CS 5602 Introduction to Cryptography
Lecture 21
Entropy

George Markowsky
Computer Science Department
Missouri University of Science & Technology

1

Perfect Security

- If every message we send requires a key as long as the message, and we never encrypt two messages with the same key, then encryption will not be very useful in everyday applications such as Internet transactions.
- This is because getting the key from one person to another will be an impossible task.
- After all one cannot encrypt it since that would require another key.
- This problem is called the key distribution problem.

2

Perfect Security

- To simplify the key distribution problem we need to turn from perfectly secure encryption algorithms to ones which are, hopefully, computationally secure.
- This is the goal of modern cryptographers, where one aims to build systems such that
  - one key can be used many times,
  - one small key can encrypt a long message.
- Such systems will not be unconditionally secure, by Shannon's Theorem, and so must be at best only computationally secure.

3

Entropy

DEFINITION 5.5 (Entropy). *Let X be a random variable which takes on a finite set of values x_i, with $1 \leq i \leq n$, and has probability distribution $p_i = p(X = x_i)$. The entropy of X is defined to be*

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i.$$

We make the convention that if $p_i = 0$ then $p_i \log_2 p_i = 0$.

Let us return to our *Yes* or *No* question above and show that this definition of entropy coincides with our intuition. Recall, X is the answer to some question with responses *Yes* or *No*. If you know I will always say *Yes* then

$$p_1 = 1 \text{ and } p_2 = 0.$$

We compute

$$H(X) = -1 \cdot \log_2 1 - 0 \cdot \log_2 0 = 0.$$

Hence, my answer reveals no information to you.

If you have no idea what I will say and I reply *Yes* with equal probability to replying *No* then

$$p_1 = p_2 = 1/2.$$

We now compute

$$H(X) = -\frac{\log_2 \frac{1}{2}}{2} - \frac{\log_2 \frac{1}{2}}{2} = 1.$$

Hence, my answer reveals one bit of information to you.

4

Entropy

- We always have H(X) ≥ 0.
- The only way to obtain H(X) = 0 is if for some i we have $p_i = 1$ and $p_j = 0$ when $i \neq j$.
- If $p_i = 1/n$ for all i then $H(X) = \log_2 n$.
- Another way of looking at entropy is that it measures by how much one can compress the information.
- If I send a single ASCII character to signal Yes or No, for example I could simply send Y or N, I am actually sending 8 bits of data, but I am only sending one bit of information.
- If I wanted to I could compress the data down to 1/8th of its original size.
- Hence, naively if a message of length n can be compressed to a proportion ϵ of its original size then it contains $\epsilon \cdot n$ bits of information in it.

5

Huffman Codes

- Traditionally, the most common character schemes have all characters the same length
- For example, all characters in ASCII are 8-bits long
- UTF-8 is now a commonly used variable length code – characters are represented by 8, 16, 24 or 32 bits
- The idea of a Huffman Code goes back to at least Morse Code and uses the shortest sequences for the most common characters!

6

Example

- Suppose we have 6 different characters represented by 3 bits each
- If a file contains 120 characters, we would need 360 bits
- Can we do better? How much better?
- Answer: depends on the distribution!

7

Example

	a	b	c	d	e	f
Frequency	55	18	17	16	9	5
Code 1	000	001	010	011	100	101
Code 2	0	100	101	110	1110	1111

len(120 characters) using Code 1 = 120 × 3 = 360 bits

len(120 characters) using Code 2 = 55× 1 + 51 × 3 + 14 × 4 = 264 bits

Code 2 reduces the length by 96/360 ≈ 27%

If the "a" appeared 120 times, we could represent it with a 1-bit code, so the total string could be represented by 120 bits, a 67% reduction

8

Prefix Codes

- A **prefix code** is a code such that the representation of any code word is not a prefix of the representation of any other code word
- A code in which all code words have the same length is a prefix code
- Optimal codes can always be made into prefix codes
- Prefix codes are always easy to decode and can be decoded by a finite state automaton!
- 1000110111101011110
- badface

	a	b	c	d	e	f
Code 2	0	100	101	110	1110	1111

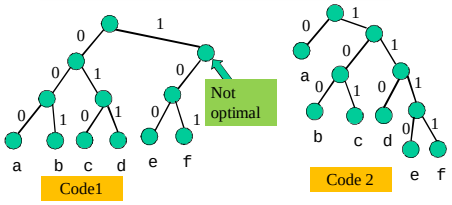
- Note that a code is optimal only for a particular set of frequencies – different frequencies produce different codes

9

Codes and Trees

- Associated with each binary code is a binary tree

	a	b	c	d	e	f
Code 1	000	001	010	011	100	101
Code 2	0	100	101	110	1110	1111

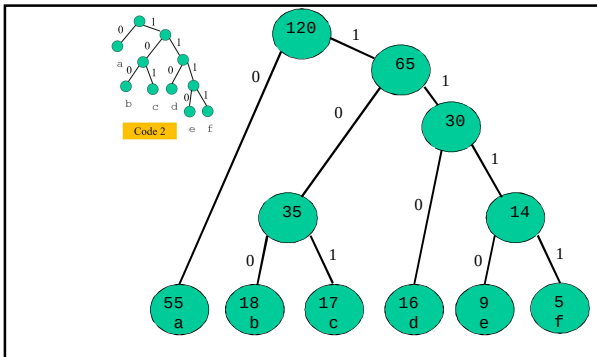


10

Optimal Encodings

- The cost of an encoding = Σ_{chars}depth(c)×freq(c)
- So the problem reduces to the following
- We have nodes with weights w_1, w_2, …, w_n and want to find a tree that has these weights as leaves so that $\Sigma_i$depth(w_i)×w_i is minimized
- **Algorithm:** pick two nodes with smallest weight and create a new parent node with a weight equal to the weight of its children, and repeat

11



12

Huffman Trees & Entropy

Items	Sums	a	b	c	d	e	f
counts	120	55	18	17	16	9	5
probabilities	1	0.458333	0.15	0.141667	0.133333	0.075	0.041667
$\log_2(p_i)$		-1.12553	-2.73697	-2.81943	-2.90689	-3.73697	-4.58496
$p_i \log_2(p_i)$	-2.18473	-0.51587	-0.41054	-0.39942	-0.38759	-0.28027	-0.19104
Entropy	2.18473						
count*entropy	262.1676						

len(120 characters) using Code 1 = 120 × 3 = 360 bits

len(120 characters) using Code 2 = 55× 1 + 51 × 3 + 14 × 4 = 264 bits

13

Baby Cryptosystem

Let us return to our baby cryptosystem considered in the previous section. Recall we had the probability spaces

$$\mathbb{P} = \{a, b, c, d\},\ \mathbb{K} = \{k_1, k_2, k_3\} \text{ and } \mathbb{C} = \{1, 2, 3, 4\},$$

with the associated probabilities:

- $p(P = a) = 0.25$, $p(P = b) = p(P = d) = 0.3$ and $p(P = c) = 0.15$,
- $p(K = k_1) = p(K = k_3) = 0.25$ and $p(K = k_2) = 0.5$,
- $p(C = 1) = p(C = 2) = p(C = 3) = 0.2625$ and $p(C = 4) = 0.2125$.

We can then calculate the relevant entropies as:

$$H(P) \approx 1.9527,$$
$$H(K) \approx 1.5,$$
$$H(C) \approx 1.9944.$$

Hence the ciphertext 'leaks' about two bits of information about the key and plaintext, since that is how much information is contained in a single ciphertext. Later we will calculate how much of this information is about the key and how much about the plaintext.

14

Upper Bound for Entropy

THEOREM 5.6 (Jensen's Inequality). *Suppose*

$$\sum_{i=1}^{n} a_i = 1$$

with $a_i > 0$ for $1 \le i \le n$. Then, for $x_i > 0$,

$$\sum_{i=1}^{n} a_i \log_2 x_i \le \log_2 \left(\sum_{i=1}^{n} a_i x_i \right).$$

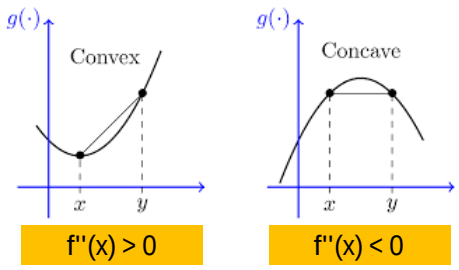
With equality occurring if and only if $x_1 = x_2 = \ldots = x_n$.

15

Proof of Upper Bound

$$\begin{aligned} H(X) &= -\sum_{i=1}^{n} p_i \log_2 p_i \\ &= \sum_{i=1}^{n} p_i \log_2 \frac{1}{p_i} \\ &\le \log_2 \left(\sum_{i=1}^{n} \left(p_i \times \frac{1}{p_i} \right) \right) \text{ by Jensen's inequality} \\ &= \log_2 n. \end{aligned}$$

16



f''(x) > 0 f''(x) < 0

17

THEOREM

Jensen's Inequality

Let a real-valued function f be convex on the interval I. Let $x_1, \ldots, x_n \in I$ and $\omega_1, \ldots, \omega_n \ge 0$. Then we have

$$\frac{\omega_1 f(x_1) + \omega_2 f(x_2) + \cdots + \omega_n f(x_n)}{\omega_1 + \omega_2 + \cdots + \omega_n} \ge f\left(\frac{\omega_1 x_1 + \omega_2 x_2 + \cdots + \omega_n x_n}{\omega_1 + \omega_2 + \cdots + \omega_n} \right).$$

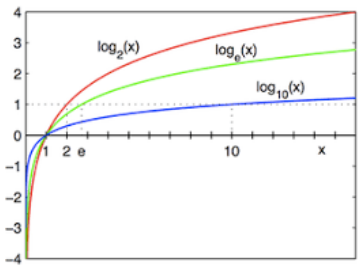
If f is concave, the direction of inequality is flipped.

In particular if we take weights $\omega_1 = \omega_2 = \cdots = \omega_n = 1$, we get the inequality

$$\frac{f(x_1) + f(x_2) + \cdots + f(x_n)}{n} \ge f\left(\frac{x_1 + x_2 + \cdots + x_n}{n} \right). \square$$

Note that f(x) is a convex function iff –f(x) is a concave function.
For concave functions just flip ≥ into ≤

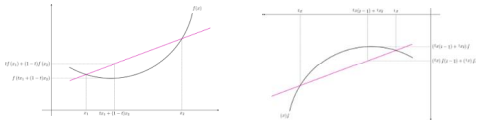
18



$\log_a n = \log_b n \times \log_a b$ so all logs are multiples of each other
logs are all concave functions

19

Jensen's Inequality



20

Entropy

THEOREM 5.7. *If X is a random variable which takes n possible values then*

$$0 \leq H(X) \leq \log_2 n.$$

The lower bound is obtained if one value occurs with probability one, the upper bound is obtained if all values are equally likely.

$$\begin{aligned} H(X) &= -\sum_{i=1}^{n} p_i \log_2 p_i \\ &= \sum_{i=1}^{n} p_i \log_2 \frac{1}{p_i} \\ &\leq \log_2 \left(\sum_{i=1}^{n} \left(p_i \times \frac{1}{p_i} \right) \right) \text{ by Jensen's inequality} \\ &= \log_2 n. \end{aligned}$$

21

Joint Entropy

The joint entropy is then obviously defined as

$$H(X,Y) = -\sum_{i=1}^{n}\sum_{j=1}^{m} r_{i,j} \log_2 r_{i,j}.$$

You should think of the joint entropy $H(X,Y)$ as the total amount of information contained in one observation of $(x,y) \in X \times Y$. We then obtain the inequality

$$H(X,Y) \leq H(X) + H(Y)$$

with equality if and only if X and Y are independent. We leave the proof of this as an exercise.

22

Conditional Entropy

Let X and Y be two random variables. Recall we defined the conditional probability distribution as

$$p(X = x|Y = y) = \text{ Probability that } X = x \text{ given } Y = y.$$

The entropy of X given an observation of $Y = y$ is then defined in the obvious way by

$$H(X|y) = -\sum_{x} p(X = x|Y = y) \log_2 p(X = x|Y = y).$$

Given this we define the conditional entropy of X given Y as

$$\begin{aligned} H(X|Y) &= \sum_{y} p(Y = y) H(X|y) \\ &= -\sum_{x}\sum_{y} p(Y = y) p(X = x|Y = y) \log_2 p(X = x|Y = y). \end{aligned}$$

This is the amount of uncertainty about X that is left after revealing a value of Y. The conditional and joint entropy are linked by the following formula

$$H(X,Y) = H(Y) + H(X|Y)$$

and we have the following upper bound

$$H(X|Y) \leq H(X)$$

with equality if and only if X and Y are independent. Again we leave the proof of these statements as an exercise.

23

Entropy & Cryptography

- $H(P|K,C) = 0$:
  If you know the ciphertext and the key then you know the plaintext. This must hold since otherwise decryption will not work correctly.
- $H(C|P,K) = 0$:
  If you know the plaintext and the key then you know the ciphertext. This holds for all ciphers we have seen so far, and holds for all the symmetric ciphers we shall see in later chapters. However, for modern public key encryption schemes we do not have this last property when they are used correctly.

24

Entropy & Cryptography

- H(K, P, C) = H(P, K) + H(C|P, K) as H(X, Y) = H(Y) + H(X|Y)
- = H(P, K) as H(C|P, K) = 0
- = H(K) + H(P) as K and P are independent
- Alternatively,
- H(K, P, C) = H(K, C) + H(P |K, C) as H(X, Y) = H(Y) + H(X|Y)
- = H(K, C) as H(P|K, C) = 0
- Hence,
- H(K, C) = H(K) + H(P)

25

Key Equivocation

- This last equality is important since it is related to the conditional entropy H(K|C), which is called the **key equivocation**.
- The key equivocation is the amount of uncertainty about the key left after one ciphertext is revealed.
- Recall that our goal is to determine the key given the ciphertext.
- Putting two of our prior equalities together we find
- H(K|C) = H(K, C) − H(C) = H(K) + H(P) − H(C)
- H(P) ≈ 1.9527, H(K) ≈ 1.5, H(C) ≈ 1.9944.
- H(K|C) ≈ 1.9527 + 1.5 − 1.9944 ≈ 1.4583
- There are only 1.5 bits of uncertainty about the key to start with, one ciphertext leaves us with 1.4593 bits of uncertainty.
- Hence, 1.5−1.4593 = 0.042 bits of information about the key are revealed by a single ciphertext.

26

Spurious Keys

- In our baby example above, information about the key is leaked by an individual ciphertext, since knowing the ciphertext rules out a certain subset of the keys.
- Of the remaining possible keys, only one is correct.
- The remaining possible, but incorrect, keys are called the **spurious keys**.
- Consider the (unmodified) shift cipher, i.e. where the same key is used for each letter.
- Suppose the ciphertext is WNAJW, and suppose we know that the plaintext is an English word.
- The only 'meaningful' plaintexts are RIVER and ARENA, which correspond to the two possible keys F and W .
- One of these keys is the correct one and one is **spurious**.

27

Analyzing English

- English is an example of a *natural language* to distinguish it from bitstreams used for computer communications such as transmitting images
- We first wish to define the entropy (or information) per letter H_L of a natural language such as English.
- Note, a random string of alphabetic characters would have entropy

$$\log_2 26 \approx 4.70.$$

- So for English we have $H_L \leq 4.70$.

28

Analyzing English

- If we let P denote the random variable of letters in the English language then we have
- p(P = a) = 0.082, . . . , p(P = e) = 0.127, . . . , p(P = z) = 0.001
- We can then compute

$$H_L \leq H(P) \approx 4.14$$

- But this is a gross overestimate, since letters are not independent.
- For example Q is always followed by U and the bigram TH is likely to be very common.
- One would suspect that a better statistic for the amount of entropy per letter could be obtained by looking at the distribution of bigrams.

29

Analyzing English

- Hence, we let P^2 denote the random variable of bigrams.
- If we let p(P = i, P′ = j) denote the random variable which is assigned the probability that the bigram 'ij' appears, then we define

$$H(P^2) = -\sum_{i,j} p(P = i, P' = j) \log p(P = i, P' = j)$$

- A number of people have computed values of $H(P^2)$ and it is commonly accepted to be given by

$$H(P^2) \approx 7.12.$$

- We want the entropy per letter so we compute

$$H_L \leq H(P^2)/2 \approx 3.56.$$

30

Analyzing English

- But again this is an overestimate, since we have not taken into account that the most common trigram is THE.
- Hence, we can also look at P^3 and compute $H(P^3)/3$. This will also be an
- overestimate and so on...
- This leads us to the following definition.

DEFINITION 5.8. *The entropy of the natural language L is defined to be*

$$H_L = \lim_{n\longrightarrow\infty} \frac{H(P^n)}{n}.$$

31

Analyzing English

- The exact value of H_L is hard to compute exactly but we can approximate it.
- In fact one has, by experiment, that for English

$$1.0 \leq \mathrm{HL} \leq 1.5$$

- So each letter in English
  - requires 5 bits of data to represent it
  - only gives at most 1.5 bits of information
- This shows that English contains a high degree of redundancy
- One can see this from the following,
- which you can still hopefully read (just) even though I have deleted two out of every four letters
- On** up** a t**e t**re **s a **rl **ll** Sn** Wh**e.

32

Redundancy

The *redundancy* of a language is defined by

$$R_L = 1 - \frac{H_L}{\log_2 \#\mathbb{P}}.$$

If we take $H_L \approx 1.25$ then the redundancy of English is

$$R_L \approx 1 - \frac{1.25}{\log_2 26} = 0.75.$$

So this means that we should be able to compress an English text file of around 10 MB down to 2.5 MB.

33

Spurious Keys Again

We now return to a general cipher and suppose $c \in \mathbb{C}^n$, i.e. c is a ciphertext consisting of n characters. We define $\mathbb{K}(c)$ to be the set of keys which produce a 'meaningful' decryption of c. Then, clearly $\#\mathbb{K}(c) - 1$ is the number of spurious keys given c. The average number of spurious keys is defined to be $\bar{s}_n$, where

$$\begin{aligned}\bar{s}_n &= \sum_{c\in\mathbb{C}^n} p(C=c)\,(\#\mathbb{K}(c)-1)\\ &= \sum_{c\in\mathbb{C}^n} p(C=c)\#\mathbb{K}(c) - \sum_{c\in\mathbb{C}^n} p(C=c)\\ &= \left(\sum_{c\in\mathbb{C}^n} p(C=c)\#\mathbb{K}(c)\right) - 1.\end{aligned}$$

34

Spurious Keys Again

Now if n is sufficiently large and $\#\mathbb{P} = \#\mathbb{C}$ we obtain

$$\begin{aligned}\log_2(\bar{s}_n+1) &= \log_2 \sum_{c\in\mathbb{C}^n} p(C=c)\#\mathbb{K}(c)\\ &\geq \sum_{c\in\mathbb{C}^n} p(C=c)\log_2\#\mathbb{K}(c) && \text{Jensen's inequality}\\ &\geq \sum_{c\in\mathbb{C}^n} p(C=c)H(K|c)\\ &= H(K|C^n) && \text{By definition}\\ &= H(K)+H(P^n)-H(C^n) && \text{Equation (8)}\\ &\approx H(K)+nH_L-H(C^n) && \text{If } n \text{ is very large}\\ &= H(K)-H(C^n)\\ &\quad + n(1-R_L)\log_2\#\mathbb{P} && \text{By definition of } R_L\\ &\geq H(K)-n\log_2\#\mathbb{C}\\ &\quad + n(1-R_L)\log_2\#\mathbb{P} && \text{As } H(C^n)\leq n\log_2\#\mathbb{C}\\ &= H(K)-nR_L\log_2\#\mathbb{P} && \text{As } \#\mathbb{P}=\#\mathbb{C}.\end{aligned}$$

So, if n is sufficiently large and $\#\mathbb{P} = \#\mathbb{C}$ then

$$\bar{s}_n \geq \frac{\#\mathbb{K}}{\#\mathbb{P}^{nR_L}} - 1.$$

35

Unicity Distance

So, if n is sufficiently large and $\#\mathbb{P} = \#\mathbb{C}$ then

$$\bar{s}_n \geq \frac{\#\mathbb{K}}{\#\mathbb{P}^{nR_L}} - 1.$$

As an attacker we would like the number of spurious keys to become zero, and it is clear that as we take longer and longer ciphertexts then the number of spurious keys must go down.

The unicity distance n_0 of a cipher is the value of n for which the expected number of spurious keys becomes zero. In other words this is the average amount of ciphertext needed before an attacker can determine the key, assuming the attacker has infinite computing power. For a perfect cipher we have $n_0 = \infty$, but for other ciphers the value of n_0 can be alarmingly small. We can obtain an estimate of n_0 by setting $\bar{s}_n = 0$ in

$$\bar{s}_n \geq \frac{\#\mathbb{K}}{\#\mathbb{P}^{nR_L}} - 1$$

to obtain

$$n_0 \approx \frac{\log_2 \#\mathbb{K}}{R_L \log_2 \#\mathbb{P}}.$$

In the substitution cipher we have

$$\#\mathbb{P} = 26,$$
$$\#\mathbb{K} = 26! \approx 4 \cdot 10^{26}$$

and using our value of $R_L = 0.75$ for English we can approximate the unicity distance as

$$n_0 \approx \frac{88.4}{0.75 \times 4.7} \approx 25.$$

So we require on average only 25 ciphertext characters before we can break the substitution cipher, again assuming infinite computing power. In any case after 25 characters we expect a unique valid decryption.

36

Unicity Distance

Now assume we have a modern cipher which encrypts bit strings using keys of bit length l, we have

$$\#\mathbb{P} = 2,$$
$$\#\mathbb{K} = 2^l.$$

Again we assume $R_L - 0.75$, which is an underestimate since we now need to encode English into a computer communications media such as ASCII. Then the unicity distance is

$$n_0 \approx \frac{l}{0.75} = \frac{4l}{3}.$$

Now assume instead of transmitting the plain ASCII we compress it first. If we assume a perfect compression algorithm then the plaintext will have no redundancy and so $R_L \approx 0$. In which case the unicity distance is

$$n_0 \approx \frac{l}{0} = \infty.$$

37

Types of Attacks & Redundancy

- So you may ask if modern ciphers encrypt plaintexts with no redundancy?
- The answer is no, even if one compresses the data, a modern cipher often adds some redundancy to the plaintext before encryption.
- The reason is that we have only considered passive attacks, i.e. an attacker has been only allowed to examine ciphertexts and from these ciphertexts the attacker's goal is to determine the key.
- There are other types of attack called **active attacks**, in these an attacker is allowed to generate plaintexts or ciphertexts of her choosing and ask the key holder to encrypt or decrypt them, the two variants being called a **chosen plaintext attack** and a **chosen ciphertext attack** respectively.

38

Public Key Cryptosystems

- In public key systems that we shall see later, chosen plaintexts attacks cannot be stopped since anyone is allowed to encrypt anything.
- We would however, like to stop chosen ciphertext attacks.
- The current wisdom for public key algorithms is to make the cipher add some redundancy to the plaintext before it is encrypted.
- In that way it is hard for an attacker to produce a ciphertext which has a valid decryption.
- The philosophy is that it is then hard for an attacker to mount a chosen ciphertext attack, since it will be hard for an attacker to choose a valid ciphertext for a decryption query.
- This is similar to having "salt" in a password encryption system

39

Summary

- A cryptographic system for which knowing the ciphertext reveals no more information than if you did not know the ciphertext is called a perfectly secure system.
- Perfectly secure systems exist, but they require keys as long as the message and a different key to be used with each new encryption.
- Hence, perfectly secure systems are not very practical.
- • Information and uncertainty are essentially the same thing.
- An attacker really wants, given the ciphertext, to determine some information about the plaintext.
- The amount of uncertainty in a random variable is measured by its entropy.

40

Summary

- The equation H(K|C) = H(K)+H(P)−H(C) allows us to estimate how much uncertainty remains about the key after one observes a single ciphertext.
- The natural redundancy of English means that a naive cipher does not need to produce a lot of ciphertext before the underlying plaintext can be discovered.

41