Ryan Stillings

Cryptography and Network Security I

Final Project Writeup

Functional Overview

* Single main file - can act as a client or server depending on who initiates connection
  + Person who sends connection request becomes client for that session; receiver becomes server
  + Makes the netcode more complicated, but allows the application to be entirely peer to peer
* Async i/o is handled via threading
  + One thread for the GUI
  + One thread for listening for incoming connections
  + One thread for listening for incoming messages
  + One temporary thread for executing the SSL protocol to establish a secure chat
  + "global" variables stored in a shared module - networking.py (thread memory is shared)
* GUI created by hand in WXPython
  + Allows for a more polished feeling application
  + Hides debug messages in the terminal so that users can ignore them during normal operation
    - Terminal left in for demo/debugging purposes
  + Easily scroll through messages sent/received
  + Automatically scrolls to bottom when a new message is sent/received
* Preferences stored in preferences.cfg config file
  + Keeps things simple and avoids the need to clutter up the GUI
  + Automatically stateful since it’s a separate file
* Common utlities stored in cryptoutil.py
  + Modular inverse, extended euclid, type conversions, etc..
* Functions related to primality stored in primality.py
  + Miller Rabin, Pollard Rho, and prime generation in a Galois Field
* Hash functions stored in hashing.py
  + HMAC, SHA1 for message authentication
* Ciphers stored in individual named packages

Technical Details

* Connecting to someone to chat
  + Negotiation - SSL
    - Client
      * Sending preferences
    - Server
      * Deciding preferences
  + Establishing a secure channel
    - Diffie Hellman - Key Exchange
    - Key is used to seed random function for producing equivalent primes, etc.. on server and client
  + Failures
    - Any unexpected failures will stop the connection process and disconnect
* Messaging passing
  + Paillier - Encryption
    - Generates two primes matching 128 bits
    - Stores the public and private key in simple structs passed to the main module
    - Encryption and decryption take advantage of python's built-in pow function for modular exponentiation
  + Blum Goldwasser - Encryption
  + Authentication

Closing remarks

* + Peer to peer (serverless) design posed some major challenges; required multiple threads to solve elegantly
  + Simple GUI design was more time consuming than expected
  + Opted away from a generic "Cipher" class as originally proposed during the white hat presentation
    - Python's weak dynamic typing meant that we could use generic code that works differently depending on the properties defined by the cipher (ie. perform the same operations on a key that might have completely different fields)
  + For future releases would be a fun challenge to pursue full SSL compatibility
  + Would also be fun to implement more, more secure ciphers