On the Foundations of Quantitative Information Flow

Geoffrey Smith

School of Computing and Information Sciences,
Florida International University, Miami, FL 33199, USA
smithg@cis.fiu.edu

Abstract. There is growing interest in quantitative theories of information flow in a variety of contexts, such as secure information flow, anonymity protocols, and side-channel analysis. Such theories offer an attractive way to relax the standard noninterference properties, letting us tolerate "small" leaks that are necessary in practice. The emerging consensus is that quantitative information flow should be founded on the concepts of *Shannon entropy* and *mutual information*. But a useful theory of quantitative information flow must provide appropriate security guarantees: if the theory says that an attack leaks x bits of secret information, then x should be useful in calculating bounds on the resulting threat. In this paper, we focus on the threat that an attack will allow the secret to be guessed correctly in one try. With respect to this threat model, we argue that the consensus definitions actually fail to give good security guarantees—the problem is that a random variable can have arbitrarily large Shannon entropy even if it is highly vulnerable to being guessed. We then explore an alternative foundation based on a concept of *vulnerability* (closely related to *Bayes risk*) and which measures uncertainty using Rényi's *min-entropy*, rather than Shannon entropy.

1 Introduction

Protecting the *confidentiality* of sensitive information is one of the most fundamental security issues:

- In *secure information flow analysis* [1] the question is whether a program could leak information about its *high* (i.e. secret) inputs into its *low* (i.e. public) outputs.
- In *anonymity protocols* [2] the question is whether network traffic could reveal information to an eavesdropper about *who* is communicating.
- In *side-channel analysis* [3] the question is whether the running time or power consumption of cryptographic operations could reveal information about the secret keys.

A classic approach is to try to enforce *noninterference*, which says that low outputs are independent of high inputs; this implies that an adversary can deduce nothing about the high inputs from the low outputs.

Unfortunately, achieving noninterference is often not possible, because sometimes we *want* or *need* to reveal information that depends on the high inputs. In an election protocol, for example, the individual votes should be secret, but of course we want to reveal the tally of votes publicly. And in a password checker, we need to reject an incorrect password, but this reveals information about what the secret password is *not*. A

L. de Alfaro (Ed.): FOSSACS 2009, LNCS 5504, pp. 288–302, 2009.

variety of approaches for dealing with these sorts of deliberate violations of noninterference are currently being explored; see Sabelfeld and Sands [4] for a survey.

One promising approach to relaxing noninterference is to develop a *quantitative* theory of information flow that lets us talk about "how much" information is being leaked, and perhaps allowing us to tolerate "small" leaks. Such a quantitative theory has long been recognized as an important generalization of noninterference (see for example Denning [5, Chapter 5] and Gray [6]) and there has been much recent work in this area, including the works of Clark, Hunt, Malacaria, and Chen [7,8,9,10,11] Clarkson, Myers, and Schneider [12], Köpf and Basin [3], Chatzikokolakis, Palamidessi, and Panangaden [2,13], Lowe [14], and Di Pierro, Hankin, and Wiklicky [15].

In this paper, we consider the foundations of quantitative information flow. The basic scenario that we imagine is a program (or protocol) that receives some high input $\mathtt{H}$ and produces some low output $\mathtt{L}$. An adversary $\mathcal{A}$, seeing $\mathtt{L}$, might be able to deduce something about $\mathtt{H}$. We would like to *quantify* the amount of information in $\mathtt{H}$ ($\mathcal{A}$'s initial uncertainty), the amount of information leaked to $\mathtt{L}$, and the amount of unleaked information about $\mathtt{H}$ ($\mathcal{A}$'s remaining uncertainty). These quantities intuitively ought to satisfy the following slogan:

"initial uncertainty = information leaked + remaining uncertainty".

Of course, for a definition of quantitative information flow to be *useful*, it must be possible to show that the numbers produced by the theory are *meaningful* with respect to security—if the theory says that an attack leaks x bits of secret information, then x should be useful in calculating bounds on the resulting threat. A natural first step along these lines is to show that a leakage of 0 corresponds to the case of noninterference. But this is just a first step—we also must be able to show that differences among nonzero leakage values have significance in terms of security. This is the key issue which we explore in this paper.

We begin in Section 2 by establishing our conceptual framework—we consider a deterministic or probabilistic program c that receives a high input $\mathtt{H}$, assumed to satisfy a publicly-known *a priori* distribution, and produces a low output $\mathtt{L}$.

In Section 3, we review definitions of quantitative information flow found in the literature; we note an emerging consensus toward a set of definitions based on information-theoretic measures such as Shannon entropy and mutual information.

Section 4 then explores the consensus definitions with respect to the security guarantees that they support. After reviewing a number of security guarantees that have appeared in the literature, we focus our attention on one specific threat model: the probability that adversary $\mathcal{A}$ can guess the value of $\mathtt{H}$ correctly in one try. With respect to this threat model, we argue that the consensus definitions actually do a poor job of measuring the threat; briefly, the problem is that the Shannon entropy $H(\mathtt{X})$ of a random variable $\mathtt{X}$ can be arbitrarily high, even if the value of $\mathtt{X}$ is highly vulnerable to being guessed.

Because of these limitations of the consensus definitions, in Section 5 we propose an alternative foundation for quantitative information flow. Our definitions are based directly on a concept of *vulnerability*, which is closely related to *Bayes risk*. The vulnerability $V(\mathtt{X})$ is simply the maximum of the probabilities of the values of $\mathtt{X}$; it is the worst-case probability that an adversary could correctly guess the value of $\mathtt{X}$ in one try.

Using vulnerability, we propose to use *min-entropy*, defined by $H_\infty(\mathtt{X}) = -\log V(\mathtt{X})$, as a better measure for quantitative information flow. (Min-entropy is an instance of *Rényi entropy* [16].)

It should be acknowledged that our observations about the limitations of Shannon entropy as a measure of uncertainty are not entirely new, having appeared in recent research on anonymity protocols, notably in papers by Tóth, Hornák, and Vajda [17] and by Shmatikov and Wang [18]; they also propose the use of min-entropy as an alternative. But these papers do not address *information flow* generally, considering only the narrower question of how to quantify an adversary's uncertainty about who is communicating in a mix network. In the literature on quantitative information flow, we believe that the ideas we propose have not previously appeared—see for instance the recent high-profile papers of Malacaria [10] and Köpf and Basin [3].

In Section 5, we also develop techniques (based on Bayesian inference) for calculating conditional vulnerability and conditional min-entropy, for both deterministic and probabilistic programs. And we illustrate the reasonableness of our definitions by considering a number of examples.

Finally, Section 6 discusses some future directions and concludes.

A preliminary discussion of the ideas in this paper—restricted to deterministic programs and not using min-entropy—was included as a part of an invited talk and subsequent invited tutorial paper at the *TGC 2007 Workshop on the Interplay of Programming Languages and Cryptography* [19].

2 Our Conceptual Framework

The framework that we consider in this paper aims for simplicity and clarity, rather than full generality. We therefore restrict our attention to total programs c with just one input $\mathtt{H}$, which is high, and one output $\mathtt{L}$, which is low. (Thus we do not consider the case of programs that receive both high and low inputs, or programs that might not terminate.) Our goal is to quantify how much, if any, information in $\mathtt{H}$ is leaked by program c to $\mathtt{L}$. More precisely, our question is how much information about $\mathtt{H}$ can be deduced by an *adversary* $\mathcal{A}$ who sees the output $\mathtt{L}$.

We further assume that there is an *a priori*, *publicly-known* probability distribution on $\mathtt{H}$. We therefore assume that $\mathtt{H}$ is a random variable with a finite space of possible values $\mathcal{H}$. We denote the *a priori* probability that $\mathtt{H}$ has value h by $P[\mathtt{H} = h]$, and we assume that each element h of $\mathcal{H}$ has nonzero probability. Similarly, we assume that $\mathtt{L}$ is a random variable with a finite space of possible values $\mathcal{L}$, and with probabilities $P[\mathtt{L} = \ell]$. We assume that each output $\ell \in \mathcal{L}$ is possible, in that it can be produced by some input $h \in \mathcal{H}$.

In general, program c could be *deterministic* or it could be *probabilistic*. We consider these two cases separately.

2.1 Deterministic Programs

If c is a deterministic program, then the output $\mathtt{L}$ is a *function* of the input $\mathtt{H}$; that is, c determines a function $f : \mathcal{H} \to \mathcal{L}$ such that $\mathtt{L} = f(\mathtt{H})$. Now, following Köpf and Basin [3], we note that f induces an *equivalence relation* $\sim$ on $\mathcal{H}$:

$$h \sim h' \text{ iff } f(h) = f(h').$$

(In set theory, $\sim$ is called the *kernel* of f.) Hence program c *partitions* $\mathcal{H}$ into the equivalence classes of $\sim$. We let $\mathcal{H}_\ell$ denote the equivalence class $f^{-1}(\ell)$:

$$\mathcal{H}_\ell = \{h \in \mathcal{H} \mid f(h) = \ell\}.$$

Notice that there are $|\mathcal{L}|$ distinct equivalence classes, since we are assuming that each output $\ell \in \mathcal{L}$ is possible. The importance of these equivalence classes is that they bound the knowledge of the adversary $\mathcal{A}$: if $\mathcal{A}$ sees that L has value ℓ, then $\mathcal{A}$ knows only that the value of H belongs to the equivalence class $\mathcal{H}_\ell$. How much does this equivalence class tell $\mathcal{A}$ about H?

In one extreme, the function determined by c is a *constant* function, with just one possible value ℓ. In this case $\sim$ has just one equivalence class $\mathcal{H}_\ell$, which is equal to $\mathcal{H}$. Hence there is no leakage of H and *noninterference* holds.

In the other extreme, the function determined by c is *one-to-one*. Here the equivalence classes of $\sim$ are all singletons, and we have *total leakage* of H. Note however that, given ℓ, adversary $\mathcal{A}$ might not be able to *compute* the equivalence class $\mathcal{H}_\ell$ efficiently; thus we are adopting a worst-case, *information theoretic* viewpoint, rather than a *computational* one.

As an intermediate example, suppose that H is a 32-bit unsigned integer, with range $0 \leq \mathtt{H} < 2^{32}$. The program

```
L := H & 037
```

copies the last 5 bits of H into L. (Here 037 is an octal constant, and & denotes bitwise "and".) In this case, $\sim$ has $2^5 = 32$ equivalence classes, each of which contains 2^{27} elements. Intuitively, c leaks 5 bits (out of 32) of H.

2.2 Probabilistic Programs

More generally, the program c might be probabilistic. In this case, each value of H could lead to more than one value of L, which means that c may not give a partition of $\mathcal{H}$.

Following [2], we can model a probabilistic program c using a matrix whose rows are indexed by $\mathcal{H}$ and whose columns are indexed by $\mathcal{L}$, where the (h, ℓ) entry specifies the conditional probability $P[\mathtt{L} = \ell | \mathtt{H} = h]$. Notice that each row of this matrix must sum to 1. Also notice that in the special case where c is deterministic, each row will have one entry equal to 1 and all others equal to 0.

3 Existing Definitions of Quantitative Information Flow

Given (deterministic or probabilistic) program c, which may leak information from H to L, we want to *define* how much information c leaks. In the literature, such definitions are usually based on information-theoretic measures, such as Shannon entropy [20,21,22].

First we briefly review some of these measures. Let X be a random variable whose set of possible values is $\mathcal{X}$. The *Shannon entropy* $H(\mathtt{X})$ is defined by

$$H(\mathtt{X}) = \sum_{x \in \mathcal{X}} P[\mathtt{X} = x] \log \frac{1}{P[\mathtt{X} = x]}.$$

(Throughout we assume that $\log$ denotes logarithm with base 2.) The Shannon entropy can be viewed intuitively as the "uncertainty about $\mathtt{X}$"; more precisely it can be understood as the expected number of bits required to transmit $\mathtt{X}$ optimally.

Given two (jointly distributed) random variables $\mathtt{X}$ and $\mathtt{Y}$, the *conditional entropy* $H(\mathtt{X}|\mathtt{Y})$, intuitively the "uncertainty about $\mathtt{X}$ given $\mathtt{Y}$", is

$$H(\mathtt{X}|\mathtt{Y}) = \sum_{y \in \mathcal{Y}} P[\mathtt{Y} = y] H(\mathtt{X}|\mathtt{Y} = y)$$

where

$$H(\mathtt{X}|\mathtt{Y} = y) = \sum_{x \in \mathcal{X}} P[\mathtt{X} = x|\mathtt{Y} = y] \log \frac{1}{P[\mathtt{X} = x|\mathtt{Y} = y]}.$$

Note that if $\mathtt{X}$ is determined by $\mathtt{Y}$, then $H(\mathtt{X}|\mathtt{Y}) = 0$.

The *mutual information* $I(\mathtt{X};\mathtt{Y})$, intuitively the "amount of information shared between $\mathtt{X}$ and $\mathtt{Y}$", is

$$I(\mathtt{X};\mathtt{Y}) = H(\mathtt{X}) - H(\mathtt{X}|\mathtt{Y}).$$

Mutual information turns out to be symmetric: $I(\mathtt{X};\mathtt{Y}) = I(\mathtt{Y};\mathtt{X})$.

The *guessing entropy* $G(\mathtt{X})$ is the expected number of guesses required to guess $\mathtt{X}$ optimally; of course the optimal strategy is to guess the values of $\mathtt{X}$ in nonincreasing order of probability. If we assume that $\mathtt{X}$'s probabilities are arranged in nonincreasing order $p_1 \geq p_2 \geq \ldots \geq p_n$, then we have

$$G(\mathtt{X}) = \sum_{i=1}^{n} i p_i.$$

Now we consider how these entropy concepts can be used to quantify information leakage. Recalling the slogan

initial uncertainty = information leaked + remaining uncertainty

we have three quantities to define. For the initial uncertainty about $\mathtt{H}$, the entropy $H(\mathtt{H})$ seems appropriate. For the remaining uncertainty about $\mathtt{H}$, the conditional entropy $H(\mathtt{H}|\mathtt{L})$ seems appropriate. Finally, for the information leaked to $\mathtt{L}$, the entropy $H(\mathtt{L})$ might appear appropriate as well, but this cannot be correct in the case where c is probabilistic. For in that case, $\mathtt{L}$ might get positive entropy simply from the probabilistic nature of c, even though there is no leakage from $\mathtt{H}$. So we need something different.

Rearranging the slogan above, we get

information leaked = initial uncertainty $-$ remaining uncertainty.

This suggests that the information leaked to $\mathtt{L}$ should be $H(\mathtt{H}) - H(\mathtt{H}|\mathtt{L})$, which is just the mutual information $I(\mathtt{H};\mathtt{L})$.

If, however, we restrict to the case where c is deterministic, then we know that $\mathtt{L}$ is determined by $\mathtt{H}$. In that case we have $H(\mathtt{L}|\mathtt{H}) = 0$ which implies that

$$I(\mathtt{H};\mathtt{L}) = I(\mathtt{L};\mathtt{H}) = H(\mathtt{L}) - H(\mathtt{L}|\mathtt{H}) = H(\mathtt{L}).$$

So, in the case of deterministic programs, the mutual information $I(\mathtt{H};\mathtt{L})$ can be simplified to the entropy $H(\mathtt{L})$.

We can apply these definitions to some example programs. If we assume that H is a uniformly-distributed 32-bit integer, with range $0 \leq \mathtt{H} < 2^{32}$, then we get the following results:

Program	$H(\mathtt{H})$	$I(\mathtt{H};\mathtt{L})$	$H(\mathtt{H}\vert\mathtt{L})$
`L := 0`	32	0	32
`L := H`	32	32	0
`L := H & 037`	32	5	27

Turning to the research literature, the definitions we have described:

- initial uncertainty = $H(\mathtt{H})$
- information leaked = $I(\mathtt{H};\mathtt{L})$
- remaining uncertainty = $H(\mathtt{H}|\mathtt{L})$

seem to be the emerging consensus. Clarke, Hunt, and Malacaria [7,8,9,10] use these definitions, although they also address the more general case where the program c receives both high and low input. Köpf and Basin [3] use these definitions in their study of side-channel attacks, but they consider only the deterministic case. (They also consider guessing entropy and marginal guesswork in addition to Shannon entropy.) Chatzikokolakis, Palamidessi, and Panangaden [2] also use these definitions in their study of anonymity protocols. However, they are especially interested in situations where it is unreasonable to assume an *a priori* distribution on H; this leads them to emphasize the *channel capacity*, which is the *maximum* value of $I(\mathtt{H};\mathtt{L})$ over all distributions on H. Finally, the framework of Clarkson, Myers, and Schneider [12] is a significant extension of what we have described here, because they consider the case when the adversary $\mathcal{A}$ has (possibly mistaken) *beliefs* about the probability distribution on H. But in the special case when $\mathcal{A}$'s beliefs match the *a priori* distribution, and when the expected flow over all experiments is considered (see Section 4.4 of their paper), then their approach reduces to using the above definitions.

4 Security Guarantees with the Consensus Definitions

Given the consensus definitions of quantitative information flow described in Section 3, we now turn our attention to the question of what security guarantees these definitions support.

A first result along these lines is proved in [8]; they show, for deterministic programs, that $H(\mathtt{L})$ (the "information leaked") is 0 iff c satisfies noninterference. This is good, of course, but it is only a sanity check—it establishes that the zero/nonzero distinction is meaningful, but not that different nonzero values are meaningful.

Really the key question with respect to security is whether the value of $H(\mathtt{H}|\mathtt{L})$ (the "remaining uncertainty") accurately reflects the threat to H.

One bound that seems promising in justifying the significance of $H(\mathtt{H}|\mathtt{L})$ is given by Clark, Hunt, and Malacaria [7] based on work by Massey [23]. It states that the guessing entropy $G(\mathtt{H}|\mathtt{L})$, which is the expected number of guesses required to guess H given L, satisfies

$$G(\mathtt{H}|\mathtt{L}) \geq 2^{H(\mathtt{H}|\mathtt{L}) - 2} + 1 \qquad (1)$$

provided that $H(\mathtt{H}|\mathtt{L}) \geq 2$. For example, consider the program discussed above,

```
L := H & 037
```

where H is uniformly distributed with range $0 \leq \mathtt{H} < 2^{32}$. Here we have $H(\mathtt{H}|\mathtt{L}) = 27$, since each equivalence class contains 2^{27} elements, uniformly distributed. So by (1) we have

$$G(\mathtt{H}|\mathtt{L}) \geq 2^{25} + 1$$

which is quite an accurate bound, since the actual expected number of guesses is

$$\frac{2^{27}+1}{2}.$$

But note however that when we assess the threat to H, the adversary's *expected* number of guesses is probably not the key concern. The problem is that even if the expected number of guesses is huge, the adversary might nonetheless have a significant probability of guessing the value of H in just one try.

A result that addresses exactly this question is the classic *Fano inequality*, which gives lower bounds, in terms of $H(\mathtt{H}|\mathtt{L})$, on the probability that adversary $\mathcal{A}$ will *fail* to guess the value of H correctly in one try, given the value of L. Let P_e denote this probability. The Fano inequality is

$$P_e \geq \frac{H(\mathtt{H}|\mathtt{L}) - 1}{\log(|\mathcal{H}| - 1)}. \tag{2}$$

Unfortunately this bound is extremely weak in many cases. For example, on the program

```
L := H & 037
```

the Fano inequality gives

$$P_e \geq \frac{27-1}{\log(2^{32}-1)} \approx 0.8125$$

But this wildly understates the probability of error, since here the adversary has no knowledge of 27 of the bits of H, which implies that

$$P_e \geq \frac{2^{27}-1}{2^{27}} \approx 0.9999999925$$

One might wonder whether the Fano inequality could be strengthened, but (as we will illustrate below) this is not in general possible.

Fundamentally, the problem is that $H(\mathtt{H}|\mathtt{L})$ is of little value in characterizing the threat that the adversary, given L, could guess H. We demonstrate this claim through two key examples. Assume that H is a uniformly distributed $8k$-bit integer with range $0 \leq \mathtt{H} < 2^{8k}$, where $k \geq 2$. Hence $H(\mathtt{H}) = 8k$.

The first example is the program

```
if H mod 8 = 0 then
   L := H
else
   L := 1
```
(3)

Since this program is deterministic, its information leakage is just $H(\texttt{L})$. Notice that the `else` branch is taken on 7/8 of the values of H, namely those whose last 3 bits are not all 0. Hence

$$P[\texttt{L} = 1] = \frac{7}{8}$$

and

$$P[\texttt{L} = 8n] = 2^{-8k}$$

for each n with $0 \leq n < 2^{8k-3}$. Hence we have

$$H(\texttt{L}) = \frac{7}{8} \log \frac{8}{7} + 2^{8k-3} 2^{-8k} \log 2^{8k} \approx k + 0.169$$

This implies that

$$H(\texttt{H}|\texttt{L}) \approx 7k - 0.169$$

suggesting that about $7/8$ of the information in H remains unleaked. But, since the `then` branch is taken $1/8$ of the time, the adversary can guess the value of H at least $1/8$ of the time! (We remark that this example shows that the Fano inequality cannot in general be strengthened significantly—here the Fano inequality says that the probability of error is at least

$$\frac{7k - 1.169}{\log(2^{8k} - 1)}$$

which is close to $7/8$ for large k.)

The second example (using the same assumptions about H) is the program

$$\texttt{L := H \& } 0^{7k-1}1^{k+1} \qquad (4)$$

where $0^{7k-1}1^{k+1}$ is a binary constant; this program copies the last $k + 1$ bits of H into L. Hence we have

$$H(\texttt{L}) = k + 1$$

and

$$H(\texttt{H}|\texttt{L}) = 7k - 1.$$

Here notice that, given L, the adversary's probability of guessing H is just $1/2^{7k-1}$.

The key point to emphasize here is that, under the consensus definitions, program (4) is actually *worse* than program (3), even though program (3) leaves H highly vulnerable to being guessed, while program (4) does not. The conclusion is that, with respect to this threat model, the consensus definitions do a poor job of measuring the threat: $H(\texttt{H}|\texttt{L})$ does not support good security guarantees about the probability that H could be guessed.

5 An Alternative Foundation: Vulnerability and Min-entropy

The limitations of the consensus definitions noted in Section 4 lead us now to explore alternative definitions of quantitative information flow, with the goal of finding a measure supporting better security guarantees with respect to the probability that the adversary could guess H in one try.

Rather than inventing a new measure and then trying to prove that it implies good security guarantees, why not define a measure of remaining uncertainty directly in terms of the desired security guarantees? To this end, we propose the concept of *vulnerability*:

Definition 1. *Given a random variable* $\mathtt{X}$ *with space of possible values* $\mathcal{X}$*, the* vulnerability of $\mathtt{X}$*, denoted* $V(\mathtt{X})$*, is given by*

$$V(\mathtt{X}) = \max_{x \in \mathcal{X}} P[\mathtt{X} = x].$$

The vulnerability $V(\mathtt{X})$ is thus the worst-case probability that an adversary $\mathcal{A}$ could guess the value of $\mathtt{X}$ correctly in one try. It is clearly a rather crude measure, because it depends only on the *maximum* probability in the distribution of $\mathtt{X}$, focusing on the single probability that brings the greatest risk. Limiting to a single guess might seem unreasonable, of course. But notice that with m guesses the adversary can succeed with probability at most $mV(\mathtt{X})$. This implies (very roughly speaking) that if the vulnerability with m guesses is "significant", where m is a "practical" number of tries, then $V(\mathtt{H})$ must itself be "significant".

Vulnerability is a probability, so its value is always between 0 and 1. But to quantify information flow, we would like to measure information in *bits*. We can convert to an entropy measure by mapping $V(\mathtt{X})$ to

$$\log \frac{1}{V(\mathtt{X})}.$$

This, it turns out, gives a measure known as *min-entropy*:

Definition 2. *The* min-entropy of $\mathtt{X}$*, denoted* $H_\infty(\mathtt{X})$*, is given by*

$$H_\infty(\mathtt{X}) = \log \frac{1}{V(\mathtt{X})}.$$

As far as we know, min-entropy has not previously been used in quantitative information flow. But, as noted in Section 1, it has been used to measure the anonymity provided by mix networks [17,18]. Also, Cachin [24] discusses its relevance in cryptographic guessing attacks.

Min-entropy is the instance of *Rényi entropy* [16]

$$H_\alpha(\mathtt{X}) = \frac{1}{1-\alpha} \log \left(\sum_{x \in \mathcal{X}} P[\mathtt{X} = x]^\alpha \right)$$

obtained when $\alpha = \infty$. Notice that if $\mathtt{X}$ is uniformly distributed among n values, then $V(\mathtt{X}) = 1/n$ and $H_\infty(\mathtt{X}) = \log n$. Hence Shannon entropy $H(\mathtt{X})$ and min-entropy $H_\infty(\mathtt{X})$ coincide on uniform distributions. But, in general, Shannon entropy can be arbitrarily greater than min-entropy, since $H(\mathtt{X})$ can be arbitrarily high even if $\mathtt{X}$ has a value with a probability close to 1.

We propose to use $H_\infty(\mathtt{H})$ as our measure of initial uncertainty. To measure the remaining uncertainty, we first consider *conditional vulnerability*, which gives the expected probability of guessing $\mathtt{X}$ in one try, given $\mathtt{Y}$:

Definition 3. *Given (jointly distributed) random variables* $\mathtt{X}$ *and* $\mathtt{Y}$*, the* conditional vulnerability $V(\mathtt{X}|\mathtt{Y})$ *is*

$$V(\mathtt{X}|\mathtt{Y}) = \sum_{y \in \mathcal{Y}} P[\mathtt{Y} = y] V(\mathtt{X}|\mathtt{Y} = y)$$

where

$$V(\mathtt{X}|\mathtt{Y}=y) = \max_{x\in\mathcal{X}} P[\mathtt{X}=x|\mathtt{Y}=y].$$

We now show that $V(\mathtt{H}|\mathtt{L})$ is easy to calculate for probabilistic programs c, given the *a priori* distribution on $\mathtt{H}$ and the matrix of conditional probabilities $P[\mathtt{L}=\ell|\mathtt{H}=h]$. In fact, $V(\mathtt{H}|\mathtt{L})$ is simply the complement of the *Bayes risk* P_e.

First we note that by Bayes' theorem we have

$$P[\mathtt{H}=h|\mathtt{L}=\ell]P[\mathtt{L}=\ell] = P[\mathtt{L}=\ell|\mathtt{H}=h]P[\mathtt{H}=h].$$

Now we have

$$\begin{aligned}
V(\mathtt{H}|\mathtt{L}) &= \sum_{\ell\in\mathcal{L}} P[\mathtt{L}=\ell]V(\mathtt{H}|\mathtt{L}=\ell)\\
&= \sum_{\ell\in\mathcal{L}} P[\mathtt{L}=\ell] \max_{h\in\mathcal{H}} P[\mathtt{H}=h|\mathtt{L}=\ell]\\
&= \sum_{\ell\in\mathcal{L}} \max_{h\in\mathcal{H}} P[\mathtt{H}=h|\mathtt{L}=\ell]P[\mathtt{L}=\ell]\\
&= \sum_{\ell\in\mathcal{L}} \max_{h\in\mathcal{H}} P[\mathtt{L}=\ell|\mathtt{H}=h]P[\mathtt{H}=h].
\end{aligned}$$

It should be noted that [13] proposes Bayes risk as a measure of protection in anonymity protocols, and also includes essentially the same calculation as above.

We next observe that the calculation of $V(\mathtt{H}|\mathtt{L})$ becomes simpler in the special case where program c is deterministic. For in that case $\mathcal{H}$ is partitioned into $|\mathcal{L}|$ equivalence classes $\mathcal{H}_\ell$, where

$$\mathcal{H}_\ell = \{h \in \mathcal{H} \mid P[\mathtt{L}=\ell|\mathtt{H}=h]=1\}.$$

Hence we have

$$\begin{aligned}
V(\mathtt{H}|\mathtt{L}) &= \sum_{\ell\in\mathcal{L}} \max_{h\in\mathcal{H}} P[\mathtt{L}=\ell|\mathtt{H}=h]P[\mathtt{H}=h]\\
&= \sum_{\ell\in\mathcal{L}} \max_{h\in\mathcal{H}_\ell} P[\mathtt{H}=h].
\end{aligned}$$

Finally, we note that in the special case where c is deterministic and $\mathtt{H}$ is uniformly distributed, the conditional vulnerability becomes very simple indeed:

$$\begin{aligned}
V(\mathtt{H}|\mathtt{L}) &= \sum_{\ell\in\mathcal{L}} \max_{h\in\mathcal{H}_\ell} P[\mathtt{H}=h]\\
&= \sum_{\ell\in\mathcal{L}} (1/|\mathcal{H}|)\\
&= |\mathcal{L}|/|\mathcal{H}|
\end{aligned}$$

Thus in this case all that matters is the *number* of equivalence classes. (We remark that Lowe [14] focuses on a quantity analogous to $|\mathcal{L}|$ in quantifying information flow in a process calculus, even though his approach is not probabilistic.)

We now define $H_\infty(\mathtt{H}|\mathtt{L})$, which will be our measure of remaining uncertainty:

Definition 4. *The* conditional min-entropy $H_\infty(\mathtt{X}|\mathtt{Y})$ *is*

$$H_\infty(\mathtt{X}|\mathtt{Y}) = \log \frac{1}{V(\mathtt{X}|\mathtt{Y})}.$$

Note that this definition of conditional min-entropy is *not* the same as the one given by Cachin [24, p. 16], but it is equivalent to the one proposed by Dodis et al. [25].

We now propose the following definitions:

- initial uncertainty = $H_\infty(\mathtt{H})$
- remaining uncertainty = $H_\infty(\mathtt{H}|\mathtt{L})$
- information leaked = $H_\infty(\mathtt{H}) - H_\infty(\mathtt{H}|\mathtt{L})$

Note that our measure of remaining uncertainty, $H_\infty(\mathtt{H}|\mathtt{L})$, gives an immediate security guarantee:

$$V(\mathtt{H}|\mathtt{L}) = 2^{-H_\infty(\mathtt{H}|\mathtt{L})}.$$

Thus the expected probability that the adversary could guess H given L decreases exponentially with $H_\infty(\mathtt{H}|\mathtt{L})$.

Also note that calculating the information leakage is easy in the case where c is deterministic and H is uniformly distributed:

Theorem 1. *If c is deterministic and* H *is uniformly distributed, then the information leaked is* $\log|\mathcal{L}|$.

Proof. Here we have $V(\mathtt{H}) = 1/|\mathcal{H}|$ and $V(\mathtt{H}|\mathtt{L}) = |\mathcal{L}|/|\mathcal{H}|$, so

$$H_\infty(\mathtt{H}) - H_\infty(\mathtt{H}|\mathtt{L}) = \log|\mathcal{H}| - \log(|\mathcal{H}|/|\mathcal{L}|) = \log|\mathcal{L}|. \qquad \square$$

Let us now revisit example programs (4) and (3) from Section 4 using our new definitions. Because these programs are deterministic and H is uniformly distributed, we only need to focus on $|\mathcal{H}|$ and $|\mathcal{L}|$. Note that $|\mathcal{H}| = 2^{8k}$, so the initial uncertainty $H_\infty(\mathtt{H})$ is $8k$, as before.

On program (4), we get the same values as before. We have $|\mathcal{L}| = 2^{k+1}$, which implies that the information leaked is $k+1$ and the remaining uncertainty is $7k-1$.

But on program (3), we have $|\mathcal{L}| = 2^{8k-3}+1$, which implies that the information leaked is about $8k-3$ and the remaining uncertainty is about 3. Thus our new measures hugely increase the leakage ascribed to this program.

It is interesting to compare program (3) with a program that always leaks all but the last 3 bits of H:

```
L := H | 07
```
(5)

(Here | denotes bitwise "or".) For this program, $|\mathcal{L}| = 2^{8k-3}$, so it is ascribed almost exactly the same leakage as program (3). Notice that while both of these programs make H highly vulnerable, the threats are different: with program (3), the adversary $\mathcal{A}$ learns H completely $1/8$ of the time, and learns very little $7/8$ of the time; with program (5), in contrast, $\mathcal{A}$ never learns H completely, but always learns it to within 8 possible values.

Is it reasonable to ascribe the same leakage to programs (3) and (5)? That seems hard to answer without further assumptions. For instance, if $\mathcal{A}$ is allowed *several* guesses, rather than just one, then program (5) is clearly worse. On the other hand, if a *wrong* guess would trigger an alert, then program (3) might be worse, since then $\mathcal{A}$ *knows* whether it knows H or not, and could choose to make a guess only when it knows. These examples suggest the difficulty of measuring a range of complex threat scenarios precisely using a single number; still, we feel that one-guess vulnerability is a sufficiently basic concern to serve as a generally useful foundation.

As another example, consider a password checker, which tests whether H (assumed uniformly distributed) is equal to some particular value and assigns the result to L. Since $|\mathcal{L}| = 2$, we get a leakage of 1 here.

We remark that Köpf and Basin [3] briefly consider *worst-case entropy measures* in addition to the "averaging" measures (like $G(\mathtt{H}|\mathtt{L})$) used in the rest of their paper. Specifically, they define the *minimum guessing entropy* by

$$\hat{G}(\mathtt{H}|\mathtt{L}) = \min_{\ell \in \mathcal{L}} G(\mathtt{H}|\mathtt{L} = \ell).$$

But this measure is not very useful, as shown by the password checker example—the password checker splits $\mathcal{H}$ into 2 equivalence classes, one of which is a singleton. Hence the minimum guessing entropy is 1. This measure thus judges a password checker to be as bad as a program that leaks H completely.

As a final example, consider an election system. Suppose that we have an election between candidates A and B with k voters, whose individual votes are represented by the k-bit random variable H. We (unrealistically) assume that each voter independently votes for either A or B with probability $1/2$, which implies that H is uniformly distributed over 2^k values. The election system reveals into L the tally of votes received by candidate A, which means that L ranges over $\{0, 1, 2, \ldots, k\}$. Here the initial uncertainty is k and the leakage is $\log(k + 1)$. And the conditional vulnerability is

$$V(\mathtt{H}|\mathtt{L}) = \frac{k+1}{2^k}$$

some of whose values are shown in the following table:

k	1	2	3	4	5	6
$V(\mathtt{H}\|\mathtt{L})$	1	$3/4$	$1/2$	$5/16$	$3/16$	$7/64$

So the adversary's ability to guess the individual votes decreases exponentially with k.

We conclude this section with a curious result, whose significance is unclear (to me, at least). In the case of a deterministic program c and uniformly-distributed H, it turns out that our new definition of information leakage exactly coincides with the classical notion of the *channel capacity* of c.

Theorem 2. *If c is deterministic and* H *is uniformly distributed, then the information leaked,* $\log |\mathcal{L}|$*, is equal to the channel capacity of c.*

Proof. In the deterministic case, the channel capacity is the maximum value of $H(\mathtt{L})$ over all distributions on H. This maximum is $\log |\mathcal{L}|$, since L has $|\mathcal{L}|$ possible values and we can put a distribution on H that makes them all equally likely. (This observation is also made in [11].) Curiously, this will typically *not* be a uniform distribution on H. □

But perhaps this result is just a coincidence—it does not generalize to the case of probabilistic programs.

6 Conclusion

In this paper, we have focused on one specific, but natural, threat model: the expected probability that an adversary could guess H in one try, given L. We have argued that the consensus definitions of quantitative information flow do poorly with respect to this threat model, and have proposed new definitions based on vulnerability and min-entropy.

We mention some important future directions. First, the reasonableness of our definitions should be further assessed, both in terms of their theoretical properties and also by applying them in various specific threat scenarios. Also the definitions need to be generalized to model explicitly allowed flows, such as from low inputs to low outputs or (perhaps) from the secret individual votes in an election to the public tally. It would seem that this could be handled through conditional min-entropy.

Second, the possibility of enforcing quantitative information flow policies through static analysis needs to be explored; in the case of the standard measures there has been progress [9], but it is unclear whether min-entropy can be handled similarly. The results presented in Section 5 on how to calculate vulnerability seem encouraging, especially in the important special case of a deterministic program c mapping a uniformly-distributed H to an output L. For there we found that the leakage is simply $\log |\mathcal{L}|$. This fact seems to give insight into the difference between the cases of *password checking* and *binary search*. For in a password checker, we test whether a guess g is equal to the secret password. If the test comes out true, we know that the password is g; if it comes out false, we know only that the password is not g. Hence such a guess splits the space $\mathcal{H}$ into two equivalence classes, $\{g\}$ and $\mathcal{H} - \{g\}$. This implies that any *tree* of guesses of height k can give only $k + 1$ equivalence classes, which means that the vulnerability of the password increases slowly. In contrast, in binary search we compare the *size* of a guess g with the secret, discovering which is larger. Hence (with a well-chosen guess) we are able to split the space $\mathcal{H}$ into two equivalence classes of roughly equal size. This implies that a tree of guesses of height k can give 2^k equivalence classes, which means that the vulnerability of the secret increases very rapidly. These examples do raise concerns about *compositionality*, however, because the tests $g = \mathtt{H}$ and $g \leq \mathtt{H}$ both have a leakage of 1 under our definitions, even though *sequences* of these tests behave so differently.

Finally, it would be valuable (but challenging) to integrate the information-theoretic viewpoint used here with the computational complexity viewpoint used in cryptography.

Acknowledgments

I am grateful to Rafael Alpízar, Zesheng Chen, Pasquale Malacaria, Ziyuan Meng, Andrew Myers, Catuscia Palamidessi, and the anonymous reviewers for helpful discussions of this work. This work was partially supported by the National Science Foundation under grants HRD-0317692 and CNS-0831114.

References

1. Sabelfeld, A., Myers, A.C.: Language-based information flow security. IEEE Journal on Selected Areas in Communications 21(1), 5–19 (2003)
2. Chatzikokolakis, K., Palamidessi, C., Panangaden, P.: Anonymity protocols as noisy channels. Information and Computation 206, 378–401 (2008)
3. Köpf, B., Basin, D.: An information-theoretic model for adaptive side-channel attacks. In: Proceedings 14th ACM Conference on Computer and Communications Security, Alexandria, Virginia (2007)
4. Sabelfeld, A., Sands, D.: Dimensions and principles of declassification. In: Proceedings 18th IEEE Computer Security Foundations Workshop (June 2005)
5. Denning, D.: Cryptography and Data Security. Addison-Wesley, Reading (1982)
6. Gray III, J.W.: Probabilistic interference. In: Proceedings 1990 IEEE Symposium on Security and Privacy, Oakland, CA, pp. 170–179 (May 1990)
7. Clark, D., Hunt, S., Malacaria, P.: Quantitative analysis of the leakage of confidential data. Electronic Notes in Theoretical Computer Science 59(3) (2002)
8. Clark, D., Hunt, S., Malacaria, P.: Quantitative information flow, relations and polymorphic types. Journal of Logic and Computation 18(2), 181–199 (2005)
9. Clark, D., Hunt, S., Malacaria, P.: A static analysis for quantifying information flow in a simple imperative language. Journal of Computer Security 15, 321–371 (2007)
10. Malacaria, P.: Assessing security threats of looping constructs. In: Proceedings 34th Symposium on Principles of Programming Languages, Nice, France, pp. 225–235 (January 2007)
11. Malacaria, P., Chen, H.: Lagrange multipliers and maximum information leakage in different observational models. In: Proc. PLAS 2008: ACM SIGPLAN Workshop on Programming Languages and Analysis for Security, Tucson, Arizona, USA, pp. 135–146 (June 2008)
12. Clarkson, M., Myers, A., Schneider, F.: Belief in information flow. In: Proceedings 18th IEEE Computer Security Foundations Workshop, Aix-en-Provence, France, pp. 31–45 (June 2005)
13. Chatzikokolakis, K., Palamidessi, C., Panangaden, P.: Probability of error in information-hiding protocols. In: Proceedings 20th IEEE Computer Security Foundations Symposium, pp. 341–354 (2007)
14. Lowe, G.: Quantifying information flow. In: Proceedings 15th IEEE Computer Security Foundations Workshop, Cape Breton, Nova Scotia, Canada, pp. 18–31 (June 2002)
15. Di Pierro, A., Hankin, C., Wiklicky, H.: Approximate non-interference. In: Proceedings 15th IEEE Computer Security Foundations Workshop, Cape Breton, Nova Scotia, Canada, pp. 1–17 (June 2002)
16. Rényi, A.: On measures of entropy and information. In: Proceedings of the 4th Berkeley Symposium on Mathematics, Statistics and Probability 1960, pp. 547–561 (1961)
17. Tóth, G., Hornák, Z., Vajda, F.: Measuring anonymity revisited. In: Liimatainen, S., Virtanen, T. (eds.) Proceedings of the Ninth Nordic Workshop on Secure IT Systems, Espoo, Finland, pp. 85–90 (2004)
18. Shmatikov, V., Wang, M.H.: Measuring relationship anonymity in mix networks. In: WPES 2006: Proceedings of the 5th ACM workshop on Privacy in Electronic Society, Alexandria, Virginia, pp. 59–62 (2006)
19. Smith, G.: Adversaries and information leaks (Tutorial). In: Barthe, G., Fournet, C. (eds.) TGC 2007. LNCS, vol. 4912, pp. 383–400. Springer, Heidelberg (2008)
20. Shannon, C.E.: A mathematical theory of communication. Bell System Technical Journal 27, 379–423 (1948)

21. Gallager, R.G.: Information Theory and Reliable Communication. John Wiley and Sons, Inc., Chichester (1968)
22. Cover, T.M., Thomas, J.A.: Elements of Information Theory, 2nd edn. John Wiley & Sons, Inc., Chichester (2006)
23. Massey, J.L.: Guessing and entropy. In: Proceedings 1994 IEEE International Symposium on Information Theory, p. 204 (1994)
24. Cachin, C.: Entropy Measures and Unconditional Security in Cryptography. PhD thesis, Swiss Federal Institute of Technology (1997)
25. Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. SIAM Journal of Computing 38(1), 97–139 (2008)