CIS 551 / TCOM 401

Computer and Network Security

Spring 2008
Lecture 14

Announcements

- Reminder:
  - Project 2 is due *TOMORROW* Friday, March 7th at 11:59 pm
- Plan for today:
  - Finish discussing intrusion detection
  - Look at the background of cryptography
- After break (no class next week!):
  - Industrial strength crypto: DES / AES / RSA
  - Cryptographic protocols

Polymorphic Viruses/Worms

- Virus/worm writers know that signatures are the most effective way to detect such malicious code.

- Polymorphic viruses mutate themselves during replication to prevent detection
  - Virus should be capable of generating many different descendents
  - Simply embedding random numbers into virus code is not enough

Strategies for Polymorphic Viruses

- Change data:
  - Use different subject lines in e-mail
- Encrypt most of the virus with a random key
  - Virus first decrypts main body using random key
  - Jumps to the code it decrypted
  - When replicating, generate a new key and encrypt the main part of the replica
- Still possible to detect decryption portion of the virus using virus signatures
  - This part of the code remains unchanged
  - Worm writer could use a standard self-decompressing executable format (like ZIP executables) to cause confusion (many false positives)

Advanced Evasion Techniques

- Randomly modify the *code* of the virus/worm by:
  - Inserting no-op instructions: subtract 0, move value to itself
  - Reordering independent instructions
  - Using different variable/register names
  - Using equivalent instruction sequences:

    y = x + x vs. y = 2 * x
  - These viruses are sometimes called "metamorphic" viruses in the literature.

- There exist C++ libraries that, when linked against an appropriate executable, automatically turn it into a metamorphic program.

- Sometimes vulnerable software itself offers opportunities for hiding bad code.
  - Example: ssh or SSL vulnerabilities may permit worm to propagate over encrypted channels, making content filtering impossible.
  - If IPSEC becomes popular, similar problems may arise with it.

Other Evasion Techniques

- Observation: worms don't need to scan randomly
  - They won't be caught by internet telescopes
- *Meta-server* worm: ask server for hosts to infect (e.g., Google for "**powered by php**")
- *Topological* worm: fuel the spread with local information from infected hosts (web server logs, email address books, config files, SSH "known hosts")
  - No scanning signature; with rich inter-connection topology, potentially very fast.
- Propagate slowly: "trickle" attacks
  - Also a very subtle form of denial of service attacks

Witty Worm

- Released March 19, 2004.
- Single UDP packet exploits flaw in the *passive analysis* of Internet Security Systems products.
- “Bandwidth-limited” UDP worm like Slammer.
- Vulnerable pop. (12K) attained in 75 minutes.
- Payload: *slowly corrupt random disk blocks*.

Witty, con't

- Flaw had been announced the *previous day*.
- Telescope analysis reveals:
  - Initial spread seeded via a *hit-list*.
  - In fact, targeted a U.S. military base.
  - Analysis also reveals "Patient Zero", a European retail ISP.
- Written by a Pro.
- "Zero-day" exploits are becoming more common

Broader View of Defenses

- Prevention -- *make the monoculture hardier*
  - Get the code right in the first place …
    - … or figure out what's wrong with it and fix it
  - Lots of active research (static & dynamic methods)
  - Security reviews now taken seriously by industry
    - E.g., ~$200M just to *review* Windows Server 2003
  - But very expensive
  - And very large Installed Base problem

- Prevention -- *diversify the monoculture*
  - Via exploiting existing heterogeneity
  - Via creating artificial heterogeneity

Broader View of Defenses, con't

- Prevention -- *keep vulnerabilities inaccessible*
  - Cisco's *Network Admission Control*
    - Examine hosts that try to connect, block if vulnerable
  - Microsoft's *Shield*
    - Shim-layer blocks network traffic that fits known *vulnerability* (rather than known *exploit*)

Detecting Attacks

- Attacks (against computer systems) usually consist of several stages:
  - Finding software vulnerabilities
  - Exploiting them
  - Hiding/cleaning up the exploit
- Attackers care about finding vulnerabilities:
  - What machines are available?
  - What OS / version / patch level are the machines running?
  - What additional software is running?
  - What is the network topology?
- Attackers care about not getting caught:
  - How detectible will the attack be?
  - How can the attacker cover her tracks?
- Programs can automate the process of finding/exploiting vulnerabilities.
  - Same tools that sys. admins. use to audit their systems…
  - A worm is just an automatic vulnerability finder/exploiter…

Attacker Reconnaissance

- Network Scanning
  - Existence of machines at IP addresses
  - Attempt to determine network topology
  - ping, tracert
- Port scanners
  - Try to detect what processes are running on which ports, which ports are open to connections.
  - Typical machine on the internet gets 10-20 port scans per day!
  - Can be used to find hit lists for flash worms
- Web services
  - Use a browser to search for CGI scripts, Javascript, etc.

Determining OS information

- Gives a lot of information that can help an attacker carry out exploits
  - Exact version of OS code can be correlated with vulnerability databases
- Sadly, often simple to obtain this information:
  - Just try telnet

```
playground~> telnet hpux.u-aizu.ac.jp
Trying 163.143.103.12 ...
Connected to hpux.u-aizu.ac.jp.
Escape character is '^]'.
HP-UX hpux B.10.01 A 9000/715 (ttyp2)

login:
```

Determining OS

- Or ftp:

```
$ ftp ftp.netscape.com 21
Connected to ftp.gftp.netscape.com.
220-36
220 ftpnscp.newaol.com FTP server (SunOS 5.8) ready.
Name (ftp.netscape.com:stevez):
331 Password required for stevez.
Password:
530 Login incorrect.
ftp: Login failed.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> system
215 UNIX Type: L8 Version: SUNOS
ftp>
```

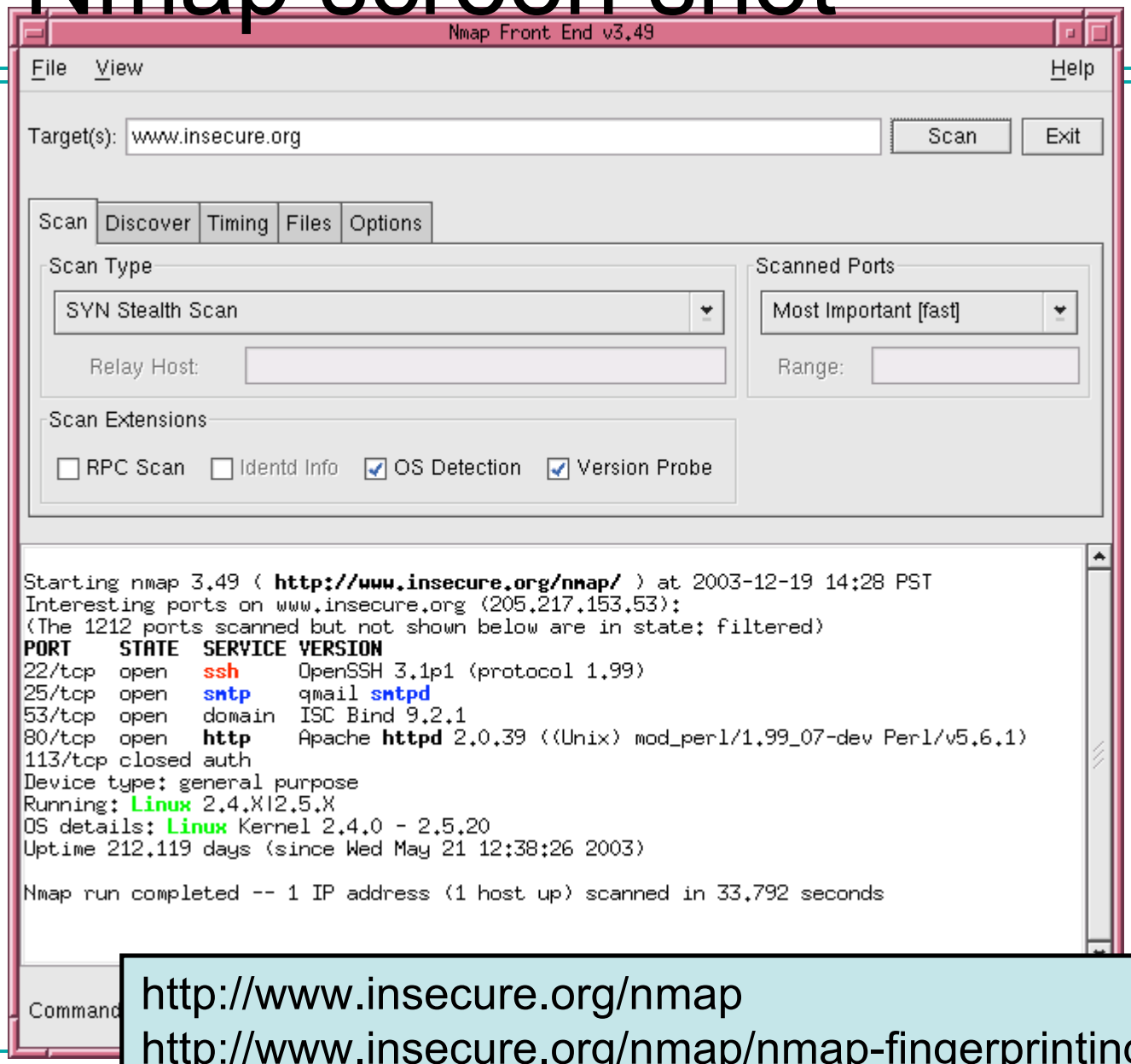
Determining OS

- Exploit different implementations of protocols
  - Different OS's have different behavior in some cases
- Consider TCP protocol, there are many flags and options, and some unspecified behavior
  - Reply to bogus FIN request for TCP port (should not reply, but some OS's do)
  - Handling of invalid flags in TCP packets (some OS's keep the invalid flags set in reply)
  - Initial values for RWS, pattern in random sequence numbers, etc.
  - Can narrow down the possible OS based on the combination of implementation features
- Tools can automate this process

Auditing: Remote auditing tools

- Several utilities available to “attack” or gather information about services/daemons on a system.
  - SATAN (early 1990’s): Security Administrator Tool for Analyzing Networks
  - SAINT - Based on SATAN utility
  - SARA - Also based on SATAN
  - Nessus - Open source vulnerability scanner
    - http://www.nessus.org
  - Nmap
- Commercial:
  - ISS scanner
  - Cybercop

Nmap screen shot



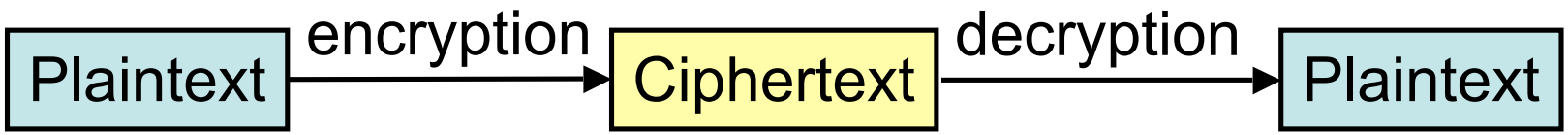
http://www.insecure.org/nmap
http://www.insecure.org/nmap/nmap-fingerprinting-article.html

Κρυπτογραφία (Cryptography)

- From the Greek "kryptos" and "graphia" for “secret writing”

- Confidentiality
  - Obscure a message from eaves-droppers
- Integrity
  - Assure recipient that the message was not altered
- Authentication
  - Verify the identity of the source of a message
- Non-repudiation
  - Convince a 3rd party that what was said is accurate

Terminology



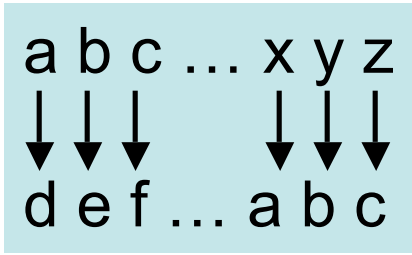
- Cryptographer
  - Invents cryptosystems
- Cryptanalyst
  - Breaks cryptosystems
- Cryptology
  - Study of crypto systems
- Cipher
  - Mechanical way of encrypting text or data
- Code
  - Semantic translation: “eat breakfast tomorrow” = “attack on Thursday” (or use Navajo!)

Kinds of Cryptographic Analysis

- Goal is to recover the key (& algorithm)
- Ciphertext only attacks
  - No information about content or algorithm
  - Very hard
- Known Plaintext attacks
  - Full or partial plaintext available in addition to ciphertext
- Chosen Plaintext attacks
  - Know which plaintext has been encrypted
- Algorithm & Ciphertext attacks
  - Known algorithm, known ciphertext, recover key

The Caesar Cipher

- Purportedly used by Julius Caesar (c. 75 B.C.)
  - Add 3 mod 26

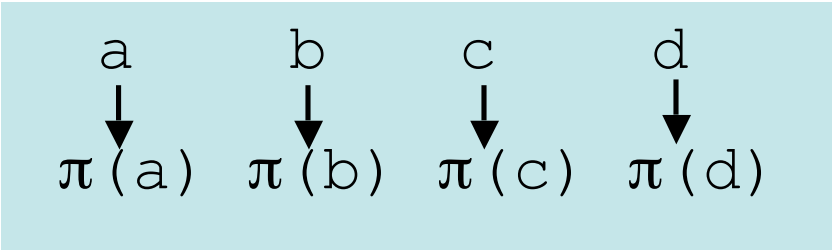


- Advantages
  - Simple
  - Intended to be performed in the field
  - Most people couldn't read anyway
- Disadvantages
  - Violates "no security through obscurity"
  - Easy to break (why?)

Monoalphabetic Ciphers

- Also called *substitution* ciphers
- Separate *algorithm* from the *key*
  - Add N mod 26
  - rot13 = Add 13 mod 26

- General monoalphabetic cipher
  - Arbitrary permutation π of the alphabet
  - Key is the permutation



Example Cipher

	a	b	c	d	e	f	g	h	i	j	k	l	...
π	z	d	a	n	c	e	w	i	b	f	g	h	...

Plaintext: **he lied**

Ciphertext: **ic hbcn**

Cryptanalysis of Monoalphabetic Ciphers

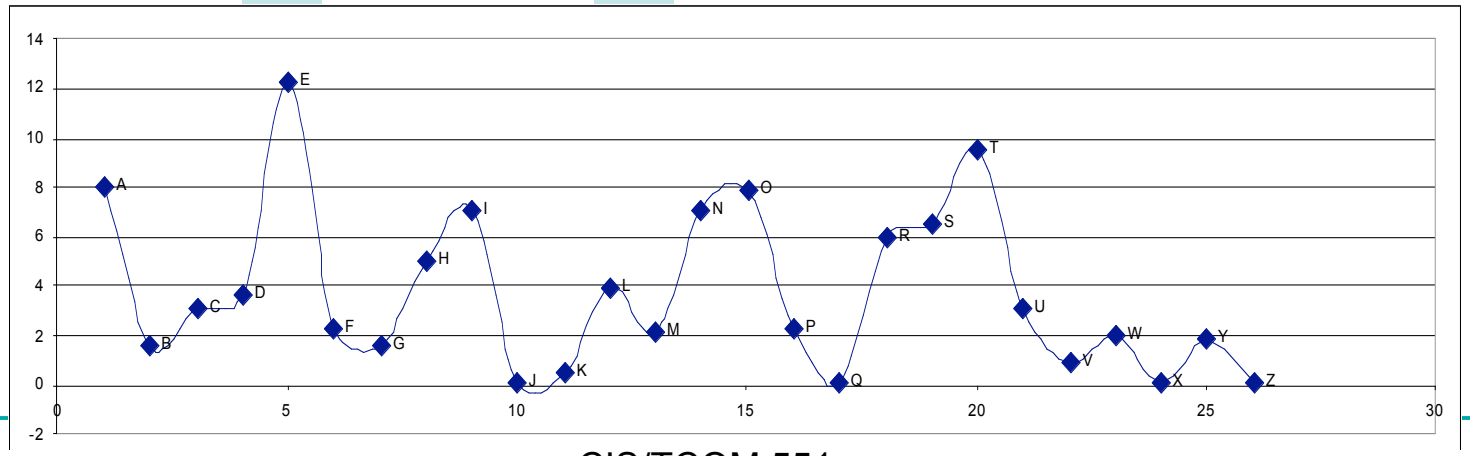
- Brute force attack: try every key
  - N! Possible keys for N-letter alphabet
  - 26! ≈ 4 x 10^{26} possible keys
  - Try 1 key per μsec … 10 trillion years

- …but (!) monoalphabetic ciphers are *easy* to solve

  - One-to-one mapping of letters is bad
  - Frequency distributions of common letters

Order & Frequency of Single Letters

E	12.31%	L	4.03%	B	1.62%
T	9.59	D	3.65	G	1.61
A	8.05	C	3.20	V	0.93
O	7.94	U	3.10	K	0.52
N	7.19	P	2.29	Q	0.20
I	7.18	F	2.28	X	0.20
S	6.59	M	2.25	J	0.10
R	6.03	W	2.03	Z	0.09
H	5.14	Y	1.88		



Monoalphabetic Cryptanalysis

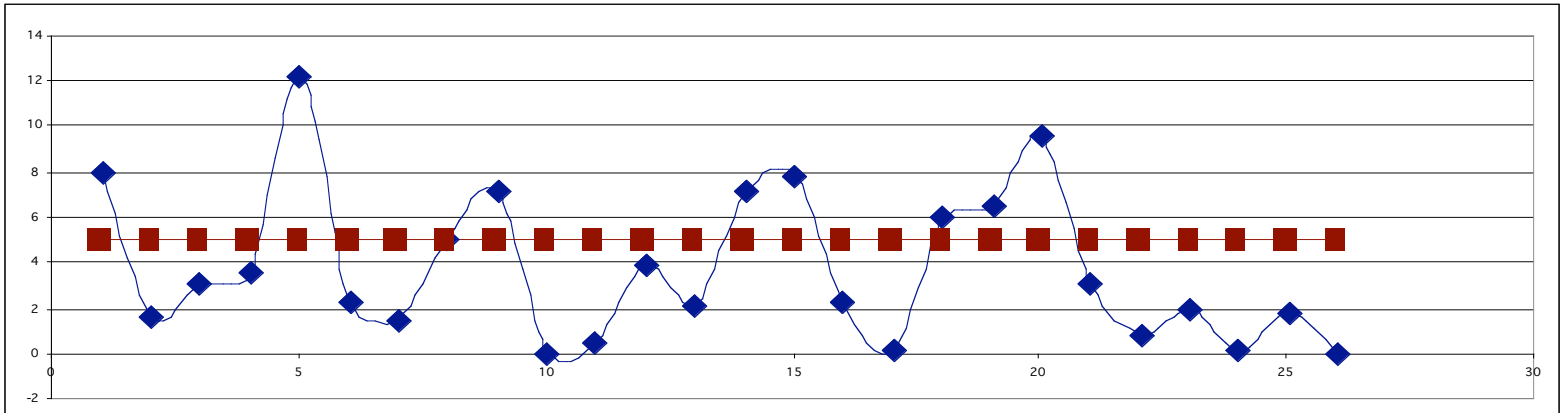
- Count the occurrences of each letter in the cipher text
- Match against the statistics of English

- Most frequent letter likely to be “e”
- 2nd most frequent likely to be “t”
- etc.

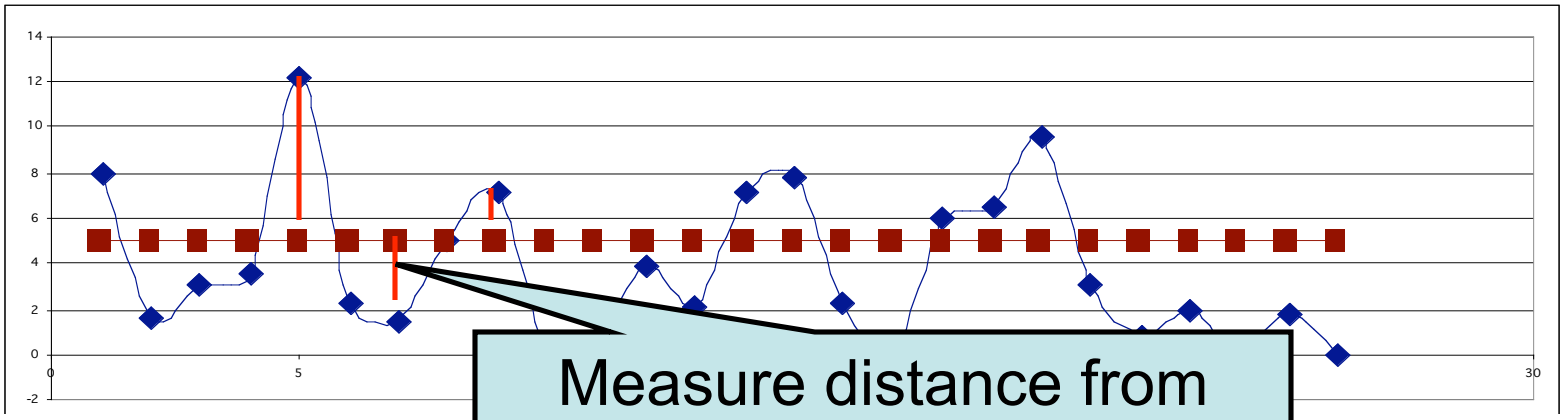
- Longer ciphertext makes statistical analysis more likely to work…

Desired Statistics

- Problems with monoalphabetic ciphers
  - Frequency of letters in ciphertext reflects frequency of plaintext
- Want a single plaintext letter to map to multiple ciphertext letters
  - "e" ⟶ "x", "c", "w"
- Ideally, ciphertext frequencies should be flat



Variance: Measure of "roughness"



$$\text{Var} = \sum_{\substack{\alpha = a \\ \alpha = z}} (\text{prob}(\alpha) - 1/26)^2$$

$$= \ldots$$

$$= \left(\sum_{\substack{\alpha = a \\ \alpha = z}} \text{prob}(\alpha)^2 \right) - 1/26$$

Polyalphabetic Substitutions

- Pick k substitution ciphers
  - $\pi_1\ \pi_2\ \pi_3\ \ldots\ \pi_k$
  - Encrypt the message by rotating through the k substitutions

m	**e**	**s**	**s**	**a**	**g**	**e**
π_1(**m**)	π_2(**e**)	π_3(**s**)	π_4(**s**)	π_1(**a**)	π_2(**g**)	π_3(**e**)
q	**a**	**x**	**o**	**a**	**u**	**v**

- Same letter can be mapped to multiple different ciphertexts
  - Helps smooth out the frequency distributions
  - *Diffusion*

Diffusion and Confusion

- Diffusion
  - Ciphertext should look random
  - Protection against statistical attacks
  - Monoalphabetic -> Polyalphabetic substitution; diffusion ⇑
- Confusion
  - Make the relation between the key, plaintext and ciphertext complex
  - Lots off confusion -> hard to calculate key in a known plaintext attack
  - Polyalphabetic substitution: little confusion

Perfect Substitution Ciphers

$$
\begin{array}{rl}
 & p_1\ p_2\ p_3\ \ldots\ p_n \\
\oplus & b_1\ b_2\ b_3\ \ldots\ b_n \\
\hline
 & c_1\ c_2\ \ c_3\ \ldots\ c_n
\end{array}
$$

- 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
  - “Numbers Stations”?
- 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
  - Not enough confusion

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 security*
  - 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.