Benjamin Fenster
CS-266 (Ling)
19 December 2005

Notes – Introduction

I. Cryptography
  a. "Secret Writing"
  b. Definition
    i. A process / technique to convert intelligible data into an unintelligible form and get it back again
    ii. Usually one-to-one in size (data is the same size in its encrypted form) but is sometimes compressed
  c. Other Aspects / Services
    i. Integrity Checking: Guarantee that a message hasn't been altered
    ii. Authentication: Prove that someone is who s/he says s/he is.
  d. Caveats
    i. We cannot *prove* that a scheme is secure; we can only try to find weaknesses and be happy if we find none.
    ii. Algorithms are public, keys are private. Always assume the bad guy (who we'll call Trudy) has the algorithm.
  e. Computational Difficulty
    i. Needs to be efficient (can't have a combination lock with ten numbers – it'd be too hard to use)
    ii. We need to know what's fast on the hardware
    iii. We want to make sure that brute force cryptanalysis (trying until it looks like plaintext) is impractical
    iv. Any scheme can be broken, given enough time. Time is money, so given enough money you can crack any encryption scheme
      1. Encryption: $O(N + 1)$
      2. Brute Force Cryptanalysis: $O(2^{N+1})$
      3. Use larger keys to make it harder to crack
    v. Cryptanalysis Tools
      1. Parallel processing
      2. Distributed computing
      3. Special, purpose-built, powerful hardware
    vi. Secret Keys vs. Secret Algorithms
      1. Could make the algorithm secret too.
      2. That produces an additional hurdle for code breakers
      3. From a military standpoint, it avoids giving the enemy good ideas
      4. If many people know the algorithm though, it's hard to keep secret.
      5. If the algorithm is published, more people can review it and find flaws

II. Trivial Cryptographic Schemes
  a. Caesar Cipher
    i. Monoalphabetic cipher
    ii. Have an alphabet of 26 letters
    iii. Map each letter to some other (unique) letter in a one-to-one correspondence
    iv. Just swap each letter for its substitute
    v. Captain Midnight Secret Decoder Ring
      1. A special case
      2. Shift by some variable n
      3. So there are 26 possible codes.
  b. Cryptanalysis
    i. Methods depend on how much information you have
    ii. Ciphertext Only:
      1. Brute force
      2. Use known properties of the English Alphabet (e.g. how often E occurs)
    iii. Known Plaintext
      1. A spy (perhaps) provides the plaintext and ciphertext

    2. Can recreate the key by comparing them
  - iv. Chosen Plaintext
    1. Compose any paragraph / message you want (include all 26 letters)
    2. This makes it even easier to reconstruct the key
  - v. Alphabet Frequencies
    1. E 0.12
    2. TAOINSHR 0.06 to 0.09
    3. DL 0.04
    4. CUMWFGYPB 0.015 to 0.028
    5. VKJXQZ < 0.01
  - vi. Digrams
    1. TH, HE, IN, ER, AN, RE, ED
    2. ON, ES, SI, EN, AT, TO, NT
    3. HA, ND, OU, EA, NG, AS, OR
    4. TI, IS, ET, IT, AR, TE, SE, HI, OF
    5. In decreasing order of frequency
  - vii. Trigrams
    1. THE, ING, AND, HER, ERE, ENT
    2. THA, NTH, WAS, ETH, FOR, DTH
- c. Vigenere Cipher
  - i. Split the text into columns, shift each column by a different amount (based on a password)
  - ii. Find pairs / tuples of encrypted letters; find the difference n | (difference)
  - iii. 1,166,236,276,286 positions, so n = 5 probably
  - iv. Kasiski Test (18005)
  - v. Could pick a longer key
    1. Would still end up with some trigrams ("the") encrypting to the same thing
    2. As long as the message is long enough you can still do it.
  - vi. Cannot use digrams to break the code anymore, since letters in columns aren't consecutive.
  - vii. A Better Method
    1. Suppose $x = x_1x_2\ldots x_n$ is a string of n alphabetic characters.
    2. The index of coincidence of x, $I_C(x)$ is defined to be the probability that two random elements of x are identical
    3. Suppose the frequencies of A, B, …, Z are f_0, f_1, …, f_{25}
    4. $I_C(x)$ = SUM(0, 25, $f_i(f_i - 1)$) / $n(n - 1)$)
    5. Each term approximates to the frequency of the ith letter squared $(f_i / n)(f_i / n)$
    6. $I_C(x)$ approximately equals SUM(0, 25, P_i^2)
    7. As long as the message is English (even if it's been subjected to a substitution cipher), we can compare the index of coincidence.
    8. From English, $I_C(x) = 0.065$
    9. If x were random (each letter occurs with equal probability), $I_C(x) = 0.038$
    10. Our goal is to determine the length of the key.
    11. If $I_C(x)$ is close to 0.065 it's probably a substitution.
    12. If not, assume the length is 2, calculate $I_C(x)$ for each column (see if they are close to 0.065).
    13. If not, try 3 columns, then 4, et cetera.
  - viii. NB: Cryptography is based on large numbers. The human inability to deal with large numbers makes it secure

III. Types of Cryptography
- a. Characterized by the number of keys
- b. Hash Functions
  - i. No key (one-way)
  - ii. Represents an arbitrarily long message in a fixed-length key

iii. Requirements
    1. Easy to compute
    2. Infeasible to find two messages with the same hash
    3. Infeasible to find the message given the hash
iv. Password Hashing
    1. Old UNIX password files
    2. Compute the password with a 'salt', store hash and salt h(p + s)
    3. At login, generate the hash using the stored salt.
    4. The salt makes dictionary attacks harder, since encrypted passwords can't be generated in advance
v. Message Integrity
    1. Compute hash(m & p) with some password
    2. So if someone changes the message, s/he can't generate the new hash since s/he won't know the password

c. Secret Key
    i. One (shared) key
    ii. Must share a key a priori
    iii. Must be reversible (which may give the enemy a clue to how it works)
    iv. Want to encrypt, decrypt efficiently
    v. Don't want to tie up the hardware or build special hardware
    vi. One-to-One relation between cipher/plain text. That means the lengths will be about the same.
    vii. Substitutions, DES, IDEA
    viii. Also called Symmetric cryptography
    ix. Uses: Transmit messages, store files
    x. Strong Authentication
        1. Prove that you know the key without proving it.
        2. Generate a random challenge r_i.
        3. Other person encrypts it using the key.
        4. The first person decrypts it; it should match the challenge
        5. Now the second person sends a challenge: Challenge Response
    xi. Integrity Check
        1. Want to be sure the message is received
        2. Could use a checksum, but the message could be altered in such a way that the checksum is the same
        3. A cryptographic checksum ensures that Trudy can't alter the message

d. Public Key: Two keys (one private, one public)
    i. One-Way trap door
    ii. Exponentiation is easy, logarithms are hard.
    iii. So if we use exponentiation to encrypt something, it will be hard to decrypt using logarithms (for example).
    iv. Asymmetric (different keys on each end)
    v. "Publicly" invented in 1975, but there's strong evidence that the UK government invented it just after WWII and just didn't tell anybody
    vi. public (e) and private (d) keys for everybody
    vii. Why d and e?
        1. Public and private both start with p
        2. e means 'encryption'
        3. d means 'decryption'
    viii. d and e are related in some mathematical manner
    ix. This is much slower than secret key
    x. Data Transmission: Four keys used (d_a, e_a, d_b, e_b
    xi. Authentication
        1. Alice encrypts r_1 using e_b
        2. Bob decrypts r_1 using d_b
        3. Only Bob could do that, so it must be him

4. Again, now do a Challenge-Response the other way around

xii. Digital Signatures

1. Compute hash, encrypt with the private key
2. Decrypt with the public key
3. Now only the author can sign stuff, but anybody can read it.