292 - Fall 2021
Quantitative Information Flow and Side Channels

Instructor: Tevfik Bultan

Lecture 2

Slides for this lecture are based on the following papers:

Geoffrey Smith. On the Foundations of Quantitative Information Flow. FOSSACS 2009: 288-302

Geoffrey Smith. Quantifying Information Flow Using Min-Entropy. QEST 2011: 159-167

UCSB

How do we quantify information leakage?

- How can we quantify information leakage from a side channel (or main channel)?
- Before we figure out how to quantify information leakage, we need answer the following question:
  - How do we quantify information?

How do we quantify information?

- Shannon Entropy
- Intuitively
  - a measure of uncertainty about a random variable X
  - expected (average) amount of information gain (i.e., the expected amount of surprise) by observing the value of the random variable expressed in terms of bits
- More precisely

  expected (average) number of bits required to transmit X optimally

Entropy example:

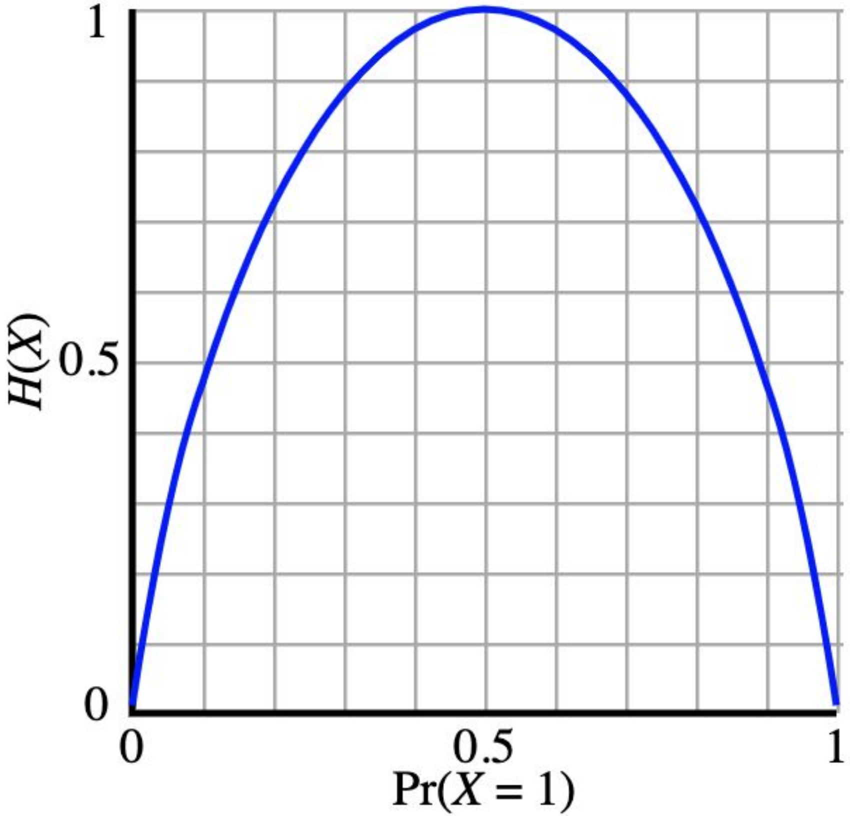
Example:

- Seattle weather, always raining: $p_{rain} = 1$
- Entropy: $H = 0$

- Costa Rica weather, coin flip: $p_{rain} = ½$, $p_{sun} = ½$
- Entropy: $H = 1$

- Santa Barbara weather, almost always beautiful: $p_{rain} = 1/10$, $p_{sun} = 9/10$
- Entropy: $H = 0.496$

Binary Entropy



UCSB

How do we quantify information?

- Random variable: X
- Set of possible values for the random variable: $\mathcal{X}$
- Probability that the random variable takes the value $x \in \mathcal{X}$

$$P[X = x]$$

- Shannon Entropy: $H(X)$

$$H(X) = \sum_{x \in \mathcal{X}} P[X = x] \log_2(1/P[X = x])$$
$$H(X) = E[\log_2(1/P[X = x])]$$

- i.e., Shannon entropy is the expected value of: $\log_2(1/P[X = x])$

How do we quantify information leakage?

- Now that we know how to quantify information, how can we quantify information leakage?
- First, let's give a simple program model:

  S is the secret input to the program. We will model it as a random variable.

  O is the public output of the program. We will model it also as a random variable

  f is a function from values of S to values of O we use to model a deterministic program

Initial uncertainty

- What is the initial uncertainty for S?
    - What is the amount of information that we need to learn about the secret?

$$H(S) = \sum_{s \in \mathcal{S}} P[S = s] \log_2 (1 / P[S = s])$$

- Assume that the probability distribution for the secret is uniform
    - so all values are equally likely
    - then, the amount of information that we need to learn is:

$$H(S) = \log_2 |\mathcal{S}|$$

UCSB

Partitioning the secret domain

- Given a program

$$f : \mathcal{S} \to \mathcal{O}$$

- The values we observe as the output of the program define an equivalence relation for the secret S

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

- So, by observing output of the program, we partition the secret values to equivalence classes

Partitioning the secret domain

- The number of equivalence classes in the partition are:

$$|\mathcal{O}|$$

- If the function is a constant function, where the output is constant, then

$$|\mathcal{O}| = 1$$

  - and, there is a single equivalence class where

$$\mathcal{S}_o = \mathcal{S}$$

UCSB

Non-interference

- So, if the output function is a constant function
  - the amount of information we need to learn remains the same

$$H(S) = \log_2 |\mathcal{S}|$$

  - means there is no information leakage

- This correspond to non-interference!
  - If the output/observable remains constant for all values of the secret then there is no information leakage!

Partitioning the secret domain

- Now, let us assume that the output values partition the secret domain to two equivalence classes with equal number of elements
  - I.e., there are two output values, half of the secret values map to one and the other half map to the other

- What is the remaining entropy?

Another example

```
f(S) { print S & 0xF; }
```

- Assume that S is a 32-bit unsigned integer
- 0xF is the hexadecimal constant corresponding to decimal 15, and & denotes bitwise “and” operation
  - So, the above code prints the last 4 bits of the secret
- The output partitions the secret domain to 16 equivalence classes, each of which has 2^{28} values in it
  - So, the remaining entropy is 28 bits

How do we quantify information leakage?

- Now that we know how to quantify information, how can we quantify information leakage
- Here is what we would expect:

  initial uncertainty = information leaked + remaining uncertainty

- Equivalently

  information leaked = initial uncertainty - remaining uncertainty

UCSB

How do we quantify the remaining uncertainty?

- Remaining uncertainty can be characterized as the conditional entropy
- Conditional entropy: What is the uncertainty about S given O?

$$H(S|O) = \sum_{o \in \mathcal{O}} P[O = o] H(S|O = o)$$

$$H(S|O = o) = \sum_{s \in \mathcal{S}} P[S = s|O = o] \log_2(1/P[S = s|O = o])$$

UCSB

Conditional Entropy uses Conditional Probability

$$H(S|O=o) = \sum_{s \in \mathcal{S}} P[S=s|O=o] \log_2(1/P[S=s|O=o])$$

$$P[S=s|O=o] = P[S=s,O=o]/P[O=o]$$

Mutual information

- Mutual information I(S;O) is the amount of information shared between S and O
- It is defined as:

$$I(S;O) = H(S) - H(S|O)$$

- Mutual information is symmetric:

$$I(S;O) = I(O;S)$$

How do we quantify information leakage?

- So, the intuitive property

  information leaked = initial uncertainty - remaining uncertainty

- is formalized as

$$I(S;O) = H(S) - H(S|O)$$

Examples

$$I(S;O) = H(S) - H(S|O)$$

`f(S) { print 10; }`	0	=	32	-	32
`f(S) { print S + 10; }`	32	=	32	-	0
`f(S) { print S & 0xF; }`	4	=	32	-	28

What about side channels?

```
f(S) { sleep(S); }

f(S) { if (S % 2 == 0) sleep(1); else sleep (2); }
```

- These programs do not return any output or print any information.
  - So, they do not leak information from the main channel of the program.
- However, they do have side channel leakage
  - They leak information from the execution time

What about side channels?

$$I(S;O) = H(S) - H(S|O)$$

```
f(S) { sleep(S); }
```

32 = 32 - 0

```
f(S) {if (S % 2 == 0)
          sleep(1);
      else
          sleep (2); }
```

1 = 32 - 31

Deterministic programs

- If we assume that the program is deterministic with only input S and only output O
  - then the value of O is determined only by the input S
  - which means $H(O|S) = 0$

Then, we have:

$$I(S;O) = I(O;S) = H(O) - H(O|S) = H(O)$$

- So, for deterministic programs with input S and output O, the information leaked is equivalent to the uncertainty of O