* Course Content
  + facwiki.cs.byu.edu/cs465
* Homework
  + Regularly assigned, due at th start of class each Tuesday
* Pro Projects
  + Due Frieday at 5PM
* Exams
  + 2 + final
* Cryptography Intro
  + Encryption
    - Hide the true meaning of something
    - Algorithms user transposition and substitution in complex ways
    - Bad Examples:
      * ROT-13
      * Vigenere Cipher
    - Strong Examples:
      * AES
      * 3DES
      * RC4
* Kerckhoff’s: everything depends on the secrecy of the KEY
* --------------------------------------------
* \*Describe Symmetric Encryption
  + draw picture
  + write a few sentences
* -Cryptographic Hash Functions
* Digest
  + Is usually a fixed size
  + The output of a cryptographic hash function
* \*Explain MAC
  + authenticity and integrity
* Cryptography
  + Encryption
  + Digital signatures
* Xtime() == times by HEX, and mOD by ‘11b’
* Access control
  + Authentication
    - You are who you claim to be
  + Authorization
    - Entities have permission to perform an action
* Confidentiality (Encryption)
  + Prevent unauthorized people from seeing something
* Integrity (No unauthorized person changing this)
  + Don’t modify this!
* Authenticity (Who someone is, ID who is invovled)
  + Is this from who it claimed to be from
* Non-repudiation
  + Can prevent someone from denying they took part in a transaction
  + Usually involves cryptographic evidence that will stand up in court
* Two requirements for strong symmetric encryption
* 1. strong algorithm (cipher)
  + attacker is unable to decrypt ciphertexxt or discover the key even if attacker has samples of ciphertext/plaintext created using the secret key
* 2. sender and receiver must securely obtain and store the secret key
* One-Time Pad
  + Works good
  + Just a little unfeasible for large data
  + Hard to get unique key everytime
* ECB
  + Weakness – it can reveal patterns in the plain text.
* Stream Modes – we don’t have to do padding.
* Padding
  + Relevate to ECB and CBC
  + Schemes
    - \*\*Read Padding article
    - Pad with the value of the padding needed
    - 0x80 followed by 0x00
    - Pad with 0x00
      * Last byte is equal to the number of padding bytes
    - Pad with 0x00 or spaces
  + FIX some of the edge cases: just always pad!
* \*\* Properties of a Hash Function
  + H can be applied to a block of data of any size
  + H produces a fixed-length output
  + H(x) si relatively easy to compute for any given x
  + For any given value h, it is computationally infeasible to find x such that H(x) = h
  + For any given block x, it is computationatonally infeasible to find y != x with h(y) = h(x) (weak collision resistance)
  + It is computationally infeasible to find ANY pair (x,Y) such that H(x) = H(y) (strong collision resistance) x != y
* CBC – Is the most common mode of systems
* \* How does AES provide integrity?
  + No not really
  + It is ment to provide confidentiality
* AES is just a simple small scope algorithm. Modes and blocks extra are all on the outside of AES.
* Know the Hash properties.
  + Talk about he attacks and their costs
* Hash is good for
  + Digital signatures
  + File downloads
  + ?
* Feistel Cipher Structure
  + Symmetric encryption algorithms
    - DES, IDEA, Blowfish, …
    - NOT AES
  + Applies to block cipher
* MAC – provide authentication and integrity
  + Sometimes called a digial signature
  + CBC-Mac
    - Use cbc mode and a block cipher
    - Expensive
  + Hash it and the Encrypt the digest
  + Hash the message along with a shared key
* HMAC – mac generated using hashing
  + Vulnerable to a message extension attack
* \* H(K||H(K||M))
  + Just prepend the key twice.
* HMAC
  + General term means any MAC that is implemented using hashing
  + Other times it’s the Government standard
* **Asymmetric Encryption**
* Why public key crypto is cool
  + Is a linear solution to the key distribution problem
    - Symmetric is expoinential
  + Send messages to people you don’t share a secret key with
    - So only they can read it
    - They know it came from you
* Number Theory
* Diffie – Hellman
  + Protects a gains a passive attacker
* Cost of Modular Exponentiation:
  + Cost of the number of one bits
  + Cost of updating the base every time
* RSA
  + Co-prime == relatively prime
* \*\*Describe public key cryptography
* \*\*Describe digital signatures
* RSA
  + P = Large Prime Number
  + Q = Large Prime Number
  + E = public exponent
    - Co-prime, Relatively prime to phi(n)
    - Common values 3,65537
  + D = private exponent
    - It’s the mulciplicative inverse of E (mod phi(n))
  + N = modulus
    - N = P\*Q
  + C=m^e%n
  + M = c^d%n
* \*computing phi(n) in RSA
  + == (p-1)(q-1)
* \*Memorize Steps for RSA Encryption
* Why is RSA Secure?
* MI
  + Extended Euclidean algorithm
* DS
  + Know digital signatures
  + Know how to describe or articulate DS diagram
  + What do we get?
    - Auth
    - Integrity
    - Non-repudiation
      * Because it is very clear who sent the message
      * But with MAC you can’t tell who sent it
  + What are the difference between MAC and Digitial Signature
  + When do you sign?
    - Sign-then-encrypt (common)
      * Surreptitious forwarding attack
        + You can decrypt it and the you have a signed message that you can send off…
    - Encrypt-then-sign
      * Authorship claim attack
        + Peal off DS and attach a new one
* RSA
  + Note there are pitfalls in the vinela version of RSA
  + Key for signature or encryption but not both
  + Padding
    - Helps us be more secure
* Certificate Verification
  + Integrity
    - Check digial signature
  + Expiration
  + Revocation Lists
  + Usage constraints
    - Basic constraints
      * Can the subject act as a CA?
      * Is there a limit to the length of the certificate chain?
      * Limitation on key use – encryption or
  + Ownership
    - Does the entity presenting the certificate have access to the associated private key?
      * Done with chalange response
      * Like a picture on a membership card
* How to recover from a lost/stolen private key? In a hiearcie..?
* College would just get a new certificate…then just resighn there immediate children…so the college would just resign all the department certificates…
* TEST REVIEW
  + HMac Attack
  + Certificate verification
    - Ownership gets interesting
  + Certificate terminology
  + Certificate Hierarchies
    - He who lost it
      * Needs a new key pair and new certificate
      * Then he must resign everyone immediately below him
  + Polynomial representation
    - Know ffMultiply
  + Talk about cryptographic one way hash
    - Know its properties
    - Describe the second pre-image attack
  + Digest comes from a hash
  + MAC – integrity and authenticity
    - 3 way to implement it
    - H(K||H(K||M))
  + SHA-1
  + Message Extension attack
  + Digital Signature
  + RSA for Encryption
  + Why do we need RSA padding and what is it?
  + Modes – give you the diagrams
    - Review the homework
* **Secure Email**
* \* How does secure email work? To provide confidentiality, integrity, and authentication
* PGP
  + Trust Model 🡪 it’s a web of trust
    - You generate keys and share them with whoever needs them.
    - No trusted 3rd party
  + Public Key Crypto
* \*Generate c on the PGP crpto functions slide
  + using DS and Zipping and Ecryption
  + this is for authentication and confidentiality
* S/MIME
  + Security extension to the mime internet email format
  + Trust model
    - Hierarchiacal, top-down
    - X.509 certificates
      * Statdard certificate format
* **TLS**
* \* TLS handshake
* \* Understand client/server auth in tls
  + rsa key exchange
  + explain ownership proofs in detail
  + what cryptographic primitives are used and why?
* \* understand session resumption
* \* understand the limitations of TLS
* TLS (View Basic TLS handshake)
* SERVER BASED AUTH
  + Gray ones are optional
  + Client\_hello
    - TLS version
    - A Random Number \*important
    - List of suggested CipherSuites (heres the encryption I want to use) (auth, encrypt, bulk ciphers(AES), and MAC(HMAC)) want to use
    - Seesion ID
  + Server\_hello Response
    - Random number
    - Chosen protocol version
    - Picks a cipher suite
  + (M) send Server Certificate
  + hello\_server\_done
  + Client\_key\_exchange
    - Client generates PreMasterSecret
    - Client encrptes PMS and sends it to the server
  + GenMaster Secret
    - INPUT: both random numbers, preMasterSecret
    - OUTPUT: MasterSeret
    - Both server and client generate this separately
    - ~ownership here….it won’t get the preMasterSecret out
  + GenKeyBlock material
    - INPUT: both random number, master secret
    - OUTPUT: generate key block material
    - Both do that
  + Keys needed?
    - Aes
    - Mac
    - IV (possibly)
    - But both generate 4 to 6 keys
      * Different keys for different directions
      * MAC, AES, IV
  + Change\_cipher\_spec
    - Nothing big her
  + Finished
    - This message is encrypted
    - Its like application data
    - Contains
      * Hash and mac over the previous handshake messages
      * Hash is a running hash of everything that has happened in the communication so far
  + Change\_cipher\_spec
    - Nothing big here
  + Finished
    - Same for the cient but now includes the clients finished…
  + Why random numbers
    - Every TLS connection will be unique. Comes in with the running communication hash…
* CLIENT AUTH
  + NEW
    - Server will send a **certificate\_request**
      * Include list of trusted CA’s
    - Client will then send
      * Certificate
      * Certificate\_verify
        + Signature over the previous handshake messages using the client’s certificates private key
        + Sign the running hash
        + Master secret is also somehow included
* \*\*Ownership thing
  + How does the server/client validate ownership.
* Session resumption
  + If we both send back the same ssid
  + Both send different radom numbers
  + Generate new key block material
  + Jump to finish messages
* Stack Smashing
  + Format of String
    - Can not have internal null values
  + Defenses
    - Write correct code
    - Non-executable buffers
      * Dosnt’ eliminate this attack. Just helps.
    - Array bounds checking
    - Code pointer integrity checking
      * Detect when a pointer is corrupted
      * stackGuard and PointerGuard
    - Address space randomization
  + StackGuard
    - How does it work?
    - What is a canary?
      * Early Warning system
        + Terminator canary – like a bunch of NULL just after the return address
        + Random cannary – pick a random number and stick it in there
        + XOR canary –
  + \* know defenses and their pro and coins
  + Principle of least privilages
    - When code runs it should run with just enough privilages that it needs to run and no more.
    - How does this relate to buffer overflow attacks
      * If you overflow something you only have the privilages of the code that you are overflowing… the less power the vulnerable code has the better
* Passwords
* \*\* UNIX and Lamports know them
* plain text -> encrypt the file
* Passwords is a misnomer
  + Do not use single words
  + Use a pass-phrase
* \*Online vs Offline attack
  + Guess the user’s password
    - online
      * attempt to login as the user would
    - offline attack
      * repeated guessing involving an encypted form of the user’s password
      * doesn’t envolve the server at all
      * they steal the hashed/encrypted file of passwords they run a bunch of guesses to see if they can guess one that hashes to one of the hashes
  + shoulder surfing
  + users write it down
  + users give away their passwords
* Help with these:
  + Exponential back off response time
  + Time lock out
* Problems w/ Passwords
  + Too many of them
* Time Est
  + (choices)^number of charaters in the password
  + 8 char password
  + 62^8
* Unix Password
  + Original password file /etc/passwd was world reasdable
    - Anyone could copy the file offline and perform a dictionary attack
    - You could fin sampel files on Gooogle courtesty of naïve sys admins
  + Moved to a shadow file /etc/shadow that required root privileges to access
* Crypt – encrypt / shaw reutein
* Create password
  + Username , password + salt, E(p+s)
* \*why is the salt there?
  + Prevents the id of identical passwords
  + All password guesses are sal-specific
    - Therefore one guess of a password will not help across any password in the file
    - \*\*\*See password slat slide
* Salt
  + Only help against offline attacks
  + Only increases it by nymber of slats in a file
  + For a tratgeted attack it doesn’t really hurt time
* Lamport’s Hash