2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks

Modeling Privacy and Tradeoffs in Multichannel Secret Sharing Protocols

Devin J. Pohly
SIIS Laboratory
Pennsylvania State University
University Park, PA 16802
Email: djpohly@cse.psu.edu

Patrick McDaniel
SIIS Laboratory
Pennsylvania State University
University Park, PA 16802
Email: mcdaniel@cse.psu.edu

***Abstract*—Privacy is an important aspect of network communications, but privacy protocols require an investment of network resources. For any such protocol to be of use, we need to understand quantitatively how much privacy to expect, as well as the tradeoff between privacy and other network properties, for any given configuration of networks and parameters. We develop a practical privacy measure and protocol model for multichannel secret sharing protocols which integrates privacy and measurable network properties, deriving optimality results for the overall privacy and performance of these protocols. After proving these results, we evaluate the effectiveness of our model by providing a reference implementation and comparing its behavior to the optimality results derived from the model. In our benchmarks, the behavior of this proof-of-concept protocol matched that which is predicted by our model; furthermore, our results demonstrate the feasibility of implementing secret sharing protocols which transmit at a rate within 3–4% of optimal. This model and its results allow us to understand quantitatively the tradeoffs between privacy and network performance in secret-sharing based protocols.**

I. Introduction

Online privacy is an ongoing battle between those who wish to be discreet in their communications and those who wish to surveil, monetize, and otherwise exploit the contents and circumstances of these activities. ISPs, for example, are situated to collect their users' data easily, and we continue to see examples of this privileged position being used to spy on Internet activities [1]. Protecting privacy from adversarial ISPs is a complex challenge involving both confidentiality and anonymity. To further complicate matters, the desired degree of privacy and what can be sacrificed to gain it depend on the context of the communication. For example, the need for privacy when listening to streaming music is not so high as to warrant significant degradation. On the other hand, the degree of privacy needed when organizing a grassroots protest against an oppressive regime merits whatever reduction in performance is necessary to protect the transmission and its sender. In either case, limited network resources must be divided between protecting privacy and improving performance.

It is therefore important that we understand the specific privacy tradeoffs inherent in different protocols under various configurations. Anonymity systems with well-understood tradeoffs include onion routers such as Tor and DC-nets such as Dissent. Tor [2] provides low-latency anonymity based on a model in which traffic from different sources appears to come from any of a variety of exit nodes, using encryption to hide the identity of the parties involved. Its privacy and network behavior have both been formally modeled so as to understand the tradeoffs involved [3], [4]. Dissent [5], on the other hand, invests more resources in privacy, sacrificing the low latency of Tor for a stronger information-theoretic anonymity model, and understanding the DC-net approach formally has led to improvements in its scalability [6]. Privacy in the form of confidentiality is typically provided by encryption via TLS or IPsec, although these solutions are not always available or practical. For example, a majority of traffic on the modern Web is served by content distribution networks (CDNs), which break the end-to-end trust assumptions of TLS in such a way that it is often simply not offered [7]. Multichannel secret sharing protocols such as SMT [8] or MICSS [9] offer a highly tunable solution, but the practical impact of their complexity (as illustrated in Figure 1) has yet to be rigorously modeled.

In this paper, we address this with three major contributions:

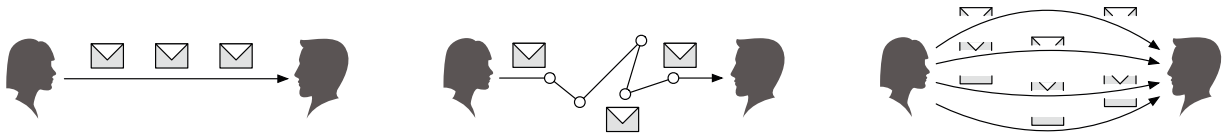
- We define a rigorous privacy measure and protocol model for multichannel secret sharing systems that quantify the expected privacy and performance for any set of network properties and protocol parameters.
- Given a formal specification of the network properties of the available channels, we use our model to derive optimality results showing the maximal achievable privacy and performance profile of a given multichannel setup in terms of privacy, loss, delay, and rate.
- We demonstrate the effectiveness of this model in expressing protocol behavior by developing a reference implementation and collecting benchmarks which come within 3–4% of the predicted optimal throughput in a quiescent network.

II. Background

A. Privacy Systems and Tradeoffs

Enforcing privacy in a public network always comes at a cost. Hiding communications or the identities of those involved adds performance and logistical overhead to otherwise fast and simple communications. Privacy systems involve complexities such as the encryption layers and additional hops of onion

978-1-4673-8891-7/16 $31.00 © 2016 IEEE
DOI 10.1109/DSN.2016.41



(a) TLS (b) Tor (c) Multichannel

Fig. 1: Why privacy and performance are complex to model for multichannel protocols vs. other privacy approaches

routing, or the grouping logic required to make DC-nets scale [6]. Understanding the balance between privacy and other factors in a given privacy system is therefore necessary when developing, deploying, and using it.

One aspect of privacy systems is anonymity: protecting the identity of the participants in a transmission. The most widely deployed and researched anonymity system is the onion-routing network Tor [2]. Tor is designed to provide anonymous network communication for privacy-conscious users while keeping latency to an acceptable minimum. In order to do this, it must strike a number of balances between privacy and performance, for example between improving latency and deterring traffic analysis attacks [10] or between long and short path lengths [11]. To better understand the interplay of these tradeoffs, research has created models to analyze the performance of onion routing [12], [13] as well as the degree of anonymity it provides [3], [14]. Other network anonymity systems, such as Dissent [6] and Crowds [15], shift the balance more heavily toward privacy, requiring further modeling and research to improve performance.

Another important aspect of privacy systems and protocols is protecting the confidentiality of communications. As with anonymity, this does not come for free. Even setting aside practical setup costs—for instance, obtaining and maintaining TLS certificates or an IPsec gateway—there are logistical security problems. Consider that in today's Internet, traffic is largely served by content distribution networks such as Akamai and CloudFlare. These networks do not coexist well with TLS. Sites must give the CDN permission to impersonate them, either by giving away a private key or by authorizing the CDN to add them as alternate names on the CDN's own certificates [7]. This lulls users into a false sense of security by giving the appearance of end-to-end encryption. The easier alternative, chosen by major sites such as the New York Times and CNN, is not to provide transport security at all, leaving their users' privacy unprotected.

B. Secret Sharing and the One-Time Pad

First articulated by Frank Miller in 1882 [16], the one-time pad has long been recognized as a strong technique for achieving secrecy. It was among the ciphers examined by Claude Shannon in his early works on information theory [17], wherein he defined perfect secrecy and proved that the one-time pad (or "Vernam system") is perfectly secure. In the same work, he showed that achieving this property requires as many bits of keying material as there is information in the message itself. The distribution of this key is typically considered a primary disadvantage of the one-time pad system.

Despite this property, perfect secrecy continues to be a topic of interest. Shannon himself states, "Perfect secrecy systems have a place in the practical picture—they may be used... where the greatest importance is attached to complete secrecy." A significant development on this front took place in 1979 with the independent invention of secret sharing by Shamir [18] and Blakley [19]. Secret sharing, like a one-time pad, provides information-theoretically provable confidentiality, but it allows for parameterization of both the multiplicity m of shares generated for each secret and the threshold k of shares required to recover the secret. Blakley explores the similarities between one-time pads and secret sharing in a subsequent paper [20], proposing what he and colleague R. D. Dixon term the "courier mode" of secret sharing. In this approach, each share is carried by a different courier to its recipient, and the robustness of the system can be described in terms of its parameters. The maximum number of abnegations (lost couriers) which can be tolerated is $m - k$, and the maximum number of betrayals to the enemy before the secret is compromised is $k - 1$. Blakley's vision is that, one day, threshold schemes may be incorporated into network protocols to provide both privacy protection and resilience to loss.

C. Secure Multichannel Protocols

Blakley's idea was later formulated by Dolev et al. [8] as the problem of *perfectly secure message transmission*, in which a sender must relay a message privately and reliably over multiple wires. The original work focused on the degree of connectivity necessary to achieve this. Further results were found which deal with Byzantine adversaries using multicast channels [21], [22] or minimize the number of exchanges required [23] or total data transmitted [24]. These works model networks as graphs, abstracting away performance characteristics such as throughput and latency.

In the meantime, recent efforts from researchers and the IETF have made multipath protocols a practical reality. These protocols use several channels simultaneously, with the goal of increasing performance and resiliency. Prominent among current research and standardization efforts is the Multi-Path TCP (MPTCP) transport protocol [25], which is already seeing widespread use in mobile devices [26] and large data centers [27]. These protocols exploit the *multiplicity* of the channels themselves—the fact that there is more than one way to transmit data between hosts—to improve communication.

There is, however, little work or analysis at the intersection of these two areas, joining the ideas of perfectly secure message transmission with the pragmatism of multipath protocols. The MICSS protocol [9], for example, provides a first step, but it is limited in terms of its flexibility. If perfect secrecy is to find a place in real networks, we must understand not only its theoretical properties but how it affects quantities such as loss, delay, and throughput.

III. Definitions and Model

If we are to understand the overall behavior of a multichannel secret sharing system, we must model each of the pieces involved in its operation. We begin by expressing a threat model in terms of risk metrics for each individual channel, then define the overall privacy and performance properties for which we derive optimal values. After establishing this context, we model the system itself. There are two major parts to this, which we will first examine independently: the set of channels which exist between endpoints, and the secret sharing scheme used to protect the data. Finally, we use all of the above definitions to model the protocol which ties them together.

A. Overall Threat and Performance Model

The purpose of a secret sharing protocol, like any mechanism for confidential communication, is to transmit data from sender to receiver in the presence of a particular adversary. At a high level, the sender gives the protocol a sequence of source symbols $x_1 x_2 x_3 \cdots$, and the protocol conveys this sequence to the receiver by transmitting share symbols y over various channels. However, there is an adversary present who may have the ability to eavesdrop on one or more of these channels. We represent the adversarial presence as a vector $\vec{z}$ of risk metrics estimated using network risk assessment techniques (e.g. [28], [29]). Each component z_i represents the likelihood that the adversary can observe a share symbol as it is being sent on channel i.

We are interested in four specific network properties which are measured or estimated for each invididual channel in the set C, and which we want to determine for the overall protocol:

- **Privacy/Risk.** As described above, the likelihood of an adversary observing any given share on each channel is modeled by the risk vector $\vec{z}$. We will derive an overall risk metric Z_C. Closely related to our threat model, this property represents the likelihood that the adversary can learn an arbitrary source symbol by observing its corresponding share symbols, given the individual z_i for each of the channels. The likelihood that any given data is communicated confidentially is then $1 - Z_C$.
- **Loss.** The lossiness vector $\vec{l}$ gives the probability for each channel that a given share symbol does not reach the receiver. Even if lost, the share may still have been observed by an attacker. We will derive the overall lossiness L_C, that is, the probability that the number of shares of a given source symbol that reach the receiver is insufficient to reconstruct that symbol.
- **Delay.** The delay vector $\vec{d}$ gives the expected amount of time for each channel that elapses from the transmission of a share symbol to its arrival at the receiver, assuming it is not lost. This is the total network delay, i.e., half of the round-trip time for the channel. We will derive the overall delay D_C, which is the expected amount of time from the sending of a source symbol to its reconstruction at the receiver. This will be affected by share loss as well.
- **Rate.** The rate vector $\vec{r}$ gives the maximum number of share symbols which can be sent on each channel in one unit time. We will derive the overall achievable rate R_C, which is the number of *source* symbols which can be sent in one unit time over the entire channel set C. This is the raw rate, not successful throughput, so it is independent of losses on the channel.

Ideally, Z_C, L_C, and D_C will be as low as possible and R_C will be as high as possible, but these properties cannot be fully optimized simultaneously. Instead, we investigate how tunable protocol parameters affect the balance among them, so that these parameters can be chosen and adjusted accordingly.

B. Channel Set

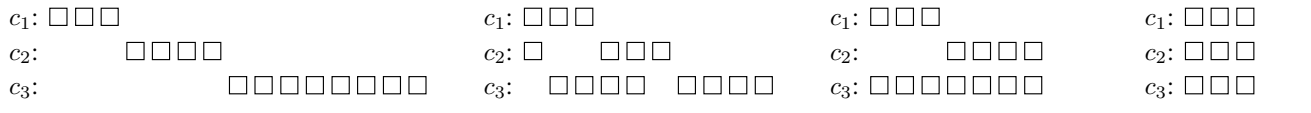
One significant part of a secret sharing protocol is the set of channels between the communicating hosts. Each channel is a distinct means of transferring data from one host to the other. In the simplest case, this is a set of some n identical channels, but the channels available in a realistic situation will typically include some amount of diversity. We choose to assume the latter for our model, as it is both the more practical and the more general case. As we will see in Section IV, this introduces more complexity in deriving optimality results, particularly for the rate at which data can be sent.

We model the available network resources as a set C of channels, where the number of channels $n = |C|$, and for each channel $i \in C$ the quadruple

$$(z_i, l_i, d_i, r_i) \in [0, 1] \times [0, 1) \times [0, \infty) \times (0, \infty)$$

represents its individual properties, as defined in Section III-A. Note the specific ranges in this definition: the lossiness of a channel is strictly less than 1, and the rate is strictly greater than 0. In other words, any channel which has zero probability of successfully transmitting shares is excluded from the set C.

For the purpose of this work, we assume that all of the channels in use are disjoint. This of course will vary from network to network, but it is a fitting assumption here because we are interested in characterizing optimal behavior. If two channels overlap, the bottleneck may reduce their combined throughput, and queueing and congestion may increase loss and delay. In terms of privacy, an attacker who is able to eavesdrop at a shared edge or vertex obtains data from multiple channels with the same effort required to eavesdrop on a single channel, leading to reduced privacy as well. The optimal case for all four channel properties, therefore, is when the channels are completely disjoint.



(a) $\mu = 1$ and $R_C = 15$ (b) $\mu = 1.5$ and $R_C = 10$ (c) $\mu = 2$ and $R_C = 7$, c_3 underutilized (d) $\mu = 3$ and $R_C = 3$, c_2 and c_3 underutilized

Fig. 2: Choosing M over one unit time to maximize rate with $\vec{r} = (3, 4, 8)$

C. Secret Sharing Scheme and Protocol

The other significant component of a secret sharing protocol is the secret sharing scheme used to provide confidentiality and reliability. In this work, we deal specifically with the original *threshold schemes* as created by Shamir and Blakley. These schemes have two integer parameters: the number of shares generated (or *multiplicity*) m, and the threshold k, such that $1 \leq k \leq m$. Recall that each share must carry at least as many bits of information as the secret itself; that is, $H(Y) \geq H(X)$, where H is Shannon's entropy function. Since $H(Y) = H(X)$ in the optimal case, our model can express rate plainly in terms of symbols per unit time rather than distinguishing between source and share symbols, as there is no need to convert between the two.

We can now use the above definitions to model the protocol which connects and coordinates these two components. This protocol communicates a sequence of source symbols from the sender to the receiver by transmitting shares on the individual channels of C in such a way as to provide additional privacy, reliability, performance, or some combination thereof. Using a secret sharing scheme, it generates m shares of a symbol, k of which are required to reconstruct that symbol. It then transmits each share on a *different* channel so that an adversary must compromise at least k channels to learn the symbol.

Sending multiple shares of the same symbol over the same channel should be avoided in this class of protocols. An adversary who compromises this channel can intercept multiple shares, effectively reducing the value of k by the number of overlapping shares. It is therefore useless to consider values of m which are greater than the number of channels, leading to the overall ordering

$$1 \leq k \leq m \leq n$$

for the parameters of the protocol. The relationship between these parameters defines the balance between privacy, reliability, and performance, with each successive pair corresponding to a different property. Privacy is characterized by $k - 1$, the number of share interceptions that can be tolerated without compromising the confidentiality of a symbol. Reliability is characterized by $m - k$, the number of share losses that can be tolerated without losing the symbol. Performance is related to $n - m$, the number of channels which can be used to send other symbols in parallel.

We now give a formal description of this protocol. To send a single source symbol $x \in X$ given integer parameters k and m, the protocol performs the following steps. First, it passes x to the secret sharing scheme to obtain a vector of shares $(y_1, \ldots, y_m)$. It then chooses a set of channels $M \subseteq C$ over which to send the shares, with $|M| = m$. Finally, it transmits each individual share over a different channel $i \in M$. The share is observed by an adversary with probability z_i, is lost with probability l_i, and if not lost takes d_i units time from transmission to receipt. On the receiving end, the protocol waits until any k of these shares $(y_{j_1}, \ldots, y_{j_k})$ have arrived successfully, then passes them to the secret sharing scheme to reconstruct the original x.

Unfortunately, with integers $1 \leq k \leq m \leq n$, the parameter space is very limited, especially given that practical values of n in some cases may be relatively small. Secret sharing schemes require that we use integral parameters k and m, but since we are sending many symbols, we can allow these parameters to vary from symbol to symbol so that the *average* threshold and multiplicity over many symbols (which we will call κ and μ) may be real numbers rather than integers. This permits operation anywhere on the available continuum of privacy/performance tradeoffs.

In order to maximize the overall rate—the number of source symbols which can be sent in one unit time—it is important to choose M intelligently. If the capacity of too many channels is exhausted quickly, the protocol cannot send any more symbols in the current unit time. It therefore needs to balance shares between channels so that it can send as many source symbols as possible before this occurs. Figure 2 illustrates this process, with rows representing channels, columns representing the choice of M for each successive source symbol, and boxes representing transmitted shares. Note that as μ increases, the number of source symbols transmitted decreases, and above a certain value it is not possible to utilize the full rate of every channel (a result proven later in Theorem 2).

This gives two decisions that the protocol must make for each symbol: a threshold k and a channel subset M. It must hold that $1 \leq k \leq m \leq n$ for each symbol, so not every combination of these parameters is valid. We define the set

$$\mathcal{M} = \left\{(k, M) \in \mathbb{N} \times \mathcal{P}(C) : 1 \leq k \leq |M|\right\}$$

to represent the acceptable combinations of k and M. Let a *share schedule* be a categorical distribution over this set of choices, defined by its probability mass function

$$p(k, M) : \mathcal{M} \to [0, 1].$$

The value of $p(k, M)$ in a given schedule is the proportion of symbols for which the parameters k and M are to be used.

We can express the resulting values of κ and μ as follows:

$$\kappa = \sum_{(k,M)\in\mathcal{M}} p(k,M)k$$
$$\mu = \sum_{(k,M)\in\mathcal{M}} p(k,M)|M|.$$

Using this construction we can define protocol parameters more precisely as a share schedule p rather than as separate k and m values. The sender provides a sequence of source symbols $x_1x_2x_3\ldots$ to the protocol. For each x, the protocol chooses an element $(k,M)\in\mathcal{M}$ independently according to the distribution given by p. It then continues with the single-symbol process to send x, using k as the threshold and M as the set of channels on which to send shares.

IV. Results

Here we use the mathematical model defined in the previous section to derive expressions for the optimal achievable privacy, loss, delay, and rate under different constraints. We first derive preliminary formulas describing the transmission of individual symbols, then use them to find values for overall properties under given sets of parameters. For each property, we give its best possible value under any choice of parameters, and its best possible value for given μ and κ. Finally, noting that fully optimizing privacy, loss, or delay can lead to significant underutilization of channel rate, we derive an expression for the maximum overall rate and show how to optimize other properties while maintaining this maximum rate.

A. Subset and Schedule Properties

To begin, we consider the properties of transmitting a single symbol as shares over a given set of channels; these results are needed to build later formulas. In this case, the protocol has already chosen k and M for a particular symbol, and we will find expressions for the expected privacy, loss, and delay when sending it. We refer to these values as the *subset* privacy, loss, and delay, as they are specific to the subset M of channels used. The expressions for these values follow logically from the protocol model, and we will then extend them to find the average over many symbols with parameters chosen according to a given share schedule.

First we examine the expected privacy of a source symbol for specified k and M. Recall that exactly one of the m shares generated for the symbol will be sent on each channel $i\in M$. If at least k of these shares are known, the source symbol can be determined; if fewer than k are known, then no information about the source symbol is known. So the expected risk metric for a source symbol is the likelihood that an adversary observes at least k of its shares. Since we assume disjoint channels, this is a set of independent trials. The probability that an adversary observes any given share on channel i is z_i, so the subset risk is the cdf of the corresponding Poisson binomial distribution,

$$z(k,M) = \sum_{\substack{K\subseteq M;\\ |K|\geq k}} \prod_{i\in K} z_i \prod_{j\in M\setminus K} (1-z_j).$$

The expression for subset loss is similar, as loss on disjoint channels is also a set of independent trials, with probability l_i for each channel i. For a source symbol to be lost, the number of shares successfully received at the destination must be fewer than k, so the subset loss is

$$l(k,M) = \sum_{\substack{K\subseteq M;\\ |K|<k}} \prod_{i\in K} (1-l_i) \prod_{j\in M\setminus K} l_j.$$

Deriving an expression for subset delay is complicated by the fact that the delay in transmitting a symbol can be affected by loss of some of its shares. It is easy to understand that, in the absence of loss, the delay is the kth smallest delay among the channels of M, since the receiver can reconstruct the source symbol as soon as the k fastest shares have arrived. In other words, if for any set of channels S we let δ_S be a nondecreasing ordering of the delays of its channels, then

$$d(k,M) = \delta_M(k)$$

when no shares are lost. When we do account for loss, we must exclude lost symbols from the calculation in order to obtain a useful result, and so the expression is heavily influenced by the equation for subset loss:

$$d(k,M) = \frac{1}{1-l(k,M)} \sum_{\substack{K\subseteq M;\\ |K|\geq k}} \delta_K(k) \prod_{i\in K} (1-l_i) \prod_{j\in M\setminus K} l_j.$$

Intuitively, this is a weighted average of the lossless delays of each subset K of M which could successfully convey a symbol (i.e., $|K|\geq k$), with each term weighted by the likelihood that K is exactly the set of channels on which the transmitted share is not lost. As expected, when all $l_i = 0$, this equation collapses to $\delta_M(k)$.

We can use the subset formulas for these network properties to find an expression for the average privacy, loss, and delay over a large number of symbols. These values will depend on how frequently each pair of k and M is chosen, and so this calculation will be specific to a share schedule. Given a share schedule, we can then calculate the *schedule* privacy as a weighted average:

$$Z(p) = \sum_{(k,M)\in\mathcal{M}} p(k,M)z(k,M)$$

and similarly with $l(k,M)$ and $d(k,M)$ for $L(p)$ and $D(p)$.

B. Optimal Privacy, Loss, and Delay

If the parameters κ and μ can be chosen freely, it is simple to completely optimize privacy, loss, or delay without regard to the other properties. The result for each property is the highest possible value it can achieve over channel set C. Typically these will be balanced with other properties (rate in particular), but there can always be scenarios where one property so outweighs the others that, relatively, they do not matter.

We begin with fully maximizing privacy. To do this, we must force the adversary to eavesdrop as many shares as possible for any symbol. This is the case when κ is as high as it can

be, namely $\kappa = \mu = n$, resulting in the share schedule with $p(n, C) = 1$. The overall risk metric Z_C is then

$$Z_C = Z(p) = z(n, C) = \prod_{i \in C} z_i.$$

To fully minimize loss, we need to add as much redundancy as possible during the sending of a symbol. This is the case when κ and μ are as far apart as they can be, namely $\kappa = 1$ and $\mu = n$. This results in the share schedule with $p(1, C) = 1$ and an overall lossiness of

$$L_C = L(p) = l(1, C) = \prod_{i \in C} l_i.$$

Delay, as explained in the previous section, is more complex in that it is affected by loss. To fully minimize delay, we will obviously choose $\kappa = 1$ since $d(1, M) \leq d(k, M)$ for any valid k and M. If there were no loss, we would only need to ensure that the channel with the smallest delay was always included in the set M, resulting in

$$D_C = \min_{i \in C} d_i$$

for this case. However, with the possibility of loss, it becomes important to set $\mu = n$ to include all of the channels. If the share with the smallest delay is lost, then we want to be sure that the channel with the second-smallest delay is in M, and so forth if that share is also lost. This leads to the following average delay:

$$D_C = \frac{1}{1 - \prod_{i \in C} l_i} \sum_{a=1}^{n} (1 - \lambda_M(a)) \delta_M(a) \prod_{b=1}^{a-1} \lambda_M(b).$$

where $\lambda_M(i)$ is the lossiness of the channel to which $\delta_M(i)$ refers. In other words, the delay is the average of the delays of the various channels, each weighted by the probability that a share will arrive on that channel but not on any channel with lower delay. As with the subset delay equation, this collapses to the lossless equation when all $l_i = 0$.

To balance these properties against one another, we may choose parameters κ and μ anywhere between 1 and n. Given these parameters, a share schedule which fully optimizes privacy, loss, or delay can be found via linear programming. The following linear program, for instance, fully optimizes privacy within the parameters κ and μ by finding values for each $p(k, M)$:

$$\begin{aligned}
&\text{Minimize } Z(p) \\
&\text{subject to } p(k, M) \geq 0 && (k, M) \in \mathcal{M}, \\
&\qquad \sum_{(k,M) \in \mathcal{M}} p(k, M) = 1, \\
&\qquad \sum_{(k,M) \in \mathcal{M}} p(k, M)(k - \kappa) = 0, \\
&\qquad \text{and } \sum_{(k,M) \in \mathcal{M}} p(k, M)(|M| - \mu) = 0.
\end{aligned}$$

This is a valid linear program, as all of the constraints are linear in p with pre-calculatable coefficients. The first two constraints ensure that p defines a valid categorical distribution over $\mathcal{M}$, and the last two ensure that the average k is κ and the average size of M is μ. The same program can be used to optimize loss or delay simply by substituting $L(p)$ or $D(p)$ for the objective function.

C. Optimal Rate

As with privacy, loss, and delay, optimizing the rate without regard for other properties is straightforward when the parameters κ and μ can be chosen freely. Since at least one share must be sent for each source symbol, we can maximize the number of source symbols sent if we limit the number of shares per symbol to exactly one by setting $\kappa = \mu = 1$. The rate, which is the total number of shares that can be sent per unit time, is

$$R_C = \sum_{i \in C} r_i.$$

This is the ideal behavior for throughput-maximizing protocols like MPTCP, and it is achieved with a share schedule which assigns shares to each channel according to what proportion of the total rate it represents:

$$p(k, M) = \begin{cases} r_i / R_C & \text{if } k = 1 \text{ and } M = \{i\} \\ 0 & \text{otherwise.} \end{cases}$$

Determining the optimal multichannel rate for a chosen κ and μ is more complex. Given that we must achieve an average share multiplicity of μ and may not transmit more than r_i shares on channel i in one unit time, we wish to find an expression for the overall multichannel rate R_C. By our definition, R_C is the maximum number of source symbols which can be sent over the channels in C per unit time. Recall that for each source symbol, the protocol chooses an $M \subseteq C$ and transmits one share on each channel in M. Channel i can be an element of this M at most r_i times over one unit time, so the protocol must choose its channels strategically if it is to achieve the maximum rate.

Let r'_i be the actual number of shares which can be sent on channel i using an optimal scheduling strategy. One way to express R_C, then, is

$$R_C = \frac{1}{\mu} \sum_{i \in C} r'_i, \tag{1}$$

since μ is the average ratio of shares to source symbols. There are two constraints governing the values of r'_i. First, the number of shares sent on each channel obviously cannot exceed the rate r_i of that channel. Second, since the protocol is not permitted to send more than one share of any given source symbol on any given channel, the number of shares which can be transmitted on any channel is at most R_C. So the number of shares to send on each channel is constrained by the following inequalities:

$$r'_i \leq r_i \qquad \text{all } i \in C \tag{2}$$

$$r'_i \leq R_C \qquad \text{all } i \in C, \tag{3}$$

where R_C is as expressed in Equation 1. Given these constraints, we can find a lower bound on the achievable multichannel rate.

Theorem 1. *The achievable multichannel rate is at least that of the channel with the $\lceil \mu \rceil$th-highest individual rate.*

Proof. Let S be a set of channels with the $\lceil \mu \rceil$ highest rates, i.e., $S \subseteq C$ with $|S| = \lceil \mu \rceil$ and

$$\min_{i \in S} r_i \geq \max_{j \in C \setminus S} r_j.$$

Equation 2 is clearly satisfied when

$$r'_i = \begin{cases} \min_{i \in S} r_i & \text{for } i \in S \\ 0 & \text{otherwise,} \end{cases}$$

and the total rate

$$\frac{1}{\mu} \sum_{i \in C} r'_i = \frac{\lceil \mu \rceil}{\mu} \min_{i \in S} r_i$$

is no smaller than any r'_i, satisfying Equation 3. Therefore

$$R_C \geq \frac{\lceil \mu \rceil}{\mu} \min_{i \in S} r_i \geq \min_{i \in S} r_i. \qquad \square$$

One implication of these constraints is that the channels cannot always be fully utilized if they have different rates. If we are interested in full utilization, we can determine the range of choices for μ which will allow it for a given set of channels.

Theorem 2. *Full utilization of every channel is possible if and only if the average share multiplicity is at most the ratio of total available rate to that of the fastest channel.*

Proof. If all of the channels are fully utilized, then $r'_i = r_i$ for all $i \in C$. This assignment will always satisfy Equation 2, so it only remains to show under what conditions it also satisfies Equation 3. If

$$r_j \leq \frac{1}{\mu} \sum_{i \in C} r_i$$

for all $j \in C$, then it is equivalent to say that

$$\max_{j \in C} r_j \leq \frac{1}{\mu} \sum_{i \in C} r_i$$

and, solving for μ,

$$\mu \leq \frac{\sum_{i \in C} r_i}{\max_{j \in C} r_j}. \qquad \square$$

Corollary 1. *A set of channels with identical rates can be fully utilized for any valid μ.*

Proof. If all r_i are equal, then

$$\frac{\sum_{i \in C} r_i}{\max_{j \in C} r_j} = n$$

and $\mu \leq n$ in any valid set of parameters. $\square$

It will be useful in later theorems to define a set to distinguish between those channels which are fully utilized and those which are limited by Equation 3.

Definition 1. The *fully-utilized set* is the set

$$A = \{i \in C : r_i \leq R_C\}$$

of all channels with rates which can be fully utilized under given protocol parameters.

Corollary 2. *The size of the fully-utilized set is greater than $n - \mu$.*

Proof. Since by Theorem 1, R_C is at least the rate of the $\lceil \mu \rceil$th fastest channel,

$$|A| \geq n - \lceil \mu \rceil + 1.$$

Equivalently, since $|A|$ and n are integers,

$$|A| > n - \mu. \qquad \square$$

Theorem 3. *The relation between the average share multiplicity μ and the optimal multichannel rate is*

$$\mu = \sum_{i \in C} \min \left\{ \frac{r_i}{R_C}, 1 \right\}.$$

Proof. We wish to find the maximum R_C for which the previously mentioned three constraints are satisfiable. From Equation 1 we note that R_C is proportional to the sum of all r'_i and is therefore maximized when each r'_i is as large as possible. Given the two inequalities, we can say that this is the case when

$$r'_i = \min\{r_i, R_C\} \qquad \text{all } i \in C. \tag{4}$$

To eliminate r'_i, we substitute this into Equation 1:

$$\sum_{i \in C} \min\{r_i, R_C\} = \mu R_C,$$

and dividing through by R_C we have

$$\mu = \sum_{i \in C} \min \left\{ \frac{r_i}{R_C}, 1 \right\}. \qquad \square$$

This is, on its own, a useful result. If we have a target multichannel rate and wish to balance this with other properties, we can use this formula to find the highest value of μ for which the overall rate is at least the target. However, we also wish to derive an expression for the reverse: the rate achievable for a specified μ. To set this up, we will use the set A to differentiate between the two cases in the min-expression.

Theorem 4. *The optimal multichannel rate for multiplicity μ and channel rates r_i is given by*

$$R_C = \min_{\substack{S \subseteq C; \\ |S| > n - \mu}} \frac{\sum_{i \in S} r_i}{\mu - n + |S|}.$$

Proof. The result from Theorem 3 can be written using Definition 1 as follows:

$$\begin{aligned} \mu &= \sum_{j \in C \setminus A} 1 + \sum_{i \in A} \frac{r_i}{R_C} \\ &= n - |A| + \frac{1}{R_C} \sum_{i \in A} r_i. \end{aligned}$$

Solving for R_C,

$$R_C = \frac{\sum_{i \in A} r_i}{\mu - n + |A|}. \tag{5}$$

The set A needs to be eliminated, as it is defined in terms of R_C. This can be accomplished by recognizing that A is the unique set of more than $n - \mu$ channels which minimizes the expression. In other words, we will show that, for any set $S \subseteq C$ with $|S| > n - \mu$,

$$R_C \leq \frac{\sum_{i \in S} r_i}{\mu - n + |S|}.$$

Starting with some simple set identities, we have

$$\sum_{i \in S} r_i = \sum_{j \in A} r_j + \sum_{g \in S \setminus A} r_g - \sum_{h \in A \setminus S} r_h,$$

and substituting from Equation 5,

$$\sum_{i \in S} r_i = (\mu - n + |A|)R_C + \sum_{g \in S \setminus A} r_g - \sum_{h \in A \setminus S} r_h.$$

Noting that $r_g > R_C$ for all $g \notin A$ and $r_h \leq R_C$ for all $h \in A$,

$$\begin{aligned}\sum_{i \in S} r_i &\geq (\mu - n + |A|)R_C + \sum_{g \in S \setminus A} R_C - \sum_{h \in A \setminus S} R_C \\ &= (\mu - n + |A| + |S \setminus A| - |A \setminus S|)R_C \\ &= (\mu - n + |S|)R_C\end{aligned}$$

and rearranging,

$$R_C \leq \frac{\sum_{i \in S} r_i}{\mu - n + |S|}.$$

Therefore,

$$R_C = \min_{\substack{S \subseteq C; \\ |S| > n - \mu}} \frac{\sum_{i \in S} r_i}{\mu - n + |S|}. \qquad \square$$

D. Optimal Privacy, Loss, and Delay at Optimal Rate

Optimizing strictly for privacy, loss, or delay, even with a pre-selected κ and μ, will usually force the protocol to transmit at a rate significantly lower than R_C. The linear program from Section IV-B often finds a single (k, M) which yields the best value and transmits using only those parameters until it has exhausted the rate of the slowest channel in M. Any channels not in this "best" M are left completely unused. This is typically not desirable behavior.

It would be more useful to know, for given parameters κ and μ, how to compute and achieve the best privacy, loss, or delay *while maintaining maximum rate*. This would ensure that the protocol is making full use of the available network resources, while still judiciously choosing its share schedule to optimize whatever property is deemed important. We can modify the earlier linear program to suit this purpose, but we will need a way to express the maximum-rate constraint as an equation linear in the values of p. Recall from Equation 4 that overall rate is maximized when the number of source symbols for which channel i carries a share is the lesser of its rate and the overall rate. To convert this to an expression using p, we note that the proportion of symbols using channel i is

$$\sum_{\substack{M \subseteq C; \\ i \in M}} p(k, M) = \frac{r'_i}{R_C} = \min\left\{\frac{r_i}{R_C}, 1\right\}.$$

This constraint ensures that the share schedule p can achieve the maximum rate R_C. Since Theorem 4 allows us to calculate R_C, we may use it as a constant in the new linear program:

$$\begin{aligned}&\text{Minimize } Z(p) \\ &\text{subject to } p(k, M) \geq 0 && (k, M) \in \mathcal{M}, \\ &\qquad \sum_{(k,M) \in \mathcal{M}} p(k, M) = 1, \\ &\qquad \sum_{(k,M) \in \mathcal{M}} p(k, M)(k - \kappa) = 0, \\ &\qquad \sum_{(k,M) \in \mathcal{M}} p(k, M)(|M| - \mu) = 0, \\ &\qquad \text{and } \sum_{\substack{(k,M) \in \mathcal{M}; \\ i \in M}} p(k, M) = \min\left\{\frac{r_i}{R_C}, 1\right\} \quad i \in C.\end{aligned}$$

With the added constraint, this program now finds a valid share schedule to optimize privacy, given a specified κ and μ, and ensuring that the overall rate is R_C. As in Section IV-B, loss and delay can be optimized by changing the objective function to $L(p)$ and $D(p)$ respectively.

E. Accommodating Previous Multichannel Threat Models

The MICSS work [9], as well as Blakley's paper on the courier mode of secret sharing [20], do not model their adversaries using probabilistic risk metrics. Rather, they simply assume that the adversary can always eavesdrop on a fixed set of channels. In those works, a constant, integral parameter k is used for all symbols, and if the adversary cannot compromise at least k channels, then no information is disclosed. Our approach, which allows for an average threshold κ, is not directly suited to the MICSS threat model. For example, if the adversary compromises two channels and $\kappa = 3$, there may or may not be information disclosure, depending on the individual k chosen from the share schedule.

Our approach can be modified in a straightforward fashion to cater to both threat models. To do this, we limit the potential share schedules to those in which only elements of the set

$$\mathcal{M}' = \left\{(k, M) \in \mathcal{M} : k \geq \lfloor \kappa \rfloor,\ |M| \geq \lfloor \mu \rfloor\right\}$$

have a nonzero probability of being chosen. By doing this, we ensure that $k \geq \lfloor \kappa \rfloor$ for every symbol, forcing the adversary to be able to compromise $\lfloor \kappa \rfloor$ channels simultaneously to learn any given source symbol. The following theorem states that limiting the share schedule in this way does not exclude any combination of κ and μ; the proof, which is a straightforward construction, is omitted for space considerations.

Theorem 5. *For any κ and μ such that $1 \leq \kappa \leq \mu \leq n$, there exists a valid share schedule p' over $\mathcal{M}'$ with average threshold $\kappa' = \kappa$ and average multiplicity $\mu' = \mu$.*

While we hoped to find that this limitation preserved the optimal values of each network property, it became apparent that this was not the case. The optimal rate does remain the same, since by Theorem 4 it depends only on the average μ and not the individual choices of k and M. Other properties, however, can have optimal values which are not achievable with a limited share schedule. Consider the case in which there are three channels with negligible loss and $\vec{d} = (2, 9, 10)$, for parameters $\kappa = 2$ and $\mu = 3$. The only possible limited share schedule is the one in which $p(2, C) = 1$, and the average delay $d(2, C) = 9$. However, this is not equal to the optimal delay achievable with a non-limited share schedule, since choosing $(1, C)$ for half of the symbols and $(3, C)$ for the other half gives the same κ and μ but with a lower average delay of 6. Similar examples can be found for privacy and loss as well.

V. Reference Protocol

In order to demonstrate the predictive abilities of our model and results, we need a usable secret sharing protocol with which to test them. Unfortunately, the assumptions of MICSS [9] are not general enough. Using perfect secret sharing schemes instead of threshold schemes leads to a simplified design which does not make sense in a more general secret sharing protocol. Essentially, MICSS only provides one configuration of κ and μ for any given channels, namely that where $\kappa = \mu = n$.

We therefore create a new protocol to serve as a reference implementation of our model of parameterized secret sharing protocols. Our goal is not to provide a perfectly tuned implementation, but to provide a means to evaluate the usefulness of our model. We call this protocol ReMICSS, as it is a significantly redesigned protocol yet still based on the concepts of MICSS. ReMICSS does not force maximum privacy, allowing instead for the selection of parameters κ and μ to define how the protocol will operate. In this section, we discuss the design of ReMICSS and highlight how it differs from MICSS, and in Section VI we will examine the behavior of this protocol over a variety of choices of κ and μ.

The fundamental difference in ReMICSS is the addition of support for threshold schemes, although this affects several other aspects of its design and implementation. One obvious benefit is the ability to lose $m - k$ shares of a packet without the need for retransmission. However, reliable share transport such as that of MICSS forces all lost shares to be retransmitted regardless, stalling the channel and wasting network resources when $k < m$. In order to benefit from threshold scheme support, ReMICSS is designed as a best-effort protocol, and instead of intercepting TCP connections, it uses the network-layer DIBS architecture [30] to flexibly and transparently intercept IP traffic. As a further benefit of changing the network semantics, ReMICSS is transport-agnostic, able to carry any IP-based communication and not only TCP.

These changes also affect the sending and receiving of shares. Without reliable share transport, ReMICSS cannot assume that one packet will be reconstructed before shares of the next begin to arrive. Due to loss, reordering, or differing channel rates, the receiver will typically be waiting for shares of many packets at once and needs to store the received shares intelligently. We borrow ideas from IP fragment reassembly algorithms, evicting shares after a set timeout and limiting the total amount of memory used, to provide time for slower shares to arrive without blocking new shares or sacrificing throughput.

On the sending side, the introduction of parameters κ and μ requires that the sender somehow choose an appropriate share schedule. In ReMICSS, to avoid the complexity of computing an explicit schedule, we implement a *dynamic share schedule*. Instead of deciding M ahead of time, the sender chooses the first m channels which are ready for writing (on Linux, we use the `epoll` mechanism). Our evaluation will show how this simplification affects different aspects of protocol performance.

VI. Empirical Evaluation

We conduct a set of experiments to demonstrate our model's usefulness in describing the performance characteristics of real multichannel secret sharing protocols, as well as to evaluate the network behavior of our reference implementation. These experiments consist of network microbenchmarks targeting rate, loss, and delay between two endpoints connected by five controlled network channels. Each experiment demonstrates how the κ and μ parameters of the protocol affect a specific network property on a given setup, and we compare these real-world results to the optimal values projected by our model. Experiments are carried out on one or more of the following pre-defined network setups, designed specifically to illustrate the effects of each property:

- **Identical.** All five channels are configured to transmit at a given rate between 100 Mbps and 800 Mbps, with negligible loss and delay.
- **Diverse.** The five channels are configured to transmit at 5, 20, 60, 65, and 100 Mbps, with negligible loss and delay.
- **Lossy.** The five channels are again configured as in the Diverse setup, but with loss of 1, 0.5, 1, 2, and 3 percent respectively in each direction.
- **Delayed.** The five channels are configured as in the Diverse setup, but with an added delay of 2.5, 0.25, 12.5, 5, and 0.5 ms respectively in each direction.

The two hosts used in these experiments are Dell Precision T7600 desktop workstations with 2.3 GHz hex-core Xeon processsors and 32 GB of memory, running Arch Linux with kernel 4.1.6. Each system is equipped with five 10 Gbps network interfaces, and each channel is a direct wired connection between two of these interfaces which is dedicated solely to experimental traffic. Channel properties are controlled using built-in features of the Linux network infrastructure, with the Hierarchical Token Bucket (htb) queueing class used to limit transmission rate and the Network Emulator (netem) queueing discipline used to introduce loss and delay.

A. Rate on Identical and Diverse Channels

In our first experiment, we measure the transmission rate achieved by ReMICSS over its parameter space, and we compare the results to the optimal values from our model. To carry out this test, we use the iperf network benchmarking

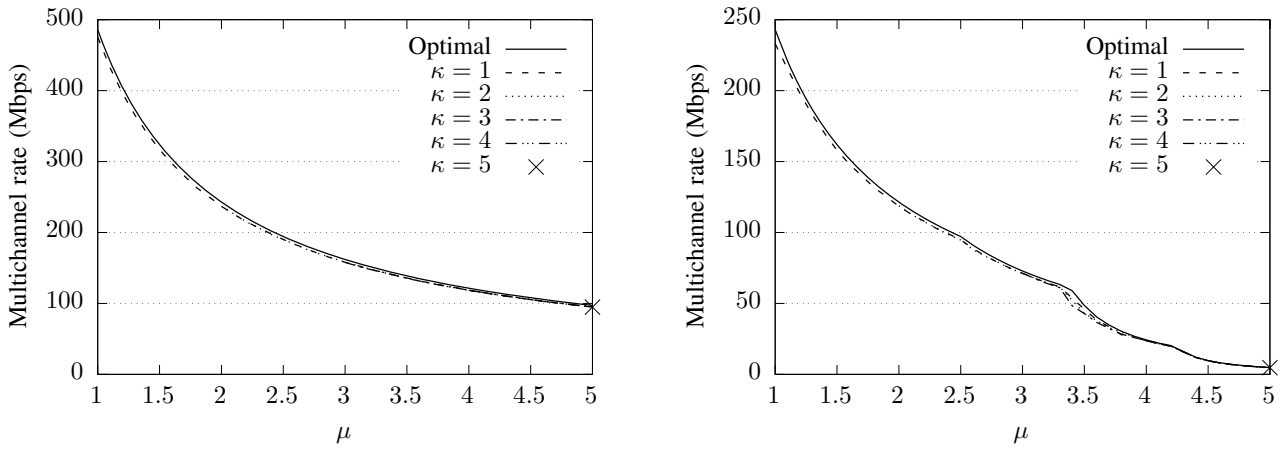


Fig. 3: Optimal and actual rate over κ and μ on 100 Mbps Identical setup (left) and Diverse setup (right)

tool to generate UDP traffic for one minute at 1000 Mbps, recording the bitrate reported by the iperf receiver. We begin by using this method to obtain an accurate rate for each individual channel, which gives us the vector $\vec{r}$ to use in calculating the optimal rate at each point. After this, we measure the protocol's transmission rate for various combinations of κ and μ.

The results of this experiment for the 100 Mbps Identical setup and the Diverse setup are shown in Figure 3. For each κ, the rate is measured at values of μ ranging from κ to 5 in steps of 0.1. For the Identical setup, the achieved rate followed the optimal predictions closely, with overhead of no more than 3% at any point. By Corollary 1, every channel is fully utilized at all values of μ, and this is evident from the smoothly curving graph for this setup. In contrast, the graph of the results for the Diverse experimental setup is bumpy. Each bump indicates a new channel which can no longer be fully utilized at higher values of μ. For this setup, aside from slightly anomalous behavior in the vicinity of $\mu = 3.4$, the rate achieved by the reference implementation in our experiments was consistently within 4% of optimal.

B. Loss and Delay at Maximum Rate

Since our model also provides a means of calculating the optimal loss and delay which can be achieved while maintaining maximum rate, we evaluate how the reference implementation compares in this regard as well. We do not anticipate that its performance will be as near to optimal for these properties as it was for rate, due to the dynamic share schedule approach described in Section V. Instead, we will use our optimality results to determine how well the simpler scheduling approach used in ReMICSS deals with loss and delay.

We will first evaluate performance in terms of overall packet loss at maximum rate, given lossy underlying channels. In this experiment, we continue to use the iperf tool, which in addition to rate reports the percentage of datagrams lost for UDP benchmarks. For each choice of parameters κ and μ, we direct iperf to generate 30 seconds of UDP traffic at the rate measured in the previous experiment. In these tests we configure our network channels in the Lossy experimental setup, so that there may now be loss of individual shares on different channels, potentially leading to the loss of the symbol depending on the current protocol parameters.

The results of the loss experiment are shown in Figure 5. As before, we measure the lossiness of each individual channel first to ensure that we have an accurate $\vec{l}$ with which to calculate optimal values. These optimal predictions, represented by the solid lines in the figure, are computed by solving the linear program in Section IV-D. We can see a variety of behavior in the results, which stems from the way in which ReMICSS chooses its share schedule. In some cases, such as when κ is 2, 4, or 5, the actual loss is extremely close to optimal. At other points, such as the pathological case where $\kappa = 3$ and $\mu = 3.8$, the procedure for selecting channels interacts negatively with the specific proportions of channel properties, leading to much higher loss than other nearby values. It is still clear to see from other parts of the graph that the predictions from our model are reinforced by the results of this experiment, and that oddities in the graph are effects of the particular implementation.

We evaluate delay similarly, although we must use a different tool because iperf does not give results for packet delay. We create a simple utility to determine average packet round-trip time for echoed UDP traffic which is generated at a specified rate. The client is based on the sending routine from iperf, but each packet includes a timestamp so that when it receives the echoed packet, it can determine its round-trip time. We run our client/server pair instead of iperf for 30 seconds, again at the rate determined in the first experiment. The client gives an average round-trip time for all of the packets sent, and since the channel delays are applied in both directions, we divide this result by 2 to find the one-way delay.

It is clear from our results that the reference implementation is much more heavily affected by delay than by loss. We plot optimal and actual delays separately in Figure 4 due to the difference in scale. It is interesting to note that each delay curve is actually well-behaved beyond a certain point. These

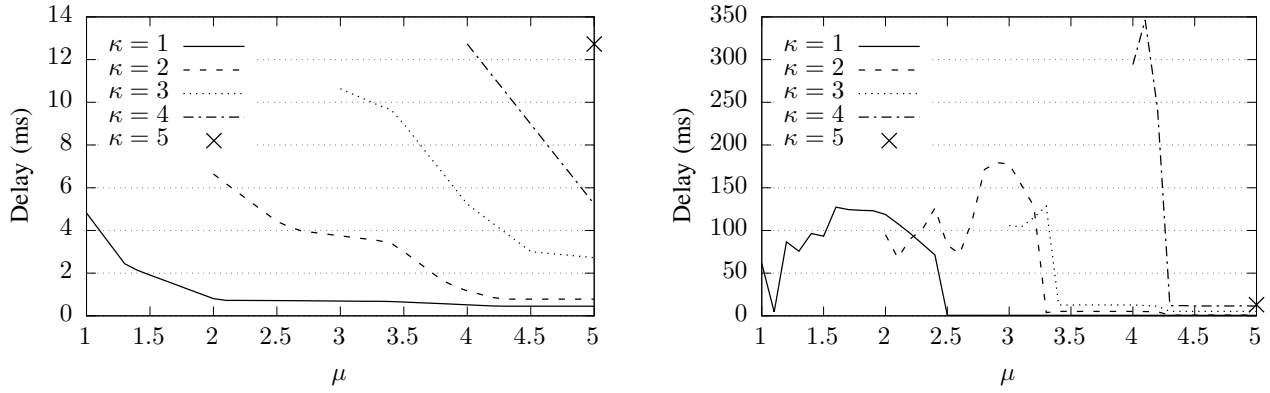


Fig. 4: Optimal (left) and actual (right) delay at maximum rate for Delayed setup

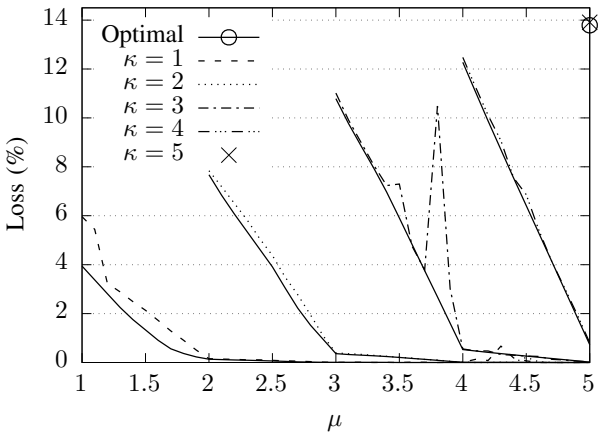


Fig. 5: Loss at maximum rate for Lossy setup

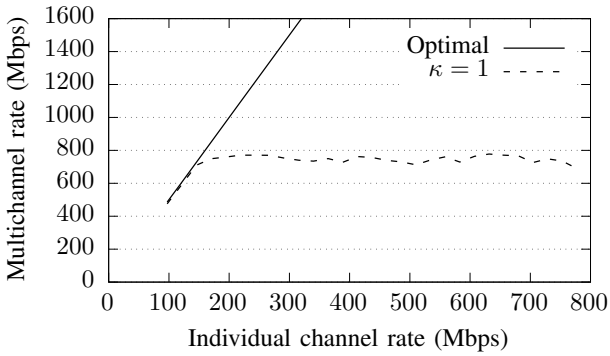


Fig. 6: Optimal and achieved rate for Identical setup with increasing channel rate and $\mu = 1$

points correspond exactly to the bumps in the rate curve, i.e., for a given κ, the effects of delay are problematic for this implementation only if there are fewer than κ underutilized channels. This seems to be a direct consequence of the simplified channel selection approach, since underutilized channels will almost always be available to send according to `epoll`, so the implementation will rarely need to wait for additional channels to become available.

C. High-Bandwidth Channels

Our final experiment is designed to push the limits of the system and implementation to see how long they are able to sustain near-optimal rates as the available bandwidth increases. In other words, we wish to see at what point the bottleneck becomes something other than the capacity of the channels. To accomplish this, we return to the Identical setup and gradually increase the channel rate from 100 Mbps to 800 Mbps in increments of 25 Mbps. The multichannel rate is measured at each point using iperf, as in the first experiment.

First we choose parameters $\kappa = \mu = 1$ so as to fully maximize the multichannel rate. As shown in Figure 6, this results in performance leveling off around 750 Mbps total, or when the individual channel capacity reaches approximately 150 Mbps. To complement this, we run another round of experiments with $\mu = 5$ and varying values of κ, for which the overall multichannel rate will be lower for the same total amount of traffic. Interestingly, even though the threshold affects the rate very little during normal operation, it makes a significant difference once the system and implementation are pushed to their limits, with large κ values causing the protocol to fall short of optimal much sooner than small values.

VII. Conclusion

It is important to understand the inherent privacy and performance tradeoffs of any approach to network privacy. In this work, we have presented a rigorous model of multi-channel secret sharing protocols in real networks, including a corresponding privacy measure. From this model, we have derived formulas and programs to calculate optimal privacy and performance metrics from the properties of individual channels and the parameters of the protocol. Benchmarks on a reference implementation demonstrated the utility of our model, reinforced the accuracy of our analysis, and showed that multichannel secret sharing protocols can be efficiently

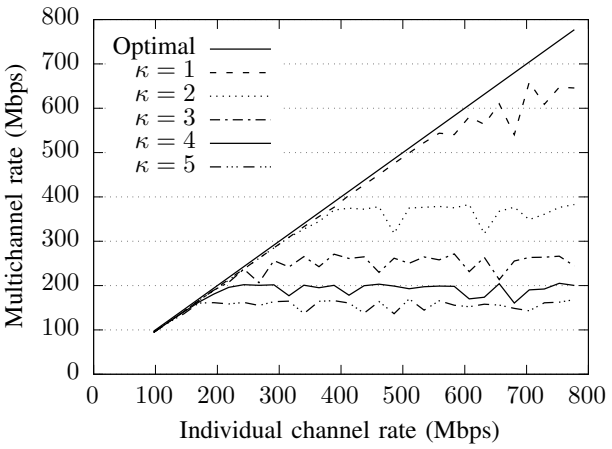


Fig. 7: Optimal and achieved rate for Identical setup with increasing channel rate and $\mu = 5$

implemented. Given these results, we can now make strong statements about the achievable privacy and performance of this class of network privacy protocols.

VIII. Acknowledgments

We wish to thank Tom La Porta for providing valuable feedback on our protocol model and Meghan Riegel for assisting us with preliminary experiments.

References

[1] J. Angwin, C. Savage, J. Larson, H. Moltke, L. Poitras, and J. Risen, "AT&T helped U.S. spy on Internet on a vast scale," Aug. 2015.

[2] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proc. 13th USENIX Security Symposium*, Aug. 2004.

[3] S. Mauw, J. H. Verschuren, and E. P. de Vink, "A formalization of anonymity and onion routing," in *Computer Security–ESORICS 2004*, pp. 109–124, Springer, 2004.

[4] P. Dhungel, M. Steiner, I. Rimac, V. Hilt, and K. W. Ross, "Waiting for anonymity: Understanding delays in the Tor overlay," in *Peer-to-Peer Computing (P2P), 2010 IEEE Tenth International Conference on*, pp. 1–4, IEEE, 2010.

[5] H. Corrigan-Gibbs and B. Ford, "Dissent: accountable anonymous group messaging," in *Proceedings of the 17th ACM conference on Computer and communications security*, pp. 340–350, ACM, 2010.

[6] D. I. Wolinsky, H. Corrigan-Gibbs, B. Ford, and A. Johnson, "Dissent in numbers: Making strong anonymity scale," in *OSDI*, pp. 179–182, 2012.

[7] J. Liang, J. Jiang, H. Duan, K. Li, T. Wan, and J. Wu, "When HTTPS meets CDN: A case of authentication in delegated service," in *Security and Privacy (SP), 2014 IEEE Symposium on*, pp. 67–82, IEEE, 2014.

[8] D. Dolev, C. Dwork, O. Waarts, and M. Yung, "Perfectly secure message transmission," *Journal of the ACM (JACM)*, vol. 40, no. 1, pp. 17–47, 1993.

[9] D. J. Pohly and P. McDaniel, "MICSS: A realistic multichannel secrecy protocol," in *Proceedings of the 2015 IEEE Global Communications Conference*, 2015.

[10] S. J. Murdoch and G. Danezis, "Low-cost traffic analysis of Tor," in *Security and Privacy, 2005 IEEE Symposium on*, pp. 183–195, IEEE, 2005.

[11] K. Bauer, J. Juen, N. Borisov, D. Grunwald, D. Sicker, and D. McCoy, "On the optimal path length for Tor," in *HotPets in conjunction with Tenth International Symposium on Privacy Enhancing Technologies (PETS 2010), Berlin, Germany*, 2010.

[12] R. Snader and N. Borisov, "A tune-up for Tor: Improving security and performance in the Tor network," in *NDSS*, vol. 8, p. 127, 2008.

[13] P. Dhungel, M. Steiner, I. Rimac, V. Hilt, and K. W. Ross, "Waiting for anonymity: Understanding delays in the Tor overlay," in *Peer-to-Peer Computing (P2P), 2010 IEEE Tenth International Conference on*, pp. 1–4, IEEE, 2010.

[14] J. Camenisch and A. Lysyanskaya, "A formal treatment of onion routing," in *Advances in cryptology–CRYPTO 2005*, pp. 169–187, Springer, 2005.

[15] M. K. Reiter and A. D. Rubin, "Crowds: Anonymity for web transactions," *ACM Transactions on Information and System Security (TISSEC)*, vol. 1, no. 1, pp. 66–92, 1998.

[16] S. M. Bellovin, "Frank Miller: Inventor of the one-time pad," *Cryptologia*, vol. 35, no. 3, pp. 203–222, 2011.

[17] C. E. Shannon, "Communication theory of secrecy systems," *Bell System Technical Journal*, vol. 28, no. 4, pp. 656–715, 1949.

[18] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[19] G. R. Blakley, "Safeguarding cryptographic keys," in *Proceedings of the National Computer Conference*, (New York), pp. 313–317, IEEE Computer Society, AFIPS Press, 1979.

[20] G. R. Blakley, "One time pads are key safeguarding schemes, not cryptosystems; fast key safeguarding schemes (threshold schemes) exist," in *1980 IEEE Symposium on Security and Privacy*, pp. 108–113, IEEE Computer Society, 1980.

[21] M. Franklin and R. N. Wright, "Secure communication in minimal connectivity models," *Journal of Cryptology*, vol. 13, no. 1, pp. 9–30, 2000.

[22] Y. Wang and Y. Desmedt, "Secure communication in multicast channels: the answer to franklin and wright's question," *Journal of Cryptology*, vol. 14, no. 2, pp. 121–135, 2001.

[23] K. Srinathan, A. Narayanan, and C. P. Rangan, "Optimal perfectly secure message transmission," in *Advances in Cryptology–CRYPTO 2004*, pp. 545–561, Springer, 2004.

[24] M. Fitzi, M. Franklin, J. Garay, and S. H. Vardhan, "Towards optimal and efficient perfectly secure message transmission," in *Theory of Cryptography*, pp. 311–322, Springer, 2007.

[25] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "TCP extensions for multipath operation with multiple addresses." RFC 6824 (Experimental), Jan. 2013.

[26] O. Bonaventure, "In Korean, Multipath TCP is pronounced GIGA Path." http://blog.multipath-tcp.org/blog/html/2015/07/24/korea.html, 2015.

[27] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: Networking data centers randomly," in *NSDI*, vol. 12, p. 17, 2012.

[28] A. Arnes, F. Valeur, G. Vigna, and R. A. Kemmerer, "Using hidden markov models to evaluate the risks of intrusions: system architecture and model validation," *Lecture notes in computer science*, pp. 145–164, 2006.

[29] D. Rios Insua, J. Rios, and D. Banks, "Adversarial risk analysis," *Journal of the American Statistical Association*, vol. 104, no. 486, pp. 841–854, 2009.

[30] D. J. Pohly, C. Sestito, and P. McDaniel, "Adaptive protocol switching using dynamically insertable bumps in the stack," in *Milcom 2015 Track 3 - Cyber Security and Trusted Computing (Milcom 2015 Track 3)*, (Tampa, USA), Oct. 2015.