AN INFORMATION-THEORETIC APPROACH TO PRACTICAL SIDE CHANNEL PROTECTION

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Benjamin Huang Wu

December 2021

AN INFORMATION-THEORETIC APPROACH TO PRACTICAL SIDE CHANNEL PROTECTION

Benjamin Huang Wu, Ph.D.

Cornell University 2021

Side channel attacks are quickly becoming a serious problem in many computational security contexts. Presently, side channel countermeasures are primarily developed per case study, and the degree of mathematical rigor varies across studies. In order to address this methodological gap in the study of side channels, we use information theory-driven mathematical models to form a more rigorous foundation for the study of common classes of side channels, including one-shot and replayable channels.

Due to the broad selection of competing side channel leakage metrics in the literature, we take both an information-theoretic and an empirical approach to analyzing the usefulness and use cases of the most common metrics, including the recently proposed maximal leakage metric. Using maximal leakage as an operationally interpretable upper bound on side channel leakage, we show that linear combinations of no more than two deterministic protection schemes against one-shot attacks are optimal, which is a surprising result [56]. Conventional approaches to protecting against side channels that add randomized noise turn out to be sub-optimal.

In addition to one-shot attacks, we also study the class of replayable attacks in which the adversary can test the protection scheme many times with the same input. Here, we develop theoretical information bounds governing how the total leakage increases with the number of replays. The total leakage must naturally be

upper bounded, but existing information-theoretic work provides only unbounded characterizations of how the total leakage increases with the number of replays. We prove that the total leakage approaches its upper limit exponentially fast, and the exponent can be relatively easily computed given the protection scheme [58].

Finally, we study inference-phase attacks on dynamically pruned convolutional neural networks (CNNs), utilizing our theoretical results. Dynamic pruning is an increasingly popular technique for optimizing the execution time of the inference phase. We show that these techniques create a highly informative side channel that can be used to perform one-input-one-secret attacks, given indirect access to the pruning decisions. Moreover, we show that protecting this channel without losing the execution time savings from the pruning is naturally difficult because of the characteristics of the channel. Utilizing our studies of the one-shot and replayable attacks, we devise protection schemes that can significantly reduce the leakage from these attacks.

BIOGRAPHICAL SKETCH

Benjamin Wu received his Bachelor of Science degree in Applied and Computational Mathematics from the California Institute of Technology in 2015. He pursued his Ph.D. in Electrical and Computer Engineering at Cornell, Ithaca, NY. His research interests include applying information-theoretic security principles to real-world computational security problems.

To my family, whose endless support made this achievement possible.

ACKNOWLEDGEMENTS

I have been fortunate to have a great deal of support from a variety of people during my time at Cornell. First, I would like to thank my two co-advisors, Professors Edward Suh and Aaron Wagner. Their unique perspectives on the problems I tackled have been endlessly illuminating. Although their areas of expertise are fundamentally different, they have both been open to learning about and adapting to the conventions of the other's fields, which has not only been inspiring to me but has also greatly enriched my research projects and experience at Cornell. They have also both been very patient during the difficult times of my Ph.D. dissertation and were always available and eager to help. Being their student has truly been a privilege.

I would also like to thank my minor committee member Professor Elaine Shi. She has provided valuable guidance at critical stages of my research.

I would like to thank my family as well, for their support during this period of my life. In particular, my father has always been eager to share his experiences as a Ph.D. graduate as well as a technical researcher by profession. Although my research was far beyond the scope of his particular area of expertise, he has always made an effort to understand the problems I studied beyond just the surface level and offer substantive advice whenever it seemed like I was stuck. My mother has also been very supportive during this process and has always reminded me to take care of my health and day-to-day living.

I would also like to thank my colleagues at CSL, especially my fellow students being advised by Professor Suh. Notably, I would like to thank Dr. Andrew Ferraiuolo, Dr. Tao Chen, and Dr. Yao Wang for their advice and support during the early stages of my Ph.D. dissertation when I was trying to find a research direction. I would like to thank Weizhe Hua and Mulong Luo for their day to

day assistance with various tools that I was unfamiliar with. I would especially like to thank Trishita Tiwari and Muhammad Umar for lending me their technical expertise in performing cache side channel attacks and simulating dynamically pruned CNNs, respectively. Finally, I would also like to give a special thank-you to Professor Wagner's former student Professor Ibrahim Issa, who was not just a helpful and inspiring mentor during his time at Cornell, but has continued to collaborate with me on my research as a professor at the American University of Beirut despite our wildly different time zones.

Lastly, I would like to thank the other friends I have made at Cornell during my stay here for their moral support during trying times, especially Kimberly Michaels who graciously invited me to stay at her home while I finished my dissertation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Motivation

The field of cybersecurity has many long-established protocols and techniques for defending secure systems against attackers seeking to break basic confidentiality guarantees. However, side channels represent a class of attack surfaces that are not always easily solved by conventional cybersecurity techniques. Broadly speaking, a side channel is a source of information about a system that arises from the implementation of the system, as opposed to algorithmic weaknesses of the system. As a result, side channel attacks are dangerous in part because they can be difficult to detect since side channel attacks often require minimal interference with the system by the attacker. Indeed, it is sometimes possible for an attacker to passively observe the victim system without leaving any traces at all. Moreover, side channels tend to be difficult to protect even when they are known to exist, because removing them entirely requires changing the system implementation without breaking the algorithmic functions. The notion of side channels is decades-old and extends to broader fields of study than computer systems, but they have only in recent years begun to gain prominence as serious threats in the field of cybersecurity. Here we provide a few examples.

For example, consider the following pseudocode:

```
if(secret==1):
    executeLong()
else:
```

```
executeShort()
```

Algorithmically, this system is secure, because the code does not cause the secret value to be printed, saved to a file, or made available to an interested attacker. However, the implementation is clearly not secure, because an attacker can simply measure the execution time of the code snippet in order to infer the value of the secret. More broadly, this example can be extended to more complex secrets, more extensive control-flow structures, and other attack surfaces (for instance, power consumption or a series of memory accesses).

Notably in recent memory, the 2017 demonstrations of the Spectre [28] and Meltdown [33] vulnerabilities affecting all x86 Intel microprocessors revealed that cache side channel attacks could feasibly allow leakage of critical private data on a wide variety of platforms by leaking cryptographic keys, giving the attacker root access to any computer.

Another example is Kocher's attack on Diffie-Hellman key exchange, the RSA cryptosystem, and Digital Signature Standard (DSS) [29]. These three algorithms are crucial cryptographic primitives used for all types of secure communication over the Internet by using a private key that cannot be easily brute-forced. While the mathematical underpinnings of all three algorithms have remained unbroken for decades, Kocher demonstrated that performance-optimized implementations could statistically leak enough information to reduce the search space of the private key to allow brute-force attacks to succeed in a reasonable amount of time.

In addition to leaking cryptographic keys, side channel attacks have also been shown to be capable of breaking previously-thought secure communications channels, such as VoIP [55, 11, 54] or SSH [47, 14], or assumed privacy/intellectual

property guarantees in machine learning algorithms [20], and a variety of other privacy-sensitive applications.

In many of these cases, completely eliminating information leaks from side channels incurs significant costs in performance such as speed, power consumption, memory bandwidth, or others. So, practical protection techniques often aim to reduce information leakage as much as possible while maintaining acceptable cost levels. Unfortunately, existing leakage analysis methods and protection designs vary even more than side channels themselves, and the state-of-the-art typically requires that custom protection schemes be specially designed to defend against individual side channel attacks. Thus, a principled approach to desigining protection schemes for side channels is needed.

1.2 Overview of Dissertation

We study the side channel protection problem from multiple angles.

First, we mathematically interpret side channels as a Markov Chain of random variables so that we can study them from a rate-distortion perspective in order to establish information-theoretic leakage and implementation cost trade-off relationships. Under our model, side channel attackers perform their attacks by measuring an observable (random) variable and making the best maximum-likelihood guess of the secret based on the relationship between the secret and the observed variable. This represents the most powerful non-trivial attacks, as we assume the attacker has the best possible understanding of the system implementation short of knowing the secret directly and makes the best possible guess on the basis of the available information.

Under this model, we consider multiple strains of attacks, distinguished by the number and nature of measurements made by the attacker before making their guess. In this research, we primarily focus on: 1) one-look-one-guess (or one-shot) attacks, where the attacker measures the observable variable one time per secret value and 2) attacks where the attacker measures the different realizations of conditionally-independent copies of the same variable before guessing a single common secret value (or replayable attacks). We acknowledge the existence of other variations, and our mathematical model is general enough to include all such variants.

In addition, we primarily focus on side channels with finite discretized manifestations. That is, we assume that the variables observed by the adversary take on discrete distributions with finite alphabet sizes. The reason for this simplification is that, in computer systems, there are rarely truly continuous numerical values. Furthermore, even if a particular case arises such that the attacker must measure continuous numerical values, it is almost always possible to discretize those values as part of the protection mechanism.

Our contributions can be summarized as four studies relevant to the side channel problem.

1. We study four metrics for side channel information leakage (mutual information, channel capacity, local differential privacy, and maximal leakage) from both a theoretical and a practical standpoint and show that maximal leakage (maxL) [24] is best-suited for providing bounded security guarantees for one-shot attacks by passive adversaries. We also establish use cases for the remaining three metrics (which are more popular in the literature than maxL) in the side channel cases, as they are still useful given the correct

understanding of their information-theoretic significance.

2. We establish and solve the trade-off problem between maxL and linear costs for one-shot attacks as a linear programming (LP) problem. We show that linear combinations of no more than two deterministic protection mechanisms (as opposed to purely stochastic ones) are optimal. This is a surprising result, because conventional approaches to protection typically involve injecting random noise into the side channel in order to make it more difficult for the attacker to infer secrets. However, we show that this is not the optimal approach.

3. We study the replayable attack model from an information-theoretic standpoint and derive a strong composition theorems that describe the asymptotic behavior of Sibson mutual information, Arimoto mutual information, and alpha maxL [32] as the number of looks increases. Since mutual information and maximal leakage are both special cases of Sibson mutual information, and channel capacity is derived from mutual information, our composition theorem naturally applies to all three. Furthermore, the similar results we obtain for different metrics are surprising from a mathematical perspective in that all three metrics approach their respective limits exponentially fast at a same rate.

4. We study side channels created by dynamic activation pruning techniques in convolutional neural networks (CNNs) and demonstrate that one-input-one-secret attacks targeting various sensitive information during the inference phase are not only possible, but also difficult to protect against cheaply due to the inherent characteristics of CNNs. In particular we find that there typically exists a large degree of correlation and overlap between dynamic pruning decisions, even when the pruning decisions are made somewhat in-

dependently.

1.3 Mathematical Interpretation of Side Channels

We interpret a side channel as follows. First, in order for a side channel to leak information, there must exist an adversary, a victim system, and a secret. In order to guess this secret, the adversary uses some measurement of the victim system in order to improve their guess of the secret.

Fundamentally, all side channels can be mathematically interpreted the same way, even if the complexity of the problem may vary:

- The secret is a random variable U, with no restrictions on whether it is discrete/continuous or on types of probability distributions.
- There exists an *intermediate*, discrete random variable X, which is a possibly stochastic function of U.
- The measurement or *output*, discrete random variable Y, is a possibly stochastic function of X. The variables U, X, and Y form a Markov chain.
- The function $X(U)$ represents the immutable aspects of the victim system that cannot be modified as part of a protection mechanism.
- The function $Y(X)$ represents the mutable aspects of the victim system that can be modified in order to obfuscate any information encoded into the output.

To put these concepts into perspective, consider an RSA timing channel that arises from the following pseudocode:

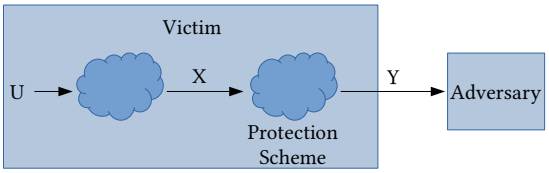


Figure 1.1: Summary of mathematical model of side channels.

```
def decrypt(ciphertext, key):
    //ciphertext and key are both bit arrays of length 1024
    for i in range(0, 1024):
        if(key[i]):
            executeLong(ciphertext)
        else:
            executeShort(ciphertext)
```

Supposing the adversary is able to noiselessly measure the total execution time of a single decryption, they could obtain the number of 1s (or *key weight*) in the key. In this example then, $X(U)$ represents the mapping of keys to key weights, which in this case happens to be a deterministic function. A protection scheme could then be implemented on top of this base implementation of the decryption code that hides the key weight by padding the total execution time in some fashion. The mapping of key weights to execution times would then be represented by $Y(X)$.

Note that this formulation of a side channel is inclusive of many types of applications. First, the introduction of the intermediate X does not cause any loss of generality. If there does not exist an immutable parts of the victim system, then we can simply let $X(U) = U$. Second, there are no restrictions on the dimensionality of any of the three random variables. If it is necessary to consider X or Y as random vectors, vectorized functions relating U, X, and Y can be defined and

analyzed, and it is even possible to interpret side channels with memory (where multiple outputs are collectively used for the adversary's guess of the secret) this way.

1.4 Sub-types of Side Channels

In this dissertation, we largely restrict our attention to the following subclasses of side channels:

- A side channel adversary is said to be *passive* if the function $Y(X)$ is independent of the adversary. In other words, the adversary cannot modify the victim's behavior to their advantage by feeding additional inputs.
- We constrain X and Y to have discrete and finite *alphabets* (i.e. the space of possible values a random variable can take). This constraint is limiting, but not unreasonably so since computer systems almost always deal with discrete and finite values.
- A side channel is *one-shot* if the adversary uses a single measurement of Y to guess U. Side channels with vectorized X and Y are included in this category.
- A side channel is *replayable* if the adversary can observe Y given the same underlying X value (i.e. $Y_1, Y_2, ...Y_n$ are conditionally independent given X). For example, for RSA decryption, consider multiple observations of the same decryption (same ciphertext, same key). Naturally, a replayable side channel is not one-shot.

CHAPTER 2

COMPARATIVE STUDY OF MAXIMAL LEAKAGE, MUTUAL INFORMATION, CHANNEL CAPACITY, AND DIFFERENTIAL PRIVACY

2.1 Existing Leakage Metrics

Numerous metrics for quantifying side channel leakage have been proposed in the past. Here we define and weigh the merits of just four proposed metrics:

- Shannon's mutual information (MI) is a standard information-theoretic metric that has been commonly used in computer architecture circles
- Shannon's channel capacity (CC) is a closely related metric to MI that is used to measure information transmission in communication over noisy channels
- Local differential privacy (LDP) [12] is a metric of plausible deniability used in cryptographic circles to analyze privacy protection
- Maximal leakage ($MaxL$) [24] is a recently-proposed, operationally-interpretable leakage metric designed to measure side channel leakage

Definition 1. ***(Basic Notation)*** *For random variables X and Y, with alphabet sizes $|\mathcal{X}| = M$ and $|\mathcal{Y}| = N$, define the following:*

- *$c(x, y)$ is the nonnegative (but not necessarily finite) cost of mapping each $x \in \mathcal{X}$ to each $y \in \mathcal{Y}$. Infinite cost entries correspond to illegal mappings. We refer to this function as the* cost function *and the corresponding matrix $\{c_{xy}\}$ as the* cost matrix.

- *$p(x)$ is the distribution of X. $p(y)$ is the distribution of Y. $p(x,y)$ is the joint distribution of X and Y. $p(y|x)$ is the conditional distribution of Y given X.*
- *$\mathscr{C}(\boldsymbol{A}) = \mathscr{C}\{a_{xy}\} = \sum_{x,y} p(x)c(x,y)a_{xy}$ is the* total cost *for any matrix $\boldsymbol{A} = \{a_{xy}\}$. We will use $\mathscr{C}$ as a shorthand for this quantity, when the parameter matrix is implied.*
- *$L(\boldsymbol{A})$ is a placeholder expression for the leakage value associated with any matrix $\boldsymbol{A} = \{a_{xy}\}$ using some pre-specified leakage metric. When necessary, we will distinguish different metrics using subscripts (e.g. $L_{MaxL}(\boldsymbol{A})$).*
- *$\boldsymbol{P} = \{p_{xy}\}$ is an $M \times N$ transition matrix such that $p_{xy} = Pr(Y = y|X = x) \quad \forall x, y$ and $\mathscr{C}(\boldsymbol{P})$ is finite. We refer to any matrix of this form interchangeably as a* protection scheme *or* protection mechanism*. It is subject to typical transition matrix constraints (rows sum to 1, non-negative entries).*

Mutual Information: As one of the most basic information-theoretic metrics, mutual information has been used to evaluate side channel protection in some previous work on side channel protection, e.g. [61, 17].

$$I(X;Y) = L_{MI}(\mathbf{P}) = \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x)p(y|x) \log \frac{p(y|x)}{\sum_{x \in \mathcal{X}} p(x)p(y|x)} \tag{2.1}$$

In plain terms, mutual information measures the amount of information shared between two random variables. In the context of a side channel, the precise significance of $I(X;Y)$ can be difficult to pin down, since under our formulation, it is agnostic to the secret U. On the other hand, directly using $I(U;Y)$ can be unwieldy in practice, since it requires the protection designer to fully understand $X(U)$ to compute it, which may be difficult for complex systems.

Furthermore, mutual information as a metric is sensitive to the marginal distributions of all random variables involved. This is potentially problematic in systems

where the marginal distributions are not necessarily static.

Channel Capacity: Another basic information-theoretic metric, channel capacity is typically used to measure the rate of reliable communication over a noisy channel.

$$C(X \to Y) = L_{CC}(\mathbf{P}) = \max_{p(x)} L_{MI}(\mathbf{P}) \tag{2.2}$$

As such, $C(X \to Y)$ is a useful metric for leakage in covert channels where a sender deliberately encodes messages to a receiver, a scenario that is analogous to standard communication over a noisy channel. Since $C(X \to Y)$ is still agnostic to U, it is again difficult to pin down its significance in a side channel, whereas $C(U \to Y)$ is just as unwieldy as $I(U;Y)$.

Local Differential Privacy: A metric of plausible deniability used in cryptographic circles, local differential privacy is defined as [12]:

$$LDP(X \to Y) = L_{DP}(X \to Y) = \max_{\substack{y \in \mathcal{Y} \\ x,x' \in \mathcal{X}}} \log \frac{p(y|x)}{p(y|x')} \tag{2.3}$$

but can also be equivalently represented as [24]:

$$L_{DP}(X \to Y) = \max_{P_X} \max_{U-X-Y} \log \frac{\max_y \max_u p(u|y)}{\max_u p(u)} \tag{2.4}$$

$$= \max_{P_X} \max_{U:U-X-Y} \log \frac{\max_{\tilde{u}(\cdot)} P(U = \tilde{u}(Y))}{\max_{\tilde{u}} P(U = \tilde{u})} \tag{2.5}$$

As we will see momentarily, local differential privacy is defined quite similarly to maximal leakage but is a more conservative upper bound.

Maximal Leakage: Maximal leakage is defined based on the multiplicative gain ratio of the probability that the adversary guesses the secret correctly with versus without access to the side channel (call this quantity *mult-leakage*). So first, define:

$$L_{mult}(\mathbf{P}) = \log \frac{\max_{\tilde{u}(\cdot)} P(U = \tilde{u}(Y))}{\max_{\tilde{u}} P(U = \tilde{u})} \tag{2.6}$$

where the notation $\tilde{u}$ is a blind guess of U, and $\tilde{u}(Y)$ is a informed guess of U after observing Y. Then, mult-leakage tells us the multiplicative gain on the adversary's probability of correctly guessing U given the side channel (the logarithm is for scaling purposes). Alternatively, mult-leakage can be interpreted as the bit-difference of information between the informed and blind guesses. Thus, if the side channel leaks no information, then the best informed guess is no better than the best blind guess and the leakage is 0. On the other hand, if the side channel does leak information, mult-leakage tells us how much the adversary's guesses have been improved by the side channel and, ultimately, how useful the side channel is to the attacker.

Then, maximal leakage is defined as follows [24]:

$$MaxL(X \to Y) = L_{MaxL}(\mathbf{P}) = \max_{U:U-X-Y} \log \frac{\max_{\tilde{u}(\cdot)} P(U = \tilde{u}(Y))}{\max_{\tilde{u}} P(U = \tilde{u})} = \max_{U:U-X-Y} L_{mult}(\mathbf{P}) \tag{2.7}$$

By definition, maximal leakage is the worst-case mult-leakage over all possible U, which means it upper bounds mult-leakage. The reason we elect to use maximal leakage in favor of mult-leakage is that the latter is not always possible to compute, depending on whether the system designer knows U. Mult-leakage requires knowledge of $p(x)$, which is often difficult to characterize, especially for complex systems. Certainly, an approximation of mult-leakage is possible through data collection of X, but we are interested in upper-bounding leakage.

At first blush, it seems that maximal leakage also requires knowledge of $p(x)$, but it turns out this is not the case. First, we note that maximal leakage is agnostic to U by definition, so it is well-defined even in side channels where it isn't clear which secret the adversary is after. Furthermore, it can be shown that maximal

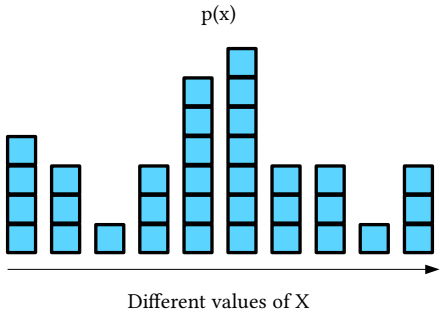


Figure 2.1: Conceptual example of shattering U.

leakage is can be computed as follows [24]:

$$L_{MaxL}(\mathbf{P}) = \log \sum_{y \in \mathcal{Y}} \max_{x \in \mathcal{X}} p(y|x) \tag{2.8}$$

In other words, maximal leakage requires knowledge of $p(x)$ only in terms of its support, the set of X-values with non-zero $p(x)$. And since we have restricted our attention to memoryless protection, we only need to find the maximum value in each column (ignoring X-values with probability of 0) of $\mathbf{P}$, sum these values, and take the log of the sum to compute it. Moreover, the maximum mult-leakage is achieved by a particularly useful U, referred to as a *shattering* U by the authors. This type of U is characterized by the following two properties.

Definition 2. ***(Shattering U)*** *For any random variable X, U is* shattering *if:*

- *U is uniformly distributed over a finite alphabet $\mathcal{U}$*
- *For each $u \in \mathcal{U}$, $x = X(u)$ is a deterministic value.*

A conceptual example of a shattered distribution is shown in Figure 2.1. Here, each blue square corresponds to an equally-probable potential value of U (34 distinct values total, in this example). An example distribution of X is shown, where each possible X-value corresponds to some number of U-values.

The significance of this shattering distribution is twofold. First, for any shattering U, maximal leakage equals mult-leakage. In such cases, maximal leakage measures the the utility gained by the adversary from the side channel under any protection scheme. Second, the shattering U is quite representative of side channels in which U is known to be an encryption key, if keys are selected uniform randomly from some (not necessarily known) space of allowed keys and the baseline side channel process $X(U)$ is not stochastic. For such channels, maximal leakage, though pessimistic by design, exactly captures the side channel leakage.

While we focus in part on demonstrating the merits of maximal leakage as a side channel metric in the next chapter, it should be noted that maximal leakage does has its own practical downsides. Notably, when U is not shattering, it is possible for maximal leakage to potentially overestimate the mult-leakage. As such, the purpose of this chapter is not necessarily to hold up maximal leakage as the best possible metric in all side channel cases, but rather to illuminate the situations in which it does excel, and how to interpret it appropriately as an upper bound.

2.2 Empirical Comparison of Leakage Metrics to Mult-Leakage

Here, we study the relationship between the target leakage metrics and mult-leakage in a more practical setting. To do so, we study the RSA decryption timing channel under two sample implementations: first, under square-and-multiply implementations of the decryption and second under GNU's multiple precision library implementation of the same function.

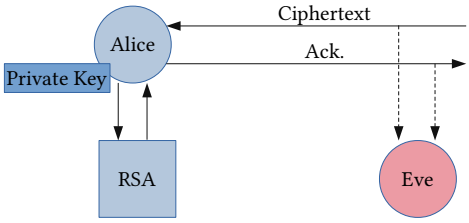


Figure 2.2: RSA timing side channel example.

2.2.1 RSA Decryption Timing Channel

For the rest of this section, we consider a timing channel involving RSA as seen in Figure 2.2. In this side channel, Alice serves many clients who need to use Alice's public key to encrypt messages to her over a network. For each such encrypted message, Alice uses her private key to decrypt the message and then responds. The adversary, Eve, is not one of Alice's clients but is capable of observing the network traffic coming to and from Alice, perhaps by employing a packet sniffer (hence, Eve is a passive adversary). Eve sees when each decryption request arrives and when Alice responds from the sequence and timing of packets, which she uses to guess Alice's private key. Here, U is the private key, X is the decryption time, and Y is the length of time between when Alice receives each message and when she sends a response to the client. To implement a protection scheme, we must choose $Y(X)$, which is how long to delay Alice's response on top of the true decryption time.

Finally, we assume that Alice's private key was chosen uniformly at random from all binary strings of a fixed length (as opposed to only legal keys based on the RSA cryptosystem). This is a choice of convenience to facilitate our experiments, but doing so does not affect our conclusions, as we will explain later in this section.

2.2.2 Square-and-Multiply Implementation of RSA

We first consider the square-and-multiply implementation of modular exponentiation, the main sensitive operation of RSA decryptions. Pseudocode for the square-and-multiply implementation of RSA decryption is as follows:

```
1: Inputs c (ciphertext), u (private key), n (modulus)
2: r ← 1
3: c ← c mod n
4: while u > 0 do
5:     if u mod 2 == 1 then
6:         r ← (m * c) mod n
7:     end if
8:     u ← u >> 1
9:     c ← (c * c) mod n
10: end while
11: return r
```

The timing channel arises from the if statement in line 5. The modular multiplication of the result and ciphertext only occurs if the next bit of the private key is 1. From this fact, the adversary can deduce the weight (the number of 1s) of the private key. We make several simplifying assumptions and define the parameters of this experiment as follows:

- Ignore confounding factors, such as system noise or network delay. Y is simply equal to X plus any delay we choose to add.
- Assume all 1024-bit sequences are valid keys.
- Assume the bits of the key are independently and identically distributed

Bernoulli random variables. This results in a uniform-randomly selected key out of all 1024-bit binary strings.

- Let $\mathbf{U} = [U_1, U_2, ...U_{1024}]$ be a random vector representing the value of the private key.
- Let $X = \sum_{i=1}^{1024} U_i$ be a random variable representing the weight of the private key.
- Assume that the multiplication in line 6 of the above pseudocode takes a fixed K milliseconds to execute each time it is called.
- Let Z be a binomial random variable with fixed probability $p = \frac{1}{2}$ and size parameter m (which we vary).
- Let $Y = X + Z$. In other words, our protection scheme is independently added binomial noise.
- Let $c(x, y) = \begin{cases} y - x & \text{if } y \geq x \\ \infty & \text{otherwise} \end{cases}$

  Note that this cost matrix enforces that any protection with finite total cost must be upper triangular. Moreover, note that with this cost matrix and independent binomial delays, the total cost is $\frac{m}{2}$.

Here, we can analytically compute mutual information, channel capacity, and maximal leakage for many different values of m (the size parameter of the binomial-distributed noise) and plot them on the same axes, as seen in Figure 2.3. We note that there exists a large gap between mutual information and maximal leakage that sharply shrinks as we approach no protection. A sizeable gap exists between maximal leakage and channel capacity as well. Recall that, since U is shattering, maximal leakage equals mult-leakage. Conversely, this implies that both mutual information and channel capacity underestimate mult-leakage in this example, and

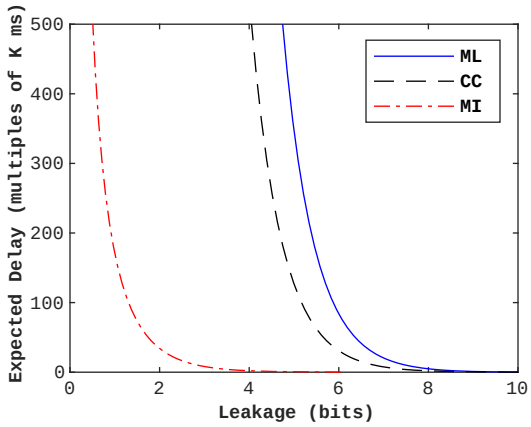


Figure 2.3: Metric comparison for square-and-multiply RSA with independent binomial noise on a 1024-bit key decryption. Legend: MaxL is maximal leakage, CC is channel capacity, and MI is mutual information. Note that MaxL equals mult-leakage since the key is a shattering U.

thus also underestimate the adversary's utility. Note that, had we chosen a key uniform-randomly from the set of all feasible keys (according to the RSA cryptosystem), U would still have been shattering.

2.2.3 GMP Implementation of RSA

Here, we consider an implementation of modular exponentiation where the weight of the key isn't directly leaked but instead the decryption time varies with the key in some other way. We show that maximal leakage's gaps with mutual information and channel capacity are still significant even in this case. We use GNU's multiple-precision (GMP) library's implementation of modular exponentiation. Here, we perform essentially the same experiment as before. The assumptions and parameters of the experiment are as follows (only ones that are different from the square-and-multiply implementation will be listed):

- Let $\mathbf{U} = [U_1, U_2, ...U_{16}]$ be a random vector representing the value of the

private key. Note that we are using a 16-bit key here so that it is possible to exhaustively collect decryption timing data for all private keys and ciphertexts, to compute the distribution of decryption times.

- Let X be the random variable representing the execution time (in cycles) of GMP's modular exponentiation on an Intel i7 core. The decryption time of each private key varies with the ciphertext, so we uniform randomly selected a fixed ciphertext for the purposes of this experiment. The distribution of X we used can be seen in Figure 2.4.
- Choose the alphabet of Y, $\mathcal{Y}$, as follows. Choose a noise *width* w. Extend $\mathcal{X}$ by w elements, each spaced by the most common difference between consecutive elements in $\mathcal{X}$ (in case of a tie, choose the smallest common difference). So for example, suppose $\mathcal{X} = \{1,\ 5,\ 7,\ 9,\ 11,\ 13\}$ and $w = 4$. Then $\mathcal{Y} = \{1,\ 5,\ 7,\ 9,\ 11,\ 13,\ 15,\ 17,\ 19,\ 21\}$ since the most common interval between consecutive elements in $\mathcal{X}$ is 2.
- Let Z be a binomial random variable with fixed probability $p = \frac{1}{2}$ and size parameter w (which we vary).
- Let $Y(X)$ be defined as follows. Given $x \in \mathcal{X}$, generate a Z-value z. Note that z is an integer; choose the zth larger element than x in $\mathcal{Y}$.

As before, we can directly compute mutual information, channel capacity, and maximal leakage for many different values of w. Plotting mutual information, channel capacity, and maximal leakage against total cost (Figure 2.5), we find that a gap exists between maximal leakage and mutual information/channel capacity. These results reaffirm our earlier observation that mutual information and channel capacity underestimate the advantage given to the adversary in practice.

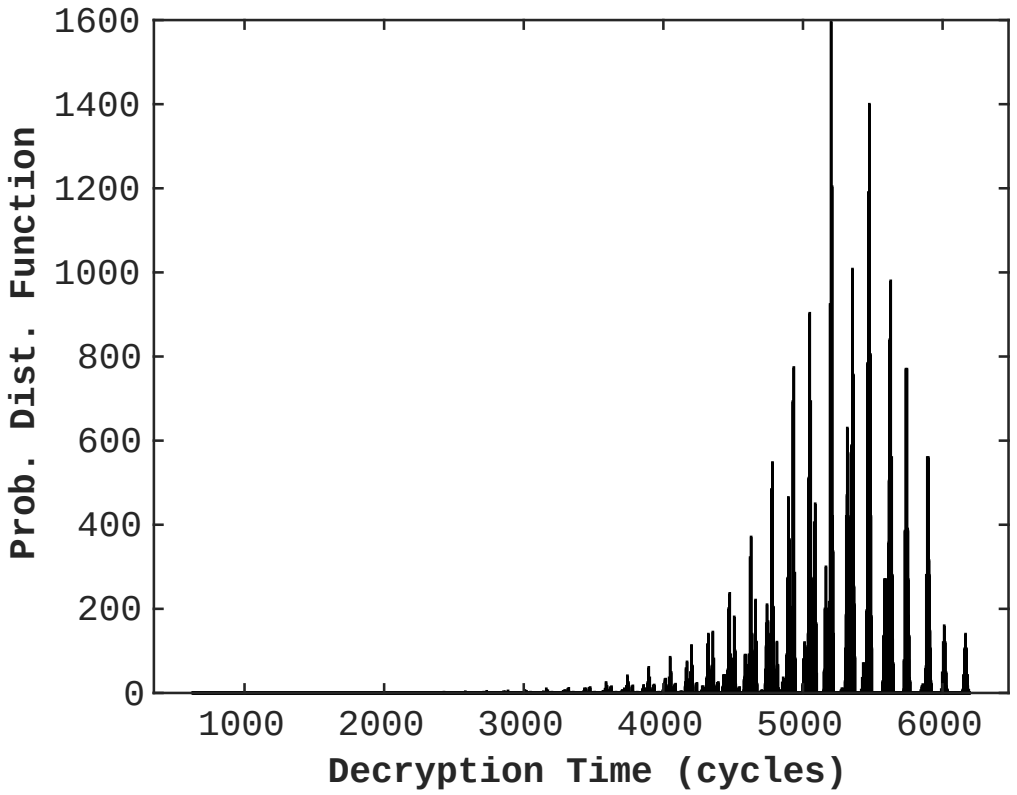


Figure 2.4: Empirical distribution of decryption time X for uniform randomly selected ciphertext 1110011001001100_2 (58956_{10}).

Here, we remark that both in the case of the GMP implementation and in the earlier square-and-multiply implementation, we obtained trade-off curves for mutual information, channel capacity, and maximal leakage with very similar shapes. So, it may be tempting to suggest that there is little difference in usage between the three metrics, since their trade-off curves are so similar in shape. However, there are two factors to keep in mind. First, in the above experiments we have only used independent (of X) random padding. Second, we compared how different metrics behave for the same protection scheme. Essentially, we have not shown that mutual information, channel capacity, and maximal leakage agree on a relative ordering of how secure an arbitrary pair of protection schemes (possibly not independent of X). Thus, we conclude this section by studying a toy optimization

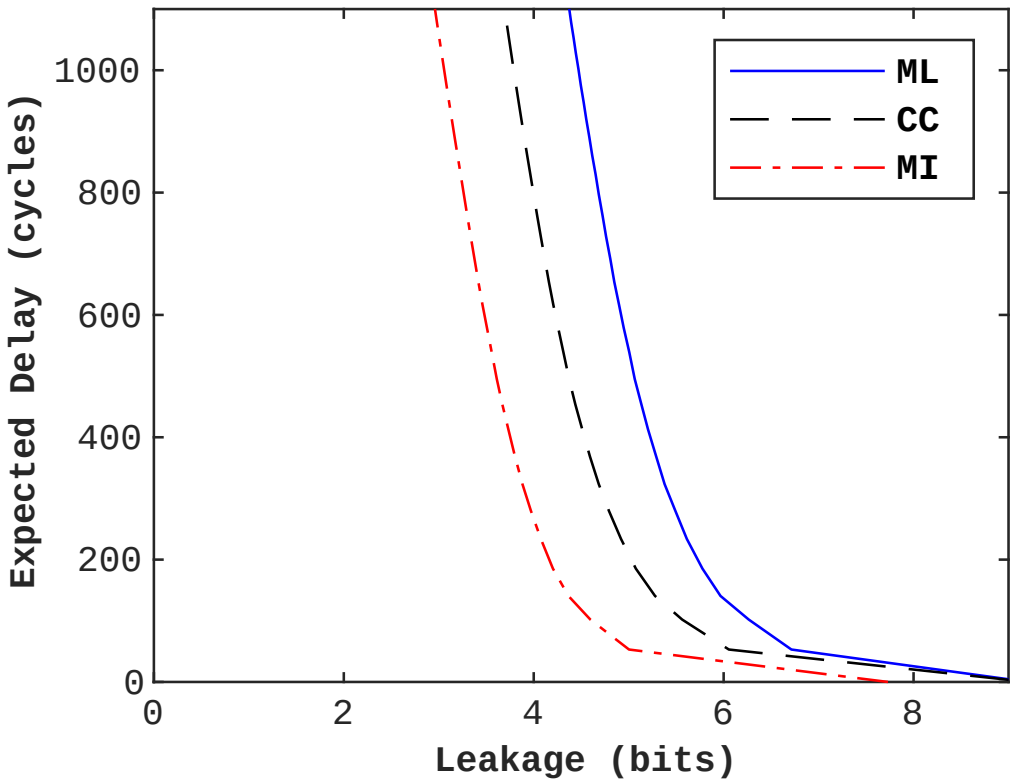


Figure 2.5: Metric comparison for GMP implementation of RSA with binomial random extension on 16-bit key decryption. Again, note that MaxL equals mult-leakage.

under each metric.

2.2.4 MI, CC, and LDP optimal protections:

First, recall the definition of mutual information as given in Equation 2.1. Since $p(x)$ is fixed for the optimization over mutual information, the only active variable in mutual information is $p(y|x)$. Since mutual information is nonlinear and convex over $p(y|x)$, it is typically optimized by protection schemes in the interior of the feasible set. In other words, MI-optimal schemes will typically be stochastic. Indeed, a simple experiment confirms this. Using the the alphabets $\mathcal{X} = \{x_1, x_2, x_3, x_4\}$

and $\mathcal{Y} = \{y_1, y_2, y_3, y_4\}$, the marginal distribution of X, $p(x) = [0.4, 0.2, 0.2, 0.2]$, and cost function

$$C = \begin{bmatrix} 1 & 2 & 3 & 4 \\ \infty & 1 & 2 & 3 \\ \infty & \infty & 1 & 2 \\ \infty & \infty & \infty & 1 \end{bmatrix}$$

the MaxL-optimal and MI-optimal solutions for 0.5 units of cost are given by:

$$P^*_{MaxL} = \begin{bmatrix} .25 & .75 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad P^*_{MI} = \begin{bmatrix} .5235 & .3031 & 0.1233 & 0.0502 \\ 0 & .4890 & .3120 & .1990 \\ 0 & 0 & .6105 & .3895 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Interestingly, the MaxL-optimal scheme is almost deterministic with the exception of the first row, while the MI-optimal scheme is entirely stochastic. Since channel capacity is itself defined as a maximization over mutual information, the CC-optimal scheme will also tend to be stochastic.

Finally, the LDP-optimal solution is simply:

$$C = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This is because local differential privacy is infinite if any column of protection scheme contains both a 0 and a non-zero value. Since for the given cost function, all columns but the rightmost one must have at least one 0, the LDP-optimal solution is always achieved by mapping all values to the maximum value.

2.3 Summary

To conclude the previous two chapters, the major takeaway is that for the purposes of operationally upper bounding the information leakage from a side channel, maximal leakage is the best option. Mutual information, channel capacity, and local differential privacy also have their uses, but are not best used as upper bounds.

CHAPTER 3

OPTIMAL TRADE-OFFS UNDER MAXIMAL LEAKAGE

3.1 Optimal MaxL-Cost Trade-offs

Using maximal leakage as our leakage metric, we will show two key facts about the minimization of total cost subject to an upper bound on maximal leakage, which we call the optimization over maximal leakage. First, we demonstrate that the optimization over maximal leakage can be written as a linear program. While this fact is relatively simple to verify, its significance lies in that it greatly simplifies the process of solving the optimization itself (which, in general, is not a trivial feat). Second, as the main theorem of this chapter, we prove that under certain constraints on the cost matrix $c(x, y)$ that are quite common among side channels, optimality under maximal leakage can be achieved with easy-to-implement deterministic protection schemes, a surprising and useful result. The remainder of the section is dedicated to stating these results rigorously and explaining their implications.

3.1.1 Formulating the Optimization

Definition 3. ***(Definitions)***

The following definitions are needed to formulate the optimization. For random variables X and Y, with alphabet sizes $|\mathcal{X}| = M$ and $|\mathcal{Y}| = N$, we define the following:

- *We retain the defined variables and functions given in Definition 1, but will at*

this point we will retire the previous notation to distinguish between various leakage metrics (e.g. $L_{MaxL}(\boldsymbol{P})$) in favor of the next item.

- $\mathscr{L}(\boldsymbol{A}) = \mathscr{L}(\{a_{xy}\}) = \sum_y \max_x a_{xy}$ *is the* exponentiated maximal leakage *(or* exp-leak*, for short) of any matrix* $\boldsymbol{A} = \{a_{xy}\}$*. We will use* $\mathscr{L}$ *as a shorthand for this quantity, when the parameter matrix is implied. Note that minimizing over exp-leak is equivalent to minimizing over maximal leakage.*
- *a given protection scheme* $\boldsymbol{P}$ *is* deterministic *if all* p_{xy} *equal 0 or 1. It is* stochastic *otherwise.*
- *an* (L, C) *pair is* achieved *by* $\boldsymbol{P}$ *if* $\mathscr{L}(\boldsymbol{P}) \leq L$ *and* $\mathscr{C}(\boldsymbol{P}) \leq C$*.*
- *an* (L, C) *pair is* achievable *if there exists such a* $\boldsymbol{P}$ *that* (L, C) *is achieved by it.*
- *the set* S *is the set of all achievable* (L, C) *pairs.*
- $C^*(L) = \inf[C : (L, C) \in S]$*. We refer to* $C^*(L)$ *evaluated for all values of* L *as the tradeoff curve and the set of points* $S_b = [(L, C) \in S | C = C^*(L)]$ *as the boundary of* S*.*
- $\boldsymbol{P}$ *is* optimizing in S *if* $\mathscr{C}(\boldsymbol{P}) = C^*(\mathscr{L}(\boldsymbol{P}))$ *(i.e. if* $\boldsymbol{P}$ *achieves a point on the boundary of* S*).*
- *the set* S_d *is the set of all points in* S *that can be achieved by a deterministic protection scheme.*
- *an* (L, C) *pair is* achievable in S_d *if there exists a deterministic protection scheme* $\boldsymbol{P}$ *that achieves* (L, C)*.*
- $C_d^*(L) = \inf[C : (L, C) \in S_d]$*.*
- $\boldsymbol{P}$ *is* optimizing in S_d *if* $\mathscr{C}(\boldsymbol{P}) = C_d^*(\mathscr{L}(\boldsymbol{P}))$*. Note that a* $\boldsymbol{P}$ *that is optimizing in* S_d *is not necessarily a deterministic protection scheme.*

Remark. ***(Set Indexing)*** *We will choose to let* $\mathcal{X} = \{x_1, x_2, ...x_M\}$ *and* $\mathcal{Y} = \{y_1, y_2, ...y_N\}$.

Here, we consider the optimization problem over maximal leakage:

$$C^*(L) = \min_{\mathbf{P}} \mathscr{C}(\mathbf{P}) \quad \text{s.t. } \mathscr{L}(\mathbf{P}) \leq L, \ \sum_y p_{xy} = 1 \ \forall x, \\ p_{xy} \geq 0 \ \forall \ x, y \tag{3.1}$$

This can be rewritten as an LP as follows:

$$C^*(L) = \min_{p_{xy}, q_y} \mathscr{C}(\mathbf{P}) \quad \text{s.t. } \sum_y q_y \leq L, \ \sum_y p_{xy} = 1 \ \forall x, \\ p_{xy} \geq 0, \ p_{xy} \leq q_y, \ \forall \ x, y \tag{3.2}$$

For this LP to be useful, we first note that the optimization problem given in Equation 3.3 is a convex problem, which can be proven using standard techniques.

Lemma 1. ***(Convexity)*** *$C^*(L)$ is a convex function of L.*

Proof of Lemma 1. Fix L_1, L_2, and $\lambda \in [0, 1]$ and let $\bar{L} = \lambda L_1 + (1 - \lambda) L_2$
Suppose $\mathbf{P}_i = \{p_i(y|x)\}$ minimizes $\mathscr{C}(\mathbf{P}_i)$ such that $\mathscr{L}(\mathbf{P}_i) \leq L_i$ for $i = 1, 2$. In other words, $\mathbf{P}_i$ is optimizing.
Note $\bar{\mathbf{P}} = \lambda \mathbf{P}_1 + (1 - \lambda)\mathbf{P}_2$ is a valid transition matrix and

$$\begin{aligned}
\mathscr{L}(\bar{\mathbf{P}}) &= \sum_y \max_x \bar{p}(y|x) \\
&\leq \lambda \sum_y \max_x p_1(y|x) + (1 - \lambda) \sum_y \max_x p_2(y|x) \\
&= \lambda \mathscr{L}(\mathbf{P}_1) + (1 - \lambda) \mathscr{L}(\mathbf{P}_2) \\
&\leq \lambda L_1 + (1 - \lambda) L_2 \\
&= \bar{L}
\end{aligned}$$

Thus $\{\bar{p}(y|x)\}$ satisfies the exp-leak bound condition $\mathscr{L}(\bar{\mathbf{P}}) \leq \bar{L}$. So by definition,

$$\begin{aligned} C^*(\bar{L}) &\leq \mathscr{C}(\bar{\mathbf{P}}) \\ &= \mathscr{C}(\lambda \mathbf{P}_1 + (1-\lambda)\mathbf{P}_2) \\ &= \sum_x p(x) \sum_y [\lambda p_1(y|x) + (1-\lambda) p_2(y|x)] c(x,y) \\ &= \lambda \mathscr{C}(\mathbf{P}_1) + (1-\lambda)\mathscr{C}(\mathbf{P}_2) \\ &= \lambda C^*(L_1) + (1-\lambda) C^*(L_2) \end{aligned}$$

Hence $C^*(L)$ is a convex function of L. □

First, the convexity of the optimal trade-off curve is significant in that it allows for a useful qualitative assessment of the optimization problem. That is, adding a little protection on top of an unprotected side channel is very costly, but relaxing a zero-leakage scheme buys more cost reduction. Second, since both the objective function and constraints (the latter is obvious) of Equation 3.3 are all convex, we have confirmed the convexity of the optimization problem.

Remark. *Note that $C_d^*(L)$ is not convex since a deterministic protection schemes necessarily has an integer exp-leak value. The space S_d is a subset of S given by all $(L, C) \in S$ pairs dominated by the set of finite points $(L, C_d^*(L))$ for integer L values . The boundary of S_d is shaped like a descending staircase, and S_d is the set of all points above and to the right of this stair-like boundary.*

3.1.2 Structural Result and Proof

Definition 4. ***(Cost Constraints)***

We refer to a cost function/matrix that satisfies the following constraints as staircase nondecreasing*:*

1. *For* $0 < i < j \leq M$ *and all* $y \in \mathcal{Y}$*, if* $c(x_i, y) = \infty$*, then* $c(x_j, y) = \infty$*. (i.e. if one matrix element is infinite, then that column is infinite all the way down).*
2. *For* $0 < i < j \leq N$ *and all* $x \in \mathcal{X}$*, if* $c(x, y_i) < \infty$*, then* $c(x, y_i) \leq c(x, y_j) < \infty$*. (i.e. excluding infinities, each row of the matrix is nondecreasing from left to right).*

Note that staircase nondecreasing cost matrices are exemplified by upper triangular cost matrices (where all entries below the diagonal are infinite cost) with ordered cost entries for each row. This special case of staircase infinite, nondecreasing cost matrices is typical of most power and timing side channels, since one cannot map power consumption or latency to a value less than itself.

Theorem 2. ***(Wu-Suh-Wagner)***

If $c(x, y)$ *is staircase nondecreasing, then*

1. $\min\limits_{(\mathscr{L},\mathscr{C})\in S} \mathscr{C} + \alpha\mathscr{L} = \min\limits_{(\mathscr{L},\mathscr{C})\in S_d} \mathscr{C} + \alpha\mathscr{L} \quad \forall \alpha > 0$
2. *For all* $L \geq 1$*,* $(L, C^*(L))$ *can be achieved by* $\boldsymbol{P} = \lambda \boldsymbol{P}_1 + (1-\lambda)\boldsymbol{P}_2$ *for some* $\lambda \in [0, 1]$ *and some deterministic protection schemes* $\boldsymbol{P}_1$ *and* $\boldsymbol{P}_2$*, such that* $\mathscr{L}(\boldsymbol{P}) \leq L$ *and* $C^*(L) \leq \lambda C_d^*(\mathscr{L}(\boldsymbol{P}_1)) + (1-\lambda) C_d^*(\mathscr{L}(\boldsymbol{P}_2))$*.*

The proof proceeds by taking an optimizing stochastic protection scheme and showing that it can be made more deterministic without losing optimality. The

proof of Theorem 2 is given in Section 3.2. First, we remark on the useful implications of this theorem.

3.1.3 Implications of Theorem 2

The main implication of Theorem 2 is that deterministic protection schemes are sufficient to achieve optimality over maximal leakage. The first part of the theorem essentially states that the supporting hyperplanes of the trade-off space are achieved by deterministic schemes. The second part follows from the first and states that the constrained optimization is solved by at least the next best thing: a mixture of at most two deterministic schemes.

First, recall the implementation benefits of deterministic protection schemes. Regardless of the specific cost function (as long as it is staircase nondecreasing) or application, any deterministic protection scheme can be compressed to an $N \times 2$ matrix (or smaller) recording which Y-value each X-value maps to. In addition, deterministic schemes are resistant to averaging attacks, where the adversary attempts to learn additional information by gathering statistics of Y, since the same X value always maps the the same Y value. In the event that a mixture of two deterministic schemes is needed, one may implement a pre-determined schedule alternating between the two deterministic schemes for each $X \to Y$ mapping. Here, while the leakage of individual observations of Y will change over time, we can enforce the desired long-run bound.

Second, the proof of the main theorem induces an algorithm by which one may take any optimizing protection scheme and convert it to a deterministic form, so that the discussed benefits can be leveraged. This algorithm simply performs the

procedures specified in Definitions 5 and 7 in Section 3.2 recursively.

Third, if it is necessary to solve the entire optimal trade-off curve (for example, if on-the-fly tuning of leakage is expected), Theorem 2 states that it is only necessary to solve for integer exp-leak points and then connect the dots so that the overall curve is convex. Also, for small deviations in the leakage bound, tuning can be done simply by changing the mixture proportion.

3.2 Proof of Theorem 2

3.2.1 Proof of Theorem 2 Part 1

First, we establish a structural result for all optimal protection schemes that will be a useful assumption for later steps of our proof of Part 1 of Theorem 2

Definition 5. ***(Water-Filling)***

Consider any protection scheme $\boldsymbol{P}$*. Define a* $1 \times N$ *vector* $\vec{p} = [p_1, p_2, ...p_N]$ *such that* $p_i = \max_{x \in \mathcal{X}} p_{xy_i}$*. In plain terms,* $\vec{p}$ *consists of the column maxima of* $\boldsymbol{P}$*. Using* $\vec{p}$ *alone, we construct a new protection scheme* $\boldsymbol{P'}$ *as follows:*

1. *Start with a* $M \times N$ *zero matrix* $\boldsymbol{P'} = \{p'_{xy}\}$*. We will assume that the members of* $\mathcal{X}$ *and* $\mathcal{Y}$ *are in some enumerated order, as previously stipulated.*
2. *For each row* $x \in \mathcal{X}$*, iterate over each* y_i*,* $i = 1, 2, ...N$*.*
   - *If* $c(x, y_i) = \infty$*, let* $p'_{xy_i} = 0$
   - *Else, set* $p'_{xy_i} = \min\{p_i, 1 - \sum_{j=1}^{i-1} p'_{xy_j}\}$*.*

In plain terms, we are constructing ***P'*** *by maintaining the column maxima of* ***P*** *and "filling" in probability mass in each row from left to right.*

We define this procedure to generate ***P'*** *from* ***P*** *as the method to convert* ***P*** *into "water-filled" form. Also, if* ***P*** *and* ***P'*** *are identical, we say that* ***P*** *is a "water-filled" protection scheme.*

Lemma 3. ***(Water-Filling Lemma)***

If the cost function satisfies definition 4, then all optimizing ***P*** *can be converted into water-filled form* ***P'*** *such that* $\mathscr{C}(\boldsymbol{P}) = \mathscr{C}(\boldsymbol{P'})$ *and* $\mathscr{L}(\boldsymbol{P}) = \mathscr{L}(\boldsymbol{P'})$.

Lemma 3. Suppose we are given optimizing $\mathbf{P}$ and its water-filled form $\mathbf{P'}$. By its construction, $\mathscr{L}(\mathbf{P}) \geq \mathscr{L}(\mathbf{P'})$ since we did not increase the total sum of column maxima. In addition, since we independently fill up each row's entries in $\mathbf{P'}$ from least cost to greatest cost, $\mathscr{C}(\mathbf{P}) \geq \mathscr{C}(\mathbf{P'})$ for any cost function that is staircase nondecreasing.

From these two statements, we also obtain the reverse inequalities:

- $\mathscr{C}(\mathbf{P}) \geq \mathscr{C}(\mathbf{P'})$ implies that $\mathscr{L}(\mathbf{P}) \leq \mathscr{L}(\mathbf{P'})$ since $\mathbf{P}$ is optimizing and $\mathbf{P'}$ cannot perform any better (have lower $\mathscr{C} + \alpha\mathscr{L}$) than $\mathbf{P}$.
- Similarly, $\mathscr{L}(\mathbf{P}) \geq \mathscr{L}(\mathbf{P'})$ implies that $\mathscr{C}(\mathbf{P}) \leq \mathscr{C}(\mathbf{P'})$, since $\mathbf{P}$ is optimizing.

Therefore, $\mathscr{C}(\mathbf{P}) = \mathscr{C}(\mathbf{P'})$ and $\mathscr{L}(\mathbf{P}) = \mathscr{L}(\mathbf{P'})$. □

Remark. ***(Proof Approach for Theorem 2 Part 1)*** *For any optimizing* ***P****, we start by assuming it is already in water-filled form, since we have already shown that doing so does not unnecessarily restrict our space of optimizing solutions.*

Then, we would like to show that there exists a special choice of $\boldsymbol{Q}$ *such that* $\mathscr{C}(\boldsymbol{P}+\delta\boldsymbol{Q})+\alpha\mathscr{L}(\boldsymbol{P}+\delta\boldsymbol{Q})$*:*

1. *is linear over some well-defined interval of* δ *values around 0.*
2. *does not change with* δ *for any fixed* α
3. *results in protection scheme* $\boldsymbol{P}+\delta\boldsymbol{Q}$ *being strictly "more deterministic" (to be defined shortly) than* $\boldsymbol{P}$ *for a particular choice of* δ*.*

Definition 6. ***(Types of Matrix Entries)***

For the sake of discourse, we will define the following classifications of matrix entries in any protection scheme:

- *An entry is* fractional *if it is not equal to 0 or 1, and* integral *otherwise. Similarly, a column is fractional if its maximum entry is fractional and integral otherwise.*
- *An entry is* maxed out *if it is equal to the maximum value in its column, and* hanging *otherwise.*

Remark. *It is true by construction that a water-filled protection scheme will have at most one hanging mass entry and at least one maxed out entry in each row. Moreover, if a row has a hanging mass entry, there do not exist other non-zero entries further to the right of that entry.*

Remark. ***(Measure of Randomness)***

In order to compare which protection scheme, between two options, is "more deterministic", we rely on the following metric for randomness of a protection scheme:

$R(\boldsymbol{P})$ = (# fractional columns in $\boldsymbol{P}$) + (# hanging entries in $\boldsymbol{P}$)

Note that $R(\boldsymbol{P}) = 0$ if and only if $\boldsymbol{P}$ is a deterministic protection scheme.

We will now propose a particular choice of $\mathbf{Q}$ and δ, and prove the desired properties about these choices after.

Definition 7. ***(Q-Generation Procedure)***

Given the water-filled protection scheme $\boldsymbol{P}$ with at least one fractional entry, we now define a procedure to generate a $\boldsymbol{Q}$ matrix. Note that any such protection scheme must also have at least one fractional column or else it would contradict the water-filled property.

1. *Start with an $M \times N$ zero matrix $\boldsymbol{Q}$ that we will populate with values.*
2. *Denote the leftmost fractional column index in $\boldsymbol{P}$ as y. Further denote the current "sign" to "+".*
3. *In the yth column of $\boldsymbol{Q}$, if the sign is "+", assign the value 1 to all entries in that column that are maxed out in $\boldsymbol{P}$. If the sign is "-", assign the value -1 instead.*
4. *If the current sign is "+", change it to "-", and vice versa.*
5. *Consider the set of rows that are maxed out in the yth column of $\boldsymbol{P}$. Do all of these rows either have hanging mass in $\boldsymbol{P}$ or already have 2 non-zero entries in $\boldsymbol{Q}$? Depending on the answer:*
   - *If yes, go to step 9.*
   - *If no, then proceed to step 6.*

6. *Again consider the set of rows that are maxed out in the yth column of $\boldsymbol{P}$. Choose the topmost row from this set that does not have hanging mass in $\boldsymbol{P}$ and has only 1 non-zero entry in $\boldsymbol{Q}$. Denote the row index of that entry as x.*
7. *Set y to be the column index of the rightmost, maxed out entry of the xth row in the $\boldsymbol{P}$ matrix. Note that y must correspond to a fractional column here.*
8. *Go to step 3.*
9. *If any rows in $\boldsymbol{Q}$ have hanging mass and an odd number of non-zero entries, assign either 1 or -1, so that each of these rows sum to 0, to the hanging mass entries of these rows.*

Lemma 4. ***(Q-Generation Properties)*** *The procedure specified by definition 7 satisfies the following:*

1. *The procedure terminates.*
2. *All of the rows in the generated $\boldsymbol{Q}$ matrix sum to 0 (so that $\boldsymbol{P} + \delta\boldsymbol{Q}$ is a protection scheme).*
3. *$\boldsymbol{P} + \delta\boldsymbol{Q}$ is a water-filled protection scheme*

Lemma 4.1. Since we never choose columns that aren't fractional, any row selected in step 6 must have a maxed out entry (because we also ignore rows with fractional entries) somewhere to the right of the current y column. Certainly, this procedure must terminate if the y value ever reaches the right-most column (and the process may terminate earlier than that due to step 5). □

Lemma 4.2. Since we only assign 1 and -1 to entries of $\mathbf{Q}$ in alternation, this is the same as saying that each row must contain an even number of non-zero entries.

We see that this is true by noting that there are three types of rows, differentiated by how their non-zero entries in $\mathbf{Q}$ (if any) are assigned during the Q-generating procedure.

If a row has hanging mass in $\mathbf{P}$, then step 9 will necessarily adjust that row to have an even number of non-zero entries by construction. In addition, we never assign mass to hanging mass entries until step 9, when the procedure terminates, which means that all hanging mass entries are free for us to use at that point. So, rows that have hanging mass in $\mathbf{P}$ will be valid rows in $\mathbf{Q}$.

If a row has no hanging mass in $\mathbf{P}$, then there are two cases, depending on whether that row was ever used in step 6 to determine the next y value (we'll refer to such a row as a "critical" one). Note that, due to steps 5 and 6 filtering out rows that already have 2 non-zero entries, no row will ever be used in step 6 twice (i.e. a row will be a critical row at most once).

- If the row is critical, it must be the topmost one that had only one non-zero entry in $\mathbf{Q}$ at that point of the procedure in the previous yth column. Step 7 guarantees that the only other non-zero entry in this row will correspond to its rightmost non-zero entry in $\mathbf{P}$. So this row will have exactly 2 non-zero entries in $\mathbf{Q}$, making it valid.
- If the row is not critical, it must either be located below one that is or not have any non-zero entries in $\mathbf{Q}$ at all. The latter case results in a trivially valid row. In the former case, the row must have at least two non-zero entries in columns shared with the previous critical row, or else it would violate our assumptions that $\mathbf{P}$ is water-filled and the cost function is staircase nondecreasing. In addition, since $\mathbf{P}$ is water-filled, each row is majorized by all rows above it (i.e. the cumulative left-to-right sum of the upper row is no less than that of

the lower row for every column). This implies that a non-critical row cannot have more than 2 non-zero entries either.

□

Lemma 4.3. We observe that due to step 3, we only ever change all of the maxed out entries in a column together. So, for small δ, $\mathbf{P} + \delta\mathbf{Q}$ will remain water-filled. □

Definition 8. ***(Stopping Conditions)***

Recall from remark 3.2.1 that we require a particular choice of δ with various properties, as already described. We now define two choices of δ and justify properties about them in later lemmas.

Let $\delta_+ = \sup[\delta \geq 0 : \boldsymbol{P} + \delta\boldsymbol{Q}$ *is stochastic and* $\boldsymbol{P}$ *and* $\boldsymbol{P} + \delta\boldsymbol{Q}$ *are maxed out for the same entries and fractional for the same entries]*

and $\delta_- = \inf[\delta \leq 0 : \boldsymbol{P} + \delta\boldsymbol{Q}$ *is stochastic and* $\boldsymbol{P}$ *and* $\boldsymbol{P} + \delta\boldsymbol{Q}$ *are maxed out for the same entries and fractional for the same entries]*

Note that, by definition $\delta_+ > 0$ *and* $\delta_- < 0$.

Lemma 5. ***(Linearity Lemma)***

If $\boldsymbol{P}$ *is water-filled for fixed* α *and* $\boldsymbol{Q}$ *is generated according to definition 7, then* $\mathscr{C} + \alpha\mathscr{L}$ *evaluated with* $\boldsymbol{P} + \delta\boldsymbol{Q}$ *is linear with respect to* $\delta \in [\delta_-, \delta_+]$.

Lemma 5. For $\delta_- < \delta < \delta_+$ and fixed α,

$$\begin{aligned}
&\mathscr{C}(\mathbf{P}+\delta\mathbf{Q})+\alpha\mathscr{L}(\mathbf{P}+\delta\mathbf{Q}) \\
&=\sum_x\sum_y p(x)c(x,y)(p_{xy}+\delta q_{xy})+\alpha\sum_y\max_x(p_{xy}+\alpha q_{xy}) \\
&=\sum_x\sum_y p(x)c(x,y)(p_{xy}+\delta q_{xy})+\alpha\sum_y(p_{x(y)y}+\alpha q_{x(y)y})
\end{aligned}$$

where $x(y)=\arg\max_x p_{xy}$.

Since $\mathscr{C}(\mathbf{P}+\delta\mathbf{Q})+\alpha\mathscr{L}(\mathbf{P}+\delta\mathbf{Q})$ is linear over (δ_-,δ_+) and continuous over $[\delta_-,\delta_+]$, it is linear over $[\delta_-,\delta_+]$. □

Lemma 6. ***(No Improvement Lemma)***

If $\boldsymbol{P}$ *minimizes* $\mathscr{C}+\alpha\mathscr{L}$ *over* S *for fixed* α *and is water-filled and* $\boldsymbol{Q}$ *is generated according to definition 7, then* $\frac{\partial}{\partial\delta}(\mathscr{C}+\alpha\mathscr{L})=0$ *at* $\delta=0$.

Lemma 6. If $\frac{\partial}{\partial\delta}(\mathscr{C}+\alpha\mathscr{L})\neq 0$, then that implies that $\mathbf{P}+\delta\mathbf{Q}$ performs strictly better for some δ close to zero, which is a contradiction. □

Lemma 7. ***(More Deterministic Lemma)***

If $\boldsymbol{P}$ *is water-filled for fixed* α *and* $\boldsymbol{Q}$ *is generated according to Definition 7, then* $R(\boldsymbol{P}+\delta\boldsymbol{Q})<R(\boldsymbol{P})$ *for both* $\delta=\delta_-$ *or* $\delta=\delta_+$ *as defined by Definition 8.*

Lemma 7. As δ increases from 0 to δ_+, some fractional entries of $\mathbf{P}+\delta\mathbf{Q}$ change, and none of the integral entries change. In addition, if one maxed out entry changes, all of the maxed out entries in that column change together. It thus follows that the set of fractional columns can only decrease with δ and that the set of hanging entries likewise can only decrease. So $R(\mathbf{P})$ is nonincreasing in δ for $\delta\in[0,\delta_+]$. From the definition of δ_+ in Definition 8, $R(\mathbf{P}+\delta_+\mathbf{Q})<R(\mathbf{P})$.

Similarly, we can show that $R(\mathbf{P} + \delta\_\mathbf{Q}) < R(\mathbf{P})$. □

Theorem 2 Part 1. Any $\mathbf{P}$ that minimizes $\mathscr{C} + \alpha\mathscr{L}$ for some α can be chosen to be optimizing and water-filled as per Lemma 3. If $\mathbf{P}$ is not a deterministic protection scheme, we can select $\mathbf{Q}$ as in Definition 7 with the properties shown in Lemma 4.

By Lemmas 5, 6, 7, we know $\mathscr{C}(\mathbf{P}+\delta\mathbf{Q})+\alpha\mathscr{L}(\mathbf{P}+\delta\mathbf{Q})$ is constant over $[\delta_-, \delta_+]$ and $R(\mathbf{P} + \delta\mathbf{Q}) < R(\mathbf{P})$.

If $\mathbf{P} + \delta\mathbf{Q}$ is not deterministic, then we can repeat the above process since it is still water-filled and minimizes $\mathscr{C} + \alpha\mathscr{L}$ for the same α.

Eventually, after repeating this process some finite number of times, $R(\mathbf{P})$ will be 0 (since the function we defined is always nonnegative), and therefore deterministic. □

3.2.2 Proof of Theorem 2 Part 2

Theorem 2.2. Using standard convex analysis (e.g. [42], chapter 12), Theorem 2.1 implies that that $C^*(L)$ and $C_d^*(L)$ have the same lower semi-continuous hull (or the closure, as defined by [42] chapter 7), which is equivalent to our definition of the boundary of S. We can see this fact as follows:

First, we note that the left and right hand sides of the equality in Theorem 2.1 are the conjugate functions of $C^*(L)$ and $C_d^*(L)$, respectively. We have shown that the conjugates are equal for any α.

Second, since $C^*(L)$ is a convex function of L, the conjugate of the conjugate of $C^*(L)$ is equal to the closure of $C^*(L)$ ([42], Corollary 13.1.1).

Third, while $C_d^*(L)$ is not a convex function, its conjugate is the same as the conjugate of the closure of its convex hull. Therefore, the conjugate of its conjugate must be equal to the closure of its convex hull.

Thus, we have shown that the convex hulls of S and S_d are the same, since the two sets are the epigraphs of (all points in $\mathbb{R}^2$ on or above the curves defined by) the functions $C^*(L)$ and $C_d^*(L)$, respectively.

From this fact, it trivially follows that any (L, C) pair on the boundary of S must also lie on the convex hull of S, and therefore on the convex hull of S_d.

Finally, as previously noted, $C_d^*(L)$ is a descending staircase-like function for $L \in [1, \infty]$. So, the convex hull of S_d is given by the largest convex linear interpolation of the outer corner points of $C_d^*(L)$ (for example, see figure)

Therefore, any (L, C) pair on the boundary of S is achievable by a convex combination of no more than two deterministic protection schemes. □

3.3 Empirical Examples of MaxL-Optimal Schemes

Here, we present empirical case studies for MaxL-optimal schemes. We will show sample MaxL-optimal schemes for the RSA decryption side channel (the GMP implementation) and for a packet size side channel based on VoIP applications. We will find that these schemes are consistent with Theorem 2 in that they are convex combinations of deterministic schemes.

Empirical MaxL-optimal protection for RSA decryption: Using the

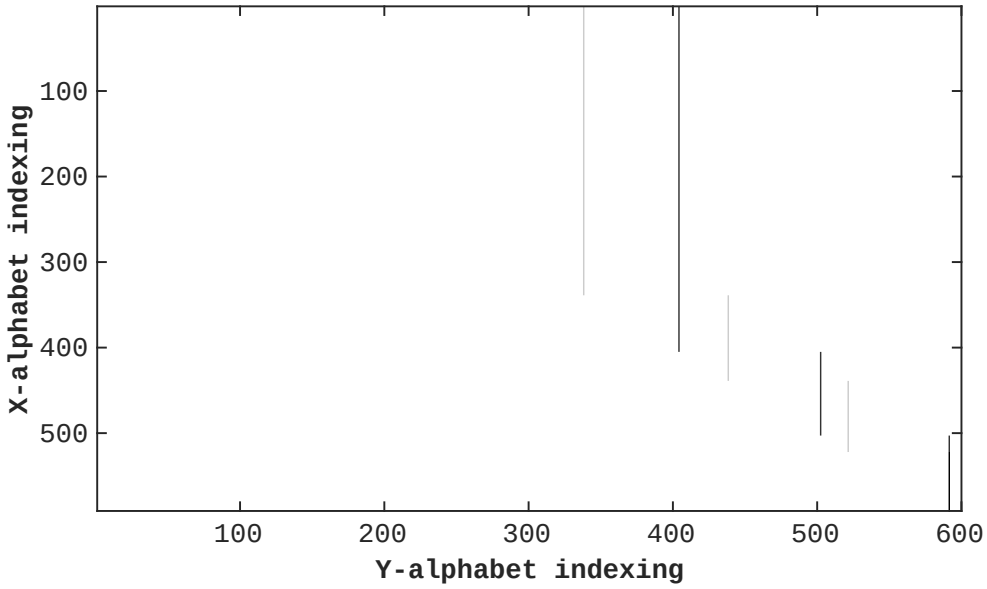


Figure 3.1: MaxL-optimal solution for 5% delay overhead on RSA decryption times. MaxL=1.6862 bits. Note MaxL equals mult-leakage

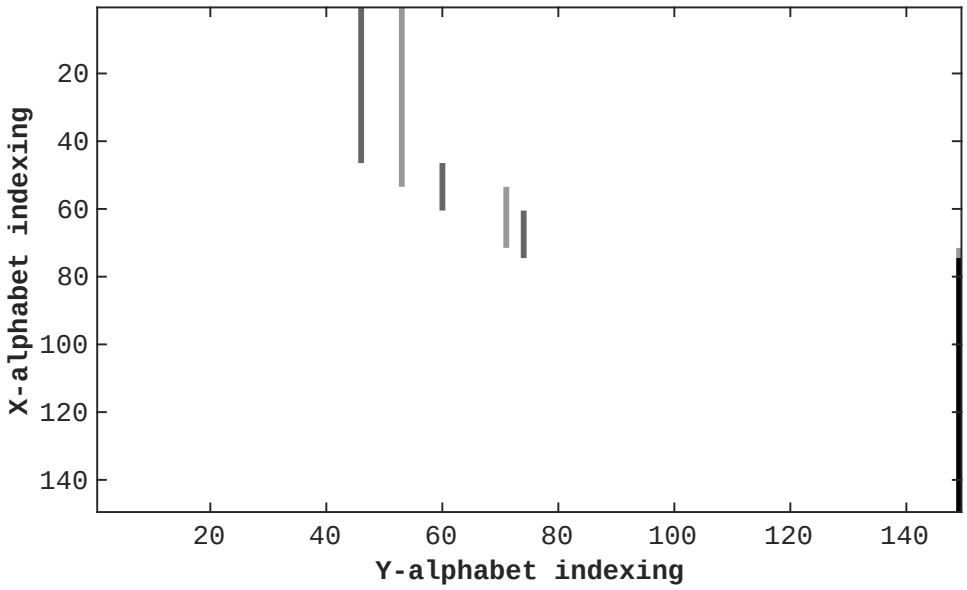


Figure 3.2: MaxL-optimal solution for 20% padding overhead on VoIP packet sizes. MaxL=1.8111 bits.

GMP-based decryption timing data from Section 4, we use Gurobi (a convex optimization solver) to solve the inverted optimization (i.e. minimize leakage subject to a cost bound) from Equation for a total cost bound of 5% delay overhead. Doing so achieves maximal leakage of 1.6862 bits. Note that in this type of side channel, one relevant class of deterministic protection schemes is *thresholding* where, for an ascending sequence of thresholds, all X values less than or equal to the smallest threshold are mapped to the first threshold, all X values greater than the first and less than or equal to the second threshold are mapped to the second, and so on. As it turns out, the resulting protection scheme from this experiment, is shown in Figure 3.1, and can be described as the combination of the thresholding schemes $\mathbf{P}_1$ with thresholds $x_{338}, x_{438}, x_{521}, x_{591}$ and $\mathbf{P}_2$ with thresholds $x_{404}, x_{502}, x_{591}$. This particular scheme can be implemented by a schedule that uses $\mathbf{P}_1$ 21.8% of the time and $\mathbf{P}_2$ 78.2% of the time.

Empirical MaxL-optimal protection for VoIP: Here we present the packet size channel for speech coding in the context of Voice-over-IP (VoIP) applications and then perform an analogous experiment to assess the MaxL-optimal scheme. Previous work demonstrated that there exists a side channel leak through the sizes of packets sent over networks[54]. It has even been shown that such side channels allow packet sniffers to partially recover or reconstruct spoken phrases [11, 55].

The victim system is a typical VoIP application, which operates by encoding fixed-length time intervals (called a "frame") of sound waveforms into one packet per time interval. In particular, VoIP system designers favor a form of variable bitrate (VBR) compression, which reduces bandwidth usage and improves recovered speech quality.

We assume that the adversary is interested in reconstructing the transcript, or the text of what was spoken. The adversary observes the final payload size of each packet, Y. X is the un-padded packet size produced by the speech codec (a coder-decoder used to compress and decompress human speech). Our protection scheme maps X to Y by padding each packet independently.

For our experiments, we use Mozilla's CommonVoice[1] English dataset. For the speech codec, we use Opus, an efficient open-source codec endorsed by the IETF, set to 24 kbps VBR with a frame size of 20 ms. Under these settings, Opus encodes each frame to one of 151 different packet sizes. For our experiments, we encoded approximately 572 hours worth of human speech to obtain the distribution $p(x)$.

Then, we set the cost matrix to be the number of bytes of padding for each packet ($c(x, y) = y - x$ if $y >= x$ and $c(x, y) = \infty$ otherwise), and again compute the inverted optimization using Gurobi. Solving the inverted optimization for a 20% padding overhead gives the solution seen in Figure 3.2, and it can be decomposed into two deterministic protection schemes in a similar fashion to the RSA optimization.

3.4 A Heuristic Algorithm

In this section, we will address the dimensionality of the LP. For alphabet $|\mathcal{X}| = N$ and $|\mathcal{Y}| = M$, the constrained optimization in Equation 3.3 is over an $N \times M$ variable matrix. So, the alphabet sizes of X and Y are intimately linked to the dimensionality of the LP and can greatly affect computational complexity. It may

[1]https://voice.mozilla.org/en

be possible to reduce the problem size by grouping symbols in X or in Y together, thereby reducing N and M. However, doing so incurs additional cost by some hard-to-measure quantity and is not always practical. For such cases, we present a heuristic algorithm that can be used to approximate the full trade-off curve.

3.4.1 Greedy Algorithm

Definition 9. *For any nonempty set $\mathcal{S} \subseteq \mathcal{Y}$ and cost matrix $\{c(x,y)\}$, we define a deterministic protection scheme $\boldsymbol{P}_{\mathcal{S}} = \{p_{xy}\}$ such that:*

$$p_{xy} = \begin{cases} 1 & \text{if } y = \min \arg\min_{y' \in \mathcal{S}} c(x,y') \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

We refer to $\boldsymbol{P}_{\mathcal{S}}$ as the deterministic protection scheme induced by *the subset $\mathcal{S}$.*

Definition 10. *For any non-empty set $\mathcal{S} \subseteq \mathcal{Y}$, let:*

$$\mathscr{L}(\mathcal{S}) = \mathscr{L}(\boldsymbol{P}_{\mathcal{S}}) \text{ and } \mathscr{C}(\mathcal{S}) = \mathscr{C}(\boldsymbol{P}_{\mathcal{S}}) \tag{3.4}$$

Definition 11. *For a given staircase nondecreasing cost matrix $\{c(x,y)\}$, we identify one (not necessarily unique) $y_0 \in \mathcal{Y}$ such that:*

$$y_0 = \arg\min_{y \in \mathcal{Y}} \mathscr{C}(\{y\}) \tag{3.5}$$

Define the subset $\mathcal{Y}' = \mathcal{Y} - \{y_0\}$.

Definition 12. *For any set $\mathcal{A} \subseteq \mathcal{Y}'$, we define the set function:*

$$f(\mathcal{A}) = -\mathscr{C}(\mathcal{A} \cup \{y_0\}) \tag{3.6}$$

Definition 13. *Here, we define a greedy algorithm to construct a sequence of deterministic protection schemes as follows:*

1. *Start with* $\mathcal{A} = \{\emptyset\}$.
2. *Choose* $y \in \mathcal{Y}' - \mathcal{A}$ *such that* $f(\mathcal{A} \cup \{y\})$ *is maximized over all such choices of* y. *If* $\mathcal{Y}' - \mathcal{A}$ *is empty or if there does not exist such* y *that* $f(\mathcal{A} \cup \{y\}) > f(\mathcal{A})$, *terminate this algorithm.*
3. *Set* $\mathcal{A} = \mathcal{A} \cup \{y\}$.
4. *Go to step 2.*

3.4.2 Bounded Sub-optimality of the Greedy Algorithm

Using standard results in combinatorial optimization [38], we can obtain bounds on how suboptimal the solutions obtained from the greedy algorithm are. We will first prove some basic facts about the set function $f(\mathcal{A})$ given in Definition 12.

Lemma 8. *$f(\mathcal{A})$ is submodular.*

Proof. For $\mathcal{A}, \mathcal{B} \subseteq \mathcal{Y}'$ such that $\mathcal{A} \cap \mathcal{B} = \{\emptyset\}$,

$$\begin{aligned} f(\mathcal{A} \cup \mathcal{B}) &= -\sum_{x \in \mathcal{X}} \min_{y \in \mathcal{A} \cup \mathcal{B} \cup \{y_0\}} p(x)c(x,y) \\ &= -\sum_{x \in \mathcal{X}} \min_{y \in \mathcal{A} \cup \{y_0\}} p(x)c(x,y) + \sum_{x \in \mathcal{X}} \min_{y \in \mathcal{A} \cup \{y_0\}} p(x)c(x,y) \\ &\qquad - \sum_{x \in \mathcal{X}} \min_{y \in \mathcal{A} \cup \mathcal{B} \cup \{y_0\}} p(x)c(x,y) \\ &= f(\mathcal{A}) + \sum_{x \in \mathcal{X}} p(x)[\min_{y \in \mathcal{A} \cup \{y_0\}} c(x,y) - \min_{y \in \mathcal{A} \cup \mathcal{B} \cup \{y_0\}} c(x,y)] \\ &\equiv f(\mathcal{A}) + D(A,B) \end{aligned}$$

Then, for $\mathcal{A} \subseteq \mathcal{Y}'$ and $b, c \in \mathcal{Y}' \backslash \mathcal{A}$,

$$\begin{aligned}
& f(\mathcal{A} \cup \{b\}) + f(\mathcal{A} \cup \{c\}) - f(\mathcal{A} \cup \{b, c\}) - f(\mathcal{A}) \\
&= D(\mathcal{A}, \{b\}) + D(\mathcal{A}, \{c\}) - D(\mathcal{A}, \{b, c\}) \\
&= \sum_{x \in \mathcal{X}} p(x)[\min_{y \in \mathcal{A} \cup \{y_0\}} c(x, y) - \min_{y \in \mathcal{A} \cup \{b, y_0\}} c(x, y) \\
&\qquad - \min_{y \in \mathcal{A} \cup \{c, y_0\}} c(x, y) + \min_{y \in \mathcal{A} \cup \{b, c, y_0\}} c(x, y)] \\
&\equiv \sum_{x \in \mathcal{X}} p(x)[C_1 - C_2 - C_3 + C_4] \\
&\geq 0
\end{aligned}$$

since C_4 is equal to C_2 or C_3 (or both), and C_1 is no smaller than either C_2 or C_3. Hence,

$$f(\mathcal{A} \cup \{b\}) + f(\mathcal{A} \cup \{c\}) \geq f(\mathcal{A} \cup \{b, c\}) + f(\mathcal{A}) \tag{3.7}$$

so $f(\mathcal{A})$ is submodular ([44], Thm 44.1).

□

Definition 14. *For integer exp-leak bound L, let $\mathcal{A}_g(L)$ be the set obtained by running the greedy algorithm unil $|\mathcal{A} \cup \{y_0\}| = L$ (for simplicity, assume the greedy algorithm does not terminate prior to this point).*

For integer exp-leak bound L, let $\mathcal{A}^*(L) \subseteq \mathcal{Y}'$ be the true optimal set such that $f(\mathcal{A})$ is maximized subject to $|\mathcal{A} \cup \{y_0\}| \leq L$.

Now, since $f(\mathcal{A})$ is submodular, we can bound the greedy algorithm for all $L \geq 2$ as follows ([38], Theorem 4.1):

$$\begin{aligned}
\frac{f(\mathcal{A}^*(L)) - f(\mathcal{A}_g(L))}{f(\mathcal{A}^*(L)) - f(\{\emptyset\})} &= \frac{\mathscr{C}(\mathcal{A}_g(L) \cup \{y_0\}) - \mathscr{C}(\mathcal{A}^*(L) \cup \{y_0\})}{\mathscr{C}(\{y_0\}) - \mathscr{C}(\mathcal{A}^*(L) \cup \{y_0\})} \\
&\leq \left(\frac{L-2}{L-1}\right)^{L-1} \leq \frac{1}{e}
\end{aligned} \tag{3.8}$$

The greedy algorithm is capable of approximating a full cost-leakage trade-off curve more quickly, compared to running as many as M individual LP optimizations. The difference in computation time increases with the size of $|\mathcal{Y}|$; the greedy algorithm runs on the order of 30 times faster than the LP on the integer exp-leakage points for our larger experiments, but only on the order of 5 times faster for our smaller experiments.

Moreover, we have shown that the cost of deterministic protection schemes computed by the greedy algorithm is bounded relative to the true optimal protection schemes at the same leakage levels. Finally, a useful side-effect of this bound is that the true optimal scheme does not perform any better than the greedy algorithm after a single iteration (when $L = 2$), which follows from Equation 3.8. As there exist many applications that require close to no leakage and since a single step of the greedy algorithm (computing for $L = 2$) merely consists of a $O(N)$ search over the space of $\mathcal{Y}'$, these protection schemes with exp-leak between 1 and 2 can be easily computed since we know from Theorem 2 that the optimal protection scheme is simply a convex combination of the two deterministic protection schemes.

3.4.3 Sub-optimality of the Greedy Algorithm in Practice

Here, we reuse the RSA decryption timing data and VoIP packet size data to demonstrate that the gap between the true optimal curve and the greedy algorithm is in fact very small. Indeed, we find that for these case studies, the error of the greedy rate is far below the projected error given by Equation 3.8. Thanks to Theorem 2, we obtain the true optimal curve by using Gurobi to optimize over maximal leakage at integer exp-leak points. We use the greedy algorithm to obtain

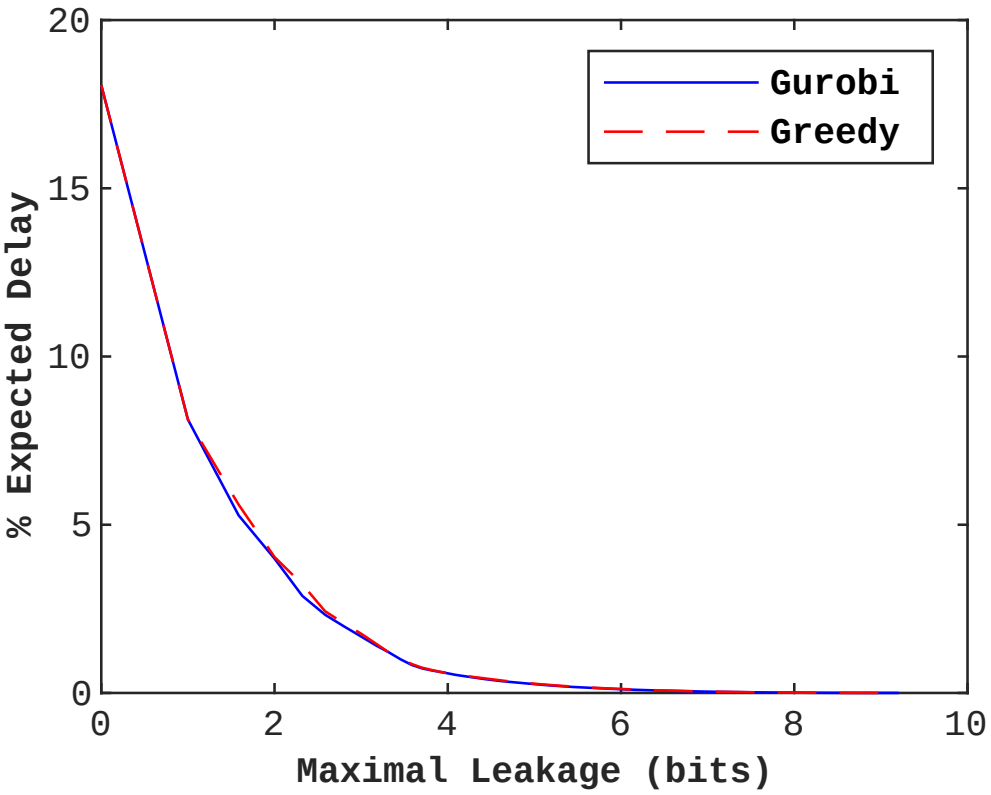


Figure 3.3: Optimal trade-off curve for 16-bit GMP RSA. The horizontal axis is the leakage bound in bits(the log of L from Equation 3.3) and the vertical axis is the percent expected delay normalized over the baseline expected decryption time (5,246.3 cycles).

an approximately optimal curve. These results can be seen in Figures 3.3 and 3.4.

3.5 Summary

To conclude the previous two chapters, the major takeaway is that one-shot side channels under maximal leakage are optimally protected by a linear combination of no more than deterministic protection schemes.

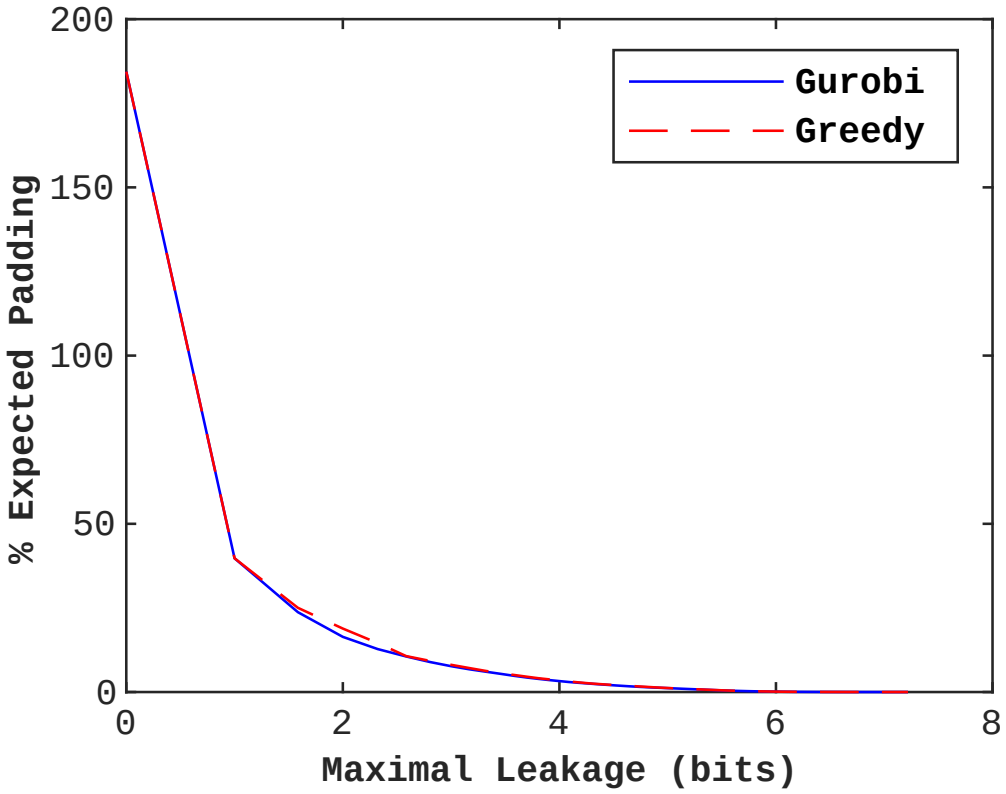


Figure 3.4: Analogous trade-off curve for VoIP packet sizes. Here, the vertical axis is the percent expected padding normalized over the baseline expected packet size (54.15 bytes).

CHAPTER 4

STRONG ASYMPTOTIC COMPOSITION THEOREMS FOR ALPHA MAXIMAL LEAKAGE

4.1 Introduction

In the context of information leakage, composition theorems characterize how leakage increases as a result of multiple, independent, noisy observations of the sensitive data. Equivalently, they characterize how security (or privacy) degrades under the "composition" of multiple observations (or queries). In practice, attacks are often sequential in nature, whether the application is side channels in computer security [29, 51, 62] or database privacy [27, 13, 36]. Thus composition theorems are practically relevant. They also raise theoretical questions that are interesting in their own right.

Various composition theorems for differential privacy and its variants have been established (e.g., [27, 13, 36]). For the information-theoretic metrics of mutual information and maximal leakage [25, 21, 23, 22] (throughout we assume discrete alphabets and base-2 logarithms)

$$I(X;Y) = \sum_{x,y} P(x,y) \log \frac{P(x,y)}{P(x)P(y)} \tag{4.1}$$

$$\mathcal{L}(X \to Y) = \log \sum_{y} \max_{x:P(x)>0} P(y|x), \tag{4.2}$$

and α-maximal leakage [32], less is known. While some results are available in the case that $P(y|x)$ is not known [46], here we assume it is known. For the metrics in (4.1)-(4.2) it is straightforward to show the "weak" composition theorem that

if $Y_1, \ldots, Y_n$ are conditionally independent given X, then

$$I(X;Y^n) \leq \sum_{i=1}^{n} I(X;Y_i)$$
$$\mathcal{L}(X \to Y^n) \leq \sum_{i=1}^{n} \mathcal{L}(X \to Y_i).$$

These bounds are indeed weak in that if $Y_1, \ldots, Y_n$ are conditionally i.i.d. given X, then as $n \to \infty$, the right-hand sides generally tend to infinity while the left-hand sides remain bounded. A "strong" (asymptotic) composition theorem would identify the limit and characterize the speed of convergence.

We prove such a result for both mutual information and maximal leakage. The limits are readily identified as the entropy and log-support size, respectively, of a minimal sufficient statistic of Y given X. In both cases, the speed of convergence to the limit is exponential, and the exponent is the same. Specifically, it is the minimum Chernoff information among all pairs of distinct distributions $Q_{Y|X}(\cdot|x)$ and $Q_{Y|X}(\cdot|x')$.

Mutual information and maximal leakage are both instances of Sibson mutual information [45, 50, 22], the former being order 1 and the latter being order ∞. The striking fact that the exponents governing the convergence to the limit are the same at these two extreme points suggests that Sibson mutual information of all orders satisfies a strong asymptotic composition theorem, with the convergence rate (but not the limit) being independent of the order. Meanwhile, Shannon mutual information can also be viewed as Arimoto mutual information of order 1 [4], and α-maximal leakage is equivalently expressed as a maximization of Sibson or Arimoto mutual information of order α over $P(X)$ for $\alpha > 1$; for $\alpha = 1$, it equals Shannon mutual information [32], as opposed to the Shannon capacity. Due to the intimate interrelation between these measures, it is reasonable to suspect that

similar strong asymptotic composition theorems obtain for them all. Indeed, we prove strong composition theorems for Sibson mutual information, Arimoto mutual information, and α-maximal leakage, for all orders of at least unity. In particular, we find that they all approach their respective limits at the same α-independent exponential rate, namely the minimum Chernoff information mentioned earlier.

The composition theorems proven here are different in nature from those in the differential privacy literature. Here we assume that the relevant probability distributions are known, and we characterize the growth of leakage with repeated looks from those distributions. We also assume that $Y_1, \ldots, Y_n$ are conditionally i.i.d. given X. Composition theorems in differential privacy consider the worst-case distributions given leakage levels for each of $Y_1, \ldots, Y_n$ individually, assuming only conditional independence.

Although our motivation is averaging attacks in side channels, the results may have some use in capacity studies of channels with multiple conditionally i.i.d. outputs given the input [7, Prob. 7.20].

4.2 Sibson, Arimoto, Rényi, and Chernoff

This study relies on both Sibson's and Arimoto's tunable mutual information metrics as well as α-maximal leakage. All random variables in the paper are assumed discrete.

Definition 15 ([45, 50])**.** *The* Sibson mutual information of order α *between random variables X and Y is defined by*

$$I_\alpha^S(X;Y) = \frac{\alpha}{\alpha - 1} \log \sum_{y \in \mathcal{Y}} \Big(\sum_{x \in \mathcal{X}} P(x) P(y|x)^\alpha \Big)^{1/\alpha}, \tag{4.3}$$

for $\alpha \in (0,1) \cap (1,\infty)$ *and for* $\alpha = 1$ *and* $\alpha = \infty$ *by its continuous extensions. These are*

$$I_1^S(X;Y) = I(X;Y)$$
$$I_\infty^S(X;Y) = \mathcal{L}(X \to Y),$$

defined in (4.1)-(4.2) above.

Definition 16 ([4])**.** *The* Arimoto mutual information of order α *between random variables* X *and* Y *is defined by*

$$I_\alpha^A(X;Y) = \frac{\alpha}{\alpha - 1} \log \sum_{y \in \mathcal{Y}} \Big(\frac{\sum_{x \in \mathcal{X}} P(x)^\alpha P(y|x)^\alpha}{\sum_{x \in \mathcal{X}} P(x)^\alpha} \Big)^{1/\alpha} \tag{4.4}$$

for $\alpha \in (0,1) \cap (1,\infty)$ *and for* $\alpha = 1$ *and* $\alpha = \infty$ *by its continuous extensions. Note that [4]*

$$I_1^A(X;Y) = I(X;Y)$$

but

$$I_\infty^A(X;Y) \neq \mathcal{L}(X \to Y).$$

Definition 17 ([32])**.** *The* α-maximal leakage *for* $\alpha \in (1,\infty]$ *is equivalently defined using either Sibson or Arimoto mutual information as:*[1]

$$\mathcal{L}_\alpha^{max}(X \to Y) = \max_{Q(X)} I_\alpha^S(X;Y) = \max_{Q(X)} I_\alpha^A(X;Y), \tag{4.5}$$

where the maxima are over all distributions of X *that have full support. For* $\alpha = 1$*, we have*

$$\mathcal{L}_\alpha^{max}(X \to Y) = I(X;Y). \tag{4.6}$$

as opposed to the (Shannon) capacity

$$\mathcal{C}(X;Y) = \max_{Q(X)} I(X;Y). \tag{4.7}$$

[1]The second equality for $1 < \alpha < \infty$ in (4.5) is apparent from (4.3) and (4.4) since the tilting of $P(x)$ in the latter can be absorbed into the maximization.

Liao *et al.* [32] define α-maximal leakage operationally. The identities in (4.5)-(4.6) are a theorem in that work, which we shall take as a definition. Likewise, Issa *et al.* [25] define maximal leakage operationally, and (4.2) is a theorem that we take as a definition.

We are interested in how $I_\alpha^S(X;Y^n)$, $I_\alpha^A(X;Y^n)$, and $\mathcal{L}_\alpha^{max}(X \to Y^n)$ grow with n when $Y_1, \ldots, Y_n$ are conditionally i.i.d. given X for $\alpha \geq 1$. The question for $\alpha < 1$ is meaningful in all cases but is not considered here because we are interested in the behavior of operational leakage measures, and the $\alpha < 1$ regime is not known to be relevant to measuring leakage. We do not consider the mutual information meaures put forward by Csiszár [8] and Lapidoth and Pfister [30, 31] for the same reason. For the quantities under study, we shall see that the limits are given by *Rényi entropy*. As they will be needed for proof later, we also define *Arimoto-Rényi conditional entropy* and *Rényi divergence*.

Definition 18. *The* Rényi entropy *of order α of a random variable X is given by:*

$$H_\alpha(X) = \frac{1}{1-\alpha} \log \sum_{x \in \mathcal{X}} P(x)^\alpha \tag{4.8}$$

for $\alpha \in (0,1) \cap (1,\infty)$ and for $\alpha = 0$, $\alpha = 1$, and $\alpha = \infty$ by its continuous extensions. These are

$$H_0(X) = \log |\{x : P(x) > 0\}| \tag{4.9}$$

$$H_1(X) = H(X) \tag{4.10}$$

$$H_\infty(X) = \log \frac{1}{\max_x P(x)}. \tag{4.11}$$

where $H(X)$ is the regular Shannon entropy.

Definition 19. *The* Arimoto-Rényi conditional entropy *of order α of a random variable X given Y is defined as:*

$$H_\alpha(X|Y) = \frac{\alpha}{1-\alpha} \log \sum_{y \in \mathcal{Y}} \Big(\sum_{x \in \mathcal{X}} P(x)^\alpha P(y|x)^\alpha \Big)^{\frac{1}{\alpha}}. \tag{4.12}$$

Remark. *One can verify that it holds*

$$I_\alpha^A(X;Y) = H_\alpha(X) - H_\alpha(X|Y). \tag{4.13}$$

Definition 20. *The* Rényi divergence *of order α between probability distributions P and Q is defined for $\alpha \in [0,\infty)$, $\alpha \neq 1$ as:*

$$D_\alpha(P||Q) = \frac{1}{\alpha - 1} \log \sum_{x \in \mathcal{X}} P(x)^\alpha Q(x)^{1-\alpha}, \tag{4.14}$$

where the continuous extension at $\alpha = 1$ is given by the standard Kullback-Leibler divergence

$$D(P||Q) = \sum_{x \in \mathcal{X}} P(x) \log \frac{P(x)}{Q(x)}. \tag{4.15}$$

The speed of convergence of $I_\alpha^S(X;Y^n)$, $I_\alpha^A(X;Y^n)$, $\mathcal{L}_\alpha^{max}(X \to Y^n)$, and $\mathcal{C}(X;Y^n)$ and to their respective limits turns out to be governed by *Chernoff information.*

Definition 21 ([7])**.** *The* Chernoff information *between two probability mass functions, P_1 and P_2, over the same alphabet $\mathcal{X}$ is given as follows. First, for all $x \in \mathcal{X}$ and $\lambda \in [0,1]$, let:*

$$P_\lambda(x) = P_\lambda(P_1, P_2, x) = \frac{P_1(x)^\lambda P_2(x)^{1-\lambda}}{\sum_{x' \in \mathcal{X}} P_1(x')^\lambda P_2(x')^{1-\lambda}}. \tag{4.16}$$

Then the Chernoff information is given by

$$\mathscr{C}(P_1||P_2) = D(P_{\lambda^*}||P_1) = D(P_{\lambda^*}||P_2), \tag{4.17}$$

where λ^* is any value of λ such that the above two relative entropies are equal. Equivalently, the Chernoff information is also given by:

$$\mathscr{C}(P_1||P_2) = -\min_{0 \leq \lambda < 1} \log \left(\sum_x P_1(x)^\lambda P_2(x)^{1-\lambda} \right) \tag{4.18}$$

Since we consider finite alphabets, the Chernoff information is infinite if and only if P_1 and P_2 have disjoint support.

Other Notation: We use $\mathcal{P}_n$ to denote the set of all possible empirical distributions of Y^n. We let $\mathcal{P}$ denote the set of all possible probability distributions over $\mathcal{Y}$ For any $P \in \mathcal{P}$, let

$$T(P) = \{y^n \in \mathcal{Y}^n | P_{y^n} = P\},$$

where P_{y^n} is the empirical distribution of y^n. Note that $T(P)$ is empty if $P \notin \mathcal{P}_n$. We use $Q(\cdot)$ to denote the true distributions of X and Y^n. We let Q_x denote the distribution of Y given x for a given $x \in \mathcal{X}$. For any $P \in \mathcal{P}$, let $x_k(P)$ denote $x \in \mathcal{X}$ such that $D(P||Q_x)$ is the k^{th} smallest relative entropy across all elements of $\mathcal{X}$. Ties can be broken by the ordering of $\mathcal{X}$.

We also define x-domains for fixed n in two slightly different ways. Let

$$D_x = \{P \in \mathcal{P} | D(P||Q_x) < D(P||Q_{x'}) \ \forall x' \neq x\} \tag{4.19}$$

$$\bar{D}_x = \{P \in \mathcal{P} | D(P||Q_x) \leq D(P||Q_{x'}) \ \forall x' \in \mathcal{X}\} \tag{4.20}$$

Note that for any $P \in \bar{D}_x$, $D(P||Q_x) = \min_{x' \in \mathcal{X}} D(P||Q_{x'})$.

4.3 The Result

Let X be a random variable with alphabet $\mathcal{X} = \{x_1, x_2, ...x_{|\mathcal{X}|}\}$. Let $Y^n = (Y_1, Y_2, ...Y_n)$ be a vector of discrete random variables with a shared alphabet $\mathcal{Y} = \{y_1, y_2, ...y_{|\mathcal{Y}|}\}$. We assume that $Y_1, Y_2, \ldots, Y_n$ are conditionally i.i.d. given X. We may assume, without loss of generality, that X and Y have full support.

We will also assume that the distributions $P_{Y|X}(\cdot|x)$ are unique over x, which we call the *unique row assumption*. For Sibson mutual information and α-max leakage, this is without loss of generality, since we can divide $\mathcal{X}$ into equivalence classes based on their respective $P_{Y|X}(\cdot|x)$ distributions and define $\tilde{X}$ to be the equivalence class of X. Then both Markov chains $X \leftrightarrow \tilde{X} \leftrightarrow Y^n$ and $\tilde{X} \leftrightarrow X \leftrightarrow Y^n$ hold and so

$$I_\alpha^S(X;Y^n) = I_\alpha^S(\tilde{X};Y^n) \tag{4.21}$$

$$\mathcal{L}_\alpha^{max}(X \to Y^n) = \mathcal{L}_\alpha^{max}(\tilde{X} \to Y^n), \tag{4.22}$$

by the data processing inequality for Sibson mutual information [41] and α-maximal leakage [32, Thm. 3]. We may then replace X with $\tilde{X}$ in the case of these measures. For Arimoto mutual information, the chain rule does not hold, and in fact an arbitrarily large discrepancy can exist between $I_\alpha^A(X;Y)$ and $I_\alpha^A(\tilde{X};Y)$, as shown in Section 4.5, where it is also shown that the unique row assumption is nonetheless still without loss of generality.

Our measures of interest satisfy the following upper bounds:

$$I(X;Y^n) \le H(X) \tag{4.23}$$

$$\mathcal{C}(X;Y^n) \le \log|\mathcal{X}| \tag{4.24}$$

$$I_\alpha^S(X;Y^n) \le H_{1/\alpha}(X) \quad \text{[50, Ex. 2 and Thm. 3]} \tag{4.25}$$

$$I_\alpha^A(X;Y^n) \le H_\alpha(X) \tag{4.26}$$

$$\begin{aligned}\mathcal{L}_\alpha^{max}(X \to Y^n) &\le \begin{cases} H(X) & \text{if } \alpha = 1 \\ \log|\mathcal{X}| & \text{if } \alpha > 1 \end{cases} \quad \text{[32, Thm. 3]} \\ &=: \mathcal{L}_\alpha(X),\end{aligned} \tag{4.27}$$

where each inequality holds for all n and all $\alpha \in [1,\infty]$. Comparing (4.25) and (4.26) suggests that perhaps the Arimoto mutual information of order α should

be associated with the Sibson mutual information of order $1/\alpha$; the identity in (4.5) suggests otherwise.

Our main result describes how fast these upper bounds are approached as $n \to \infty$.

Theorem 9. *Under the unique row assumption, for all* $\alpha \in [1, \infty]$,

$$\min_{x \neq x'} \mathscr{C}(Q_x || Q_{x'}) = \lim_{n\to\infty} -\frac{1}{n} \log \Big(H(X) - I(X; Y^n) \Big) \tag{4.28}$$

$$= \lim_{n\to\infty} -\frac{1}{n} \log \Big(\log |\mathcal{X}| - \mathcal{C}(X; Y^n) \Big) \tag{4.29}$$

$$= \lim_{n\to\infty} -\frac{1}{n} \log \Big(H_{1/\alpha}(X) - I^S_\alpha(X; Y^n) \Big) \tag{4.30}$$

$$= \lim_{n\to\infty} -\frac{1}{n} \log \Big(H_\alpha(X) - I^A_\alpha(X; Y^n) \Big) \tag{4.31}$$

$$= \lim_{n\to\infty} -\frac{1}{n} \log \Big(\mathcal{L}_\alpha(X) - \mathcal{L}^{max}_\alpha(X \to Y^n) \Big). \tag{4.32}$$

Thus the Chernoff information governs the exponential rate-of-approach for all measures and for all values of α. This Chernoff information is infinite if Q_x and $Q_{x'}$ have disjoint support for all $x \neq x'$; in this case, the bounds in (4.23)-(4.27) are met with equality already for $n = 1$. Channels with this property arise naturally in certain applications [57].

Observe that (4.30)-(4.32) coincide with (4.28) when $\alpha = 1$. Also, (4.30) and (4.32) coincide for $\alpha = \infty$; otherwise the assertions are independent.

For continuous random variables, it is meaningful and interesting to study how $I^S_\alpha(X; Y^n)$, $\mathcal{C}(X; Y^n)$, and $\mathcal{L}^{max}_\alpha(X \to Y^n)$ grow with n. The behavior would be fundamentally different from the discrete case, however. See Aishwarya and Madiman [2] for a discussion of Arimoto mutual information in the continuous case.

The remainder of the paper is devoted to proving the various assertions contained within Theorem 9. The assertions are evidently asymptotic in nature, and our proofs are not opimtized to provide the best finite-n bounds. Numerical experiments show that in many cases our lower and upper bounds are quite far apart for moderate values of n.

4.4 An Ancillary Lemma

Recall that Q_x denotes the distribution of Y given x, and for any $P \in \mathcal{P}$, $x_k(P)$ denotes $x \in \mathcal{X}$ such that $D(P||Q_x)$ is the k^{th} smallest relative entropy across all elements of $\mathcal{X}$.

Lemma 10.

$$\inf_{P \in \mathcal{P}} D(P||Q_{x_2(P)}) = \min_{x \neq x'} \mathscr{C}(Q_x||Q_{x'}), \tag{4.33}$$

where both quantities may be infinite.

Proof. We will separately prove that

$$\inf_{P \in \mathcal{P}} D(P||Q_{x_2(P)}) \leq \min_{x \neq x'} \mathscr{C}(Q_x||Q_{x'}) \tag{4.34}$$

and

$$\inf_{P \in \mathcal{P}} D(P||Q_{x_2(P)}) \geq \min_{x \neq x'} \mathscr{C}(Q_x||Q_{x'}). \tag{4.35}$$

To prove the upper bound, fix $x \neq x'$ and consider $P_\lambda(y) = P_\lambda(Q_x, Q_{x'}, y)$ as defined in (4.16). Choose λ^* such that $D(P_{\lambda^*}||Q_x) = D(P_{\lambda^*}||Q_{x'})$. Then, certainly

$$D(P_{\lambda^*}||Q_{x_2(P_{\lambda^*})}) \leq \mathscr{C}(Q_x||Q_{x'}) \tag{4.36}$$

since we know of two X-values whose corresponding $Q(Y|X)$ distributions are equidistant to P_{λ^*}, from which (4.34) follows.

For the lower bound, we first define subsets of $\mathcal{P}$:

$$E_x = \{P \in \mathcal{P} \mid D(P||Q_x) \leq \mathscr{C}(Q_x||Q_{x'})\} \tag{4.37}$$

$$E_{x'} = \{P \in \mathcal{P} \mid D(P||Q_{x'}) \leq \mathscr{C}(Q_x||Q_{x'})\} \tag{4.38}$$

Note that E_x and $E_{x'}$ are convex sets since $D(\cdot||\cdot)$ is convex and that P_{λ^*} achieves the minimum distance to $Q_{x'}$ in E_x and the minimum distance to Q_x in $E_{x'}$ [7, Sec. 11.9].

Choose any $P \in \mathcal{P}$. There are three cases to consider, depending on the location of P in $\mathcal{P}$-space.

Case 1: $P \notin E_x$ and $P \notin E_{x'}$. By construction, $D(P||Q_x) \geq \mathscr{C}(Q_x||Q_{x'})$ and $D(P||Q_{x'}) \geq \mathscr{C}(Q_x||Q_{x'})$.

Case 2: $P \in E_x$. Using the Pythagorean theorem for relative entropy [7, Thm. 11.6.1],

$$D(P||Q_{x'}) \geq D(P||P_{\lambda^*}) + D(P_{\lambda^*}||Q_{x'}) \tag{4.39}$$

Case 3: $P \in E_{x'}$. By the same argument,

$$D(P||Q_x) \geq D(P||P_{\lambda^*}) + D(P_{\lambda^*}||Q_x) \tag{4.40}$$

Hence, for any $P \in \mathcal{P}$,

$$\max\{D(P||Q_x), D(P||Q_{x'})\} \geq \mathscr{C}(Q_x||Q_{x'}) \tag{4.41}$$

Since $D(P||Q_{x_2(P)}) = \min\limits_{x \neq x'} \max\{D(P||Q_x), D(P||Q_{x'})\}$,

$$\inf_{P\in\mathcal{P}} D(P||Q_{x_2(P)}) \geq \min_{x\neq x'} \mathscr{C}(Q_x||Q_{x'}). \tag{4.42}$$

□

The following result is standard; we provide a proof for completeness.

Lemma 11. *For any discrete distributions P_1 and P_2 on a common alphabet $\mathcal{X}$,*

$$\mathscr{C}(P_1^n||P_2^n) = n\mathscr{C}(P_1||P_2) \tag{4.43}$$

Proof. From (4.18),

$$\mathscr{C}(P_1||P_2) = -\min_{0\leq\lambda<1} \log\left(\sum_x P_1(x)^\lambda P_2(x)^{1-\lambda}\right). \tag{4.44}$$

Furthermore,

$$\log\left(\sum_{x^n} P_1(x^n)^\lambda P_2(x^n)^{1-\lambda}\right) \tag{4.45}$$

$$= \log\left(\sum_{x_1}\sum_{x_2}\ldots\sum_{x_n}\prod_i^n P_1(x_i)^\lambda P_2(x_i)^{1-\lambda}\right) \tag{4.46}$$

$$= \log\left(\prod_i^n\sum_{x_i} P_1(x_i)^\lambda P_2(x_i)^{1-\lambda}\right) \tag{4.47}$$

$$= \log\left(\sum_{x\in\mathcal{X}} P_1(x)^\lambda P_2(x)^{1-\lambda}\right)^n. \tag{4.48}$$

Hence,

$$\mathscr{C}(P_1^n||P_2^n) = -\min_{0\leq\lambda<1} \log\left(\sum_{x^n} P_1(x^n)^\lambda P_2(x^n)^{1-\lambda}\right) \tag{4.49}$$

$$= -\min_{0\leq\lambda<1} n\log\left(\sum_{x\in\mathcal{X}} P_1(x)^\lambda P_2(x)^{1-\lambda}\right) \tag{4.50}$$

$$= n\mathscr{C}(P_1||P_2). \tag{4.51}$$

□

4.5 Data Processing for Arimoto Mutual Information

As a generalization of Shannon conditional entropy, Arimoto-Rényi conditional entropy satisfies a number of desirable properties. In particular, the rule that conditioning cannot increase entropy carries over to the Arimoto-Rényi version [4], [15, Thm. 2], [2, Corr. 1], [3, Prop. 2]:

$$H_\alpha(X|Y,Z) \leq H_\alpha(X|Y). \tag{4.52}$$

It follows from the definition of Arimoto mutual information that a "right-hand" data processing inequality therefore holds: if $X \leftrightarrow Y \leftrightarrow Z$ form a Markov chain, then

$$I_\alpha^A(X;Z) \leq I_\alpha^A(X;Y). \tag{4.53}$$

To reduce our problem to an instance satisfying the distinct row assumption using the technique in Section 4.3, we require a "left-hand" version of the inequality, i.e.,

$$I_\alpha^A(X;Z) \leq I_\alpha^A(Y;Z)? \tag{4.54}$$

In fact, this inequality can fail dramatically.

Proposition 1. *For any $1 < \alpha < \infty$, there exist random variables X, Y, and Z such that $X \leftrightarrow Y \leftrightarrow Z$ and $Y \leftrightarrow X \leftrightarrow Z$ with $I_\alpha^A(X;Z)$ being arbitrarily small and $I_\alpha^A(Y;Z)$ being arbitrarily large.*

Proof. Fix positive integers K and L and $0 < \epsilon < 1/L$. Let Y and Z be jointly

distributed as

$$P(Y=i) = \begin{cases} \epsilon & \text{if } i \in \{1,\dots,L\} \\ \frac{1-L\epsilon}{K} & \text{if } i \in \{L+1,\dots,L+K\} \end{cases} \tag{4.55}$$

$$P(Z=j|Y=i) = \begin{cases} 1 & \text{if } j=i \text{ and } i \in \{1,\dots,L\} \\ \frac{1}{L} & \text{if } i \in \{L+1,\dots,L+K\} \\ 0 & \text{otherwise.} \end{cases} \tag{4.56}$$

We then couple X to Y and Z via

$$X = \min(Y, L+1). \tag{4.57}$$

From (4.4), as $\epsilon \to 0$, we have that $I_\alpha^A(X;Z) \to 0$. Fix ϵ so that $I_\alpha^A(X;Z)$ is as small as desired. If we then let $K \to \infty$, we have

$$I_\alpha^A(Y;Z) \to \frac{\alpha}{\alpha-1}\log L. \tag{4.58}$$

But L was arbitrary. □

For Sibson mutual information and α-maximal leakage, we could reduce our problem to one satisfying the unique row assumption by dividing $\mathcal{X}$ into equivalence classes based on $P_{Y|X}(\cdot|x)$ and assigning to a "leader" realization in each equivalence class the probability of all of the x realizations in that class. This approach fails for Arimoto mutual information, due to the above result, but the reduction is still possible if one accounts for the exponential tilting of $P(x)$ in (4.4).

Proposition 2. *Fix $\alpha > 0$. If (X,Y) does not satisfy the unique row assumption then there exists $\tilde{X}$ such that*

(i) The support of $\tilde{X}$ is strictly contained within the support of X;

*(*ii*)* $P_{Y|X}(y|x) = P_{Y|\tilde{X}}(y|x)$ *for all* x *and* y*;*

*(*iii*)* $(\tilde{X}, Y)$ *satisfies the unique row assumption; and*

*(*iv*)* $I_\alpha^A(X;Y) = I_\alpha^A(\tilde{X};Y)$.

Proof. For $\alpha = 1$, this follows directly from the chain rule for mutual information. For $\alpha \neq 1$, without loss of generality, we may assume that there exists a $k < |\mathcal{X}|$ such that

$$P_{Y|X}(\cdot|x_j) \neq P_{Y|X}(\cdot|x_i) \tag{4.59}$$

for all $1 \leq i < j \leq k$, and for all $k < j \leq |\mathcal{X}|$ there exists $1 \leq i \leq k$ such that

$$P_{Y|X}(y|x_j) = P_{Y|X}(y|x_i) \text{ for all } y. \tag{4.60}$$

That is, the first k rows of $P_{Y|X}$, viewed as a stochastic matrix, are unique, and every other row is a copy of one of those k rows. For each $1 \leq i \leq k$, define the set of X realizations

$$C_i = \left\{ x \in \mathcal{X} : P_{Y|X}(y|x) = P_{Y|X}(y|x_i) \text{ for all } y \right\}, \tag{4.61}$$

and note that $C_1, \ldots, C_k$ are nonempty and form a partition of $\mathcal{X}$. Define $\tilde{X}$ to have support $\{x_1, \ldots, x_k\}$ with marginal distribution

$$P(\tilde{X} = x_i) = \frac{1}{\Gamma} \left(\sum_{x \in C_i} P(X = x)^\alpha \right)^{1/\alpha}, \tag{4.62}$$

where

$$\Gamma = \sum_{i=1}^{k} \left(\sum_{x \in C_i} P(X = x)^\alpha \right)^{1/\alpha}. \tag{4.63}$$

Define the joint distribution between $\tilde{X}$ and Y through (ii). Then (i)-(iii) clearly

hold and we have

$$I_\alpha^A(X;Y) \tag{4.64}$$

$$= \frac{\alpha}{\alpha-1}\log\sum_y\left(\frac{\sum_{i=1}^k\sum_{x\in C_i}P(x)^\alpha P(y|x)^\alpha}{\sum_{i=1}^k\sum_{x\in C_i}P(x)^\alpha}\right)^{1/\alpha} \tag{4.65}$$

$$= \frac{\alpha}{\alpha-1}\log\sum_y\left(\frac{\sum_{i=1}^k\sum_{x\in C_i}(P(x)^\alpha/\Gamma^\alpha)P(y|x)^\alpha}{\sum_{i=1}^k\sum_{x\in C_i}(P(x)^\alpha/\Gamma^\alpha)}\right)^{1/\alpha} \tag{4.66}$$

$$= \frac{\alpha}{\alpha-1}\log\sum_y\left(\frac{\sum_{i=1}^k P(\tilde{X}=x_i)^\alpha P(y|x)^\alpha}{\sum_{i=1}^k P(\tilde{X}=x_i)^\alpha}\right)^{1/\alpha} \tag{4.67}$$

$$= I_\alpha^A(\tilde{X};Y). \tag{4.68}$$

□

4.6 Proof for Mutual Information and Capacity

We begin by proving (4.28) and (4.29), starting with the former. For this, we derive separate upper and lower bounds on $-H(X|Y^n)$. For the lower bound,

$$
\begin{aligned}
-H(X|Y^n) &\equiv \sum_{y^n\in\mathcal{Y}^n} Q(y^n)\sum_{x\in\mathcal{X}} Q(x|y^n)\log Q(x|y^n) && (4.69)\\
&= \sum_{P\in\mathcal{P}_n}\sum_{y^n\in T(P)} Q(y^n)\sum_{x\in\mathcal{X}}\frac{Q(y^n|x)Q(x)}{Q(y^n)}\log\frac{Q(y^n|x)Q(x)}{Q(y^n)} && (4.70)\\
&= \sum_{P\in\mathcal{P}_n}\sum_{y^n\in T(P)}\sum_{x\in\mathcal{X}}\frac{1}{|T(P)|}Q(T(P)|x)Q(x)\\
&\qquad\cdot\log\frac{\frac{1}{|T(P)|}Q(T(P)|x)Q(x)}{\sum_{x'\in\mathcal{X}}\frac{1}{|T(P)|}Q(T(P)|x')Q(x')} && (4.71)\\
&= \sum_{P\in\mathcal{P}_n}\sum_{x\in\mathcal{X}} Q(T(P)|x)Q(x)\log\frac{Q(T(P)|x)Q(x)}{\sum_{x'\in\mathcal{X}}Q(T(P)|x')Q(x')} && (4.72)\\
&= -\sum_{\substack{P\in\mathcal{P}_n:\\ Q(T(P))>0}}\Bigg[Q(T(P)|x_1(P))Q(x_1(P))\\
&\qquad\cdot\log\frac{\sum_{x'\in\mathcal{X}}Q(T(P)|x')Q(x')}{Q(T(P)|x_1(P))Q(x_1(P))}\\
&\quad+\sum_{\substack{x\neq x_1(P):\\ Q(T(P)|x)>0}} Q(T(P)|x)Q(x)\\
&\qquad\cdot\log\frac{\sum_{x'\in\mathcal{X}}Q(T(P)|x')Q(x')}{Q(T(P)|x)Q(x)}\Bigg], && (4.73)
\end{aligned}
$$

due to the convention that $0\log 0 = 0$. Then, replacing weighted sums over x with their largest summand gives

$$
\begin{aligned}
&\geq -\sum_{\substack{P\in\mathcal{P}_n:\\ Q(T(P))>0}}\Bigg[Q(T(P)|x_1(P))Q(x_1(P))\\
&\qquad\cdot\log\Big(1+\frac{\sum_{x'\neq x_1(P)}Q(T(P)|x')Q(x')}{Q(T(P)|x_1(P))Q(x_1(P))}\Big)\\
&\quad+\max_{\substack{x\neq x_1(P):\\ Q(T(P)|x)>0}}\Big\{Q(T(P)|x)\log\frac{\max_{x'\in\mathcal{X}}Q(T(P)|x')}{Q(T(P)|x)Q(x)}\Big\}\Bigg]. && (4.74)
\end{aligned}
$$

Note that the entire expression inside the summation over P is 0 if $Q(T(P)|x_2(P)) = 0$. Letting $Q_{\min}(X) = \min_{x\in\mathcal{X}} Q(x)$ and using $\ln(1+x) \le x$ for the $x = x_1(P)$ term,

$$\begin{aligned}
&\ge -\sum_{\substack{P\in\mathcal{P}_n:\\ Q(T(P))>0}} \Bigg[\frac{1}{\ln 2}\sum_{x'\neq x_1(P)} Q(T(P)|x')Q(x') \\
&\quad + \max_{\substack{x\neq x_1(P):\\ Q(T(P)|x)>0}} \Big\{Q(T(P)|x)\Big\} \\
&\qquad \cdot \log\frac{1}{\min\limits_{\substack{x\neq x_1(P):\\ Q(T(P)|x)>0}} Q(T(P)|x)\cdot Q_{\min}(X)}\Bigg] \qquad (4.75)\\
&\ge -\sum_{\substack{P\in\mathcal{P}_n:\\ Q(T(P))>0}} \Bigg[\frac{1}{\ln 2}2^{-nD(P||Q_{x_2(P)})} + 2^{-nD(P||Q_{x_2(P)})} \\
&\qquad \cdot \Big[nD_{sup} + \log\frac{(n+1)^{|\mathcal{Y}|}}{Q_{\min}(X)}\Big]\Bigg] \qquad (4.76)
\end{aligned}$$

where

$$\begin{aligned}
D_{sup} &\equiv \sup_{\substack{x,P'\in\mathcal{P}\\ D(P'||Q_x)<\infty}} D(P'||Q_x) && (4.77)\\
&= \sup_{\substack{x,P'\in\mathcal{P}:\\ D(P'||Q_x)<\infty}} \sum_{y\in\mathcal{Y}} P'(y)\log\frac{P'(y)}{Q(y|x)} && (4.78)\\
&= \sup_{\substack{x,P'\in\mathcal{P}:\\ D(P'||Q_x)<\infty}} \sum_{y\in\mathcal{Y}} P'(y)\log\frac{1}{Q(y|x)} - H(P') && (4.79)\\
&\le \sup_x \log\frac{1}{\min_{Q(y|x)>0} Q(y|x)} < \infty. && (4.80)
\end{aligned}$$

Hence,

$$\begin{aligned}
&- H(X|Y^n) \\
&\ge -(n+1)^{|\mathcal{Y}|}2^{-nD_n^*}\Big[\frac{1}{\ln 2} + \log\frac{(n+1)^{|\mathcal{Y}|}}{Q_{\min}(X)} + nD_{sup}\Big] \qquad (4.81)
\end{aligned}$$

where

$$D_n^* = \min_{P\in\mathcal{P}_n} D(P||Q_{x_2(P)}) \qquad (4.82)$$

and P_n^* is its minimizer.

For the upper bound,

$$
\begin{aligned}
&- H(X|Y^n) \\
&= \sum_{P\in\mathcal{P}_n}\sum_{x\in\mathcal{X}} Q(T(P)|x)Q(x)\log\frac{Q(T(P)|x)Q(x)}{\sum_{x'\in\mathcal{X}}Q(T(P)|x')Q(x')} && (4.83)\\
&\le \sum_{x\in\mathcal{X}} Q(T(P_n^*)|x)Q(x)\log\frac{Q(T(P_n^*)|x)Q(x)}{\sum_{x'\in\mathcal{X}}Q(T(P_n^*)|x')Q(x')} && (4.84)\\
&\le Q(T(P_n^*)|x_1(P_n^*))Q(x_1(P_n^*)) \\
&\qquad \cdot\log\frac{Q(T(P_n^*)|x_1(P_n^*))Q(x_1(P_n^*))}{\sum_{x'\in\mathcal{X}}Q(T(P_n^*)|x')Q(x')} && (4.85)\\
&= Q(T(P_n^*)|x_1(P_n^*))Q(x_1(P_n^*)) \\
&\qquad \cdot\log\big[1-\frac{\sum_{x'\neq x_1(P_n^*)}Q(T(P_n^*)|x')Q(x')}{\sum_{x'\in\mathcal{X}}Q(T(P_n^*)|x')Q(x')}\big] && (4.86)
\end{aligned}
$$

recalling that $-\ln(1-x)\ge x$,

$$
\begin{aligned}
&\le -Q(T(P_n^*)|x_1(P_n^*))Q(x_1(P_n^*)) \\
&\qquad \cdot\frac{\sum_{x'\neq x_1(P_n^*)}Q(T(P_n^*)|x')Q(x')}{\sum_{x'\in\mathcal{X}}Q(T(P_n^*)|x')Q(x')}\cdot\frac{1}{\ln 2} && (4.87)\\
&\qquad \le -Q(T(P_n^*)|x_1(P_n^*))Q(x_1(P_n^*)) \\
&\qquad\qquad \cdot\frac{Q(T(P_n^*)|x_2(P_n^*))Q(x_2(P_n^*))}{\max_{x'\in\mathcal{X}}Q(T(P_n^*)|x')}\cdot\frac{1}{\ln 2} && (4.88)\\
&\qquad \le -\frac{1}{(n+1)^{|\mathcal{Y}|}}2^{-nD(P_n^*||Q_{x_1(P_n^*)})}Q(x_1(P_n^*)) \\
&\qquad\qquad \cdot\frac{2^{-nD_n^*}Q(x_2(P_n^*))}{(n+1)^{|\mathcal{Y}|}2^{-nD(P_n^*||Q_{x_1(P_n^*)})}}\cdot\frac{1}{\ln 2} && (4.89)\\
&\qquad = -\frac{Q(x_1(P_n^*))Q(x_2(P_n^*))}{(n+1)^{2|\mathcal{Y}|}\ln 2}2^{-nD_n^*}. && (4.90)
\end{aligned}
$$

As we have now shown that mutual information is upper and lower bounded by expressions of the form $H(X)-K_n\cdot 2^{-nD_n^*}$ for some subexponential sequence

K_n, it remains to be shown that this exponent approaches the minimum Chernoff information as $n \to \infty$.

First, it can be shown using standard continuity arguments that

$$\lim_{n\to\infty} \inf_{P\in\mathcal{P}_n} D(P||Q_{x_2(P)}) = \inf_{P\in\mathcal{P}} D(P||Q_{x_2(P)}) \tag{4.91}$$

since $D(P||Q_{x_2(P)})$ is a continuous function of P. Finally, we arrive at the desired result using Lemma 10 in Section 4.4.

Turning to the result for capacity, let Q_u denote the uniform distribution over $\mathcal{X}$. Then by (4.28) we have

$$\liminf_{n\to\infty} -\frac{1}{n} \log \left(\log |\mathcal{X}| - C(X;Y^n)\right) \tag{4.92}$$

$$\geq \liminf_{n\to\infty} -\frac{1}{n} \log \left(\log |\mathcal{X}| - I(X;Y^n)\Big|_{Q_u}\right) \tag{4.93}$$

$$= \min_{x\neq x'} \mathscr{C}(Q_x||Q_{x'}). \tag{4.94}$$

For the reverse inequality, for each n, let Q_n be a maximizer of $I(X;Y^n)$. Then from the previous observation, eventually we have

$$H(X)\Big|_{Q_n} - H(X|Y)\Big|_{Q_n} \geq I(X;Y^n)\Big|_{Q_u} \tag{4.95}$$

$$\geq \log |\mathcal{X}| - e^{-\frac{n}{2}\min_{x\neq x'} \mathscr{C}(Q_x||Q_{x'})}. \tag{4.96}$$

Dropping the second term from the left-hand side and using the fact that

$$D(Q_n||Q_u) = \log |\mathcal{X}| - H(X)\Big|_{Q_n} \tag{4.97}$$

this implies that, eventually,

$$e^{-\frac{n}{2}\min_{x\neq x'} \mathscr{C}(Q_x||Q_{x'})} \geq D(Q_n||Q_u). \tag{4.98}$$

Thus Q_n tends to Q_u as $n \to \infty$. Combining this fact with the bound in (4.90),

we have that, eventually,

$$\mathcal{C}(X;Y^n) = I(X;Y^n)\Big|_{Q_n} \tag{4.99}$$

$$= H(X)\Big|_{Q_n} - H(X|Y)\Big|_{Q_n} \tag{4.100}$$

$$\leq H(X)\Big|_{Q_n} - \frac{Q_n(x_1(P_n^*))Q(x_2(P_n^*))}{(n+1)^{2|\mathcal{Y}|}\ln 2}2^{-nD_n^*}, \tag{4.101}$$

$$\leq H(X)\Big|_{Q_n} - \frac{1}{4|\mathcal{X}|^2(n+1)^{2|\mathcal{Y}|}\ln 2}2^{-nD_n^*} \tag{4.102}$$

$$\leq \log|\mathcal{X}| - \frac{1}{4|\mathcal{X}|^2(n+1)^{2|\mathcal{Y}|}\ln 2}2^{-nD_n^*}, \tag{4.103}$$

which establishes the result since D_n^* converges to the Chernoff information as shown above.

4.7 Proof for Maximal Leakage

We turn to proving (4.30) for the case $\alpha = \infty$. While the lower bound on $I_\infty^S(X;Y^n)$ can be proven directly, we will instead note that it can be obtained from Lemma 12 by letting $\alpha \to \infty$ and then $n \to \infty$.

For the upper bound, recalling the x-domains defined in (4.19) and (4.20), fix $x_a \neq x_b \in \mathcal{X}$ and a $P \in D_{x_b}$ and let $\{P_n\}_{n=1}^\infty$ be a sequence such that $P_n \in \mathcal{P}_n$ for each n and $P_n \to P$. Then $P_n \in D_{x_b}$ eventually and

$$I_\infty^S(X;Y^n) \leq \log \sum_{x\in\mathcal{X}} \sum_{P\in\bar{D}_x\cap\mathcal{P}_n} Q(T(P)|x) \tag{4.104}$$

$$= \log\Big[|\mathcal{X}| - \sum_{x\in\mathcal{X}} \sum_{P\in\mathcal{P}_n\setminus\bar{D}_x} Q(T(P)|x)\Big] \tag{4.105}$$

$$\leq \log\Big[|\mathcal{X}| - \sum_{P\in\mathcal{P}_n\setminus\bar{D}_{x_a}} Q(T(P)|x_a)\Big] \tag{4.106}$$

$$\leq \log\big[|\mathcal{X}| - Q(T(P_n)|x_a)\big], \tag{4.107}$$

eventually. Thus for sufficiently large n,

$$\begin{aligned} & I_\infty^S(X;Y^n) \\ & \leq \log\left[|\mathcal{X}| - \frac{1}{(n+1)^{|\mathcal{Y}|}} 2^{-nD(P_n||Q_{x_a})}\right] && (4.108) \\ & \leq \log\left[|\mathcal{X}|\right] - \frac{1}{(\ln 2)|\mathcal{X}|(n+1)^{|\mathcal{Y}|}} 2^{-nD(P_n||Q_{x_a})} && (4.109) \end{aligned}$$

and

$$\begin{aligned} & \limsup_{n\to\infty} -\frac{1}{n}\log\left(|\mathcal{X}| - I_\infty^S(X;Y^n)\right) \\ & \leq \lim_{n\to\infty} D(P_n||Q_{x_a}) = D(P||Q_{x_a}). && (4.110) \end{aligned}$$

Since $x_a \neq x_b$ and P were arbitrary, the result follows by Lemma 10 in Appendix 4.4.

4.8 Proof for Sibson ($\alpha \in (1,\infty)$)

We turn to (4.30), focusing on the regime $\alpha \in (1,\infty)$, since the $\alpha = 1$ case is established in (4.28) and the $\alpha = \infty$ case will be proven subsequently. First, we derive a lower bound of $I_\alpha^S(X;Y^n)$ for $\alpha > 1$ that will be useful in this and subsequent proofs.

Lemma 12.

$$I_\alpha^S(X;Y^n) \geq H_{1/\alpha}(X) - \frac{\alpha}{(\alpha-1)\ln 2}\left(\Gamma_n + \frac{\Gamma_n^2}{2(1-\Gamma_n)}\right) \tag{4.111}$$

for $\alpha > 1$, where

$$\Gamma_n = \min(1, (n+1)^{|\mathcal{Y}|} \cdot 2^{-n\cdot\min_{x\neq x'} \mathscr{C}(Q_x||Q_{x'})}). \tag{4.112}$$

Remark. *If Q_x and $Q_{x'}$ have disjoint support for every $x \neq x'$, then $\Gamma_n = 0$ and this lemma establishes that $I_\alpha^S(X;Y^n) = H_{1/\alpha}(X)$ for any $n \geq 1$.*

Proof. We use the D_x sets defined in (4.19) and (4.20):

$$I_\alpha^S(X;Y^n) \equiv \frac{\alpha}{\alpha-1}\log \sum_{y^n\in\mathcal{Y}^n}\Big(\sum_{x\in\mathcal{X}} Q(x)Q(y^n|x)^\alpha\Big)^{1/\alpha} \tag{4.113}$$

$$= \frac{\alpha}{\alpha-1}\log \sum_{P\in\mathcal{P}_n}\Big(\sum_{x\in\mathcal{X}} Q(x)Q(T(P)|x)^\alpha\Big)^{1/\alpha} \tag{4.114}$$

$$\geq \frac{\alpha}{\alpha-1}\log \sum_{x\in\mathcal{X}}\sum_{P\in D_x\cap\mathcal{P}_n}\Big(\sum_{x'\in\mathcal{X}} Q(x')Q(T(P)|x')^\alpha\Big)^{1/\alpha} \tag{4.115}$$

$$\geq \frac{\alpha}{\alpha-1}\log \sum_{x\in\mathcal{X}} Q(x)^{1/\alpha}\sum_{P\in D_x\cap\mathcal{P}_n} Q(T(P)|x) \tag{4.116}$$

$$= \frac{\alpha}{\alpha-1}\log \sum_{x\in\mathcal{X}} Q(x)^{1/\alpha}\Big(1-\sum_{P\in\mathcal{P}_n\setminus D_x} Q(T(P)|x)\Big) \tag{4.117}$$

$$= \frac{\alpha}{\alpha-1}\log \Big(\sum_{x\in\mathcal{X}} Q(x)^{1/\alpha} \tag{4.118}$$

$$-\sum_{x\in\mathcal{X}}\sum_{P\in\mathcal{P}_n\setminus D_x} Q(x)^{1/\alpha}Q(T(P)|x)\Big).$$

Define

$$\gamma_n = \frac{\sum_{x\in\mathcal{X}}\sum_{P\in\mathcal{P}_n\setminus D_x} Q(x)^{1/\alpha}Q(T(P)|x)}{\sum_{x\in\mathcal{X}} Q(x)^{1/\alpha}} \leq 1. \tag{4.119}$$

Then we can write

$$I_\alpha^S(X;Y^n) \geq \frac{\alpha}{\alpha-1}\log\Big\{\Big(\sum_{x\in\mathcal{X}} Q(x)^{1/\alpha}\Big)(1-\gamma_n)\Big\} \tag{4.120}$$

$$= H_{1/\alpha}(X) + \frac{\alpha}{\alpha-1}\log(1-\gamma_n). \tag{4.121}$$

Note that

$$\ln(1-\epsilon) = -\sum_{i=1}^{\infty}\frac{\epsilon^i}{i} \tag{4.122}$$

$$\geq -\epsilon - \frac{\epsilon}{2}\Big(\sum_{i=1}^{\infty}\epsilon^i\Big) = -\epsilon - \frac{\epsilon^2}{2(1-\epsilon)} \tag{4.123}$$

for $0 < \epsilon < 1$. Hence,

$$I_\alpha^S(X;Y^n) \geq H_{1/\alpha}(X) + \frac{\alpha}{(\alpha-1)\ln 2}(-\gamma_n - \frac{\gamma_n^2}{2(1-\gamma_n)}). \tag{4.124}$$

The right-hand side in decreasing in γ_n over $[0,1]$. We also have

$$\gamma_n \leq \frac{\sum_{x\in\mathcal{X}} Q(x)^{1/\alpha}(n+1)^{|\mathcal{Y}|} \cdot \max_{P\in\mathcal{P}_n\setminus D_x} Q(T(P)|x)}{\sum_{x\in\mathcal{X}} Q(x)^{1/\alpha}} \tag{4.125}$$

$$\leq \frac{\sum_{x\in\mathcal{X}} Q(x)^{1/\alpha}(n+1)^{|\mathcal{Y}|} \cdot \max\limits_{x'\in\mathcal{X}} \max\limits_{P\in\mathcal{P}_n\setminus D_{x'}} Q(T(P)|x')}{\sum_{x\in\mathcal{X}} Q(x)^{1/\alpha}} \tag{4.126}$$

$$= (n+1)^{|\mathcal{Y}|} \cdot \max_{x\in\mathcal{X}} \max_{P\in\mathcal{P}_n\setminus D_x} Q(T(P)|x) \tag{4.127}$$

$$\leq (n+1)^{|\mathcal{Y}|} \cdot 2^{-n(\min_{x\in\mathcal{X}} \min_{P\in\mathcal{P}_n\setminus D_x} D(P||Q_x))} \tag{4.128}$$

$$\leq (n+1)^{|\mathcal{Y}|} \cdot 2^{-n(\min_{x\neq x'} \inf_{P\in\bar{D}_{x'}} D(P||Q_x))} \tag{4.129}$$

$$= (n+1)^{|\mathcal{Y}|} \cdot 2^{-n\cdot\min_{x\neq x'} \mathscr{C}(Q_x||Q_{x'})}, \tag{4.130}$$

where we have used Lemma 10 in Section 4.4. □

We next prove an analogous upper bound.

Lemma 13. *For $\alpha > 1$, define*

$$F(x,P) = Q(x)Q(T(P)|x)^\alpha. \tag{4.131}$$

For each n, let $\{E_{x_i}^{(n)}\}_{i=1}^{|\mathcal{X}|}$ be a partition of $\mathcal{P}_n$ such that $P \in E_x^{(n)}$ implies $F(x,P) = \max_{x'\in\mathcal{X}} F(x',P)$. Then

$$I_\alpha^S(X;Y^n) \leq H_{1/\alpha}(X) + \frac{\alpha}{(\alpha-1)\ln 2} \frac{1}{\sum\limits_{x\in\mathcal{X}} Q(x)^{1/\alpha}} \sum_{x\in\mathcal{X}} \sum_{P\notin E_x^{(n)}} \cdot (F(x_1(P),P)^{1/\alpha-1} - F(x,P)^{1/\alpha-1})F(x,P)), \tag{4.132}$$

where for the remainder of this section we redefine $x_k(P)$ so that they are ordered by $F(x,P)$ instead of relative entropy. Note that this ordering now depends on n.

Proof. We have

$$I_\alpha^S(X;Y^n) = \frac{\alpha}{\alpha-1}\log\sum_{x\in\mathcal{X}}\sum_{P\in E_x^{(n)}}\Big(\sum_{x'\in\mathcal{X}}F(x',P)\Big)^{1/\alpha} \tag{4.133}$$

$$= \frac{\alpha}{\alpha-1}\log\sum_{x\in\mathcal{X}}\sum_{P\in E_x^{(n)}}F(x,P)^{1/\alpha}\Big(1+\sum_{x'\neq x}\frac{F(x',P)}{F(x,P)}\Big)^{1/\alpha} \tag{4.134}$$

Using the Taylor series expansion of $(1+x)^{1/\alpha}$ and discarding x^2 and higher-order terms (since $\frac{1}{\alpha}<1$), we have

$$\leq \frac{\alpha}{\alpha-1}\log\sum_{x\in\mathcal{X}}\sum_{P\in E_x^{(n)}}F(x,P)^{1/\alpha}\Big(1+\frac{1}{\alpha}\sum_{x'\neq x}\frac{F(x',P)}{F(x,P)}\Big) \tag{4.135}$$

$$\leq \frac{\alpha}{\alpha-1}\log\sum_{x\in\mathcal{X}}\sum_{P\in E_x^{(n)}}\Big(F(x,P)^{1/\alpha} + F(x,P)^{1/\alpha-1}\sum_{x'\neq x}F(x',P)\Big), \tag{4.136}$$

where we have used the fact that $\alpha>1$. Continuing,

$$= \frac{\alpha}{\alpha-1}\log\sum_{x\in\mathcal{X}}\Big(\sum_{P\in E_x^{(n)}}F(x,P)^{1/\alpha} + \sum_{P\notin E_x^{(n)}}F(x_1(P),P)^{1/\alpha-1}F(x,P)\Big) \tag{4.137}$$

$$= \frac{\alpha}{\alpha-1}\log\sum_{x\in\mathcal{X}}\Big(\sum_{P\in E_x^{(n)}}F(x,P)^{1/\alpha} + \sum_{P\notin E_x^{(n)}}F(x_1(P),P)^{1/\alpha-1}F(x,P) + \sum_{P\notin E_x^{(n)}}F(x,P)^{1/\alpha} - \sum_{P\notin E_x^{(n)}}F(x,P)^{1/\alpha}\Big) \tag{4.138}$$

$$= \frac{\alpha}{\alpha-1}\log\sum_{x\in\mathcal{X}}\Big(\sum_{P\in\mathcal{P}_n}F(x,P)^{1/\alpha} + \sum_{P\notin E_x^{(n)}}\big(F(x_1(P),P)^{1/\alpha-1}-F(x,P)^{1/\alpha-1}\big)F(x,P)\Big) \tag{4.139}$$

Using $\ln(1+x)\leq x$ then gives the result. □

The lower bound in (4.30) for $\alpha \in (1,\infty)$ follows directly from Lemma 12. For the upper bound, pick $x_a \neq x_b$ and $P^* \in D_{x_b}$. Let $\{P_n\}_{n=1}^{\infty}$ be a sequence of types converging to P^*. From Lemma 13 we have

$$I_\alpha^S(X;Y^n) \leq H_{1/\alpha}(X) + \frac{\alpha}{(\alpha-1)\ln 2}\frac{1}{\sum\limits_{x\in\mathcal{X}} Q(x)^{1/\alpha}} \sum_{x\in\mathcal{X}} \sum_{P\notin E_x^{(n)}} \cdot (F(x_1(P),P)^{1/\alpha-1} - F(x,P)^{1/\alpha-1})F(x,P)). \tag{4.140}$$

Note that eventually $P_n \in E_{x_b}^{(n)}$, $x_1(P_n) = x_b$ and $F(x_b,P_n)^{1/\alpha-1} < \frac{1}{2}F(x_a,P_n)^{1/\alpha-1}$. Thus, eventually,

$$\leq H_{1/\alpha}(X) + \frac{\alpha}{(\alpha-1)\ln 2}\frac{1}{\sum\limits_{x\in\mathcal{X}} Q(x)^{1/\alpha}} \cdot (F(x_1(P_n),P_n)^{1/\alpha-1} - F(x_a,P_n)^{1/\alpha-1})F(x_a,P_n)). \tag{4.141}$$

$$\leq H_{1/\alpha}(X) - \frac{\alpha}{2(\alpha-1)\ln 2}\frac{1}{\sum\limits_{x\in\mathcal{X}} Q(x)^{1/\alpha}} F(x_a,P_n)^{1/\alpha} \tag{4.142}$$

$$\leq H_{1/\alpha}(X) - \frac{\alpha}{2(\alpha-1)\ln 2}\frac{1}{\sum\limits_{x\in\mathcal{X}} Q(x)^{1/\alpha}} Q_{\min}(X)^{1/\alpha} \cdot \frac{1}{(n+1)^{|\mathcal{Y}|}} 2^{-nD(P_n||Q_{x_a})} \tag{4.143}$$

where $Q_{\min}(X) = \min_{x\in\mathcal{X}} Q(x)$. This implies:

$$\limsup_{n\to\infty} -\frac{1}{n}\log\left(H_{1/\alpha}(X) - I_\alpha^S(X;Y^n)\right) \leq \lim_{n\to\infty} D(P_n||Q_{x_a}) = D(P^*||Q_{x_a}). \tag{4.144}$$

Since $x_a \neq x_b$ and $P \in D_{x_b}$ were arbitrarily chosen, this implies:

$$\limsup_{n\to\infty} -\frac{1}{n}\log\left(H_{1/\alpha}(X) - I_\alpha^S(X;Y^n)\right) \leq \min_{x\neq x'} \inf_{P\in\bar{D}_x} D(P||Q_{x'}) = \min_{x\neq x'} \mathscr{C}(Q_x||Q_{x'}), \tag{4.145}$$

where the last step used Lemma 10 in Section 4.4.

4.9 Proof for Arimoto

Note that (4.31) for the case $\alpha = 1$ has already been proven. We prove the lower and upper bounds for the $\alpha > 1$ case as follows.

4.9.1 Proof of Lower Bound

Proof. Let $|\mathcal{X}| = M$ and

$$\epsilon_{X|Y^n} = \min_{f:\mathcal{Y}^n \to X} P(X \neq f(Y^n)) \tag{4.146}$$

$$= 1 - E_{Y^n}[\max_x Q(x|Y^n)] \tag{4.147}$$

$$\leq 1 - p_{max} \text{ where } p_{max} = \max_X Q(X) \tag{4.148}$$

$$\leq 1 - \frac{1}{M}. \tag{4.149}$$

For $1 < \alpha < \infty$,

$$H_\alpha(X|Y^n) \leq \log M - d_\alpha(\epsilon_{X|Y^n}||1 - \frac{1}{M}) \tag{4.150}$$

where $d_\alpha(p||q)$ is the binary Renyi divergence ([43], Thm. 3):

$$d_\alpha(p||q) = \frac{1}{\alpha - 1} \log (p^\alpha q^{1-\alpha} + (1-p)^\alpha (1-q)^{1-\alpha}). \tag{4.151}$$

So,

$$I_\alpha^A(X;Y^n) = H_\alpha(X) - H_\alpha(X|Y^n) \tag{4.152}$$

$$\geq H_\alpha(X) - \log M + d_\alpha(\epsilon_{X|Y^n}||1 - \frac{1}{M}) \tag{4.153}$$

$$\begin{aligned} &= H_\alpha(X) - \log M \\ &\quad + \frac{1}{\alpha - 1} \log \Big(\epsilon_{X|Y^n}^\alpha (1 - \frac{1}{M})^{1-\alpha} \\ &\quad + (1 - \epsilon_{X|Y^n})^\alpha (\frac{1}{M})^{1-\alpha} \Big) \end{aligned} \tag{4.154}$$

$$\begin{aligned} &\geq H_\alpha(X) - \log M \\ &\quad + \frac{1}{\alpha - 1} \log \left((1 - \epsilon_{X|Y^n})^\alpha (\frac{1}{M})^{1-\alpha} \right) \end{aligned} \tag{4.155}$$

$$= H_\alpha(X) + \frac{\alpha}{\alpha - 1} \log \left(1 - \epsilon_{X|Y^n}\right) \tag{4.156}$$

Hence,

$$I_\alpha^A(X;Y^n) - H_\alpha(X) \geq \frac{\alpha}{\alpha - 1} \log \left(1 - \epsilon_{X|Y^n}\right) \tag{4.157}$$

which gives

$$\frac{\alpha - 1}{\alpha} [H_\alpha(X) - I_\alpha^A(X;Y^n)] \leq \log \frac{1}{1 - \epsilon_{X|Y^n}}. \tag{4.158}$$

For $0 < \epsilon \leq 1/2$,

$$\log \frac{1}{1-\epsilon} = \log(1 + \frac{\epsilon}{1-\epsilon}) \tag{4.159}$$

$$\leq \frac{\epsilon}{1-\epsilon} \frac{1}{\ln 2} \tag{4.160}$$

$$\leq \frac{2\epsilon}{\ln 2}. \tag{4.161}$$

For all sufficiently large n, we have $\epsilon_{X|Y^n} \leq 1/2$ by the unique row assumption. Thus, combining (4.158) and (4.161), for all $1 < \alpha < \infty$,

$$\frac{2\epsilon_{X|Y^n}}{\ln 2} \geq \frac{\alpha - 1}{\alpha} [H_\alpha(X) - I_\alpha^A(X;Y^n)] \tag{4.162}$$

$$-\frac{1}{n} \log(\frac{2\epsilon_{X|Y^n}}{\ln 2}) \leq -\frac{1}{n} \log(\frac{\alpha - 1}{\alpha} [H_\alpha(X) - I_\alpha^A(X;Y^n)]), \tag{4.163}$$

and, taking $\alpha \to \infty$ in (4.163),

$$-\frac{1}{n}\log(\frac{2\epsilon_{X|Y^n}}{\ln 2}) \leq -\frac{1}{n}\log(H_\infty(X) - I_\infty^A(X;Y^n)). \tag{4.164}$$

Note that $\epsilon_{X|Y^n}$ is bounded as [43, Thm. 15]

$$\epsilon_{X|Y^n} \leq (M-1)\exp\left(-\min_{x\neq x'}\mathscr{C}(Q_x^n||Q_{x'}^n)\right). \tag{4.165}$$

Then, using Lemma 11 in Section 4.4, for any $\alpha \in (1,\infty]$,

$$\begin{aligned}&\min_{x\neq x'}\mathscr{C}(P_x||P_{x'})\\ &\quad\leq \liminf_{n\to\infty} -\frac{1}{n}\log\left[H_\alpha(X) - I_\alpha^A(X;Y^n)\right]\end{aligned} \tag{4.166}$$

□

4.9.2 Proof of Upper Bound

Proof. For $\alpha \in [0,\infty]$ [43, (165)],

$$H_\alpha(X|Y^n) \geq \log\frac{1}{1-\epsilon_{X|Y^n}}. \tag{4.167}$$

Thus,

$$I_\alpha^A(X;Y^n) \leq H_\alpha(X) - \log\frac{1}{1-\epsilon_{X|Y^n}} \tag{4.168}$$

$$\log\frac{1}{1-\epsilon_{X|Y^n}} \leq H_\alpha(X) - I_\alpha^A(X;Y^n) \tag{4.169}$$

$$\frac{\epsilon_{X|Y^n}}{\ln 2} \leq H_\alpha(X) - I_\alpha^A(X;Y^n) \tag{4.170}$$

and so

$$\limsup_{n\to\infty} -\frac{1}{n}\log\epsilon_{X|Y^n} \geq \limsup_{n\to\infty} -\frac{1}{n}\log[H_\alpha(X) - I_\alpha^A(X;Y^n)]. \tag{4.171}$$

It remains to show that

$$\limsup_{n\to\infty} -\frac{1}{n}\log \epsilon_{X|Y^n} \leq \min_{x\neq x'} \mathscr{C}(Q_x||Q_{x'}). \tag{4.172}$$

To this end, for any $i \neq j$, we have

$$\begin{aligned}
\epsilon_{X|Y^n} &= E_{Y^n}[1 - \max_x Q(x|Y^n)] && (4.173)\\
&\geq \sum_{y^n} Q(y^n) \min(Q(x_i|y^n), Q(x_j|y^n)) && (4.174)\\
&\geq \min(Q(x_i), Q(x_j)) \sum_{y^n} Q(y^n) \min\left(\frac{Q(x_i|y^n)}{Q(x_i)}, \frac{Q(x_j|y^n)}{Q(x_j)}\right) && (4.175)\\
&= 2\min(Q(x_i), Q(x_j))\epsilon_{n,i,j}, && (4.176)
\end{aligned}$$

where

$$\epsilon_{n,i,j} = \frac{1}{2}\sum_{y^n} \min(Q(y^n|x_i), Q(y^n|x_j)) \tag{4.177}$$

is the error probability for the alternative problem in which X assumes only two values, x_i and x_j, which are equally likely, and we seek to guess X from Y^n. By [7, Thm. 11.9.1], we have

$$\lim_{n\to\infty} -\frac{1}{n}\log \epsilon_{n,i,j} = \mathscr{C}(Q_{x_i}||Q_{x_j}). \tag{4.178}$$

But i and j were arbitrary.

□

4.10 Proof for α-Maximal Leakage

Note that for $\alpha = 1$, α-maximal leakage is given by regular mutual information, so that case is already proven.

4.10.1 Proof of Lower Bound

Proof. We obtain the lower bound by choosing $X \sim Q_u$, where $Q_u(X)$ denotes the uniform distribution over $\mathcal{X}$. Then

$$\mathcal{L}_\alpha^{max}(X \to Y) = \max_{Q(X)} I_\alpha^S(X;Y^n) \geq I_\alpha^S(X;Y^n)|_{Q_u(X)}. \tag{4.179}$$

Then by (4.30),

$$\liminf_{n\to\infty} -\frac{1}{n}\log(\log|\mathcal{X}| - \mathcal{L}_\alpha^{max}(X \to Y)) \geq \min_{x\neq x'} \mathscr{C}(Q_x||Q_{x'}) \tag{4.180}$$

□

4.10.2 Proof of Upper Bound

Proof. As with the proof for Shannon capacity, the idea is to show that the maximizing $Q(X)$ must eventually be contained in a neighborhood of the uniform distribution. Over this neighborhood, we can use Lemma 13 to uniformly bound the difference

$$\log|\mathcal{X}| - \max_{Q(X)} I_\alpha^S(X:Y^n). \tag{4.181}$$

First, for each n, let

$$Q_n(X) \in \arg\max_{Q(X)} I_\alpha^S(X;Y^n). \tag{4.182}$$

We have [50, Ex. 2 and Thm. 3]

$$H_{1/\alpha}(X)|_{Q_n(X)} \geq I_\alpha^S(X;Y^n)|_{Q_n(X)}, \tag{4.183}$$

and thus, by Lemma 12,

$$\begin{aligned} H_{1/\alpha}(X)|_{Q_n(X)} &\geq I_\alpha^S(X;Y^n)|_{Q_u(X)} && (4.184)\\ &\geq H_{1/\alpha}(X)|_{Q_u(X)} \\ &\quad - \frac{\alpha}{(\alpha-1)\ln 2}(\Gamma_n + \frac{\Gamma_n^2}{2(1-\Gamma_n)}). && (4.185) \end{aligned}$$

Then,

$$H_{1/\alpha}(X)|_{Q_n(X)} \geq H_{1/\alpha}(X)|_{Q_u(X)} - \frac{\alpha}{(\alpha-1)\ln 2}(\Gamma_n + \frac{\Gamma_n^2}{2(1-\Gamma_n)}) \tag{4.186}$$

$$H_{1/\alpha}(X)|_{Q_u(X)} - H_{1/\alpha}(X)|_{Q_n(X)} \leq \frac{\alpha}{(\alpha-1)\ln 2}(\Gamma_n + \frac{\Gamma_n^2}{2(1-\Gamma_n)}) \tag{4.187}$$

$$D_{1/\alpha}(Q_n(X)||Q_u(X)) \leq \frac{\alpha}{(\alpha-1)\ln 2}(\Gamma_n + \frac{\Gamma_n^2}{2(1-\Gamma_n)}) \equiv \epsilon_n, \tag{4.188}$$

where we have used the fact that $H_{1/\alpha}(X)|_{Q_u(X)} - H_{1/\alpha}(X)|_{Q_n(X)} = D_{1/\alpha}(Q_n(X)||Q_u(X))$. Note that $\lim_{n\to\infty} \epsilon_n = 0$. Then, using the Rényi version of Pinsker's Inequality ([49, Thm. 31]),

$$D_{1/\alpha}(Q_u(X)||Q_n(X)) \geq \frac{2}{\alpha} \sup_A |Q_n(A) - Q_u(A)|^2 \tag{4.189}$$

$$\geq \frac{2}{\alpha} \sup_x |Q_n(x) - Q_u(x)|^2 \tag{4.190}$$

and so

$$\epsilon_n \geq \frac{2}{\alpha} \sup_x |Q_n(x) - Q_u(x)|^2. \tag{4.191}$$

It also follows that, under this constraint,

$$\epsilon_n \geq \frac{2}{\alpha}(Q_u(x) - \min_{x'} Q_n(x'))^2 \tag{4.192}$$

$$\sqrt{\frac{\alpha\epsilon_n}{2}} \geq Q_u(x) - \min_{x'} Q_n(x') \tag{4.193}$$

$$\min_{x'} Q_n(x') \equiv Q_{\min,n}(X) \geq \frac{1}{|\mathcal{X}|} - \sqrt{\frac{\alpha\epsilon_n}{2}} \tag{4.194}$$

and similarly,

$$\max_{x'} Q_n(x') \equiv Q_{\max,n}(X) \leq \frac{1}{|\mathcal{X}|} + \sqrt{\frac{\alpha\epsilon_n}{2}} \tag{4.195}$$

Let A_n be the set of distributions over X that satisfy both (4.194) and (4.195) and note that $Q_n \in A_n$ eventually. Recalling (4.131), define

$$F(x, P, \tilde{Q}) = \tilde{Q}(x) Q(T(P)|x)^{\alpha}, \tag{4.196}$$

where we now indicate the dependence on the input distribution $\tilde{Q}(x)$. Similarly, we let $\{E^{(n)}_{x_i,\tilde{Q}}\}$ be a partition of $\mathcal{P}_n$ such that $P \in E^{(n)}_{x,\tilde{Q}}$ implies $F(x, P, \tilde{Q}) = \max_{x'} F(x', P, \tilde{Q})$ and we let $x_1(P, \tilde{Q})$, $x_2(P, \tilde{Q})$, $\ldots$, denote the letters of $\mathcal{X}$ in decreasing order of (4.196). By Lemma 13, we have, eventually

$$\begin{aligned}
&\max_{\tilde{Q}} I^S_{\alpha}(X; Y^n) \\
&= \max_{\tilde{Q} \in A_n} I^S_{\alpha}(X; Y^n) \\
&\leq \max_{\tilde{Q} \in A_n} H_{1/\alpha}(X) + \frac{\alpha}{(\alpha - 1) \ln 2} \frac{1}{\sum\limits_{x \in \mathcal{X}} \tilde{Q}(x)^{1/\alpha}} \sum_{x \in \mathcal{X}} \sum_{P \notin E^{(n)}_{x,\tilde{Q}}} \\
&\qquad \cdot (F(x_1(P, \tilde{Q}), P, \tilde{Q})^{1/\alpha - 1} - F(x, P, \tilde{Q})^{1/\alpha - 1}) F(x, P, \tilde{Q})).
\end{aligned} \tag{4.197}$$

Fix $x_a \neq x_b$ and $P^* \in D_{x_b}$ and let P_n be a sequence of types converging to P^*. Then for all sufficiently large n, we have that $P_n \in E^{(n)}_{x_b,\tilde{Q}}$ for all $\tilde{Q} \in A_n$. Then because the summands in (4.197) are nonpositive, we have

$$\begin{aligned}
&\max_{\tilde{Q} \in A_n} I^S_{\alpha}(X; Y^n) \\
&\leq \max_{\tilde{Q} \in A_n} H_{1/\alpha}(X) + \frac{\alpha}{(\alpha - 1) \ln 2} \frac{1}{\sum\limits_{x \in \mathcal{X}} \tilde{Q}(x)^{1/\alpha}} \\
&\quad \cdot (F(x_1(P_n, \tilde{Q}), P_n, \tilde{Q})^{1/\alpha - 1} - F(x_a, P_n, \tilde{Q})^{1/\alpha - 1}) F(x_a, P_n, \tilde{Q})).
\end{aligned} \tag{4.198}$$

Note that, eventually, $x_1(P_n, \tilde{Q}) = x_b$ for all $\tilde{Q} \in A_n$ and $F(x_b, P_n, \tilde{Q})^{1/\alpha - 1} < \frac{1}{2} F(x_a, P_n, \tilde{Q})^{1/\alpha - 1}$ for all $\tilde{Q} \in A_n$. The remainder of the argument proceeds anal-

ogously to the Sibson proof. Eventually, we have

$$\max_{\tilde{Q}\in A_n} I_\alpha^S(X;Y^n)$$

$$\leq \max_{\tilde{Q}\in A_n} H_{1/\alpha}(X) - \frac{1}{2}\frac{\alpha}{(\alpha-1)\ln 2}\cdot\frac{1}{\sum_{x\in\mathcal{X}}\tilde{Q}(x)^{1/\alpha}} \tag{4.199}$$

$$\cdot F(x_a, P_n, \tilde{Q})^{1/\alpha} \tag{4.200}$$

$$\leq \max_{\tilde{Q}\in A_n} H_{1/\alpha}(X) - \frac{1}{2}\frac{\alpha}{(\alpha-1)\ln 2}\cdot\frac{1}{|\mathcal{X}|\left(\frac{1}{|\mathcal{X}|}+\sqrt{\frac{\alpha\epsilon_n}{2}}\right)^{1/\alpha}} \tag{4.201}$$

$$\cdot\left(\frac{1}{|\mathcal{X}|}-\sqrt{\frac{\alpha\epsilon_n}{2}}\right)^{1/\alpha}\frac{1}{(n+1)^{|\mathcal{Y}|}}2^{-nD(P_n||Q_{x_a})} \tag{4.202}$$

$$\leq \log|\mathcal{X}| - \frac{1}{2}\frac{\alpha}{(\alpha-1)\ln 2}\frac{1}{|\mathcal{X}|\left(\frac{1}{|\mathcal{X}|}+\sqrt{\frac{\alpha\epsilon_n}{2}}\right)^{1/\alpha}} \tag{4.203}$$

$$\cdot\left(\frac{1}{|\mathcal{X}|}-\sqrt{\frac{\alpha\epsilon_n}{2}}\right)^{1/\alpha}\frac{1}{(n+1)^{|\mathcal{Y}|}}2^{-nD(P_n||Q_{x_a})}. \tag{4.204}$$

This implies that

$$\lim_{n\to\infty} -\frac{1}{n}\log\left(\log|\mathcal{X}| - \max_{\tilde{Q}(X)} I_\alpha^S(X;Y^n)\right) \leq \min_{x\neq x'}\mathscr{C}(Q_x||Q_{x'}) \tag{4.205}$$

by Lemma 10 in Section 4.4, which implies the result for $1<\alpha<\infty$. The $\alpha=\infty$ case follows from (4.30) since $I_\infty^S(X;Y^n)$ does not depend on $Q(X)$, and $H_{1/\alpha}(X)=\log|\mathcal{X}|$ in that case. □

4.11 Numerical Approximation Sample Plots

Here, we present numerical examples and show how the bounds an exponential approximations compare to the true mutual information and maximal leakage values. We give 5 numerical examples as follows:

- Binary symmetric channel with flipping probability of 0.2

- Binary asymmetric channel with uniform distribution on X, with $0 \to 1$ flip probability of 0.1 and $1 \to 0$ flip probability of 0.4

- Uniform 3×3 upper triangular channel with uniform trinary X and row-wise uniform $Q(Y|X)$. The used $Q(Y|X)$ can be seen in Figure(4.1

- Random 3×3 upper triangular channel with uniform trinary X, but randomly-generated $Q(Y|X)$. The generated $Q(Y|X)$ can be seen in Figure 4.2

- Random 2 channel with uniform binary X and randomly-generated $Q(Y|X)$ where Y is trinary. The generated $Q(Y|X)$ can be seen in Figure 4.3

$$\begin{bmatrix} 0.3333 & 0.3333 & 0.3333 \\ 0 & 0.5 & 0.5 \\ 0 & 0 & 1 \end{bmatrix}$$

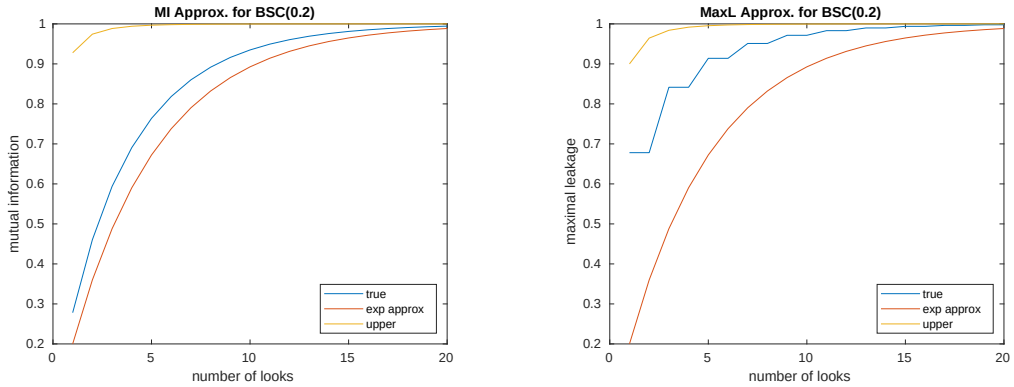
Figure 4.1: Row-wise uniform 3×3 upper triangular $Q(Y|X)$

$$\begin{bmatrix} 0.1528 & 0.6266 & 0.2206 \\ 0 & 0.4931 & 0.5069 \\ 0 & 0 & 1 \end{bmatrix}$$

Figure 4.2: Randomly generated 3×3 upper triangular $Q(Y|X)$

$$\begin{bmatrix} 0.1359 & 0.3106 & 0.5535 \\ 0.3703 & 0.4675 & 0.1623 \end{bmatrix}$$

Figure 4.3: Randomly generated 2×3 $Q(Y|X)$



(a) MI approx. (b) MaxL approx.

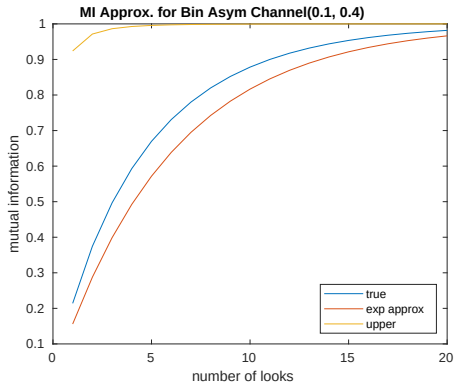
Figure 4.4: Binary symmetric channel approximation plots

In all cases, the lower bounds for both mutual information (Equation 4.81) and maximal leakage (Equation 4.111) could be computed, for relatively low n values, the sub-exponential factors dominate too much and the cited equations do not generally serve as useful approimations.

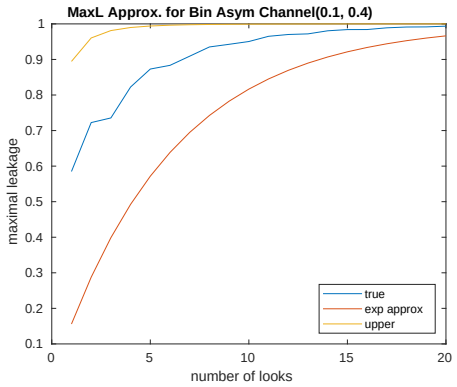
The upper bounds for mutual information (Equation 4.90) and maximal leakage (Equation 4.109) are computed and labeled in the following plots as "upper".

The exponential approximation for both mutual information and maximal leakage are simply computed as $2^{-n \min_{x \neq x'} \mathscr{C}(Q_x || Q_{x'})}$ and are labeled in the following plots as "exp approx".

Finally, the true values of mutual information and maximal leakage are directly computed according to their definitions.

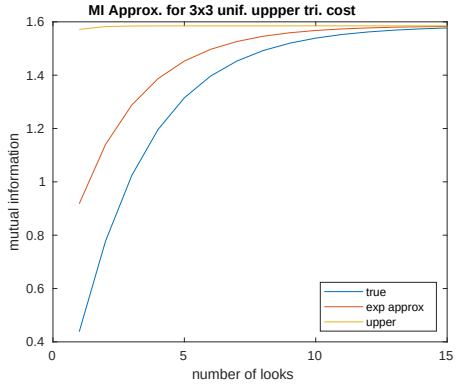


(a) MI approx.

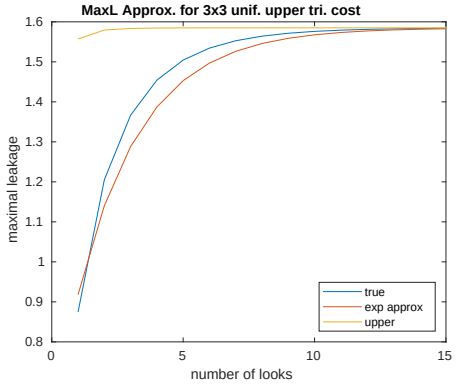


(b) MaxL approx.

Figure 4.5: Binary asymmetric channel approximation plots



(a) MI approx.

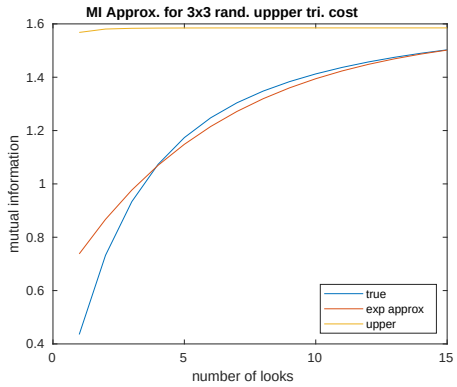


(b) MaxL approx.

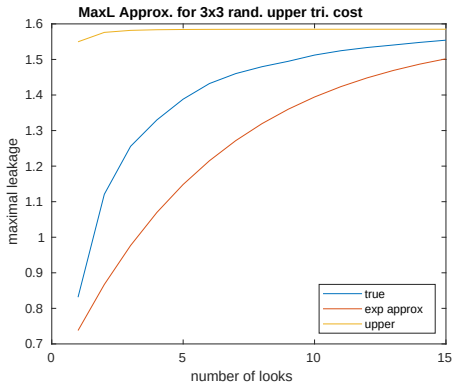
Figure 4.6: Uniform upper triangular channel approximation plots

4.12 Summary

To conclude this chapter, the major takeaways are as follows. Our primary result is that, for a large enough number of looks in the replayable attack, the composition bound on Sibson mutual information and alpha maximal leakage approaches its asymptotic upper limit exponentially with the number of looks, with a decay rate of the smallest Chernoff information between any two rows of the protection transition matrix. In practice, this bound is not suitable for a small number of looks, so

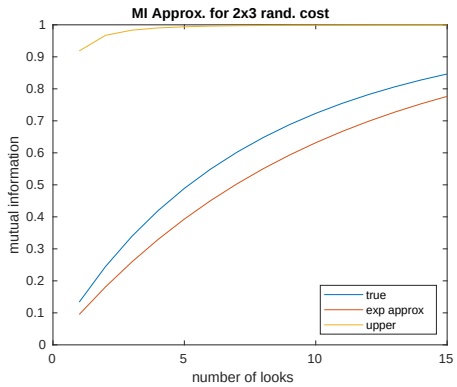


(a) MI approx.

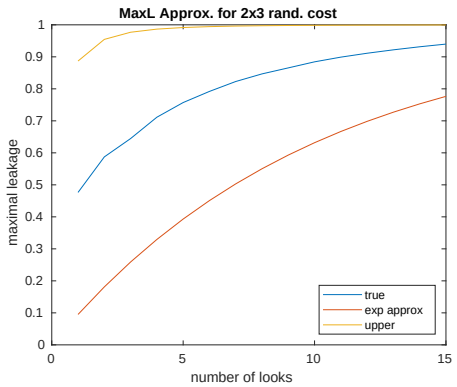


(b) MaxL approx.

Figure 4.7: Random upper triangular channel approximation plots



(a) MI approx.



(b) MaxL approx.

Figure 4.8: Random 2×3 channel approximation plots

the previously established additive bounds may be used in the small number of looks regime, whereas our bound is more useful for higher number of looks. In terms of protection against replayable attacks, this implies that the best protection schemes either a) are simply deterministic or b) when the protection scheme must be stochastic, their transition matrix rows have low relative entropy with each other (i.e. in layman's terms, the rows should be similar to each other).

CHAPTER 5

MEMORY ACCESS-BASED SIDE CHANNEL ATTACKS/DEFENSES IN DYNAMICALLY PRUNED NEURAL NETWORKS

Deep learning has gained popularity in many applications, including image/video recognition, recommender systems, self-driving vehicles, and more[9]. In particular, convolutional neural networks (CNNs) are widely used for many machine learning (ML) tasks on images and videos, and many custom hardware accelerators have been proposed to efficiently handle the large amount of computation needed in complex CNNs.

In particular, dynamic pruning refers to a class of optimization that skips unnecessary or less important computation or memory accesses at run-time. For example, feature maps in CNNs often have many zeros due to the ReLU activation function, and studies [16, 52, 18, 6] have shown that the computation and data movements can be significantly reduced by ignoring zeros or other less important features during runtime. In fact, dynamic pruning has been shown to achieve upwards of 2x speed up during the inference phase for Resnet and upwards of 5x for VGG, both of which are popular CNN architectures [16].

We show that dynamic pruning techniques introduce a side-channel vulnerability that allows an attacker to infer various runtime information, most notably the outputs of the CNN. Unlike the model/structure-stealing attacks [20], which learn a single static secret by observing memory accesses of multiple inputs, our attack learns one secret value per input. Moreover, our attack does not require an adversarial presence during the victim training as in the input-reconstruction attacks [53, 37], but instead only requires black-box access to a copy of the vic-

tim's model and trained weights. The presence of this vulnerability is potentially alarming in light of the demonstrated benefits of the dynamic pruning technique, as it may lead to unexpected privacy leaks in CNN accelerator architectures and similar shared resource contexts.

Previous work has demonstrated a number of privacy attacks on CNNs already. Here, we broadly distinguish two categories of CNN privacy attacks: API attacks (which require access to a victim CNN's API but are generally not used for pruning attacks) and side-channel attacks. This side-channel attack belongs to the latter group, in which previous studies have shown that memory access patterns can leak the victim model's structure and parameters [20] and that power side channels can be used to reconstruct inputs[53, 37].

We investigate the side-channel attack through memory access patterns of dynamically-pruned CNNs, using image classifier CNNs as our primary example. This chapter can be conceptually broken down into four parts.

First, we provide some background on the inner workings on convolutional layers and dynamic pruning. We explain the fundamental mechanics that allow for a potential side channel to exist and why dynamic pruning techniques cannot be cheaply modified to prevent the information leakage altogether. We also provide a threat model for an attacker with limited access to the victim's resources.

Second, we show that, given only information on the victim CNN's dynamic pruning decisions and a set of labels, an adversary can train a supervised machine learning (ML) model to predict at two types of attack targets with surprising accuracy: the classification label and the dominant color of the input image. We also perform experiments to probe the nature of the attack space and study the

efficacy of rudimentary protection mechanisms here. The following sections are devoted to a deeper dive on protection.

Third, using information theory, we define the concept of information overlap as a general, inter-feature property of all ML problems. This information overlap (or simply *overlap*) is a measure of the amount of shared information among any group of two or more features with respect to a learning target (in this case, an *attack target*). Moreover, we propose information theory-driven techniques to probe this property across a large feature set. As a result, we find that for the CNN dynamic pruning problem, not only does there tend to be a high degree of overlap among the features for the studied targets, it also manifests as each feature being a close-to-independent look at the target in a similar fashion to the problem studied in the previous chapter.

Finally, we more deeply investigate cost-effective protection schemes designed for the CNN problem. Here we find that, remarkably, the simplest and general-purpose protection schemes do not perform significantly worse than protection schemes custom-made for a particular attack target. In some cases, the simplest schemes even outperform attack-specific schemes.

5.1 Background on Dynamically Pruned CNNs

Here, we provide some necessary background on the inner workings of convolutional layers and dynamic pruning.

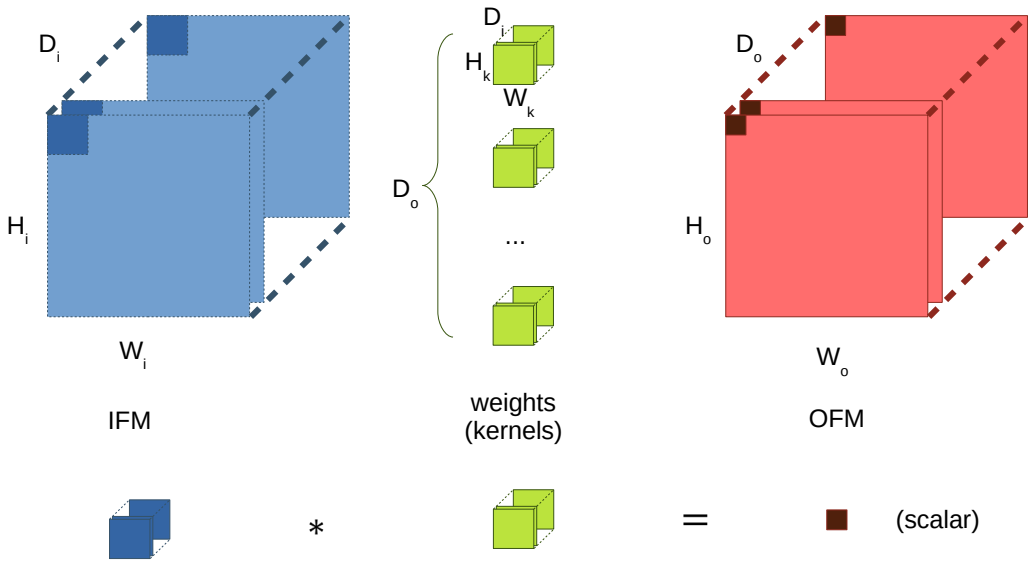


Figure 5.1: Matrix representation of a convolutional layer (batch size=1) and example of one dot product producing a single output activation

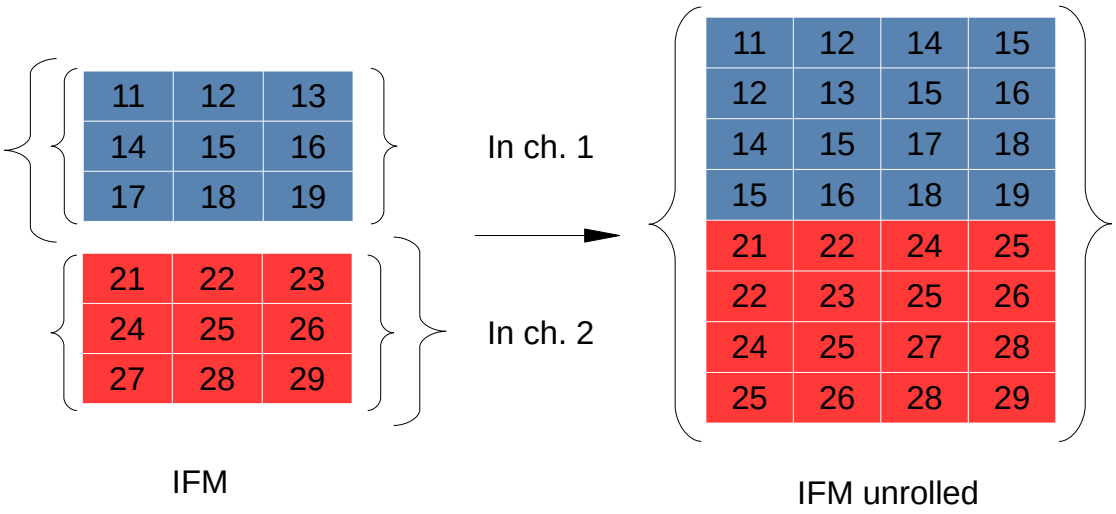


Figure 5.2: Numerical example of IFM unrolling by im2col

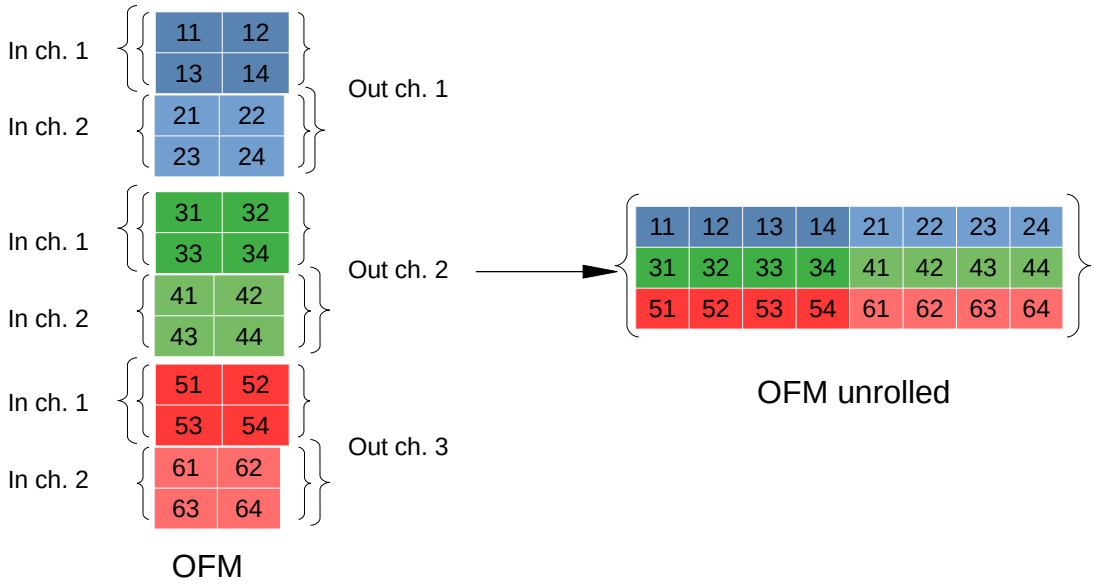


Figure 5.3: Numerical example of weight unrolling by im2col

5.1.1 Convolutional Layers

Mathematically, a vanilla (no pruning) convolutional layer can be represented as a series of 3D matrix operations. Figure 5.1 shows how a single convolutional layer is computed for a single input. The input feature map (IFM) consists of a 3D matrix of size $H_i \times W_i \times D_i$. Similarly, the output feature map (OFM) of the layer is also a 3D matrix of size $H_o \times W_o \times D_o$. Each 2D slice of the IFM and of the OFM along their depth dimensions are commonly referred to as *input channels* and *output channels*, respectively. For convenience, we also distinguish the feature maps from *activations*, in that the former refer to the matrices as a whole while the latter refers the individual scalar values that compose them.

Since a CNN consists of multiple convolutional layers linked in sequence, the OFM of one convolutional layer will become the IFM to the next layer. Finally, the weights can be thought of as a 4D matrix of size $D_i \times H_k \times W_k \times D_o$. Each 3D slice of the weights along the fourth dimension is commonly referred to as a

kernel.

Put simply, the convolution itself consists of a systematic series of inner products between submatrices of the IFM and weight kernels, accumulated over the entire IFM and all kernels. For instance, Figure 5.1 shows how the top-leftmost pixel in each of the output channels is computed. Take the inner product of the top-leftmost $D_i \times H_k \times W_k$ part of the IFM and the first weight kernel; the resulting scalar is the value of the top-leftmost pixel in the first output channel. Similarly, the inner product of the same part of the IFM with the second kernel results in the top-leftmost pixel in the second output channel. The rest of the convolutional layer proceeds similarly, performing a slilding window sweep over the entire IFM.

Note however that CNNs are not usually implemented directly this way, because doing so is inefficient in terms of memory access order when the IFM, weights and OFM are collectively too large to fit in on-chip memory [39]. That is because the pictured procedure requires redundantly reading from/writing to the same memory addresses many times, thereby incurring unnecessary execution time.

Instead, standard implementations of CNNs employ vectorization to speed up the process via im2col (image block to column) [40, 1], a process that unrolls the IFM and weights into 2D versions of themselves, reducing the process to a 2D matrix multiplication. A small numerical example of how im2col is used to unroll both the IFM and weight kernels can be seen in Figures 5.2 and 5.3, respectively. In particular, note that the activations within any given input channel of the IFM will end up being unrolled into a contiguous block of memory, and the weights for a particular input/output channel pair will similarly be unrolled into contiguous memory blocks. Then, if any activations are pruned, knowledge of the victim's memory accesses will reveal which activations were skipped during either

the im2col unrolling process or the 2D matrix multiplication that follows, because the corresponding memory locations will not be accessed. We have somewhat simplified the details here, as the precise order of memory accesses within each layer can be further streamlined to reduce memory accesses (*dataflow implementations*) [39], but these streamlined implementations do not affect the efficacy of our attack.

5.1.2 Dynamic Pruning

Broadly speaking, dynamic activation pruning is any technique that reduces per-layer computations and memory accesses by zeroing out activations in either the IFM or OFM. Since zeros in the IFM of a layer do not contribute to the inner products that compose the OFM, multiplications with those zeros can be skipped, and values very close to zero have been found to be skippable with minimal impact on prediction accuracy as well. In particular, only input-dependent pruning techniques can leak information on a per-input basis, but input-independent techniques are not usually considered dynamic.

While dynamic pruning can theoretically be performed on either the IFM or the OFM, most practical implementations operate on the OFM, since doing so allows for OFM compression in order to reduce the total volume of memory accesses. Thus, we will presume dynamic pruning of the OFM in the rest of this paper.

In general, dynamic pruning is performed by deciding whether individual activations (or groups of activations) should be pruned or not pruned. Each of these single-bit (prune/don't prune) decisions are called *pruning decisions*, and a CNN may make many pruning decisions for a single input. We call a collection of pruning decisions spawned from a single input, a *pruning vector*. We can further separate

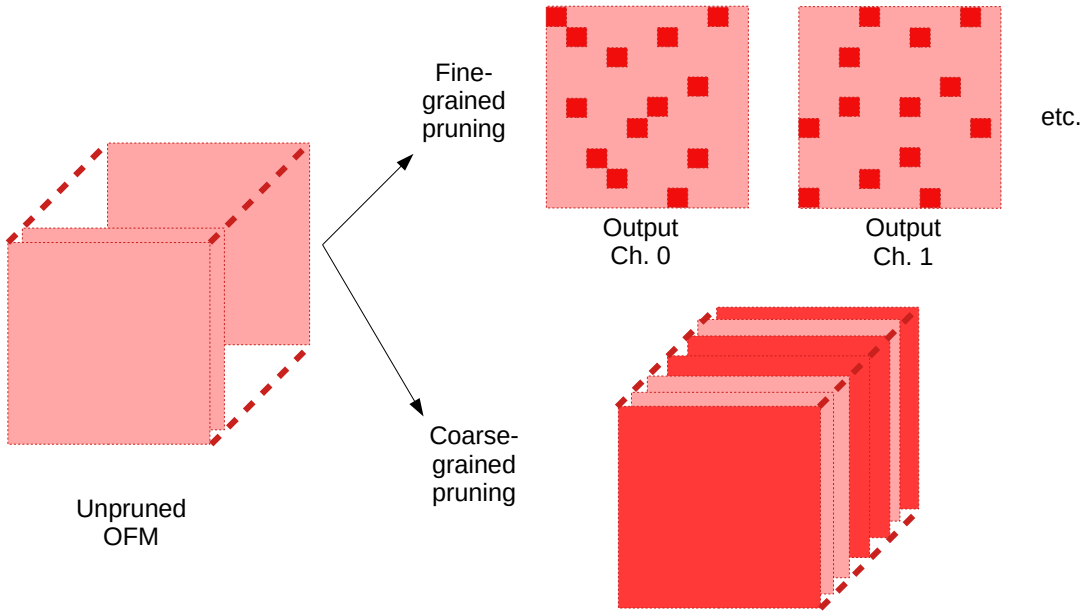


Figure 5.4: Conceptual examples of fine-grained and coarse-grained (channel) pruning. Darkened activations represent pruned values.

dynamic pruning into two categories (illustrated in Figure 5.4), depending on how pruning decisions are performed: fine-grained and coarse-grained.

Fine-grained dynamic pruning makes individual (not necessarily independent) pruning decisions for each activation. One example of a simple fine-grained pruning method is to use an activation function with a built-in threshold, such as ReLU, wherein any output activation below the threshold (in the case of ReLU, this threshold is 0) is pruned [10]. Other simple fine-grained pruning techniques generally make pruning decisions on each activation based on the value of the activation itself, but more sophisticated techniques do exist [19, 60, 34].

In each layer, as parts of the pruned OFM are computed and then typically compressed to reduce the total number of memory accesses [48]. The simplest compression method is to compress each output channel individually, using a sparse matrix format such as *compressed sparse row* format (*CSR*). While the full pruning

vector in this case may consist of one bit per every activation in the entire network, it is typically more practical to consider the *compressed* pruning vector, where each element corresponds to a different output channel and contains the number of nonzeros (NNZs) of each channel.

Coarse-grained pruning is a special case of fine-grained pruning, in which pruning decisions are made on a per-channel basis (i.e. either the entire channel is pruned or it isn't). Since coarse-grained pruning takes many activations into account for each pruning decision, the decision logic is often more complex than fine-grained pruning, and one common approach is to train/operate a decision network in parallel with the CNN [16, 35, 52, 6]. Despite the more complex implementation, the advantage of coarse-grained over fine-grained pruning is that skipping memory accesses and computations is easier to implement, since channels are stored contiguously in memory.

We will consider both cases for the attack demonstration, but focus on the fine-grained case when dealing with protection schemes.

5.2 Threat Model

In this study, we perform ML-driven attacks on dynamically-pruned CNNs used for image classification. We consider a victim CNN that classifies input images using a publically known model. The victim's input images, the model weights, and all IFM/OFM intermediate values are stored in off-chip DRAM memory and loaded into on-chip memory as needed. We assume that the feature maps of at least one of the convolutional layers are too large to fit entirely in on-chip memory, since an off-chip memory access based attack is simply impossible without off-chip memory

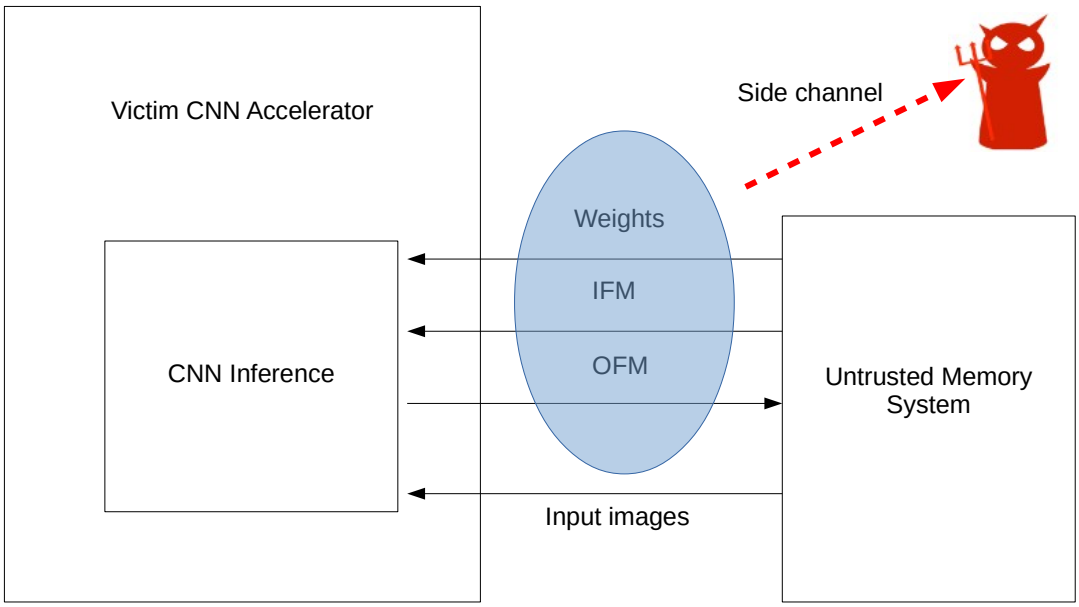


Figure 5.5: Threat model

accesses. The off-chip memory is also assumed to be encrypted, or else the attack is trivial. Finally, the adversary has access to a copy of the trained victim CNN that is used to run offline experiments using its own input data, and they can monitor the addresses of all victim off-chip memory accesses (both reads and writes) via physical access to the memory bus between on-chip and off-chip memory or by other means (such as memory side channels). If any protection mechanisms have been implemented, we assume the attacker is also aware of them. For instance, this threat model may be applicable in CNN accelerators or on cloud computing platforms shared by the adversary and victim.

An architecture for such a system can be seen in Figure 5.5, and the attack objective is to use the victim's memory access patterns to infer the victim's classifications.

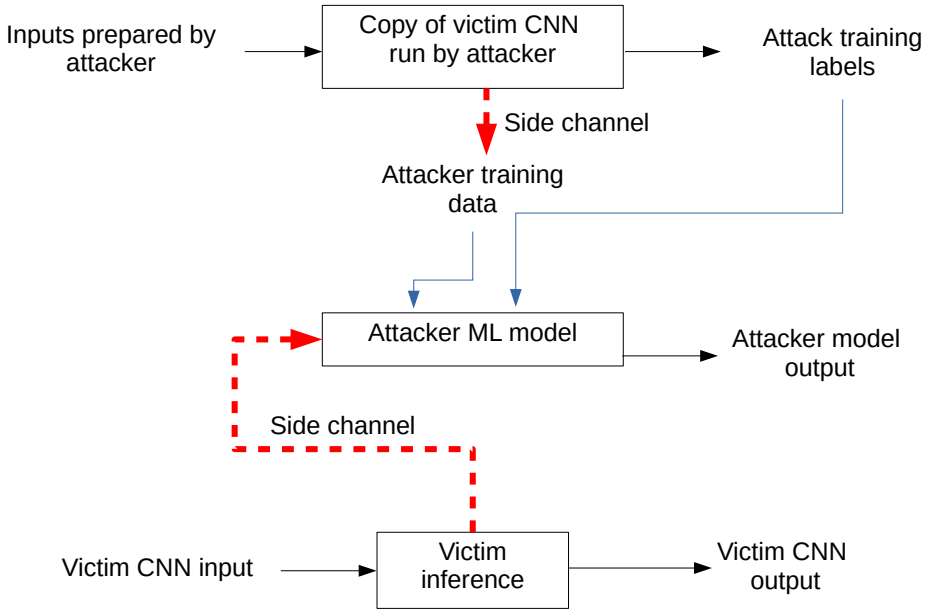


Figure 5.6: Attacker workflow

5.3 Attack Methodology

To perform the attack (summarized in Figure 5.6), the attacker first uses the victim copy to classify their own input dataset and collects memory access patterns for each input. Using these memory access patterns, they extract the pruning decisions made in all convolutional layers (as a fixed-format *pruning vector*) in order to obtain a training set of pruning vectors and labels. The attacker then trains a supervised ML model to infer labels from pruning vectors, which they use on the victim's memory access patterns. The attack is considered successful if the attacker's ML model can predict victim classifications with accuracy significantly exceeding a random guess.

5.3.1 Attacker Models

Naturally, a key step in this attack is choosing an appropriate attacker ML model. From the attacker's point of view, they are dealing with a classification learning problem with the following properties.

First, the attacker is free to generate an arbitrarily large amount of training data. Second, the pruning vector size will typically be much smaller than the input size(s) used by the victim, so the attacker model does not need to be as complex as the victim model. Third, pruning vectors may or may not have strong internal spatial correlations but are generally sectioned-off by layer, so the best attacker model may change depending on the pruning technique used by the victim.

For this study, we use two types of attacker models: multilayer perceptrons (MLPs) and gradient boosting. Since CNNs are generally thought to be effective on applications where the input data has strong spatial relationships (they are used for image classification for this very reason), it is not clear that they would be effective in general for this attack where such relationships do not necessarily exist. Thus, MLPs may be used since they thrive on abundant training data and do not require spatial correlations within the pruning vector to work.

Alternatively, since the pruning vectors are fixed-format, specific locations within the vectors may hold individual significance. Gradient boosting and random decision forest models excel at identifying these decision points within the input vector. The ensemble decision tree model of XGBoost [5], for example, is a quick and easy general-purpose model for this attack.

5.3.2 Obtaining Pruning Vectors

For fine-grained pruned CNNs, there is a separate pruning decision made for every individual scalar activation, which means that there are potentially as many dimensions to the attacker training data as there are activations across all layers. While it is certainly possible to train the attacker ML model using such detailed pruning vectors, such low-level representation of the pruning decisions is generally unnecessary and difficult even with physical access to the memory bus. Instead, we let the pruning vector contain the number of unpruned activations (*number of nonzeros* or *NNZs*) in each channel in each convolutional layer. The NNZs of each channel can be inferred by making use of the previously-described OFM compression method, in which each channel is individually compressed using a sparse matrix compression format such as CSR. Crucially, the most efficient sparse matrix compression formats reveal the NNZ of the original matrix through the size of the compressed matrix, which can be detected by counting the number of memory accesses needed to load the compressed matrix.

For coarse-grained pruned CNNs, there is a separate pruning decision for each channel in each convolutional layer, where each individual pruning decision is a simple pruned/not pruned bit. Thus the pruning vector simply takes the form of a bit array for each channel in each layer of the victim CNN. These bits can be inferred by observing whether or not the memory locations corresponding to each channel was accessed by the victim, for each layer. If a particular channel is skipped, then the attacker will be able to detect a contiguous block of missing memory accesses by the victim, indicating that the channel was pruned.

5.4 Attack Characterization Study

In this section we present a series of attack experiments. First, we show that the proposed class of attacks is effective, by demonstrating its efficacy on a variety of victims, representing both fine-grained and coarse-grained pruning networks, targeting the victim classifications. Second, we empirically analyze the nature of the attack space by performing the above attack while allowing the attacker access to selected subsets of the pruning decisions or implementing rudimentary protection. Finally, we conduct experiments exploring attack model specificity; that is, we study whether protection schemes can thwart the attack when the attacker does not have knowledge of the protection scheme.

This section will therefore motivate the remainder of this study, which focuses on developing protection against knowledgeable attackers who are aware of any designed protection scheme.

5.4.1 Attack Demonstration

To demonstrate our attack, we simulated attacks on ReLU-based (fine-grained) pruning using AlexNet, Resnet20, and Resnet18 as victims, and we simulated software-based attacks on coarse-grained pruning using the Feature-Boosting and Suppression (FBS) [16] technique. In order to generate the attacker's training dataset, we used mutually exclusive attack training/test sets (extracted using ImageNet and CIFAR10/100 test sets), and used the victim's classifications as the training labels. Note that we chose not to use the ImageNet or CIFAR training sets to ensure that the attacker model does not gain any advantage from training with the same dataset as the victim itself. All victim models are quantized as

32-bit floating point numbers. For the attacker model, we used both a 3-layer MLP and XGBoost [5], either of which were sufficient to recover classifications with significant accuracy. In this section, we discuss our experimental procedures and results for fine-grained pruning and coarse-grained pruning separately.

Experimental Procedure

For our attacks on fine-grained pruning, we used the previously described ReLU-based pruning technique. It is performed by allowing ReLU's built-in threshold to determine which activations to skip. Then, for each output channel, we counted the number of positive activations (*number of nonzeros* or *NNZs*). The pruning vector is an array of NNZs where each element corresponds to a different output channel for a different layer. This pruning vector format represents the CSR-compressed size of each output channel, inferred from the number of victim memory accesses per channel.

For an attack on coarse-grained pruning, we used the FBS channel-pruning technique, which uses per-layer trained networks to infer which channels are most important. The pruning vector is a binary array, where each bit indicates whether FBS decided to prune a particular channel in a particular layer.

For the victim models, we experimented with AlexNet trained on ImageNet and ResNet trained on CIFAR10/100 for the fine-grained attacks, and CifarNet trained on CIFAR10 for the coarse-grained attacks.

We used two different attacker models for all of these experiments. First, we used a 3-layer MLP (i.e. input layer, hidden layer, and output layer) with a couple different hidden layer sizes, both with and without basic normalized pruning

vectors (if the pruning vectors are normalized, then each entry is the NNZs divided by the total number of activations in that channel). The activation function used is negative log-likelihood, and the training algorithm used is gradient descent (50 epochs for all experiments). For XGBoost experiments, we simply used the toolbox's default gradient boost training algorithm, with a max tree depth of 6 with 10 boosting rounds. These experimental results can be seen in Tables 5.1 and 5.2. We measured the top1/3/5 accuracy by checking whether the attacker's guess matches with one of the top 1, 3, or 5 labels according to the victim.

Note that for the MLP, choosing more complex hidden layers and normalizing the pruning vector does have some positive effects for the attacker. However, the 512×1024 hidden layer also caused training to take significantly longer, especially for larger networks, for relatively small benefits.

Experimental Observations

We observe that, if left unmitigated, the attacker is able to predict the victim's classifications from these pruning vectors, both in the fine-grained and coarse-grained cases, with significantly better-than-random accuracy. In addition, simple grouping-based protection based on cache behavior does not sufficiently mitigate our attack. From these experiments we conclude that the pruning vectors are information-rich and that simple protections are not enough to prevent such attacks to any reasonable degree. Moreover, since information embedded in the pruning vectors is not restricted to the victim's classifications only, the potential for further privacy-violating attacks is high.

Vic/Data/Atk	norm?	top1 %	top3 %	top5 %
AlexNet/ImageNet/MLP(256×512)	n	24.36%	39.38%	46.67%
AlexNet/ImageNet/MLP(512×1024)	n	28.17%	44.17%	51.57%
AlexNet/ImageNet/MLP(256×512)	y	28.03%	42.65%	50.19%
AlexNet/ImageNet/MLP(512×1024)	y	30.95%	46.24%	53.07%
ResNet20/CIFAR10/MLP(256×512)	n	71.16%	90.76%	96.5%
ResNet20/CIFAR10/MLP(256×512)	y	77.37%	93.33%	97.57%
ResNet20/CIFAR100/MLP(256×512)	n	57.93%	83.53%	92.67%
ResNet20/CIFAR100/MLP(256×512)	y	64.10%	89.47%	95.8%
ResNet18/CIFAR100/XGB	n	40.03%	63.13%	74.00%

Table 5.1: Test accuracy of output attacks on fine-grained pruning. Used AlexNet and ResNet as experimental victim models trained on ImageNet or CIFAR datasets. Attacker model is either a 3-layer MLP or XGBoost (XGB) with a max depth of 6 and 10 boosting rounds. MLP hidden layer sizes are indicated as input layer size times output layer size. Top1/3/5 indicates attack success rate against the top 1/3/5 most probable victim classifications.

Model/Dataset	top1 %	top3 %	top5 %
FBS-CifarNet/CIFAR10/MLP	61.63%	86.8%	94.47%
FBS-CifarNet/CIFAR10/XGB	61.63%	85.57%	94.77%

Table 5.2: Test accuracy results of output attacks on coarse-grained pruning. Used CifarNet with Feature Boosting and Suppression (FBS) pruning as victim model, trained on CIFAR10 (CF10).

L1 Cache Attack Procedure

We also present a detailed methodology and obstacles to successfully performing this attack using an L1 cache attack to obtain the pruning vectors. Here, referring to Figure 5.5, we assume the adversary is a process running on the same physical core as the victim process in order to ensure that both processes share an L1 cache. We also assume that the adversary process spawns immediately when the victim process begins running, although in an actual attack, the attacker must have other means of synchronizing with the victim.

Then, the attacker simply performs a standard L1 cache prime-and-probe at-

tack. An example procedure is outlined below using an 8-way set associative L1 cache with 64 cache sets and 64-byte cachelines:

1. Allocates 8 pages of its own data
2. Access all 8 pages worth of data, thereby filling up the the L1 cache
3. Wait for the victim to run. When the victim accesses the activation and weight data, it evicts the adversary's data
4. Access all 8 pages worth of data again, this time monitoring how long each access takes. Short accesses indicate cachelines untouched by the victim; long accesses indicate cachelines accessed by the victim.
5. Steps 2-4 are repeated for as long as the victim executes each classification

In order to distinguish between short and long accesses, the attacker must first conduct a controlled experiment on the sandbox copy of the system to measure the duration of a cache hit vs the duration of a cache miss. From these experiments, a duration threshold can be determined.

Then, the result of this procedure produces a cache trace over time for the victim's classification. In order to convert this trace into a pruning vector, the attacker must have knowledge of how each channel in each layer maps to different cachelines. For instance, if it is known that in a particular layer, the first channel's activations occupy cache sets 0-5, then a cluster of cache misses in those cache sets indicates that that channel was not pruned.

5.4.2 Attack Space Study

As we have now established that the proposed attack works, we now investigate its robustness. That is, how sensitive is the attack success rate to incomplete or protected features in the pruning vector? We investigate this question from two angles. First, we impose simplistic protection on the pruning vector equivalent to what already exists by default if the attacker tries to obtain the pruning vectors from a cache side channel. Second, we attempt the attack using special subsets of the layers' channels. In both cases we find that the attack is relatively robust.

Granularity Reduction Technique and Experiments

The protection scheme we applied to the fine-grained pruned victims is granularity reduction. Mathematically, this involves reducing the granularity of the alphabet of the observable NNZ for each channel, by grouping different possible values of NNZs into uniformly-sized bins. Indeed, in later sections we will refer to this same technique as *uniform binning*.

In terms of the actual implementation, this protection scheme entails grouping neighboring activations within each channel into fixed-size groups, so that each group of activations are treated pruned or not pruned together. From the perspective of the attacker, this results in a reduced NNZ alphabet, thereby reducing the amount of information per channel. Moreover, this style of protection mimics natural cache behavior in which multiple activations fit on a single cacheline and the attacker is only able to count the number of cachelines accessed per channel.

For our experiments we assumed various cacheline sizes in powers of 2. These results can be seen in Table 5.3. While not shown, we also tried cacheline sizes

that were not powers of 2, and oddly these cases resulted in close no improvement for the attacker (thereby thwarting the attack), but this is likely a property of the MLP used, not of the attack itself.

Vic/Data/Atk	cacheline sz.	top1 %	top3 %	top5 %
AlexNet/ImageNet/MLP(256×512)	4	22.33%	36.50%	42.90%
AlexNet/ImageNet/MLP(256×512)	8	19.71%	33.38%	40.16%
ResNet20/CIFAR10/MLP(256×512)	2	71.17%	90.77%	96.50%
ResNet20/CIFAR10/MLP(256×512)	4	63.43%	86.63%	94.67%
ResNet20/CIFAR10/MLP(256×512)	8	51.46%	81.17%	92.17%
ResNet20/CIFAR100/MLP(256×512)	2	57.93%	83.53%	92.67%
ResNet20/CIFAR100/MLP(256×512)	4	48.80%	78.73%	90.57%
ResNet20/CIFAR100/MLP(256×512)	8	40.63%	72.87%	87.47%
ResNet18/CIFARF100/XGB	16	31.50%	54.30%	65.53%

Table 5.3: Test accuracy of output attacks on fine-grained pruning with granularity reduction. Where noted, RG-N is a granularity reduction scheme that groups activations into groups of N

We remark here that the reduction in attack accuracy of these reduce-granularity attacks is small relative to the group size. Maximal leakage for instance, would tell us that the uniform binning protection scheme reduces the leakage by a factor equal to the group size, but in reality the guessing probabilities have only decreased by a few percentage points in each case. These experiments therefore suggest that our attack is robust at least against the information loss caused by obtaining the pruning vectors through a cache.

Selective Layers Experiments

We study the impact of missing elements from the pruning vector on the output attack success rate. For these experiments, we focus on ResNet18, which has a total of 20 convolutional layers, which can be further categorized into 4 groups. ResNet18 contains 5 layers each with 64 channels with 1024 activations, 5 layers

each with 128 channels with 256 activations, 5 layers each with 256 channels with 64 activations, and 5 layers each with 512 channels with 16 activations, in that order. The full, unprotected pruning vector therefore contains 4800 attack training features (NNZs) with four different alphabet sizes of 1025, 257, 65, and 17. We then conducted the following experiments by selectively allowing the attacker to see subsets of these 20 layers.

First, we allowed the attacker to only see one unprotected layer at a time. The results can be seen in Table 5.4. From these results, we can clearly see that there is clear upward trend in the relative usefulness of each layer as we move towards the later layers. In fact, the last two layers in particular seem to be significantly more useful to the attacker than the first 18, as one might expect for output attacks. Moreover, using the last layer alone is sufficient to achieve the same roughly 40% accuracy achived in Table 5.1 using the entire pruning vector.

Second, in order to better characterize just how important the last few layers are for the output attack, we allowed the attacker access to only the last 10 layers and used uniform binning to gradually reduce the alphabet sizes of those layers. These results can be seen in Table 5.5. We find that no significant reduction in the attack accuracy can be achieved with uniform binning without fully protecting layers 16-20 (represented by when the layer 16-20 alphabet sizes reach 1). Moreover, even after fully protecting layers 16-20, layers 11-15 continue to leak information comparable to accuracy levels found in rows 11-15 in Table 5.4 until they too are fully protected. This experiment shows that, even with the intuition that the later layers are more important to the attacker, naive protection of the last 10 layers is not sufficient to attain a useful trade-off.

Finally, in order to probe the possibility of inter-layer relationships within the

Layer	No prot. acc %
1	11.70%
2	15.97%
3	14.77%
4	14.47%
5	11.10%
6	15.03%
7	14.13%
8	15.97%
9	16.23%
10	13.43%
11	17.03%
12	17.87%
13	16.03%
14	19.00%
15	17.30%
16	16.87%
17	16.63%
18	17.10%
19	29.13%
20	42.97%

Table 5.4: ResNet18 one layer at a time selective layer experiment results

Lay 11-15 alph sz	Lay 16-20 alph sz	accuracy %
65	17	40.87%
46	12	41.17%
33	9	40.97%
23	6	39.37%
16	4	40.27%
11	3	38.70%
8	2	39.83%
6	2	41.37%
4	1	19.27%
3	1	18.83%
2	1	18.83%
1	1	1.93%

Table 5.5: ResNet18 last 10 layers (out of 20 layers total) experiment results

pruning vector, we tried removing every other layer (resulting in a pruning vector with only the even-numbered layers) and then applying cacheline-like protection

as in the experiments from Table 5.3. These results are in Table 5.6, and we find that the attack is still relatively robust against this form of protection.

cacheline sz.	accuracy %
1	28.17%
2	29.50%
4	29.40%
8	23.40%
16	19.53%
32	17.73%
64	17.00%

Table 5.6: ResNet18 even layers only (out of 20 layers total) experiment results. Cacheline sizes are given in the number of activations that fit on each cacheline. Attack model is XGBoost.

Takeaways of the Attack Study

We have seen that a supervised ML approach to attacking dynamically pruned CNNs is effective on output attacks, specifically the victim classifications, over multiple CNN architectures. Simple granularity reduction (or uniform binning) is largely ineffective at preventing these attacks unless taken to extremes approaching full protection. We can confirm that the intuition that later layers are more useful for output attacks is true, but it is not sufficient to simply protect the last few layers. A more sophisticated analysis of the protection space will be presented in the following sections.

5.5 Information Characteristics of Feature Sets

For any machine learning-based protection problem, not necessarily restricted to our CNN attack, it is useful to consider the nature of the information about the

learning target contained within the feature set. For instance, two or more individual features may contain a mixture of the same and distinct information about the target, or they may be individually independent of the target but very informative only when viewed together. And of course, it is possible for some features to be completely irrelevant to a particular attack target. Within a feature set, these types of inter-feature relationships may be very complex and not necessarily obvious or intuitive.

However, even without knowing every single inter-feature relationship in detail, insight into these behaviors within a feature set is useful for devising protection schemes for that feature set. In this section, we aim to consolidate the information characteristics of a large, unprotected feature set into a small number of analyzable properties, which are used to provide a meaningful look into the nature of a many-feature learning problem. We will consider the class of protection problems in which an adversary uses machine learning to infer a secret from a large feature set, and we wish to prevent them from learning this secret by modifying the features with a protection scheme.

To this end, we define three general properties: *feature importance*, *feature dispersion*, and *information overlap*. We use mutual information to quantify these properties and study how these measures impact the protection problem. We propose one diagnostic test to analyze feature importance and dispersion and two diagnostic tests to analyze information overlap. Then, we use synthetically generated data to construct test cases to empirically validate these tests.

5.5.1 Feature Importance and Dispersion Definitions

We first define feature importance and importance dispersion.

Definition 22. ***(Feature Importance)***

For a given attack target, we define individual feature importance *as a measure of the amount of information about the target contained in a single feature. Similarly, we define the group feature importance as a measure of the joint amount of information about the target contained in a group of features. The group importance of the entire feature set is effectively a (non-bounded) measure of the overall leakage with respect to the target.*

We elect to use mutual information as a metric of feature importance instead of maximal leakage, because maximal leakage is typically too conservative to distinguish between the individual usefulness of features in the unprotected case. By contrast, mutual information is a reasonable measure of information about the target contained in the feature(s), provided that the data sample used to estimate it has representative marginal distributions. Hence, let individual importance of feature F with respect to target T be measured as $I(F;T)$ and the group importance of features $(F_1, F_2, ...F_n)$ be measured as $I(F_1, F_2, ...F_n; T)$.

Note however, that since estimating multi-variable mutual information requires a large amount of data, scaling exponentially with the number of features considered, it is typically not feasible to measure large-group importances in practice. While group importance is a useful theoretical concept, in practice we will mostly use individual importance for our analyses.

Definition 23. ***(Dispersion)***

For a given attack target, its information dispersion *is a measure of how widespread the target information is among all individual features. There are two subproperties of interest:*

- *Dispersion Count: The number of features with high (i.e. greater than some threshold) individual importance*
- *Dispersion Distribution: The distribution of importance across all features.*

5.5.2 Impact of Dispersion on the Protection Problem

It stands to reason that dispersion impacts the protection problem by defining the trade-off space. For instance, if a feature set contains a number of highly important features with respect to the attack target, then all of those features must be protected to achieve a high degree of protection. In other words, dispersion count determines the total amount of cost that can be usefully incurred in order to reduce the overall leakage. For instance, a feature set with only one important feature has a relatively small cost range, because it is only necessary to incur cost on protecting the one important feature. On the other hand, a feature set where every feature is important has a large cost range, because protection on any and all features is theoretically useful in reducing the overall leakage.

Dispersion distribution affects the relative efficiency of protecting certain features compared to others. For instance, consider two features with unequal individual importance. Then, the feature with greater importance has a greater impact on the overall leakage when protected (by definition). When normalized against unit cost of each of the features, the dispersion distribution can be used to determine which features are most cost-efficient to protect.

5.5.3 Information Overlap Definitions

We introduce the concept of information overlap between two or more features with respect to a target in order to analyze the degree of shared information about the target between the features. The overlap must capture whether the features contain shared information about the target, distinct information, or leak information jointly, and to this end, we define the following.

Definition 24. ***(Information Overlap)***

For any pair of features F_1 and F_2, a target T, and a generic leakage metric L, define information overlap *between F_1 and F_2 with respect to T as $OL(F_1, F_2; T) = L(F_1 \to T) + L(F_2 \to T) - L(F_1, F_2 \to T)$. Then,:*

- *If $OL(F_1, F_2; T) > 0$, we say F_1 and F_2 are* overlapping *with respect to T*
- *If $OL(F_1, F_2; T) = 0$, we say F_1 and F_2 are* non-overlapping *with respect to T*
- *If $OL(F_1, F_2; T) < 0$, we say F_1 and F_2 are* anti-overlapping *with respect to T*

Similarly define the group overlap *of features $(F_1, F_2, ...F_n)$ as $L(F_1 \to T) + L(F_2 \to T) + ... + L(F_n \to T) - L(A, B, C, ... \to T)$.*

From this point on, we will again use mutual information in place of L when measuring overlap. Also, in practice, it is useful to apply a close-to-zero threshold when distinguishing non-overlapping features from overlapping/anti-overlapping ones.

We further introduce special terminology relating to group overlap behaviors as follows:

Definition 25. ***(Group Overlap Behaviors)***

Non-overlapping items *within a feature set are either single features or groups of features that are pairwise non-overlapping. Note that two groups of features* $(F_1, ...F_n)$ *and* $(G_1, ...G_m)$ *are pairwise non-overlapping if* $L(F_1, ...F_n \rightarrow T) + L(G_1, ...G_m \rightarrow T) - L(F_1, F_2...F_n, G_1...G_m \rightarrow T) = 0$.

An overlap group *is a group of features such that each feature is pairwise overlapping with at least one other feature in the group, but are pairwise non-overlapping/anti-overlapping with all features outside of the group.*

An anti-overlap group *is a group of features such that their group overlap is negative and does not contain a subgroup with a negative group overlap.*

Crucially, note that the definitions of overlap groups and anti-overlap groups are constructed fundamentally differently. An overlap group contains a network of features that are overlapping with each other, but it is possible to find a pair of features that are non-overlapping or even anti-overlapping within the group. By contrast, an anti-overlap group cannot contain overlapping pairs.

5.5.4 Overlapping Features: Independent Looks vs. Duplicate Features

Within the overlapping features case, there exists a further distinction. There are generally two ways for a pair of features to be overlapping with respect to a target.

Either the features are partial duplicates of each other or they are conditionally independent looks at the same target.

Definition 26. ***(Independent Looks vs Duplicate Features)***

For important and overlapping features F_1 and F_2 with respect to target T, F_1 and F_2 are independent looks *if they are independent, conditioned on T. Otherwise they are considered to be (partially) duplicative.*

Note that $I(F_1; F_2|T)$ is an appropriate metric of conditional independence. If it is zero, that means F_1 and F_2 are conditionally independent given T. However, if it is nonzero, that fact alone is not sufficient to conclude that F_1 and F_2 are duplicative; it must also be known that the features are individually important and overlapping.

5.5.5 Impact of Overlap on the Protection Problem

The presence of overlapping important features increases the difficulty of the protection problem. This is because when there exists two overlapping features and only one of them is protected, some shared information is still present in the pruning vector. Hence the impact of protecting only one feature is reduced by the presence of the other feature. So, when there exists a overlap group, every member of the overlap group must be simultaneously protected to achieve significant leakage reduction. Further note that duplicative features further enhance this effect, compared to independent looks, because in the independent looks case, the attack is exponentially weakened by reducing the number of looks, as governed by the theorems presented in the previous chapter. By contrast, protecting one duplicate feature does not reduce leakage at all.

On the other hand, the presence of anti-overlapping important features decreases the difficulty of the problem. This is because removing or protecting any one member of an anti-overlap group will greatly reduce the group's leakage. This effectively reduces the number of features that must be protected, and an entire anti-overlap group can be protected simply by protecting one of its members.

Finally, the presence of non-overlapping features neither increases nor decreases the difficulty of the protection problem. The impact of protecting a feature that is non-overlapping with every other feature is isolated to that feature alone, and will reduce the overall leakage by some amount related to the feature's individual importance. However, it should be noted that in most practical learning problems, it is unlikely for a large feature set to contain a large number of non-overlapping features relative to the alphabet size of the target, since each non-overlapping item contains a slice of the sum total information about the target. The more non-overlapping features there are, the more diluted the total information about the target is. Unless the target itself is a collection of individual targets, typically learning targets will not be so conveniently divided up.

5.5.6 Synthetic Data Generation for Empirical Analyses

In this part, we discuss how to synthetically construct groups of features falling under different overlap cases. For each case, we will start by explaining how to generate two features with the desired overlap behavior and then discussing how to expand that procedure to multiple features, as well as how these procedures scale with the desired number of features. Note that in all cases, the generated features and targets will have pre-determined alphabet sizes.

Generating Overlapping Features

To generate two features and a target such that the features are overlapping and important with respect to the target:

1: Inputs: P_T (distr. of target), P_n (distr. of noise)

2: Generate $T \sim P_T$

3: Generate $Z_1, Z_2 \sim P_n$

4: $F_1 \leftarrow T + Z_1$

5: $F_2 \leftarrow T + Z_2$

6: **return** F_1, F_2, T

This algorithm will result in a simple pair of overlapping features, with the degree of overlap controlled by the noise distribution relative to the target distribution. Finer control over the alphabet sizes of the features can be achieved by re-scaling the $T + Z$ steps by any linear factor. Moreover, this algorithm can be easily scaled to many more features if desired; simply generate more Z variables and generate additional features similarly.

Note that the features generated are independent looks at the target. A slightly modified algorithm to generate (partially) duplicative features is as follows:

1: Inputs: P_T (distr. of target), P_n (distr. of noise)

2: Generate $T \sim P_T$

3: Generate $Z \sim P_n$

4: Generate $S_1, S_2 \sim Gaussian(0, 1)$

5: $F_1 \leftarrow Round(T + Z * S_1)$

6: $F_2 \leftarrow Round(T + Z * S_2)$

7: **return** F_1, F_2, T

Again, fine control over the feature alphabet sizes can be achieved by linear re-scaling. Note however, that if the alphabet size of the features is too small, rounding effects may break the duplicative nature of the generated features. Again, any number of duplicative features can be generated this way.

Generating Non-overlapping Features

To generate a pair of non-overlapping important features:

1: Inputs: P_F (distr. of features)

2: Generate $F_1, F_2 \sim P_F$

3: Define N as max possible value achievable by P_F

4: $T \leftarrow N * F_1 + F_2$

5: **return** F_1, F_2, T

This algorithm results in a pair of non-overlapping features with nonzero importance. Finer control over the alphabet size of the target can be achieved by taking roots of the features and re-defining N and T accordingly. Crucially, note that this algorithm (without feature-scaling) will result in an exponetially-growing target alphabet size as more features are added. Even taking roots of the features to construct the target has its limits because doing so greatly skews the target distribution. Fundamentally, these limitations are caused by the nature of generating a large number of overlapping features. Either the target must contain a large amount of information, or the individual importance of the generated features must be greatly diluted.

Generating Anti-overlapping Features

To generate a pair of anti-overlapping features with positive group importance:

1: Inputs: P_F (distr. of features)

2: Generate $F_1, F_2 \sim P_F$

3: Define N as max possible value achievable by P_F

4: $T \leftarrow (F_1 + F_2)$ modulo N

5: **return** F_1, F_2, T

This results in a pair of anti-overlapping features such that their group importance is positive but their individual importance is zero. Finer control over the alphabet size of the target can be achieved by linear scaling of T, and the algorithm can be scaled to more features by extending the pre-modulo sum to more features.

Combining Non-overlapping Items

With the above algorithms, it is possible to synthetically generate an overlapping group, a collection of non-overlapping features, or an anti-overlapping group. When executed individually, each of these algorithms will give their own feature sets and targets. Then, in order to combine multiple feature sets and targets as non-overlapping items, one can simply re-purpose the non-overlapping features algorithm to combine multiple targets into a single target. The features are simply concatenated together into one larger feature set.

In this way, we can construct a diversity of example feature sets exhibiting mixtures of overlap behavior. In the following subsections, we will use these techniques for some of our experiments.

5.5.7 Analyzing Importance, Dispersion, and Overlap

In practice, in order to gauge the individual feature importances, it is necessary to compute mutual information from potentially limited data. The problem of estimating functionals of discrete random variables with unknown marginal distributions is a nontrivial one and has been the subject of a subfield of study in information theory ([26, 59]). However, for the purposes of a first-order analysis of feature importance, it is typically not necessary to obtain high-precision estimates of mutual information.

Instead, we simply consider the "plug-in" estimate of mutual information, in which the marginal distributions of all relevant random variables are approximated as the sampled empirical distribution, and then these distributions are directly plugged into the mutual information formula. The mean square error of the plug-in single-variable entropy for n data samples is given by [59]:

$$E_{mse}(\tilde{H(X)}) = \frac{|\mathcal{X}|^2}{n^2} + \frac{\log^2 |\mathcal{X}|}{n} \tag{5.1}$$

When n is relatively small compared to the alphabet size of X, the first term dominates the error. Note, when estimating $I(F;T) = H(T) - H(T|F)$, the error is dominated by $H(T|F)$ where the *joint alphabet size* is $|\mathcal{T}| \times |\mathcal{F}|$. Thus for a first-order approximation of $I(F;T)$, mathematical theory dictates that we need the number of data samples to be the same order of magnitude (preferably greater than) the joint alphabet size of all random variables involved, or else the estimate will be too imprecise. Similarly, when dealing with mutual information of 3 or more variables, the joint alphabet size grows exponentially with the number of features so the necessary number of data points grows accordingly.

Synthetic Data Scenarios

In order to validate this rule, we use synthetically generated data using the techniques described in the previous subsection. In particular, for all of the experiments done to validate feature importance estimation, we generated test cases with the following overlap behaviors:

- Features A and B are non-overlapping with respect to target TarAB
- Features A and C are overlapping with respect to target TarAC. The degree of overlap between A and C will be specified per-experiment.
- Features A and D are anti-overlapping with respect to target TarAD.
- Feature E is irrelevant to tarAE, while feature A has high importance.

Estimation of Feature Importance

In the following set of experiments, we use feature A and target TarAC. We plot the estimated mutual information $I(A; TarAC)$, which has a joint alphabet size (JAS) of $|\mathcal{A}| \times |TarAC|$. In each of Figures 5.7, 5.8, and 5.9, we select a JAS with orders of magnitude of 10^2, 10^3, and 10^4 respectively and plot the estimated mutual information against the number of data samples.

We observe that our theoretically established rule of using as many data samples as joint alphabet size is typically sufficient to reduce the error of the estimate to within 0.2 bits or less. However, it should also be noted that the error rises rapidly if the number of data samples even slightly smaller than the JAS. Thus, we conclude that the plug-in method is not recommended if sufficient data cannot be obtained to exceed the JAS. This is typically not a problem for 2-variable mutual

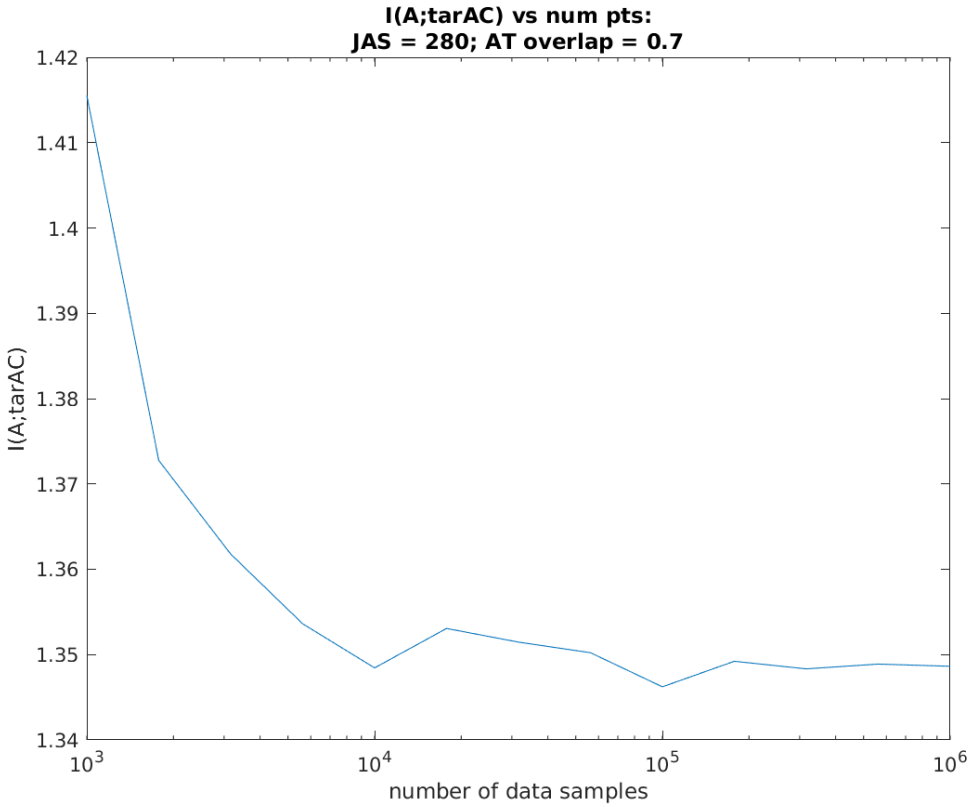


Figure 5.7: JAS order 10^2 feature importance estimates

information estimates (such as for feature importance) within the context of CNN pruning attacks, because both the feature and target alphabet sizes typically do not exceed an order of magnitude of 10^3.

5.5.8 Analyzing Overlap Behavior

We also test the plug-in method to compute overlap under different overlap cases in Figure 5.10. We find that the same general conclusion as for estimating feature importance can be reached; having the number of data samples equal or exceed the JAS is sufficient for reasonably accurate estimates of overlap. However, since overlap involves mutual informations with a minimum of 3 random variables

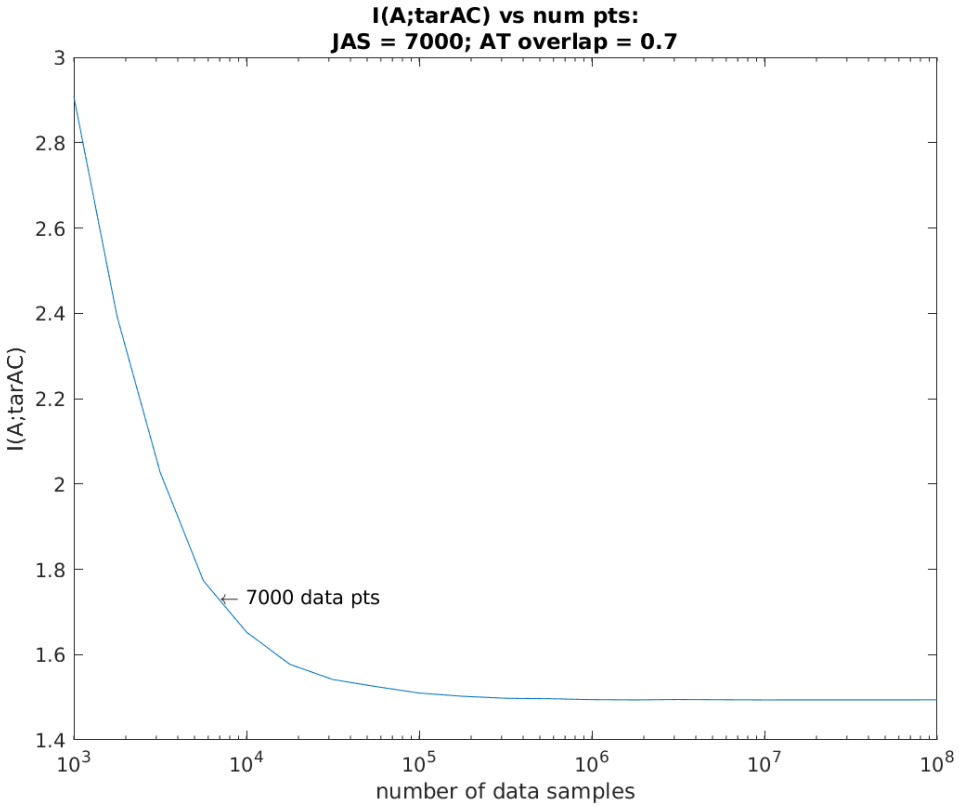


Figure 5.8: JAS order 10^3 feature importance estimates

(two features and a target), large joint alphabet sizes can sometimes occur, if the variables' alphabet sizes are sufficiently large. For instance, in ResNet18, feature alphabet sizes range from as small as 17 to as large as 1025. So, it is not always practical to directly compute pairwise overlap, except for the smallest alphabet features.

Thus in this subsection, we propose a total of three empirical techniques to analyzing the overall overlap behavior within the feature set. The first technique is to simply use the plug-in method, but as it is unreliable for very large features, it can only be used to analyze the smaller features, depending on the amount of data available. The second technique is to simply generate randomized protection and observe the shape of the resultant cost/leakage trade-off curve when the protected

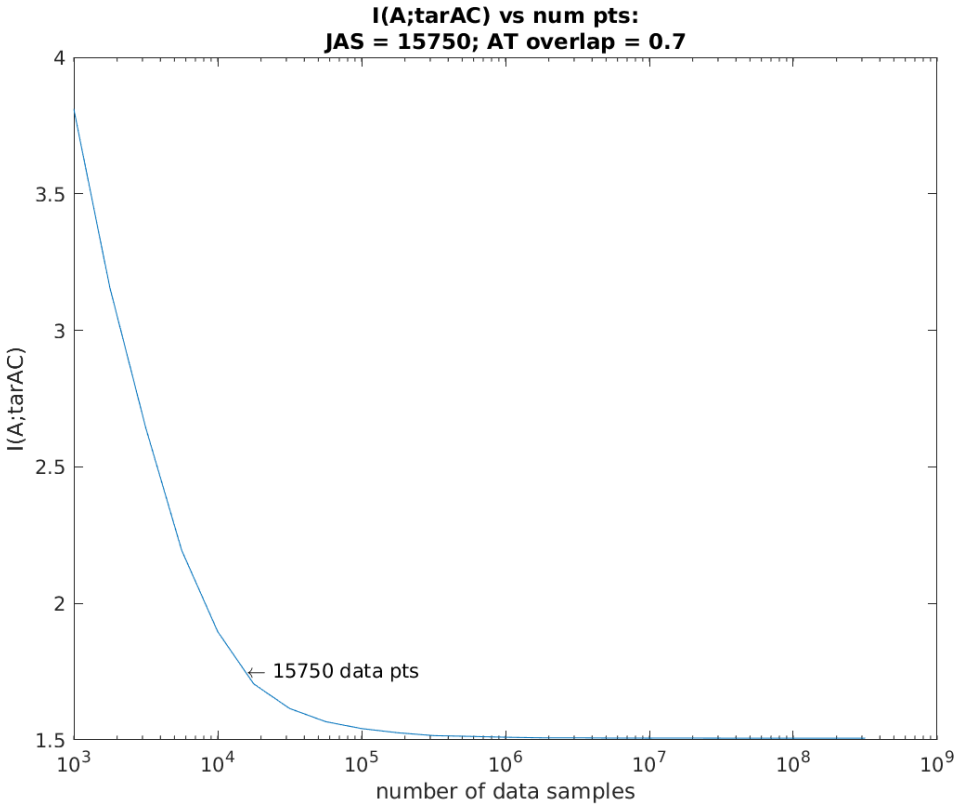


Figure 5.9: JAS order 10^4 feature importance estimates

feature vector is used to perform the attack. We call this technique the Random N-Set Test. Finally, the third technique is to measure the pairwise conditional independence of the features with respect to the target. Note that the second and third techniques are not interchangeable; instead they are meant to be used in tandem when the first technique is not feasible.

In the next two subsections we will focus on the latter two techniques.

Random N-Set Test

The procedure of the Random N-Sets test is as follows. First, take the feature set and randomly partition it into N subsets. The value of N should be relatively small,

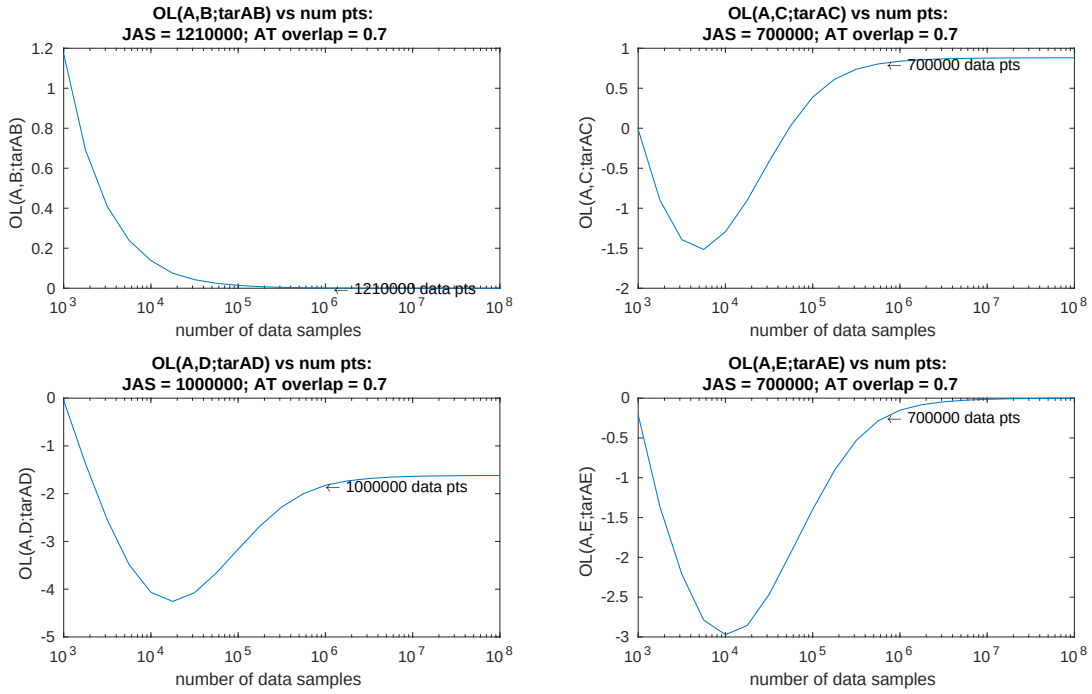


Figure 5.10: JAS order 10^6 overlap estimates. Subplots: 1) A is non-overlapping with B, 2) A is overlapping with C, 3) A is anti-overlapping with D, 4) E is independent of the target

such that each subset is at least 20 features. Then, starting with the unprotected features, progressively fully protecting each subset one by one from the feature vector until full protection on the entire feature set is achieved. In essence, the Random N-Set Test represents random protection of the features. For each step, use an attacker ML model to perform the attack (such as XGBoost).

The intuition of the Random N-Set Test is that when the protection is randomized in this fashion, the predominant overlap behavior of all the features will determine the effectiveness of the randomized protection scheme. A large number of overlapping features will manifest as a highly concave trade-off curve, where almost no leakage reduction is attained until full protection of the entire feature vector is reached. In particular, a distinction between independent looks and duplicate features can potentially be drawn here as well, since independent looks will

result in exponentially fast back-off from the upper limit as features are removed (as shown in the previous chapter) whereas duplicate features will result in faster-than-exponential back-off. By contrast, a predominantly non-overlapping feature set or anti-overlapping feature set will result in a linear or convex trade-off curve.

For the test to work, it is crucial that each subset contains a significant number of features, since we are relying on the low probability that a random partition will result in, for example, all of the overlapping features being in the same subset. Roughly 20 features per subset is more than enough for this probability to be practically negligible, but a few re-randomizations can further reduce this chance to be safe.

One useful metric to represent the Random N-Set Test is the *normalized curve area* or *NCA* of the obtained trade-off curve.

Definition 27. *Given a curve $(x, f(x))$ with well-defined end points $(x_l, f(x_l))$ and $(x_r, f(x_r))$ (where $x_r > x_l$), the normalized curve area is given as:*

$$NCA = \frac{area\ under\ curve}{area\ of\ bounding\ rectangle} \tag{5.2}$$

where the area under the curve is simply the integral of f from x_l to x_r and the area of the bounding rectangle is $[max_x f(x) - min_x f(x)] \times [x_r - x_l]$ (or undefined when f is a constant function).

For monotonic curves, the NCA is a measure of concavity, such that the most concave curve has an NCA of 1 and the most convex curve has an NCA of 0.

To validate this technique under different cases, we used used synthetically generated feature sets with the desired predominant overlap behaviors. We then applied the Random 8-Set Test to these feature sets and averaged the results over

8 subset randomizations. The attack model used was XGBoost. These feature set cases and their corresponding experimental results are shown below.

1. 320-features set with a single overlapping group of varying size (remainder of features are independent of the target).
2. 320-features set with four non-overlapping groups of varying size that are internally overlapping within each group.
3. 320-features set with varying number of single non-overlapping features.
4. 320-features set with four non-overlapping groups of anti-overlapping pairs.
5. 320-features set a total of five non-overlapping items: four internally overlapping groups, and a fifth group of non-overlapping features.

# overlapping feats	# indep. feats	monotonic?	NCA
160	160	yes	0.8749
80	240	yes	0.8715
40	280	yes	0.8526
20	300	yes	0.8138
10	310	yes	0.6494
5	315	yes	0.5660

Table 5.7: Single overlapping group Random 8-Set Test results

The results for Case 1 can be seen in Table 5.7. Without exception, all of the plotted Random 8-Set Test curves are monotonic and concave, as expected. Moreover, as we decrease the number of overlapping features, the NCA also decreases. Note that while not shown in the table, the degenerate case where only one important feature exists in the entire feature set will have an NCA of 0.5 when averaged over many subset randomizations. This is because, when there is only one important feature, the Random N-Set Test curve will be a step function that

# overlapping feats per grp	# indep. feats	monotonic?	NCA
40	160	yes	0.8429
20	240	yes	0.7634
10	280	yes	0.6400
5	300	yes	0.5174
2-2-2-4 split	310	yes	0.3774

Table 5.8: Four overlapping groups Random 8-Set Test results

jumps from the highest possible leakage to the lowest, and where this step occurs is uniformly distributed across the subsets.

The results for Case 2 can be seen in Table 5.8. Again, without exception the plotted curves are monotonic, but they are not always concave. When the number of overlapping features is large, the behavior of the Random N-Set Test curve is similar to the previous case. However, when the number of important features is small, we approach a similar degenerate case as before. In the final line in Table 5.8 where the overlapping groups are very small, individual runs of the test result in front-weighted step functions (more steps tend to occur earlier), thus resulting in a convex curve when averaged. In the next case, we will see this effect with greater clarity, because the last line of this table most closely resembles the case when there are only non-overlapping features.

# non-overlapping feats	# indep. feats	monotonic?	NCA
32	288	no, flat & jumpy	0.5053
27	293	mostly, but jumpy	0.2897
22	298	mostly, but jumpy	0.5609
16	304	mostly, but jumpy	0.3066
8	312	yes, step-like function	0.1402

Table 5.9: Only non-overlapping features Random 8-Set Test results

The results for Case 3 can be seen in Table 5.9. At first glance these results may seem somewhat unusual, given that one may have expected linear curves. However, these results are a manifestation of the nature of the case. Recall that when the

alphabet size of the target is restricted, increasing the number of non-overlapping features necessarily requires reducing the amount of information contained in each individual feature. Indeed, this is what we observe in the first two lines of Table 5.9, where the highest leakage is lower than the other lines'. While not shown, experiments with slightly more than 32 non-overlapping features start failing altogether, where XGBoost is incapable of learning anything from the feature set due information dilution. For the last 3 lines, we start seeing individual curves exhibit random step-like behavior as before.

# non-overlapping feats per grp	# indep. feats	monotonic?
2	312	no, flat

Table 5.10: Four anti-overlapping groups Random 8-Set Test results

The results for Case 4 can be seen in Table 5.10. Here, for the anti-overlapping case, the attack simply fails altogether, where XGBoost is incapable of learning anything about the target. While this is partially a property of the tree-boosting model used by XGBoost, it is also symptomatic of other common machine learning models, such as neural networks or nearest neighbors. In short, it requires a specialized learning model to detect anti-overlap groups since each individual feature within an anti-overlap group looks irrelevant (or at least, low importance). Thus, traditional learning models that rely on individual feature importance to get started will struggle to learn from a feature set that only contains anti-overlap groups. While this does mean that this portion of our study is theoretically incomplete, we consider this case to be a degenerate case in the context of ML-based attacks. In other words, there is no need for the Random N-Set Test when the attack itself fails.

Finally, the results for Case 5 can be seen in Table 5.11. In this case, the important features are equally split between overlapping features and non-overlapping

# OLing feats per grp	# non-OLing features	# indep. feats	monotonic?	NCA
20	80	160	yes	0.5578
10	40	240	yes	0.5332
5	20	280	yes	0.4572
2	8	304	yes	0.3731

Table 5.11: Four overlapping groups and non-overlapping features mixture Random 8-Set Test results

features to study the race condition that exists between cases 1/2 and 3. Here, we simply remark that the properties of non-overlapping features can dominate over overlapping features when they coexist.

To summarize, the Random N-Set Test can help detect whether the predominant behavior within the feature set is overlapping features. If the majority of important features are overlapping with each other to some degree (whether a single large overlapping group exists or it is broken into a few separate overlapping groups), then a monotonic Random N-Set Test curve and a high NCA will uniquely result – no other predominant overlap behavior will cause the same behavior.

On the other hand, an NCA of 0.5 or less indicates that either there is a significant number of non-overlapping features or the amount of overlapping features is low. While the Random N-Set Test cannot easily distinguish these two cases, other techniques presented in this section can be used to supplement the analysis.

Detecting Conditional Independence

Once we know that a feature set contains predominantly overlapping features, we may be interested in whether these overlapping features are mostly characterized as independent looks or duplicate features. To do this, we simply use the plug-in method to estimate $I(F; F'|T)$. Unlike computing overlap, estimating this con-

ditional mutual information does not suffer the same kind of joint alphabet size issues, since it is computed as a series of two-variable mutual informations (each conditioned on a fixed target value), and then averaged over all target values. In other words:

$$I(F; F'|T) = \sum_{t \in \mathcal{T}} P(T = t) * I(F; F')|_{T=t} \tag{5.3}$$

While it is possible for the 2-variable joint alphabet size to still be too large, this estimation requires orders of magnitude fewer data samples than the estimation of overlap.

5.5.9 Probing Overlap Behavior of Pruned CNN

At this point, we can apply our diagnostics to the CNN pruning problem (ResNet18, Cifar100) in order to get a better idea of the nature of the problem.

First, we perform a single-feature importance sweep of all of the 4800 features by measuring the single-feature mutual information for the victim classification attack. The largest-alphabet features in ResNet18 are 1025, and the alphabet size of Cifar100 is 100. Thus using 10,000 test images is sufficient for a reasonable approximation of two-variable mutual informations here. The resulting dispersion distribution can be seen in Figure 5.11. We note a few interesting points about this distribution. The single-feature importances are somewhat correlated to the alphabet sizes of the features (recall that earlier layers have larger alphabet channels), but there are no layers that are entirely irrelevant. Whereas the earlier layers' features tend to have individually more important features, the later layers have more relevant features overall. Moreover, there is a noticeable increase in overall importance in the last layer (features 4288-4800) compared to other layers

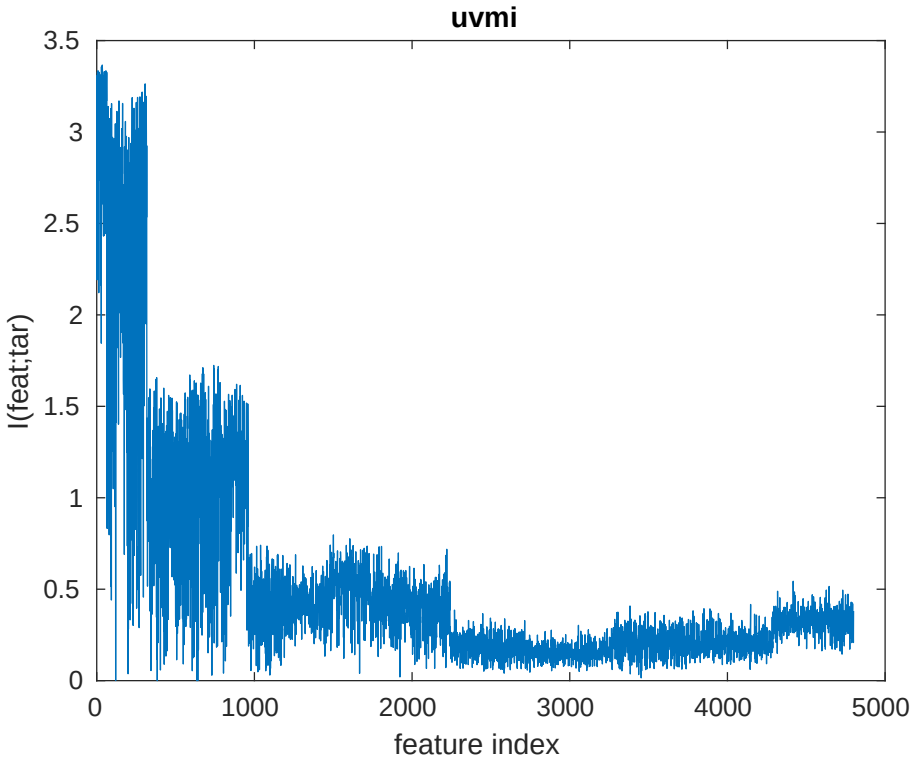


Figure 5.11: ResNet18 single-feature importance sweep

with the same feature alphabet sizes. Overall, this suggests to us that, due to the sheer number of features, one contributing factor to the nature of the protection problem we observed from the attack study is high dispersion count and relatively uniform-across-layers dispersion distribution in terms of order of magnitude.

We perform the Random 8-Set and Random 32-Set Tests and plot two different views of the same experiments in Figures 5.12 and 5.13. Figure 5.12 is a normal plot of the trade-off curve with an NCA of 0.8218 for 8 sets and 0.8833 for 32 sets. Since this curve is monotonic with a high degree of concavity, we conclude that the predominant behavior in the ResNet18 feature set is a large number of overlapping features.

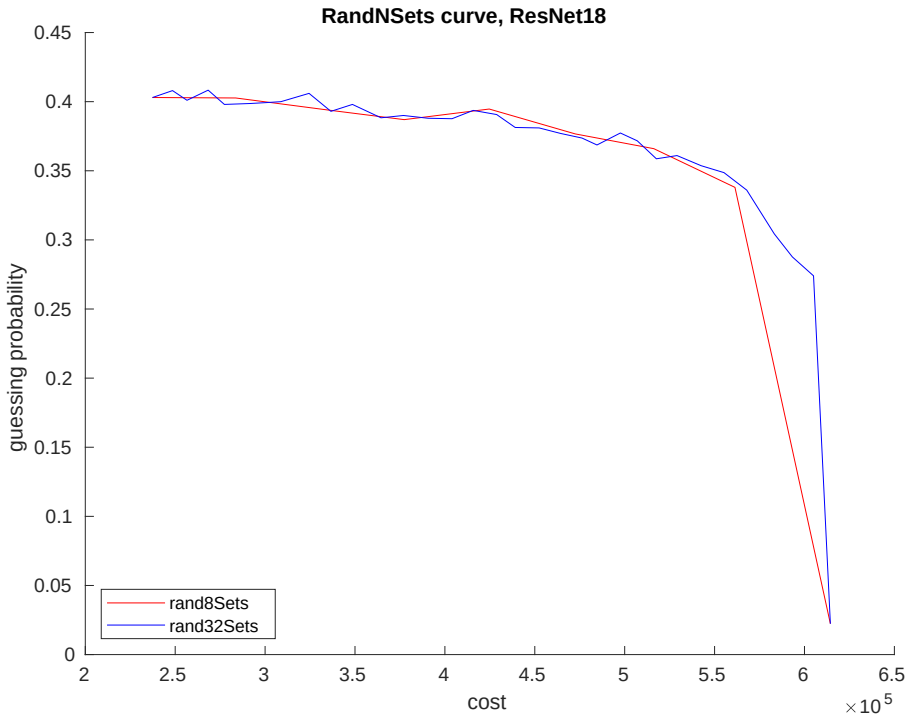


Figure 5.12: ResNet18 Random 8/32 Sets Test. NCA = 0.8218/0.8833

Figure 5.13 is a plot of the same two experiments, but with both the horizontal and vertical axes linearly flipped, and then plotted semilog in the vertical axis. This is a useful technique to observe how quickly the guessing probability (i.e. leakage) approaches the upper limit when removing protection; if the resulting semilog plot is linear, that indicates that the rate of approach is exponential. Recall that if this is the case, that suggests a high degree of independent looks overlap as opposed to In this case, we find that the semilog plot is roughly linear, although there is some expected experimental test noise in the low cost domain for the 32 sets test. Thus, these results suggest that for ResNet18, the primary form of overlap is potentially independent looks, but we will reserve judgement for the moment and use the next test to draw a conclusion.

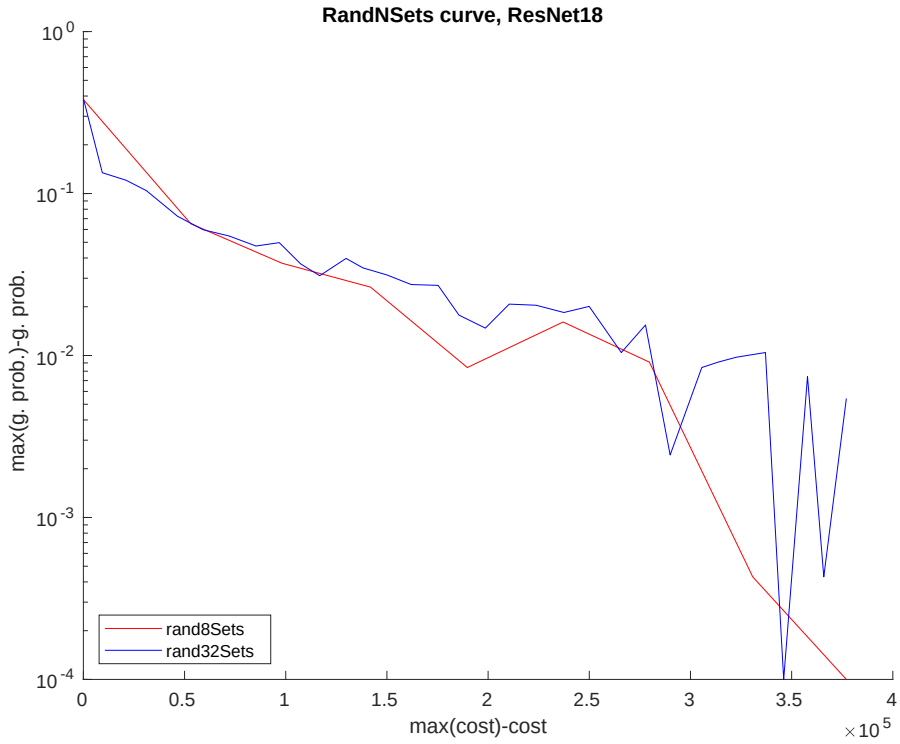


Figure 5.13: ResNet18 Random 8/32 Sets Test Semilog Plot

Finally, we perform multiple conditional mutual information experiments. Note that, since there are 4800 features, computing every single pairwise conditional mutual information is somewhat impractical. Instead, we present four experiments to probe the conditional mutual informations within the features. Recall that conditional mutual information close to zero indicates pairwise conditional independence between the features, while high conditional mutual information indicates duplicative features when both features are important and overlapping.

Figure 5.14 computes 2000 random pairs of features selected from the first layer alone. Here, the distribution is right skewed and the minimum conditional mutual information is over 7 bits. This suggests that within the first layer alone, there is a significant degree of feature duplication.

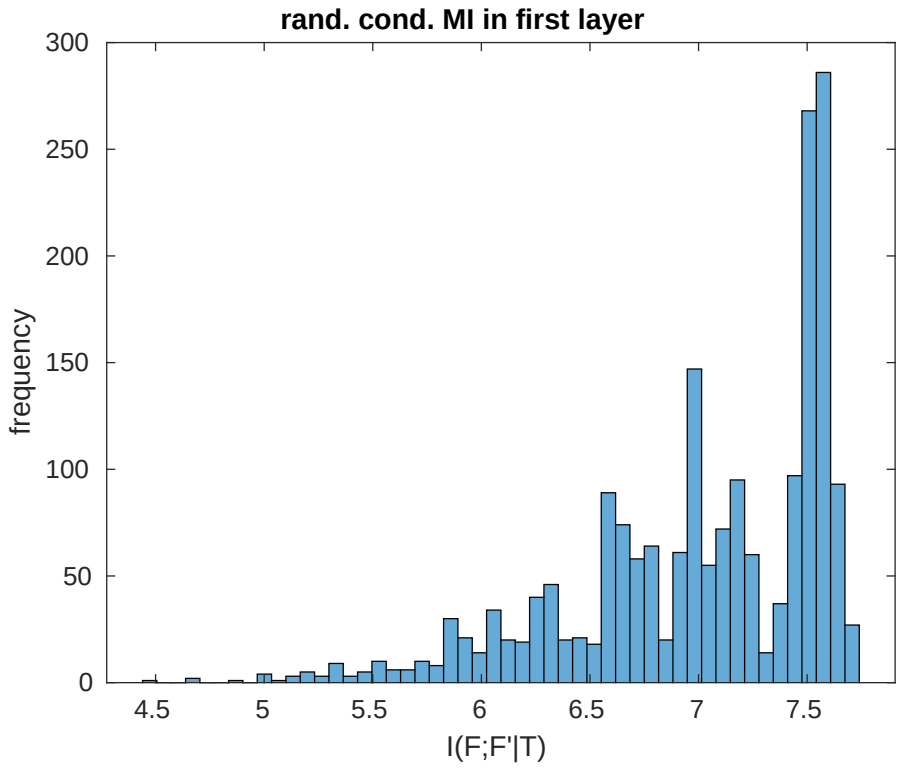


Figure 5.14: First Layer Cond. MI

Figure 5.15 computes 10,000 random pairs of features selected from the last layer alone. Here we find the distribution to be centered around 0.8 bits, which is fairly low, suggesting that within the last layer alone, the dominant overlapping behavior may be closer to independent looks. Between Figures 5.14 and 5.15, it seems that the CNN naturally distills information from the input in order to make its classifications. This means that earlier layers will tend to be internally more duplicative, while later layers will tend to be internally independent.

Figure 5.16 computes 10,000 random pairs of features selected from distant layers. Specifically, each pair consists of one feature from the first 5 layers and one feature from the last 5 layers. Here we find that the dominant overlapping

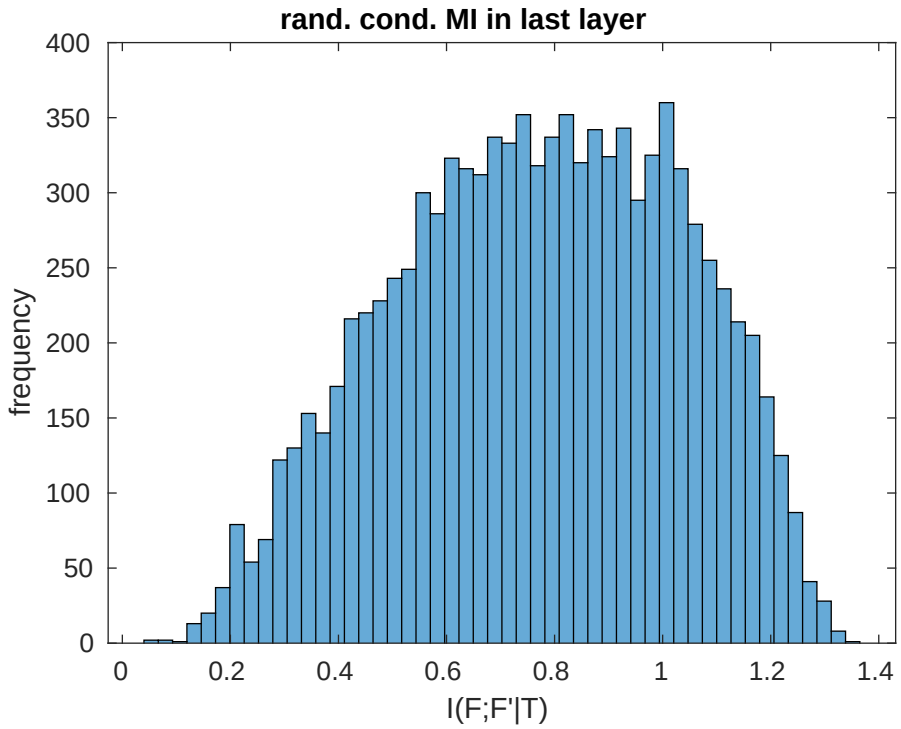


Figure 5.15: Last Layer Cond. MI

behavior is likely to be duplicate features.

Finally, Figure 5.17 computes 10,000 random pairs of features selected from different layers (in each pair, each feature is from a different layer). There is a significant amount of feature duplication.

Overall, it seems that there is a mix of both independent looks and feature duplication within ResNet18. These observations likely contribute to the robustness of the attack that we observed in the previous attack study.

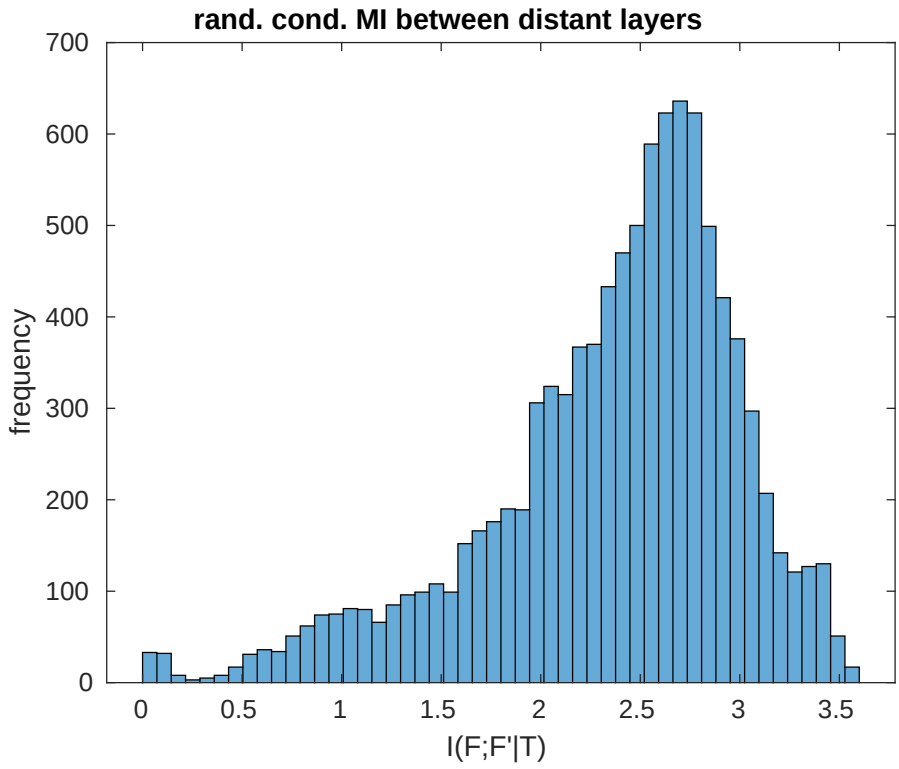


Figure 5.16: Distant Layers (first 5 layers vs last 5 layers) Cond. MI

5.6 Protection Mechanisms

Now that we have a better idea of the protection problem in the dynamically-pruned CNN side channel, we investigate protection in this section.

5.6.1 Algorithmic Description of Protection Schemes and Cost

As we have done in previous chapters, we will represent all protection schemes as transition matrices mapping numbers of nonzeros (NNZs) to larger, obfuscated val-

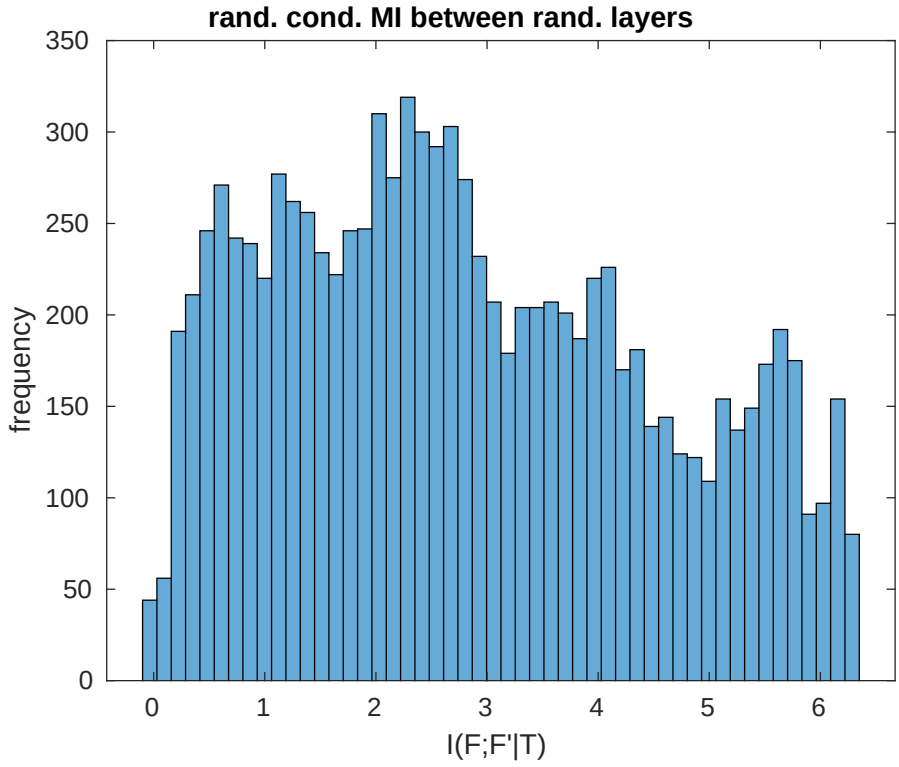


Figure 5.17: Random Different Layers Cond. MI

ues. In order to avoid unduly impacting the accuracy of the victim CNN, we do not reduce the NNZs. Unlike the previous chapters however, here we treat each entry in the pruning vector as its own side channel, so a complete protection scheme is actually represented as a 3D matrix where traversing the depth dimension equates to traversing the protection scheme by channel. This algorithmic modeling of the full protection scheme is then general enough to capture all possible protection schemes in this context. For the sake of ease of implementation (and on the basis of our work on optimality under maximal leakage), we will restrict our attention to deterministic schemes only.

In addition, for the bulk of this section we use *algorithmic cost*, which is simply

the difference between the mapped value and the original NNZ. Naturally, since we do not map NNZs to values smaller than the original, the cost matrix for each channel is upper triangular. This algorithmic cost is a useful analytical metric because it is directly related to the total inference runtime by the way the NNZs impact both the computation time and the memory access time of each layer. We refer to this cost as "algorithmic" because, while it is directly linked to true runtime costs, actual implementations of CNNs are slightly more complex so the algorithmic cost is not necessarily one-to-one with true execution time.

5.6.2 General-Purpose Protection Schemes

We present two forms of general-purpose protection schemes that are not specific to any one attack target.

Uniform Binning

The Uniform Binning scheme generation algorithm, applied independently to each feature is as follows:

1: Inputs: R (target alph. ratio, common across features)
2: Set $N =$ unprotected feature alph. size
3: Set $M = round(R * N)$
4: Choose M NNZ values spaced evenly over range $[1, N]$, rounding as necessary
5: Map NNZs to the first larger value from among this set. Call this mapping $tempScheme$
6: Return $tempScheme$

While it is not exactly the same due to rounding, Uniform Binning is fairly similar to the granularity reduction schemes discussed in the attack study. A trade-off can be established by tuning the input R, with values close to 1 corresponding to less protection and values close to 0 corresponding to more protection. The obvious weakness of Uniform Binning is that it does not take into account any characteristics of the protection problem into account. However, in exchange, not only is the Uniform Binning algorithm easy to perform with minimal computation, but it is also typically easier to implement, which we will discuss in a later subsection.

Equal Lambda

Inspired by our work in the previous chapters, we also propose an Equal Lambda scheme, which simultaneously optimizes all features in the pruning vector using the supporting hyperplane technique, while assuming that they are independent looks. The algorithm, applied independently to each feature is as follows:

1: Inputs: λ_{tar} (target slope of trade-off, common across features)

2: Set $N =$ unprotected feature alph. size. Define no protection as $tempScheme$.

3: Set $M = N - 1$, $maxL1 = log(N)$, $cost1 =$ cost of $tempScheme$

4: Find a subset of the alphabet, of size M symbols, that minimizes the total cost when that removed symbols are mapped to the next largest available symbol. Re-define this as $tempScheme$.

5: Set $maxL2 = log(M)$, $cost2 =$ cost of $tempScheme$

6: Set $\lambda = -\frac{cost2-cost1}{maxL2-maxL1}$

7: If $\lambda < \lambda_{tar}$, go to step 3. Else, continue

8: Return *tempScheme*

The strength of the Equal Lambda scheme is that it is provably optimal under the stated assumptions (maximal leakage as the metric, and independent channels), and a trade-off can be established by tuning the λ_{tar} parameter. Small λ_tar closer to 0 correspond to less protection and large λ_{tar} correspond to more. Note that there is no self-evident maximum λ_{tar}, but a practical maximum that achieves full protection can be discovered experimentally. A weakness of the Equal Lambda scheme is that it treats the features as if they are independent of one another (even though we know experimentally that this is not the case). However, in exchange Equal Lambda is a relatively low-computation general-purpose scheme that takes individual feature distributions into account.

5.6.3 Target-Specific Protection Schemes

Aside from general-purpose protection, we may also be interested in protection schemes specifically designed for one attack target.

All-Or-Nothing Schemes

Using the notion of feature importance, an All-Or-Nothing protection scheme simply operates by selecting a subset of features to fully protect. The intuition here is that, when some features are more important than others, it may make sense to focus protection on the most important features. Thus a trade-off can be established by changing the number of features to protect. The ordering of features to prioritize can be determined either standard measures of feature importance,

such as univariate mutual information (agnostic to attack model) or feature gain importance (specific to attack model), which can be obtained through standard machine learning toolboxes such as XGBoost or Sci-kit.

Of all the protection schemes presented here, All-Or-Nothing schemes are computationally the easiest to generate as well as the easiest to implement. However, its primary weakness is that it implicitly assumes that the dispersion distribution for the given attack target is non-uniform. In other words, All-Or-Nothing schemes only create effective trade-off curves when a significant number of features are low importance to the target, and the larger the set of important features, the less effective these schemes will be overall.

Weighted Binning Schemes

Here, we present an improvement over Uniform Binning and All-Or-Nothing schemes by combining the two. In essence, the Weighted Binning scheme uses a feature importance vector (assumed to be non-negative and increasing with greater importance) to generate alphabet ratios, which are then used to apply Uniform Binning with different ratios to each feature. To generate these ratios, we solve the following optimization problem:

$$\max_{a,b} E[a * (-log(imp)) + b] \quad \text{s.t. } \alpha \leq a * (-log(imp)) + b \leq \beta \quad \forall imp_i \qquad (5.4)$$

where imp is the importance vector, α is the floor alphabet ratio and β is the ceiling. In particular, we set $\alpha = 0$ and $\beta = \max\{R_o, R_{thr}\}$ where the parameter R_o is a target average alphabet ratio and the parameter R_{thr} is a threshold to ensure a minimum amount of protection on all features. Note that we elect to use the negative log of the importance vector to flip the importance order (so that

alphabet ratios are decreasing with greater importance) and to dampen the effects of outlier importance values.

This optimization is solved analytically by the following scheme generation algorithm:

1: For any $imp_i = 0$, set $imp_i = 0.5 * min_{imp_i>0}(imp_i)$. This is a heuristic step to avoid divide-by-zeros
2: Set $S = -log(imp)$
3: Set $\alpha = 0$ and $\beta = \max\{R_o, R_{thr}\}$
4: Set $a = \min((\alpha - 1)/(\min(S) - mean(S)), (\beta/R_o - 1)/(\max(S) - mean(S)))$
5: Set $b = 1 - a * mean(S)$
6: Set $\lambda = -\frac{cost2-cost1}{maxL2-maxL1}$
7: Return $R = (a * S + b) * R_o$

Then, alphabet ratios R are applied using the same algorithm per feature as was used for Uniform Binning. In this way, Weighted Binning takes feature importance into account without the pitfalls of All-Or-Nothing schemes.

5.6.4 Algorithmic Evaluation

We measure the trade-off curves for all of the proposed protection schemes for three attack targets. The full list of protection schemes used is:

1. Uniform Binning (unif. bin)
2. Equal Lambda (eq. ld)
3. All-Or-Nothing using XGBoost's gain importance for unprotected attack (gain AON)

4. All-Or-Nothing using scikit-learn's univariate mutual information for unprotected attack (MI AON)
5. Weighted Binning using XGBoost's gain importance for unprotected attack (gain WBin)
6. Weighted Binning using scikit-learn's univariate mutual information for unprotected attack (MI WBin)
7. Random 8-Set Test curve for reference

The studied attack targets are:

1. Victim classification labels (Figure 5.18)
2. Ground truth classification labels (Figure 5.19)
3. Dominant color of input image, an alph. size 3 target indicating which color channel has the greatest L2 norm (Figure 5.20)

Since victim class and ground truth are similar attack targets, we will address them together. Using the Random 8-Set Test curve as a worst-protection reference curve, we find that all of the studied protection schemes achieve obviously lower NCAs, which means that at the very least all of our protection schemes have partially overcome the extensive overlapping behavior of the feature set.

However, as anticipated, the All-Or-Nothing schemes perform relatively poorly compared to the other schemes; again this is likely because, as we have seen, a vast majority of the features are relevant to the attack target in these cases. As a result, All-Or-Nothing schemes operate on faulty implicit assumptions about non-uniform dispersion and still result in highly concave trade-offs.

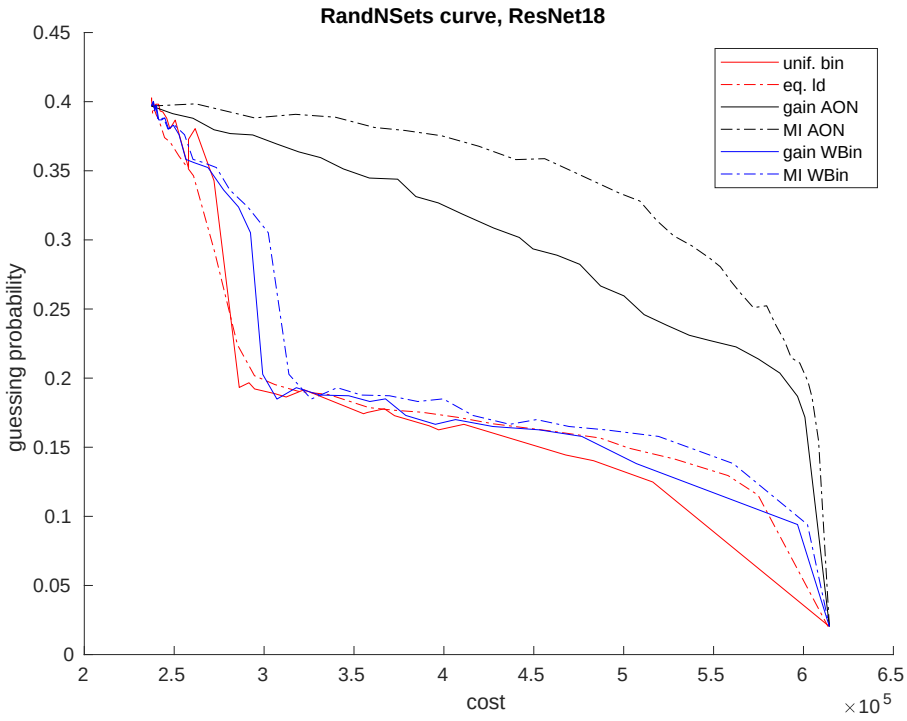


Figure 5.18: Victim classification protection scheme comparison

On the other hand, somewhat surprisingly, Uniform Binning, Equal Lambda, and both Weighted Binning schemes perform relatively similarly to each other. In fact, Uniform Binning actually slightly outperforms all other schemes in the high cost domain, while Equal Lambda performs best in the low cost domain. This is surprising because both Uniform Binning and Equal Lambda are general-purpose schemes that are agnostic to the attack target, yet the attack-specific Weighted Binning schemes cannot perform significantly better.

The reason for this surprising result is somewhat nuanced, but can be explained on the basis of our feature analyses from the previous two sections. Recall that when we analyzed the pairwise conditional mutual information between features, feature pairs selected from the last layer (as we have seen, the most important layer

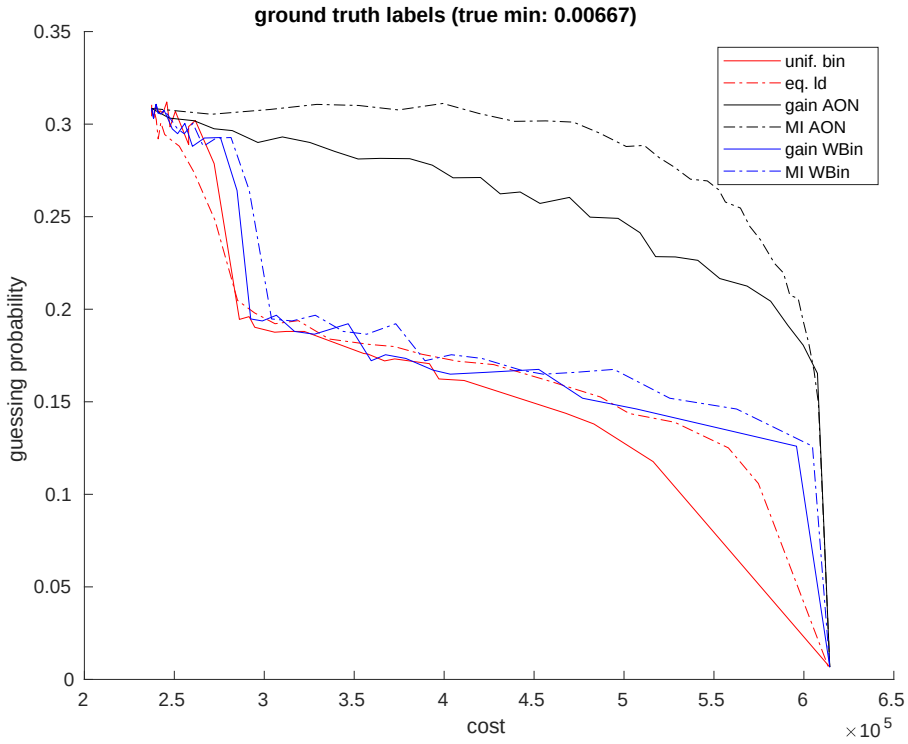


Figure 5.19: Ground truth protection scheme comparison

for output attacks) tend to be remarkably close to independent looks. This means that simultaneously scaling protection on all features from the last layer will result in exponential drop-off of leakage from the last layer alone. In addition, the last 5 layers of ResNet18 also have the smallest feature alphabets, making them cheaper to protect overall. On the other hand, we found that feature pairs across different layers tend to be duplicative rather than independent looks, which explains why, in our attack study, it was not possible to thwart the attack by only focusing on the later layers – because all of the layers are to an extent duplicates of each other.

Put together, the reason that Uniform Binning, Equal Lambda, and Weighted Binning schemes perform similarly is that there is a fundamental limit to the effectiveness of all schemes that simultaneously scale all features in the feature set

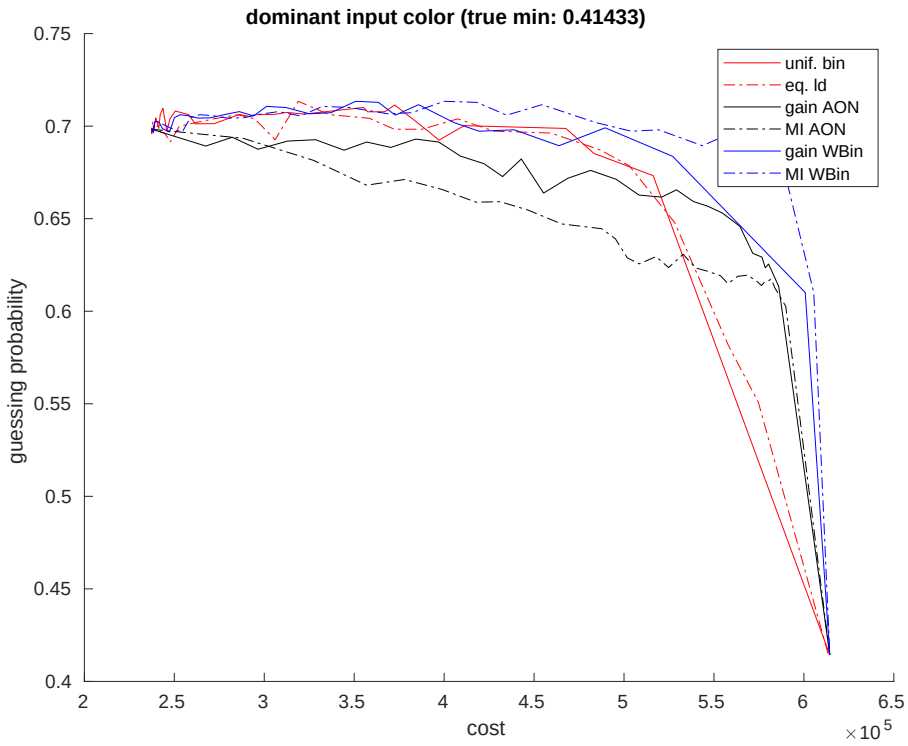


Figure 5.20: Dominant color protection scheme comparison

when the feature set has the following properties:

- High dispersion count and close-to-uniform dispersion distribution
- High degree of overlap of both types (independent looks and duplicative features) between features

5.6.5 Comparison Against Random Noise Schemes

We have primarily focused on deterministic protection schemes so far in this study, but here we briefly consider randomized noise as a potential alternative. For our randomized noise scheme, we generate i.i.d. uniform random integers for each

channel and use these integers to pad the values of the unprotected pruning vector. A plot of this scheme's performance compared to our best deterministic scheme (uniform binning) is shown in Figure 5.21. Surprisingly, the performance of the uniform random noise scheme is comparable to uniform binning low cost ranges, but a fatal flaw of independent random noise schemes at large is that it is very costly to reach the minimum guessing probability due to the large number of important features.

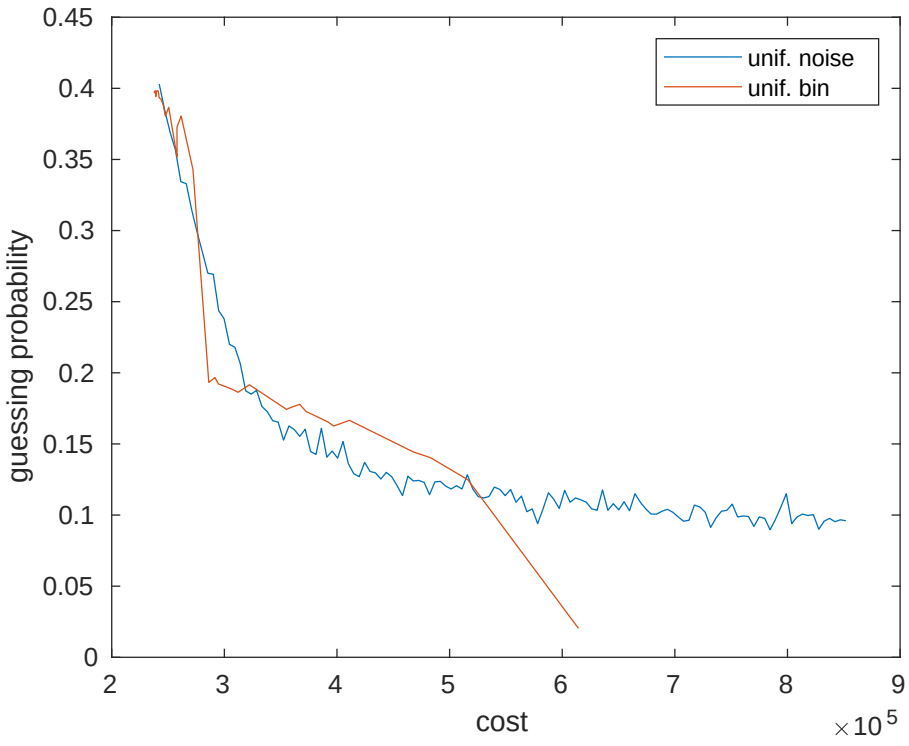


Figure 5.21: Victim class comparison between uniform random noise and uniform binning

5.6.6 Attack Model Reusability Under Different Protection

For the bulk of this study, our primary threat model assumes that the attacker has black-box access to both the trained victim CNN as well as any protection schemes that may be implemented on the pruning vector. This threat model where the attacker can abstract away any forms of protection by training on protected pruning vectors is generally the more difficult protection problem, but here we also briefly consider one in which the attacker has access to the unprotected trained CNN but not the protection scheme.

In that case, the attack proceeds by training on the unprotected victim only. The protection trade-off is then established by testing the attack model trained on the unprotected victim against our protection schemes. These experimental results for the victim class attack can be seen in Figure 5.22

Somewhat predictably, trained XGBoost models cannot be easily reused against scaled protection schemes. With the exception of All-Or-Nothing using mutual information, all other protection schemes are highly effective if the attacker is not allowed to train against the protection scheme.

5.6.7 Implementing Algorithmic Protection Schemes

Here, we discuss the practical concerns of implementing protection schemes in practical settings, such as on a CNN accelerator with off-chip memory. Recall that a full protection scheme is a 3D matrix describing the mapping of true NNZs to obfuscated values for every single channel in the victim CNN. For ResNet18,

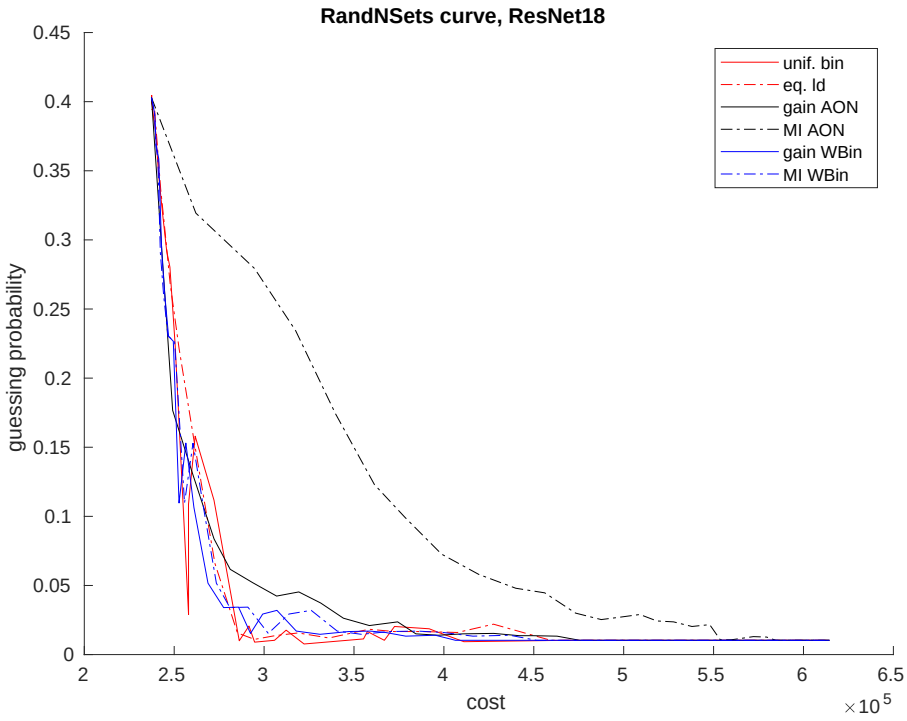


Figure 5.22: Attack model reuse experiment

this would be a $4800 \times 1025 \times 1025$ transition matrix of floats in the general case, which would be a large storage burden on the system. However, there are some implementation tricks we can employ to reduce this burden significantly.

First, since we have restricted our attention to deterministic schemes only, there is no need to work with float values. Instead, we can work with threshold values, which condenses our algorithmic scheme to a 4800×1025 matrix of mappings (e.g. row is a lookup table where the Nth column entry indicates the mapped NNZ value). This greatly reduces the size of the protection scheme data structure, but we can make further optimizations to shrink the stored data structure.

First, the reason there are 1025 columns is that the largest feature alphabet size is 1025. However, features with alphabets of this size actually form a minority

of the feature set. We could condense the 4800×1025 matrix into smaller lookup tables:

- 320 lookup tables of size 1025
- 640 lookup tables of size 257
- 1280 lookup tables of size 65
- 2560 lookup tables of size 17

Then, there is also no reason to store entire lookup tables, when we can simply store thresholds and incorporate logic to search for the next highest threshold. The number of thresholds depends on how the protection in the channel is scaled; for example Uniform Binning uses an alphabet ratio while Equal Lambda uses lambda values. This results in our protection scheme being represented as an ordered collection of threshold lists.

If necessary, this collection of threshold lists can be even further compressed as integers. To do so, envision the threshold list as a binary array of length feature-alphabet-size, and simply store the that bit array as a collection of integers (e.g. for feature alphabet size 1025, a single lookup table could be compressed into 33 32-bit integers.) However, if compressed this way, it is also necessary to unpack lookup tables during convolutional layers – depending on on-chip memory constraints, this compression technique may or may not be worth the unpacking time and execution overhead.

Finally, some of our proposed protection schemes can be implemented without storing their transition matrices at all. Uniform Binning is simple enough by construction that the protection mapping can be performed on the fly with some

simple arithmetic (.e.g. compute bin size and compute $ceil(NNZ/(\text{Bin Size}))$). All-Or-Nothing schemes are also relatively simple to implement, as one only needs to maintain a pre-determined list of features to protect fully. Again, this can be represented as a 4800-bit array.

5.7 Summary

To conclude this chapter, the major takeaways are as follows. First, we have shown that a supervised ML-based attack on dynamically pruned CNNs during the inference phase is not only possible but also robust against various forms of protection and granularity reduction. In order to better understand this attack robustness, we have studied information dispersion and overlap as properties of the feature set and found that in the CNN setting, the pruning vector has high dispersion count with relatively uniform distribution, while many of the features tend to be overlapping. On the protection side, this means that Uniform Binning is typically a competitive general purpose scheme, because the nature of the CNN pruning vector prevents schemes that are more feature importance aware from significantly outperforming it.

BIBLIOGRAPHY

[1] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

[2] Gautam Aishwarya and Mokshay Madiman. Remarks on Rényi versions of conditional entropy and mutual information. In *2019 IEEE International Symposium on Information Theory (ISIT)*, pages 1117–1121, 2019.

[3] E. Arikan. An inequality on guessing and its application to sequential decoding. *IEEE Transactions on Information Theory*, 42(1):99–105, 1996.

[4] S. Arimoto. Information measures and capacity of order α for discrete memoryless channels. *Topics in Information Theory Proc. Coll. Math Soc. Janos Bolyai*, pages 41–52, 1975.

[5] Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, August 2016. arXiv: 1603.02754.

[6] Z. Chen, T. Xu, C. Du, C. Liu, and H. He. Dynamical Channel Pruning by Conditional Accuracy Change for Deep Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2020. Conference Name: IEEE Transactions on Neural Networks and Learning Systems.

[7] Thomas M Cover and Joy A Thomas. *Elements of Information Theory*. 2nd edition, 2006.

[8] I. Csiszár. Generalized cutoff rates and Rényi's information measures. *IEEE Transactions on Information Theory*, 41(1):26–34, 1995.

[9] Shaveta Dargan, Munish Kumar, Maruthi Rohit Ayyagari, and Gulshan Kumar. A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning. *Archives of Computational Methods in Engineering*, 27(4):1071–1092, September 2020.

[10] Xuanyi Dong, Junshi Huang, Yi Yang, and Shuicheng Yan. More is Less: A More Complicated Network with Less Inference Complexity. *arXiv:1703.08651 [cs]*, May 2017. arXiv: 1703.08651.

[11] Goran Doychev, Dominik Feld, Jonas Eckhardt, and Stephan Neumann. Yes We Can: Uncovering Spoken Phrases in Encrypted VoIP Conversations. page 30, May 2009.

[12] John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. Local Privacy, Data Processing Inequalities, and Statistical Minimax Rates. *arXiv:1302.3203 [cs, math, stat]*, February 2013. arXiv: 1302.3203.

[13] Cynthia Dwork and Guy N. Rothblum. Concentrated differential privacy.

[14] Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. Peek-a-Boo, I Still See You: Why Efficient Traffic Analysis Countermeasures Fail. In *2012 IEEE Symposium on Security and Privacy*, pages 332–346, May 2012. ISSN: 1081-6011.

[15] Serge Fehr and Stefan Berens. On the conditional Rényi entropy. *IEEE Transactions on Information Theory*, 60(11):6801–6810, 2014.

[16] Xitong Gao, Yiren Zhao, ukasz Dudziak, Robert Mullins, and Cheng-zhong Xu. Dynamic Channel Pruning: Feature Boosting and Suppression. *arXiv:1810.05331 [cs]*, January 2019. arXiv: 1810.05331.

[17] Xun Gong and Negar Kiyavash. Quantifying the Information Leakage in Timing Side Channels in Deterministic Work-Conserving Schedulers. *IEEE/ACM Transactions on Networking*, 24(3):1841–1852, June 2016.

[18] Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic Network Surgery for Efficient DNNs. *arXiv:1608.04493 [cs]*, November 2016. arXiv: 1608.04493.

[19] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. AMC: AutoML for Model Compression and Acceleration on Mobile Devices. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision ECCV 2018*, volume 11211, pages 815–832. Springer International Publishing, Cham, 2018. Series Title: Lecture Notes in Computer Science.

[20] Weizhe Hua, Zhiru Zhang, and G. Edward Suh. Reverse Engineering Convolutional Neural Networks Through Side-channel Information Leaks. In *2018*

55th ACM/ESDA/IEEE Design Automation Conference (DAC), pages 1–6, San Francisco, CA, June 2018. IEEE.

[21] Ibrahim Issa, Sudeep Kamath, and Aaron B. Wagner. Maximal leakage minimization for the Shannon cipher system. In *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, pages 520–524, 2016.

[22] Ibrahim Issa, Sudeep Kamath, and Aaron B. Wagner. An operational measure of information leakage. In *Proc. Conf. Inf. Sci. and Sys. (CISS)*, pages 234–239, 2016.

[23] Ibrahim Issa and Aaron B. Wagner. Operational definitions for some common information leakage metrics. In *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, pages 769–773, 2017.

[24] Ibrahim Issa, Aaron B. Wagner, and Sudeep Kamath. An Operational Approach to Information Leakage. *arXiv:1807.07878 [cs, math]*, July 2018. arXiv: 1807.07878.

[25] Ibrahim Issa, Aaron B. Wagner, and Sudeep Kamath. An operational measure of information leakage. *IEEE Trans. Inf. Theory*, to appear.

[26] Jiantao Jiao, Kartik Venkat, Yanjun Han, and Tsachy Weissman. Minimax Estimation of Functionals of Discrete Distributions. *IEEE Transactions on Information Theory*, 61(5):2835–2885, May 2015. Conference Name: IEEE Transactions on Information Theory.

[27] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The composition theorem for differential privacy. *IEEE Trans. Inf. Theory*, 63(6):4037–4049, June 2017.

[28] Paul Kocher, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre Attacks: Exploiting Speculative Execution. *arXiv:1801.01203 [cs]*, January 2018. arXiv: 1801.01203.

[29] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '96, pages 104–113, London, UK, UK, 1996. Springer-Verlag.

[30] Amos Lapidoth and Christoph Pfister. Two measures of dependence. In

2016 IEEE International Conference on the Science of Electrical Engineering (ICSEE), pages 1–5, 2016.

[31] Amos Lapidoth and Christoph Pfister. Testing against independence and a Rényi information measure. In *2018 IEEE Information Theory Workshop (ITW)*, pages 1–5, 2018.

[32] J. Liao, O. Kosut, L. Sankar, and F. P. Calmon. A Tunable Measure for Information Leakage. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 701–705, June 2018.

[33] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. Meltdown. *arXiv:1801.01207 [cs]*, January 2018. arXiv: 1801.01207.

[34] Christos Louizos, Karen Ullrich, and Max Welling. Bayesian Compression for Deep Learning. *arXiv:1705.08665 [cs, stat]*, November 2017. arXiv: 1705.08665.

[35] Jian-Hao Luo and Jianxin Wu. AutoPruner: An end-to-end trainable filter pruning method for efficient deep model inference. *Pattern Recognition*, 107:107461, November 2020.

[36] Ilya Mironov. Rényi differential privacy. In *Proc. IEEE Comp. Sec. Found. Symp.*, pages 263–275, 2017.

[37] Shayan Moini, Shanquan Tian, Jakub Szefer, Daniel Holcomb, and Russell Tessier. Remote Power Side-Channel Attacks on CNN Accelerators in FPGAs. *arXiv:2011.07603 [cs]*, November 2020. arXiv: 2011.07603.

[38] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical Programming*, 14(1):265–294, Dec 1978.

[39] Angshuman Parashar, Minsoo Rhu, Anurag Mukkara, Antonio Puglielli, Rangharajan Venkatesan, Brucek Khailany, Joel Emer, Stephen W. Keckler, and William J. Dally. SCNN: An Accelerator for Compressed-sparse Convolutional Neural Networks. *arXiv:1708.04485 [cs]*, May 2017. arXiv: 1708.04485.

[40] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca

Antiga, Alban Desmaison, Andreas Kpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *arXiv:1912.01703 [cs, stat]*, December 2019. arXiv: 1912.01703.

[41] Yury Polyanskiy and Sergio Verdú. Arimoto channel coding converse and Rényi divergence. In *Proc. Ann. Allerton Conf. on Comm., Control, and Computing*, pages 1327–1333, 2010.

[42] R. Tyrrell Rockafellar. *Convex analysis*. Princeton Mathematical Series. Princeton University Press, Princeton, N. J., 1970.

[43] I. Sason and S. Verd. ArimotoRnyi Conditional Entropy and Bayesian M -Ary Hypothesis Testing. *IEEE Transactions on Information Theory*, 64(1):4–25, January 2018. Conference Name: IEEE Transactions on Information Theory.

[44] A. Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer, 2003.

[45] Robin Sibson. Information radius. *Zeitschrift for Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 14(2):149–160, 1969.

[46] David M. Smith and Geoffrey Smith. Tight Bounds on Information Leakage from Repeated Independent Runs. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 318–327, August 2017. ISSN: 2374-8303.

[47] Dawn Xiaodong Song, David Wagner, and Xuqing Tian. Timing Analysis of Keystrokes and Timing Attacks on SSH. page 16.

[48] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S. Emer. Efficient Processing of Deep Neural Networks: A Tutorial and Survey. *Proceedings of the IEEE*, 105(12):2295–2329, December 2017.

[49] Tim van Erven and Peter Harremos. R\'enyi Divergence and Kullback-Leibler Divergence. *IEEE Transactions on Information Theory*, 60(7):3797–3820, July 2014. arXiv: 1206.2459.

[50] Sergio Verdu. alpha-mutual information. In *2015 Information Theory and*

Applications Workshop (ITA), pages 1–6, San Diego, CA, USA, February 2015. IEEE.

[51] C. Wampler, S. Uluagac, and R. Beyah. Information Leakage in Encrypted IP Video Traffic. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7, December 2015.

[52] Yulong Wang, Xiaolu Zhang, Xiaolin Hu, Bo Zhang, and Hang Su. Dynamic Network Pruning with Interpretable Layerwise Channel Selection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):6299–6306, April 2020.

[53] Lingxiao Wei, Bo Luo, Yu Li, Yannan Liu, and Qiang Xu. I Know What You See: Power Side-Channel Attack on Convolutional Neural Network Accelerators. In *Proceedings of the 34th Annual Computer Security Applications Conference on - ACSAC '18*, pages 393–406, San Juan, PR, USA, 2018. ACM Press.

[54] C. V. Wright, L. Ballard, S. E. Coull, F. Monrose, and G. M. Masson. Spot Me if You Can: Uncovering Spoken Phrases in Encrypted VoIP Conversations. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 35–49, May 2008.

[55] Charles V. Wright, Lucas Ballard, Scott E. Coull, Fabian Monrose, and Gerald M. Masson. Uncovering Spoken Phrases in Encrypted Voice over IP Conversations. *ACM Transactions on Information and System Security*, 13(4):1–30, December 2010.

[56] Benjamin Wu, Aaron B. Wagner, and G. Edward Suh. A Case for Maximal Leakage as a Side Channel Leakage Metric. *arXiv:2004.08035 [cs, math]*, April 2020. arXiv: 2004.08035.

[57] Benjamin Wu, Aaron B. Wagner, and G. Edward Suh. Optimal mechanisms under maximal leakage. In *2020 IEEE Conference on Communications and Network Security (CNS)*, pages 1–6, 2020.

[58] Benjamin Wu, Aaron B. Wagner, G. Edward Suh, and Ibrahim Issa. Strong Asymptotic Composition Theorems for Sibson Mutual Information. *arXiv:2005.06033 [cs, math]*, May 2020. arXiv: 2005.06033.

[59] Yihong Wu and Pengkun Yang. Minimax rates of entropy estimation on large alphabets via best polynomial approximation. *arXiv:1407.0381 [cs, math, stat]*, February 2016. arXiv: 1407.0381.

[60] Hao Zhou, Jose M. Alvarez, and Fatih Porikli. Less Is More: Towards Compact CNNs. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision ECCV 2016*, volume 9908, pages 662–677. Springer International Publishing, Cham, 2016. Series Title: Lecture Notes in Computer Science.

[61] Y. Zhou, S. Wagh, P. Mittal, and D. Wentzlaff. Camouflage: Memory Traffic Shaping to Mitigate Timing Attacks. In *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 337–348, February 2017.

[62] Y. Zhu, Y. Lu, and A. Vikram. On Privacy of Encrypted Speech Communications. *IEEE Transactions on Dependable and Secure Computing*, 9(4):470–481, July 2012.