

Programming Project # 1

CS255

Due: Monday, February 7th

(11:59 pm)

Communication – 60,000 feet

Arguments – 60,000 feet

Big idea

- Broker wants to broadcast “hot tips”
- But doesn’t want *just anyone* to be able to recover the plaintext
- So relies on Server to handle acct mgmt
- Client obtains some Broker-generated ciphertext
- Needs to go through Server in order to get key to recover the plaintext

BrokerGUI – arguments

- Name of file containing info to encrypt (info to encrypt is hereafter “hottip”)
- Name of output file to write (ciphertext) to
- A password which he shares with the AuthorityServer (hereafter “sharedPwd”)

BrokerGUI – actions I

- Generates an encryption key (**K-BC**) and a MAC key (**K-MAC**) from the sharedPwd
- Generates new, random key material (K)
- Generates a new encryption key (**K-temp**) and MAC key (**K-MAC-temp**) from K
- Reads in the input file specified in order to get String which is the hottip

BrokerGUI – actions II

- Encrypts hottip using new random key material
 - $E[\mathbf{K-temp}, \text{hottip}]$
- MACs this
 - $MAC[\mathbf{K-MAC-temp}, E[\mathbf{K-temp}, \text{hottip}]]$
- Encrypts new random key material using the key he shares with the AuthorityServer
 - $E[\mathbf{K-BC}, K]$
- MACs this
 - $MAC[\mathbf{K-MAC}, E[\mathbf{K-BC}, K]]$

BrokerGUI – actions III

- Writes all of the preceding to a file
 - The name of which is specified when you run the BrokerGUI
- **10,000 feet:** what did Broker do?
 - Encrypted his hottip with a new & random key
 - Encrypted that new & random key using the key he shares with the Server
 - Provided integrity over all of this
- Done with Broker for the meantime...

AuthorityServer – arguments

- sharedPwd : the password that the server and broker share
- adminPwd : the password that the server alone has
- port : port # to run on

AuthorityServer – actions I

- Reads in a plaintext file consisting of username-password pairs (you create this file; you add code for server to read it in)
 - Just does once
- Encrypts and MACs that file using keys generated from his adminPwd
 - Just does once
- Waits for client connections ... (in a loop)

BrokerClient – arguments

- Client's username
- Client's password
- Name of file that Broker wrote ciphertext to
- Host server running on
- Port server listening on

BrokerClient – actions I

- Reads in the ciphertext file provided by the BrokerGUI
 - Generates an encryption key
 - **K-BC-user1**
 - and a MAC key
 - **K-MAC-user1**
- ...from his password

BrokerClient – actions II

- Generates an authentication token for the server (note: R is a new random)
 - R || username ||
 $MAC[\mathbf{K-MAC-user1}, R || \text{username}]$
- Sends to server:
 - $E[\mathbf{K-BC}, K]$
 - $MAC[\mathbf{K-MAC}, E[\mathbf{K-BC}, K]]$
 - Authentication token from above

AuthorityServer – actions II

- Takes incoming client connection
- Spawns new thread **T** for this
- **T** checks username in client message; generates appropriate BC and MAC keys
- **T** verifies integrity of authentication token
- **T** checks R value (new?) then updates R value for client (in hashtable)
- If satisfied, **T** generates & sends to client:
 - $E[\mathbf{K-BC-user1}, K]$
 - $MAC[\mathbf{K-MAC-user1}, E[\mathbf{K-BC-user1}, K]]$

BrokerClient – actions III

- Receives data from server
- Decrypts key material K; verifies integrity
- Generates **K-temp** & **K-MAC-temp** from K
- Decrypts and verifies original hottip
- Prints that hottip to the screen
- ...then exits

Some questions

- How to generate block cipher and MAC keys from a shared password?
- How to use block cipher? Which to use?
- How does multithreading work?
- How to detect replay?
- Other?

Generating keys from a password

- One way:
 - Convert `char[] pwd` to `byte[] pwdBytes`
 - Take MD5 of `pwdBytes[]` to get `seedBytes[]`
 - Create new `SecureRandom(seedBytes)`
 - `byte[] desKeyBytes` = next 24 bytes of SR
 - `byte[] hmacKeyBytes` = next 64 bytes of SR
 - Create new `DESedeKeySpec` passing `desKeyBytes`; then `KeyGenerator` to get a key
 - Similar for HMAC but use `SecretKeySpec`

How to use block cipher?

- Gotta use CBC with a new, random IV
- Can use: 3DES
- Think AES might not be supported by SunJCE crypto provider
- Think RC4 also not supported by SunJCE crypto provider
- Which brings us back to 3DES, ...

First: multithreading, a couple small changes

- See next two slides
- You will need to modify (slightly) AuthServ.java
- You'll need to add a couple hashtables
- Both of which may take username as key (yes, that means can assume usernames unique)
- One HT contains (username,pwd)
- Other contains (username,lastRValue)
- The AuthorityServer creates these HTs when reading in the PT file
- Then passes them to each thread it spawns

Sample change to AuthSvr:

```
/* spawn new thread to handle client sock */  
new AuthorityServerThread(  
    clientSocket,  
    adminPwd.getPassword(),  
    sharedPwd.getPassword(),  
    usersToPasswords, // new  
    usersToRs // new  
).start();
```

Sample: alter AST constructor

```
Hashtable userPwds = null; // new
Hashtable userRs = null;   // new

public AuthorityServerThread(Socket sock,
                             char[] adminPassword,
                             char[] sharedPassword,
                             Hashtable usersToPwds,
                             Hashtable userToRs) {

    clientSocket = sock;
    adminPwd = adminPassword;
    sharedPwd = sharedPassword;
    userPwds = usersToPwds; // new
    userRs = userToRs; // new
}
```

How to detect replay?

- For you to figure out
- But is possible that instead of new, random have initial random sent by client then increment by one each time (more modest memory reqs on server)

Side note about replay

- Point of detecting replay and revoking user whose msg was replayed is to avoid server having to invest computational cycles needlessly ... so don't have server do a lotta work if msg is eventually going to be dropped

Other...

- Output of encipherment is usually of form `byte[]`
- So not necessary printable chars
- But you need to write the output to a file and/or to a socket; what to do?
- Hex ... write routines to convert from `byte[]` to hex (`String`) and back

Misc.

- May need to resize GUI windows ... e.g. after you enter all of the information asked and click “Submit,” while waiting for response

Questions?

- Use newsgroup first
 - Simply the most efficient form of communication