Shannon's Theory of Secure Communication

CS 6750 Lecture 2

September 17, 2009

Riccardo Pucella

Introduction

- Last time, we have seen various cryptosystems, and some cryptanalyses
- How do you determine the security of a cryptosystem?
- Some reasonable ideas:
  - Unconditional Security: Cryptosystem cannot be broken even with infinite computation power
  - Computational Security: best alg takes a long time
    - No one knows how to get that (impossible?)
    - Possible against specific attacks (brute-force search)
  - Provable Security: reduce the security of a cryptosystem to a problem believed (or known) to be hard

Review of Probability Theory

- Security generally expressed in terms of probability
  - An attacker can always guess the key!
  - True for any cryptosystem, and unavoidable
- We focus on discrete probabilities

Probability Distributions

- Probability space: (Ω, Pr)
  - Ω, the sample space, is a finite set of possible states (possible outcomes)
  - Pr is a function $Pow(\Omega) \to [0,1]$ such that
    - $Pr(\Omega) = 1$
    - $Pr(\emptyset) = 0$
    - $Pr(A \cup B) = Pr(A) + Pr(B)$ when $A \cap B = \emptyset$ [additivity]
  - Pr is called a probability distribution, a probability measure, or just a probability
- Additivity implies that Pr is determined by $Pr(\{a\})\ \forall a$

Examples

- Single die:
  - $\Omega = \{1,2,3,4,5,6\}$
  - $Pr(\{4\}) = 1/6$
  - $Pr(\{1,3,5\}) = 3/6 = 1/2$

- Pair of dice:
  - $\Omega = \{(1,1),(1,2),(1,3),(1,4),\ldots,(6,5),(6,6)\}$
  - $Pr(\{(1,1)\}) = 1/36$
  - $Pr(\{(1,a) : a=1,2,3,4\}) = 4/36 = 1/9$

Joint Probabilities

- Suppose (Ω_1, Pr_1) is a probability space
- Suppose (Ω_2, Pr_2) is a probability space
- Can create the **joint probability space** ($\Omega_1 \times \Omega_2$,Pr) by taking:
  - $Pr(\{a,b\}) = Pr_1(\{a\})Pr_2(\{b\})$
  - Extend by additivity

Conditional Probability

- $\Pr(A \mid B) = \Pr(A \cap B) / \Pr(B)$
  - Only defined when $\Pr(B) > 0$
- More easily understood with a picture...

Bayes' Theorem: $\Pr(B \mid A) = \Pr(A \mid B) \Pr(B) / \Pr(A)$

Random Variables

- A random variable is a function from states to some set of values
- Given probability space and a random variable X, the probability that the random variable X takes value x is:

$$Pr(\{w : X(w)=x\})$$

- This is often written Pr(X=x) or Pr[x]   (YUCK)

- The probability space is often left implicit

- Conditional probabilities:
  $Pr(X=x \mid Y=y) = Pr(\{w : X(w)=x\} \mid \{w : Y(w)=y\})$

- X and Y are independent if $P(X=x \cap Y=y) = Pr(X=x)\, Pr(Y=y)\ \forall x,y$

Examples

- Consider the single die example again:
  - $\Omega = \{1,2,3,4,5,6\}$
  - Pr(a) = 1/6 for a=1,2,3,4,5,6

- Take Even(i) = true if i is even, and false otherwise
  - Pr(Even=true) = Pr({w : Even(w)=true})
    = Pr({2,4,6})
    = 1/2

- Take Double(i) = 2i
  - Pr(Double=10) = Pr({w : Double(w)=10})
    = Pr({5})
    = 1/6

Application to Cryptography

- Suppose a probability space (Ω, Pr) with:
  - Random variable K (i.e., the key)
  - Random variable P (i.e., the plaintext)
  - Random variable C (i.e., the ciphertext)
  - K and P are independent random variables
    - Simple example: states are (key, plaintext) pairs
- Key probability is $Pr(K=k)$
- Plaintext probability is $Pr(P=x)$
- Ciphertext probability is $Pr(C=y)$

Ciphertext Probability

- Can derive a probability over ciphertexts:

$$Pr(C = y) = \sum_{x, k \bullet e_k(x) = y} Pr(P = x) Pr(K = k)$$

- Can compute conditional probabilities:

$$Pr(C = y \cap P = x) = Pr(P = x) \sum_{k \bullet e_k(x) = y} Pr(K = k)$$

$$Pr(C = y \mid P = x) = \sum_{k \bullet e_k(x) = y} Pr(K = k)$$

$$Pr(P = x \mid C = y) = \frac{Pr(P = x) \sum_{k \bullet e_k(x) = y} Pr(K = k)}{\sum_{x', k \bullet e_k(x') = y} Pr(P = x') Pr(K = k)}$$

Perfect Secrecy

- We say a cryptosystem has perfect secrecy if

$$Pr(P=x \mid C=y) = Pr(P=x) \quad \text{for all } x,y$$

  - The probability that the plaintext is x given that you have observed ciphertext y is the same as the probability that the plaintext is x (without seeing the ciphertext)
  - Depends on key probability and plaintext probability

Characterizing Perfect Secrecy

Theorem: The shift cipher, where all keys have probability 1/26, has perfect secrecy if we use the key only once, for any plaintext probability.

- Can we characterize those cryptosystems with perfect secrecy?

Theorem: Let (P,C,K,E,D) be a cryptosystem with |K| = |P| = |C|. This cryptosystem has perfect secrecy if and only if all keys have the same probability 1/|K| and

$$\forall x \in P \ \forall y \in C \ \exists k \in K \bullet e_k(x) = y$$

Vernam Cipher

- Also know as the one-time pad
- $P = C = K = (Z_2)^n$
  - Strings of bits of length n
- If $K=(k_1, ..., k_n)$:
  - $e_K (x_1, ..., x_n) = (x_1+k_1 \pmod 2), ..., x_n+k_n \pmod 2))$
  - $d_K (x_1, ..., x_n) = (x_1-k_1 \pmod 2), ..., x_n-k_n \pmod 2))$
- To encrypt a string of length N, choose a one-time pad of length N

Conclusions

- If ciphertexts are short (same length as key), can get perfect security
  - Approach still used for very sensitive data (embassies, military, etc)
- But keys get very long for long messages
- And there is the whole key distribution problem
- Modern cryptosystems: one key used to encrypt long plaintext (by breaking it into pieces)
  - We will see more of these next time
- Need to be able to reason about reusing keys

10 minutes break

A Detour: Entropy

- Entropy: measure of uncertainty (in bits) introduced by Shannon in 1948
  - Foundation of Information Theory

- Intuition
  - Suppose a random variable that takes value {1,...,n} with some nonzero probability
  - Consider the string of values generated by that probability distribution
  - What is the most efficient way (in number of bits) to encode every value to minimize how many bits it take to encode a random string?

  - Example: {A,...,H} where H is much more likely than others

Definition of Entropy

- Let random variable take values in finite set V

$$H(X) = -\sum_{v \in V} Pr(X = v) \log_2 Pr(X = v)$$

- Weighted average of $-\log_2 Pr\ (X=v)$

Theorem: Suppose X is a random variable taking n values with nonzero probability, then

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

- When do we have equality?

Huffman Encoding

Algorithm to get a {0,1} encoding that takes less than H(X) bits on average

1. Start with a table of letter probabilities
2. Create a list of trees, initially all trees with only a letter and associated probability
3. Iteratively:
   a. Pick the two trees T_1, T_2 with smallest probabilities from the list
   b. Create a small tree with edge 0 leading to T_1 and edge 1 leading to T_2
   c. Add that tree back to the list, with probability the sum of the original probabilities
4. Stop when you get a single tree giving the encoding

Conditional Entropy

- Let X and Y be random variables

- Fix a value y of Y
- Define the random variable X|y such that
  Pr (X|y = x) = Pr (X=x | Y=y)

$$H(X \mid y) = -\sum_{v \in V} Pr(X = v \mid Y = y) \log_2 Pr(X = v \mid Y = y)$$

- Conditional entropy, written H(X|Y):

$$H(X \mid Y) = \sum_{y} Pr(Y = y) H(X \mid y)$$

- Intuition: average uncertainty about X that remains after observing Y

Application to Cryptography

- Key equivocation H(K | C): amount of uncertainty of the key that remains after observing the ciphertext

Theorem: $H(K \mid C) = H(K) + H(P) - H(C)$

- A spurious key is a possible key, but incorrect
  - E.g., shift cipher, with ciphertext WNAJW
  - Possible keys: k=5 (RIVER) or k=22 (ARENA)
- Many spurious keys ⟶ Good!

How Many Spurious Keys?

- Question: how long of a message can we permit before the number of spurious keys is 0?
  - That is, before the only key that is possible is the right one?
- This depends on the underlying language in which plaintexts are taken
  - Cf: cryptanalysis of substitution cipher, where we took advantage that not all letters have equal probability in English messages

Entropy of a Language

- H_L = number of information bits per letter in language L
  - For English, based on probabilities of letters, a first approximation to H_L is H(P) = 4.19
  - For pairs of letters? $H(P^2)/2$
  - For triplets of letters? $H(P^3)/3$
- Entropy of L:

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

- Redundancy of L:

$$R_L = 1 - \frac{H_L}{\log_2 |P|}$$

  - Amount of "wasted space" in text in language L

Unicity Distance

Theorem: Suppose (P,C,K,E,D) is a cryptosystem with |C| = |P| and keys are chosen equiprobably, and let L be the underlying language. Given a ciphertext of length n (sufficiently large), the expected number of spurious keys s_n satisfies

$$s_n \geq \frac{|K|}{|P|^{nR_L}} - 1$$

- The unicity distance of a cryptosystem is the value n_0 after which the number expected number of spurious keys is 0.
  - Average amount of ciphertext required for an adversary to be able to compute the key (given enough time)

- Substitution cipher: n_0 = 25
  - So have a chance to recover the key if encrypted message is longer than 25 characters