Quantitative Information Flow and Side Channels

Instructor: Tevfik Bultan
Lecture 2
Slides for this lecture are based on the following papers:


Geoffrey Smith. Quantifying Information Flow Using Min-Entropy. QEST 2011: 159-167
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
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: $\log_2(1/P[X = x])$
Entropy example:

Example:

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

- Costa Rica weather, coin flip: $p_{\text{rain}} = \frac{1}{2}$, $p_{\text{sun}} = \frac{1}{2}$
- Entropy: $H = 1$

- Santa Barbara weather, almost always beautiful: $p_{\text{rain}} = \frac{1}{10}$, $p_{\text{sun}} = \frac{9}{10}$
- Entropy: $H = 0.496$
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?
    \[
    H(S) = \sum_{s \in S} P[S = s] \log_2 \left( \frac{1}{P[S = s]} \right)
    \]

- 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:
    \[
    H(S) = \log_2 |S|
    \]
Partitioning the secret domain

- Given a program
  \[ f : S \rightarrow \mathcal{O} \]

- 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:
  \[ |O| \]

- If the function is a constant function, where the output is constant, then
  \[ |O| = 1 \]
  
  - and, there is a single equivalence class where
    \[ S_o = S \]
Non-interference

- So, if the output function is a constant function
  - the amount of information we need to learn remains the same
    \[ H(S) = \log_2 |S| \]
  - 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

```c
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
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 $O$?

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

$$H(S|O = o) = \sum_{s \in S} P[S = s|O = o] \log_2 \left( \frac{1}{P[S = s|O = o]} \right)$$
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
  \[
  \text{information leaked} = \text{initial uncertainty} - \text{remaining uncertainty}
  \]
- is formalized as
  \[
  I(S; O) = H(S) - H(S|O)
  \]
Examples

\[ I(S; O) = H(S) - H(S|O) \]

\[
\begin{align*}
\text{f(S) \{ print 10; \} } & \quad 0 \quad = \quad 32 \quad - \quad 32 \\
\text{f(S) \{ print S + 10; \} } & \quad 32 \quad = \quad 32 \quad - \quad 0 \\
\text{f(S) \{ print S \& 0xF; \} } & \quad 4 \quad = \quad 32 \quad - \quad 28
\end{align*}
\]
What about side channels?

```java
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?

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

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

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

\[
\begin{align*}
32 &= 32 - 0 \\
1 &= 32 - 31
\end{align*}
\]
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
Other definitions of entropy

- Note that Shannon entropy is about the expected value which averages over all possibilities.
- This may not be suitable if the goal is to assess vulnerability of a software system.
- Rather than evaluating how much information leaks on average, we may want to evaluate how much information leaks in the worst case.
- There are different entropy definitions which may be more suitable if the goal is to assess the vulnerability of a software system.
Guessing entropy

- Guessing entropy $G(S)$ is defined as the expected number of guesses required to guess $S$ optimally.
- Optimal strategy is to guess the values of $S$ in nonincreasing order of probability.

If we assume:

$$p_1 \geq p_2 \geq \ldots \geq p_n$$

then

$$G(S) = \sum_{i=1}^{n} ip_i$$
Guessing entropy vs. remaining uncertainty

- Remaining entropy $H(S|O)$ provides a bound for guessing entropy $G(S|O)$ (expected number of guesses required to guess $S$ given $O$) assuming that $H(S|O)$ is greater than or equal to 2.

$$G(S|O) \geq 2^{H(S|O) - 2} + 1$$

assuming that $H(S|O)$ is greater than or equal to 2.
Fano inequality

Let $P_e$ denote the probability that an adversary will fail to guess the value of $S$ correctly in one try, given the value of $O$.

Then, we have

$$P_e \geq (H(S|O) - 1) / \log_2(|\mathcal{S}| - 1)$$
Vulnerability and min-entropy

Let vulnerability of $S$ be $V(S)$, defined as

$$V(S) = \max_{s \in S} P[S = s]$$

Vulnerability $V(S)$ is the worst-case probability that an adversary could guess the value of the secret correctly in one try.

Then, min-entropy $H_\infty(S)$ is defined as

$$H_\infty(S) = \log_2 \left( 1 / V(S) \right)$$
Min-entropy

Min-entropy is an instance of Renyi-entropy

\[ H_\alpha(X) = \frac{1}{(1 - \alpha)} \log_2 \left( \sum_{x \in \chi} P[X = x]^\alpha \right) \]

where \( \alpha = \infty \)

So, min-entropy is also called Renyi min-entropy

If the distribution is uniform, then min-entropy is equal to the Shannon entropy
Shannon entropy vs. min-entropy

\[ S = \{a, b\} \]

\[ p(a) = x \quad p(b) = 1 - x \]
Conditional vulnerability

Conditional vulnerability $V(S|O)$ is defined as:

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

where

$$V(S|O = o) = \max_{s \in S} P[S = s | O = o]$$

then

$$V(S|O) = \sum_{o \in \mathcal{O}} \max_{s \in S} P[O = o | S = s] P[S = s]$$
Conditional min-entropy

Conditional min-entropy $H_\infty(S|O)$ is defined as:

$$H_\infty(S|O) = \log_2(1/V(S|O))$$

So, now we can use the following alternative definitions

initial uncertainty: $H_\infty(S)$

remaining uncertainty: $H_\infty(S|O)$

Information leaked (min-mutual information):

$$I_\infty(S;O) = H_\infty(S) - H_\infty(S|O)$$

and, we also have:

$$V(S|O) = 2^{-H_\infty(S|O)}$$
Deterministic programs

For deterministic programs where the secret is uniformly distributed, we have

\[ V(S) = 1/|S| \text{ and } V(S|O) = |S|/|O| \]

then the information leakage can be computed as:

\[ H_\infty(S) - H_\infty(S|O) = \log_2 |S| - \log_2 (|S|/|O|) = \log_2 |O| \]
Comparing Shannon and min entropy

Assume S is a 8k-bit integer value, uniformly distributed

Program 1:

\[ f(S) \{ \text{if } (S \mod 8 == 0) \text{ print } H; \text{ else print } 1; \} \]

Program 2:

\[ f(S) \{ \text{print } S \& C \} \]

where C is a binary constant and its least significant k+1 bits are one, rest are 0
Shannon entropy for programs 1 and 2

Since the input is a uniformly distributed 8k-bit integer value, for both programs 1 and 2 we have:

\[ H(S) = 8k \]

Since the programs 1 and 2 are deterministic, we also have:

\[ H(O|S) = 0 \]

which implies that

\[ I(S;O) = H(O) \quad \text{and} \quad H(H|O) = H(S) - H(O) \]
Shannon entropy for programs 1 and 2

We can compute $H(O)$ for program 1 by noting that:

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

and

$P[O=8n] = \frac{1}{2^{8k}}$ for each $n$ where $1 \leq n < 2^{8k-3}$

Then, $H(O) = \frac{7}{8} \log_2(\frac{8}{7}) + 2^{8k-3}\frac{1}{2^{8k}}\log_2(\frac{1}{2^{8k}}) \approx k + 0.169$

which means $I(S;O) = k + 0.169$

and $H(H|O) = H(S) - H(O) = 8k - (k + 0.169) = 7k - 0.169$
Shannon entropy for programs 1 and 2

We can compute $H(O)$ for program 2 by noting that $k+1$ bits of $S$ is copied to $O$, so

$$H(O) = k + 1$$

which means $I(S;O) = k + 1$

and $H(H|O) = H(S) - H(O) = 8k - (k + 1) = 7k - 1$
Shannon entropy for programs 1 and 2

So, Program 2 leaks more information according to Shannon entropy.

Note that, program 1 leaks the full secret $\frac{1}{8}$ of the time, whereas for program 2, $7k-1$ bits of information remains uncertain for all cases.

So, in the worst case, program 1 leaks much more information than program 2, but since Shannon entropy focuses on average case, it concludes that program 2 leaks more information.
Min entropy for programs 1 and 2

Since the input is a uniformly distributed 8k-bit integer value, for both programs 1 and 2 we have:

$$H_\infty (S) = 8k$$
Min entropy for programs 1 and 2

Since secret is uniformly distributed and the programs are deterministic, for both programs 1 and 2, the information leakage is

$$\log_2 |\mathcal{O}|$$

For program 2:

$$\log_2 |\mathcal{O}| = k + 1$$

For program 1:

$$\log_2 |\mathcal{O}| = 8k - 3$$
Min-entropy for programs 1 and 2

According to min-entropy, amount of information leaked and remaining uncertainty for programs 1 and 2 are:

Program 1:  
leakage: 8k - 3  
\( H_\infty(H|O) = 3 \)

Program 2:  
leakage: k + 1  
\( H_\infty(H|O) = 7k - 1 \)
Min-entropy for programs 1 and 2

Since min-entropy focuses on the worst-case probability that an adversary could guess the value of the secret correctly in one try, the leakage computes for program 1 increases significantly.

According the min-entropy, program 1 leaks much more information than program 2.