# Web Security Basics

## CounterMeasures: Encryption I

# Cryptography

- Symmetric Key algorithms
  - encrypt with a key
  - decrypt with the same key
  - Symmetric algorithms are strong
  - The problem is that you need a different key for each private conversation, and you have the problem of key exchange
  - DES (Data Encryption Standard) from IBM is an example
  - Fixed key length of 56 bits
  - DES now considered weak, but now have 3DES, AES

# Cryptography

- AES announced to replace DES
- National Institute of Standards and Technology chose the Rijndael cipher as the AES algorithm
- variable length keys 128, 192, or 256 bits
- variable length blocks 128, 192, or 256 bits
- AES is more efficient than DES
- AES relatively young: 3DES is more trusted because of the maturity of the algorithm

# Asymmetric Key Algorithms

- Two matched keys: one to encrypt, the other to decrypt
- Called private and public keys
- To send private message, encrypt it with the recipient's public key
- To receive private message, decrypt it with your private key

# Asymmetric Key Algorithms

- Or, to send a private message, encrypt first with your (sender's) private key, then encrypt with recipient's public key
- Recipient decrypts first with her private key, then with sender's public key
- What's the advantage of this?

# Hashing Algorithms

- One way function easy to compute but hard to reverse
- Hashing Algorithms compute a fixed-length digest (fingerprint) of output data
- Cryptographically strong; that is, it is infeasible to recover original text from the digest
- can ensure data didn’t change accidentally, but can’t ensure data wasn’t deliberately changed

# Hashes

- Message Digest 5: MD5
  - 128 bit digest
  - considered strong until recently
  - collision resistant: two messages with same hash are very unlikely to occur
- Secure Hash Algorithm 1: SHA-1
  - 160 bit digest
  - published by NIST (National Institute of Standards and Technology)
  - similar design to MD5, but stronger, not strong enough
  - slightly slower than MD5

# MD5 Example

https://www.mkyong.com/java/java-md5-hashing-example/

```
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
public class PasswordMD5 {
  public static void main(String[] args) throws NoSuchAlgorithmException {
    String password = "123456";
    MessageDigest md = MessageDigest.getInstance("MD5");
    byte[] hashInBytes = md.digest(password.getBytes(StandardCharsets.UTF_8));
    StringBuilder sb = new StringBuilder();
    for (byte b : hashInBytes) {
        sb.append(String.format("%02x", b));
     }
     System.out.println(sb.toString());
   }
 }
```

# SHA: Secure Hash Algorithm

- NSA developed this family of algorithms
- SHA1 : not considered strong since 2010
- SHA2: A family of two similar hash functions, with different block sizes, known as SHA-256 and SHA-512
  - SHA-256: 32-bit words
  - SHA-512: 64 bit words

# Java SHA-256

- Use ”SHA-256” instead of “MD5” in above code

# SSL and TLS

- SSL originally developed by Netscape
- now accepted by the WWW as the standard for authenticated and encrypted communication between clients and servers
- SSL is application independent (ie HTTP, FTP, Telnet, etc, can be layered on top of it)
- Negotiates encryption keys and authenticates the server before data is exchanged

# SSL Handshake (two phases)

- SSL Handshake first phase
  - In response to client request, server sends its certificate and its cipher preferences
  - Client then generates a master key, which it encrypts with the server's public key, and transmits it to the server
  - Server recovers the master key and authenticates itself to the client by returning a message authenticated with the master key
  - Subsequent data is encrypted and authenticated with keys derived from this master key

- SSL Handshake second phase (optional)
  - Server sends a challenge to the client
  - On the challenge, the client authenticates itself to the server by returning the clients digital signature and its public-key certificate

# TLS

- Transport Layer Security is an improved version of SSL
- SSL is still the method supported by all web servers and web browsers

# Digital Certificates

- Key management is often considered the most difficult task in designing and implementing cryptographic systems
- Public Key Infrastructure (PKI) designed to help in this regard

# PKI countermeasures

- Three primary security vulnerabilities in communicating over a public network:
  - Identity Theft – Intruder gains access by posing as an individual who can access secured resources
  - Eavesdropping – Intruder listens to the traffic between two parties during communications
  - Man in the Middle – Intruder interrupts a dialog and modifies the data between the two parties (or intruder takes over entire session)

# Digital Certificate Characteristics

- PKI provides a hierarchical framework for managing digital security attributes
- Each PKI participant holds a digital certificate that has been issued by a Certificate Authority (CA)
- The certificate contains a number of attributes
  - certificate validity period
  - end-host identity information
  - encryption keys that will be used
  - the signature of the issuing CA

# Certificate Authorities

- CA can be a trusted third party, such as VeriSign or Entrust, or a private in-house CA established within your organization
- To validate the CA's signature, the receiver must know the CA's public key
- Normally this is handled out-of-band
  - most web browsers are configured with the root certificates of the CA's used for SSL/TLS
  - you can add certificates to your browser's list of trusted certificates

# Enrolling in a CA

- Enrollment is enacted between the end host that needs the certificate, and the authority in the PKI that provides certificates
- Step 1: the end host generates a private-public key pair
- Step 2: the end host generates a certificate request to give to the CA
- Step 3: Manual human intervention is required to approve the enrollment request
- Step 4: After the CA operator approves the request, the CA signs the certificate request with its private key and returns the completed certificate to the end host
- The end host installs the certificate