CSE331: Introduction to Networks and Security

Lecture 21
Fall 2006
Announcements

• Homework 2 has been assigned:
  – **NEW DUE DATE**
  – It's now due on Friday, November 3rd.

• Midterm 2 is Friday, November 10th
  – **NEW DATE**
  – It covers just the material since Midterm 1
Variance: Measure of “roughness”

\[ \text{Var} = \sum (\text{prob}(\alpha) - 1/26)^2 \]

\[ \alpha = a \]

\[ = \ldots \]

\[ \alpha = z \]

\[ = \left( \sum \text{prob}(\alpha)^2 \right) - 1/26 \]

\[ \alpha = a \]
Estimate Variance From Frequency

• $\text{prob}(\alpha)^2$ is probability that any two characters drawn from the text will be $\alpha$
• Suppose there are $n$ ciphertext letters total
• Suppose $\text{freq}(\alpha)$ is the frequency of $\alpha$
• What is likelihood of picking $\alpha$ twice at random?
  – $\text{freq}(\alpha)$ ways of picking the first $\alpha$
  – $(\text{freq}(\alpha) - 1)$ ways of picking the second $\alpha$
  – But this counts twice because $(\alpha, \beta) = (\beta, \alpha)$
  – So $\frac{\text{freq}(\alpha) \times (\text{freq}(\alpha) - 1)}{2}$
Index of Coincidence

- But there are \( \frac{n \times (n-1)}{2} \) pairs of letters
- \( \ldots \) so \( \text{prob}(\alpha)^2 \) is roughly \( \frac{\text{freq}(\alpha) \times (\text{freq}(\alpha) - 1)}{n \times (n-1)} \)

- Index of coincidence: approximates variance from frequencies

\[
\text{IC} = \sum_{\alpha = a}^{Z} \frac{\text{freq}(\alpha) \times (\text{freq}(\alpha) - 1)}{n \times (n-1)}
\]
What’s it good for?

- If the distribution is flat, then $IC \approx 0.0384$
- If the distribution is like English, then $IC \approx 0.068$
- Can verify key length:

```
keylen  1   2   3   4   5   many
     IC  0.068 0.052 0.047 0.044 0.044  ... 0.038
```
Summary: Cracking Polyalphabets

- Use Kasiski method to guess likely key lengths
- Compute the Index of Coincidence to verify key length $k$
- $k$-Slices should have similar IC to English

- Note: digram information harder to use for polyalphabetic ciphers…
  - May want to consider “split digrams”
  - Example: if tion is a common sequence $k=2$ then “t?o” and “i?n” are likely “split digrams”
Perfect Substitution Ciphers

- Choose a string of random bits the same length as the plaintext, XOR them to obtain the ciphertext.

- Perfect Secrecy
  - Probability that a given message is encoded in the ciphertext is unaltered by knowledge of the ciphertext
  - Proof: Give me any plaintext message and any ciphertext and I can construct a key that will produce the ciphertext from the plaintext.
One-time Pads

• Another name for Perfect Substitution
• Actually used by US agents in Russia
  – Physical pad of paper
  – List of random numbers
  – Pages were torn out and destroyed after use
• Vernam Cipher
  – Used by AT&T
  – Random sequence stored on punch tape
• Not practical for computer security…
Problems with “Perfect” Substitution

• Key is the same length as the plaintext
  – Sender and receiver must agree on the same random sequence
  – Not any easier to transmit key securely than to transmit plaintext securely

• Need to be able to generate many truly random bits
  – Pseudorandom numbers generated by an algorithm aren’t good enough for long messages

• Can’t reuse the key
Computational Security

• Perfect Ciphers are *unconditionally secure*
  – No amount of computation will help crack the cipher (i.e. the *only* strategy is brute force)

• In practice, strive for *computationally secure*
  – Given enough power, the attacker could crack the cipher (example: brute force attack)
  – But, an attacker with only *bounded resources* is extremely unlikely to crack it
  – Example: Assume attacker has only polynomial time, then encryption algorithm that can’t be inverted in less than exponential time is secure.
Kinds of Industrial Strength Crypto

- Shared Key Cryptography
- Public Key Cryptography
- Cryptographic Hashes

- All of these aim for computational security
  - Not all methods have been proved to be intractable to crack.
**Shared Key Cryptography**

- Sender & receiver use the same key
- Key must remain private
- Also called *symmetric* or *secret key* cryptography
- Often are *block-ciphers*
  - Process plaintext data in blocks
- Examples: DES, Triple-DES, Blowfish, Twofish, AES, Rijndael, …
Shared Key Notation

• Encryption algorithm
  \[ E : \text{key} \times \text{plain} \rightarrow \text{cipher} \]
  Notation: \( K\{\text{msg}\} = E(K, \text{msg}) \)

• Decryption algorithm
  \[ D : \text{key} \times \text{cipher} \rightarrow \text{plain} \]

• \( D \) inverts \( E \)
  \[ D(K, E(K, \text{msg})) = \text{msg} \]

• Use capital “\( K \)” for shared (secret) keys

• Sometimes \( E \) is the same algorithm as \( D \)
Secure Channel: Shared Keys

Alice

Bart

$K_{AB}\{\text{Hello!}\}$

$K_{AB}\{\text{Hi!}\}$
Data Encryption Standard (DES)

• Adopted as a standard in 1976
• Security analyzed by the National Security Agency (NSA)
• Key length is 56 bits
  – padded to 64 bits by using 8 parity bits
• Uses simple operators on (up to) 64 bit values
  – Simple to implement in software or hardware
• Input is processed in 64 bit blocks
• Based on a series of 16 *rounds*
  – Each cycle uses permutation & substitution to combine plaintext with the key
DES Encryption
One Round of DES (f of previous slide)

Expansion

Permuted choice of key

S-box

Permutation

R (32 BITS)

48 BITS

K (48 BITS)

32 BITS
Types of Permutations in DES

Permutation

Permuted Choice

Expansion Permutation
DES S-Boxes

• Substitution table
• 6 bits of input replaced by 4 bits of output
• Which substitution is applied depends on the input bits

• Implemented as a lookup table
  – 8 S-Boxes
  – Each S-Box has a table of 64 entries
  – Each entry specifies a 4-bit output
DES Decryption

• Use the same algorithm as encryption, but use $k_{16} \ldots k_1$ instead of $k_1 \ldots k_{16}$

• Proof that this works:
  – To obtain round $j$ from $j-1$:
    (1) $L_j = R_{j-1}$
    (2) $R_j = L_{j-1} \oplus f(R_{j-1}, k_j)$
  – Rewrite in terms of round $j-1$:
    (1) $R_{j-1} = L_j$
    (2) $L_{j-1} \oplus f(R_{j-1}, k_j) = R_j$
    \[ L_{j-1} \oplus f(R_{j-1}, k_j) \oplus f(R_{j-1}, k_j) = R_j \oplus f(R_{j-1}, k_j) \]
    \[ L_{j-1} = R_j \oplus f(R_{j-1}, k_j) \]
    \[ L_{j-1} = R_j \oplus f(L_j, k_j) \]