, we have chosen \( M = 1 \) - however, it is possible to model higher order Markov chains as a first order Markov chain, by a simple re-definition of the state to include multiple outputs.

The average queue length (or average delay) is convex with respect to the distribution \( p(S) \). To establish this, we will make use of a result from [9]. First, observe that \( \mu = E[S] \) is linear in \( p(S) \); therefore, if the average queue length is convex in \( \mu \), then it must be convex in \( p(S) \). [9] studies the trade-off between average queue length and the minimum required \( \mu = E[S] \) as a rate distortion problem using average queue length as a measure of distortion. It has been shown that the minimum required \( \mu \) needed to keep the mean queue length under a fixed constant \( Q \), is decreasing and convex in \( Q \). We use this result by observing that it implies that given a \( \mu \), the average queue length must be decreasing and convex in \( \mu \). Now, \( E[S] = \sum_{i \in S} i \cdot p(S = i) \) is a linear function of the distribution \( p(S) \), and by a convexity preserving argument [10], if a function \( f(\cdot) : \mathbb{R} \to \mathbb{R} \) is convex and non-increasing and another function \( g(\cdot) : \mathbb{R}^n \to \mathbb{R} \) is concave, then the composite function \( f(g(x)) \) is convex in \( x \in \mathbb{R}^n \). Here, the

\(^2\)By an application of Little’s law, mean queue length (in bits) is proportional to the mean delay in the discrete-time queue we are using.
distribution \( p(S) \) plays the role of \( x \in \mathbb{R}^n \).

**Conjecture 1:** The mutual information rate \( I(\{A\}; \{D\}) \) is convex in the distribution \( p(s) \).

While we have been unable to conclusively prove this conjecture analytically, our conjecture is based on the observation that mutual information \( I(A^N; D^N) \) is convex in the conditional distribution \( p(D^N|A^N) \), which can be expressed in terms of \( p(S) \) via \( p(D|X) \). Therefore, controlling \( p(S) \) provides a way to simply control \( p(D^N|A^N) \), and thus it is intuitive that mutual information \( I(A^N; D^N) \) would be a convex function of \( p(S) \). We believe the hardness of proving this result conclusively arises from the infinite memory in the queue that forms the obfuscation channel. In the following subsection, we will find that the intractability of analyzing the use of buffering as a means for obfuscation can be alleviated if small amounts of padding are also allowed.

**C. Using a combination of buffering & padding**

The use of buffering and padding to obfuscate packet streams is complementary and amenable to much simpler analysis than the preceding case of a purely buffering-based obfuscation system.

First, let us focus on the use of buffering and padding to achieve perfect obfuscation, i.e. \( I(\{A\}; \{D\}) = 0 \). In this case, the obfuscated stream \( \{D\} \) gives absolutely no information about the true stream \( \{A\} \). This is achieved by making the size of each departing packet \( D_i \), completely unrelated to the sequence \( A^i \).

**Theorem 4:** Given the class of queue control strategies \( D_n = \min(X_n, S_n) \) the trade-off between delay and padding in the \( I(\{A\}; \{D\}) = 0 \) plane for different choices of \( p(S) \) is obtained by using an average padding of \( E[S] - E[A] \).

**Proof:** The critical observations are that (i) \( \{D\} \) is completely independent of \( \{A\} \) if \( D_i = S_i \), and (ii) utilizing padding bits when the buffer is not empty is counterproductive. Therefore, padding bits only need to be added when \( S_i > X_i \), in the amount of \( S_i - X_i \). The departing packet is made up of bits from the buffer, and if needed, some padding bits, but the size of the departing packet is always independent of the arrival process. Therefore, we have:

\[
D_i = \min(X_i, S_i) + Z_i = \min(X_i, S_i) + (S_i - X_i, 0)^+ = S_i
\]

In the \( I(\{A\}; \{D\}) = 0 \) plane, the output sequence is \( \{D\} = \{S\} \) and therefore \( E[D] = \mu = E[S] > \lambda \). Since the queue is stable (i.e. queue length does not blow up to infinity), by conservation of flow, we must have \( E[A] + E[Z] = E[S] \), and therefore \( E[Z] = E[S] - E[A] \).

Now, let us analyze the general problem of using padding and buffering to obfuscate the stream \( \{A\} \).

**Theorem 5:** The obfuscation of a stream of packet sizes with constraints on the average bit-padding (14) and the average-delay (13) is a convex optimization problem.

**Proof:** We wish to prove that the following problem is a convex optimization problem.

\[
\min_P I(\{A\}; \{D\})
\]

such that

\[
\sum_j P_{ij} = 1 \quad \forall \ i = 1, \ldots, N
\]

\[
E[X] \leq Q
\]

\[
E[Z] \leq B
\]

\[
P_{ij} \in [0,1]
\]

Let us construct a discrete channel, with input alphabet \( A \) and output alphabet \( D = A \), such that the channel is defined by the transition probability matrix \( P \), whose \( (i,j)^{th} \) element \( P_{ij} = pr(D = A_j | A_i = A_i) \). Unlike the channel defined in Section IV however, here \( P_{ij} \) need not be zero for \( j < i \). We will allow a packet of size \( A_i \) to be transformed to a smaller packet \( A_j < A_i \) by keeping the left over bits \( A_i - A_j \) stored in the buffer. The number of bits in the buffer is therefore updated as

\[
X_{i+1} = X_i + (A_i - A_j)
\]

When a packet \( A_i \) is transformed to a larger packet \( A_j > A_i \), the extra \( A_j - A_i \) bits are obtained from the bits held by the buffer at that time, and if the bits in the buffer are not sufficient to make a packet of size \( A_j \), then the required number of dummy bits are padded. Therefore for both the cases \( A_j > A_i \) and \( A_j < A_i \), we may write

\[
Z_i = ((A_j - A_i) - X_i)^+ \\
X_{i+1} = (X_i + (A_i - A_j))^+
\]

The proof is based on the fact that an obfuscator that uses a combination of padding and buffering delay can in fact be modeled as a discrete memoryless channel with a queue inside it, rather than a pure queue that we have dealt with in the previous subsection, which is a channel with infinite memory. In order to show that the problem is a convex optimization problem, we need to show that the quantities \( E[Z], E[X] \) and \( I(\{A\}; \{D\}) \) are convex in the variable of optimization, namely \( P \). From the flow conservation argument used in the proof of Theorem 4, it can be seen that we have the relationship

\[
E[Z] = E[D] - E[A] = \sum_i \sum_j p_{ij} (A_i - A_j),
\]

which makes \( E[Z] \) a linear function of \( P \). Further, the fact that \( E[X] \) is a convex function of \( p(S) \) in the previous model can be used to show that it is also a convex function of \( P \). This can be shown as follows. In the previous model, the random variable \( S \) represented the amount of bits requested from the buffer in order to form a departing packet in each time slot. In the present model, the number of bits requested from the buffer in each time slot is given by the output (with alphabet \( A \) of the discrete channel we have constructed. The distribution of this quantity is given by \( p(D = A_j) = \sum_i p_{ij} \), or as a vector:

\[
p_D = p^T P
\]

Therefore, the distribution of the quantity requested from the buffer is linear in the variable \( P \). Since \( E[X] \) is convex in this
quantity, $E[X]$ must therefore also be convex in $P$. Finally, $I(\{A\}; \{D\})$ is also convex in $P$. Mutual information rate may be expanded as

$$I(\{A\}; \{D\}) = \lim_{n \to \infty} H(A_n | A_1, \ldots, A_n) - H(A_n | A_1, \ldots, A_{n-1}, D_n)$$

and therefore we simply need to show that $H(A_n | A_1, \ldots, A_{n-1}, D_n)$ is concave in $P$. But this has already been shown in Section IV, both for a channel with memory, when $P$ is $p(D_n | A_n)$ as well as for a DMC, when $P$ is $p(D_n | A_n)$.

If we plot every single combination of padding-budget $B$, mean buffering delay budget $Q$ and the mutual information corresponding to this pair, we would get a surface such as the one in Figure 7. Various points on this surface correspond to different choices of $P$.

Corollary 3: The surface formed by mutual-information rate corresponding to various combinations of padding and buffering delay (Figure 7) is a convex surface. Consequently, the mutual-information rate is a convex function of the amount of padding when delay is fixed, and it is a convex function of the mean delay when the amount of padding is fixed.

Proof: This is simply a consequence of the fact that mutual information rate $I(\{A\}; \{D\})$ is convex in $P$. Consider two different matrices $P_1$ and $P_2$ and let these correspond to the points $(B_1, Q_1, I_1)$ and $(B_2, Q_2, I_1)$ in Figure 7. From the convexity of $I(\{A\}; \{D\})$ with respect to $P$, we know that the mutual information rate $I_0$, corresponding to $P_0 = \alpha P + (1-\alpha)P$ is such that $I_0 \leq \alpha I_1 + (1-\alpha)I_2$. This proves that the three-dimensional surface in Figure 7 is convex. Points in the region under the surface cannot be achieved, while points on the surface and above it, can. The achievable region is therefore an epigraph of the convex function. Further since $I(B, Q) : \mathbb{R}^2 \to \mathbb{R}$ is a convex of $(B, Q)$, it must be a convex function in each direction $B$ and $Q$ separately. This follows from the fact that a convex function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if and only if it is convex along any line. An important interpretation of the convexity of the surface is the observation that obfuscating using one of the resources (delay or padding) becomes much more effective when some amount of the other resource is thrown in (see Figure 7).

VI. CONCLUSIONS

Although streams of packets may be encrypted, such mechanisms are not sufficient to ensuring the secrecy of the content contained within the packets. Recent evidence has shown that traffic analysis of encrypted packets can reveal significant information, and consequently in order to achieve heightened levels of confidentiality, it is necessary to employ appropriate countermeasures to prevent traffic analysis. In this paper we have examined the problem of padding and delaying packets of a communication flow as a defense against traffic analysis. We started by examining the problem of protecting the size of a single packet transmission, where we sought to minimize mutual information between the original and outgoing packet size. We showed that maximizing the obfuscation level with a bound on the padding budget is a convex optimization problem and further that the output sizes should be chosen from the same set as the original packet sizes. We then examined the more general case involving a stream of packets, where the objective becomes minimizing the mutual information rate. For the packet stream case, we show how it is possible to split and fuse packets by using a buffer that stores the remaining bits of a packet that has been split earlier. We separately analyze the strategies of only padding packets, and only delaying packets before presenting a general traffic-obfuscation queue control strategy that jointly combines delay and padding. As in the single packet case, we show that there is an underlying tradeoff between delay and padding, and ultimately culminate in the observation that delay and padding are synergistic: a strategy involving small amounts of delay and padding can create far more obfuscation that a strategy involving only delay or only padding.

REFERENCES