CS255

Programming Assignment #1

Programming Assignment #1

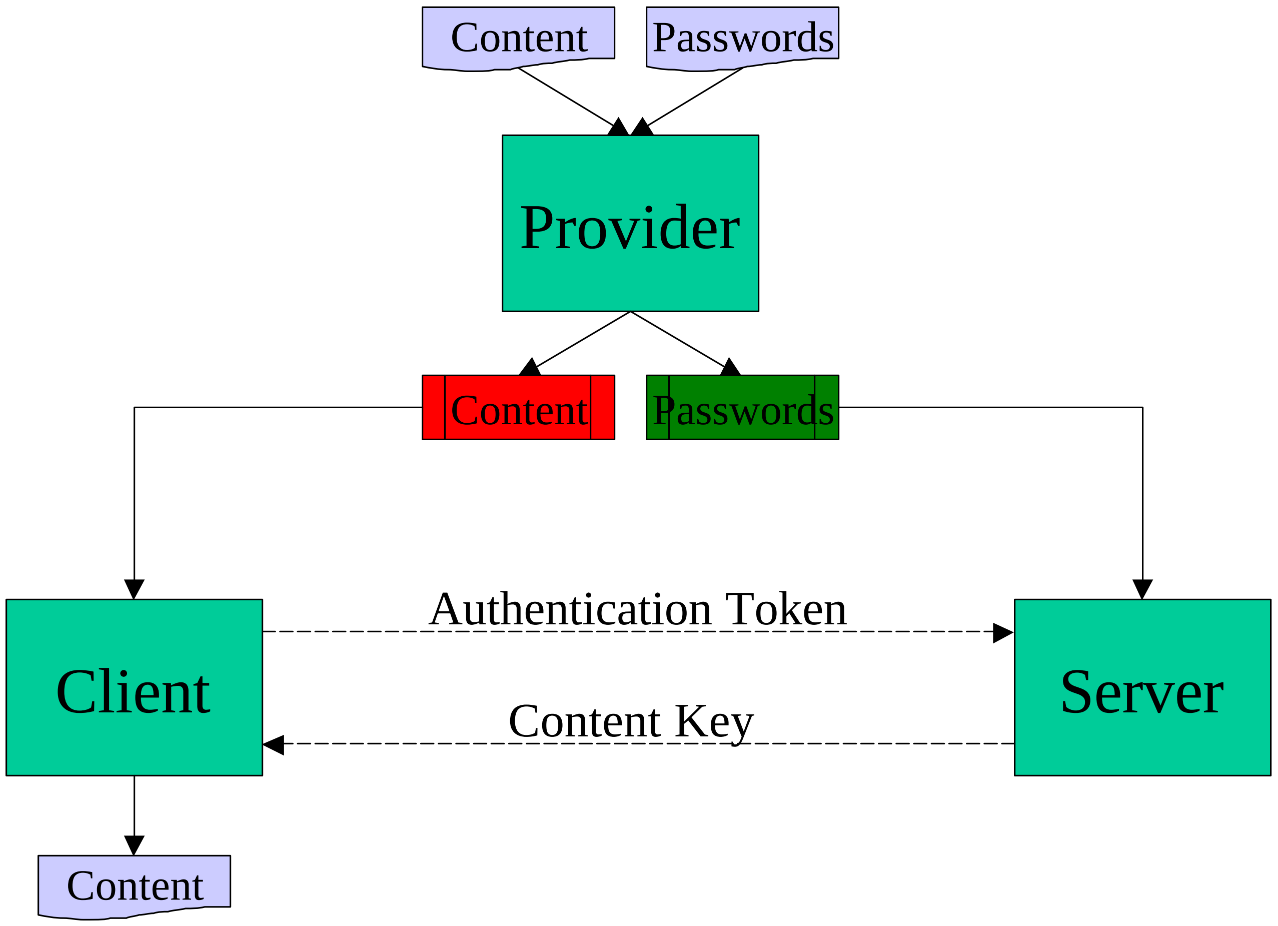
- Due: Friday Feb 10th (11:59pm)
  - Can use extension days
- Can work in pairs
  - One solution per pair
- Test and submit on Sweet Hall machines
  - SCPD students: get SUNet ID!

    sunetid.stanford.edu

Big Picture

- Provider distributes content in freely available encrypted files
- Clients obtain decryption keys from the Authority Server
- Authority Server authenticates Clients based on their username and password

Execution Scenario



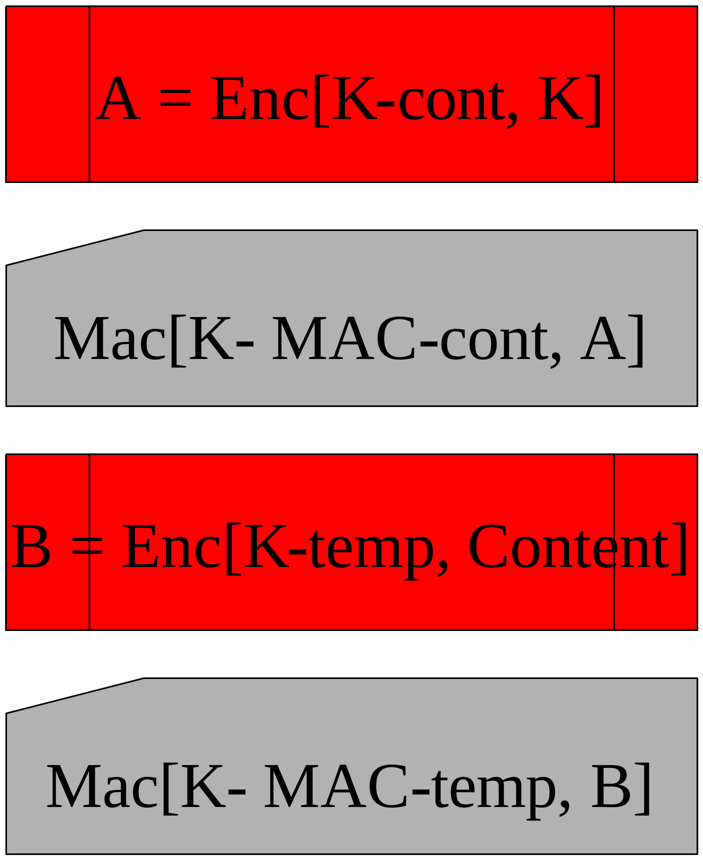
Security Requirements

- Attacker cannot obtain content or passwords
  - Encryption
- Attacker cannot modify content or passwords
  - MAC
- Only registered users can obtain content
  - Authentication
- Prevent replay attacks on the Server
  - Server does not respond to same token twice

Components: Provider

1. Generates three key pairs:
   - K-temp, K-MAC-temp (from randomness K)
   - K-cont, K-MAC-cont (from masterPwd)
   - K-pass, K-MAC-pass (from masterPwd)
2. Protects content with K-temp
   - Includes K in the header protected with K-cont
3. Protects passwords with K-pass
   - You choose the design

Protected Content



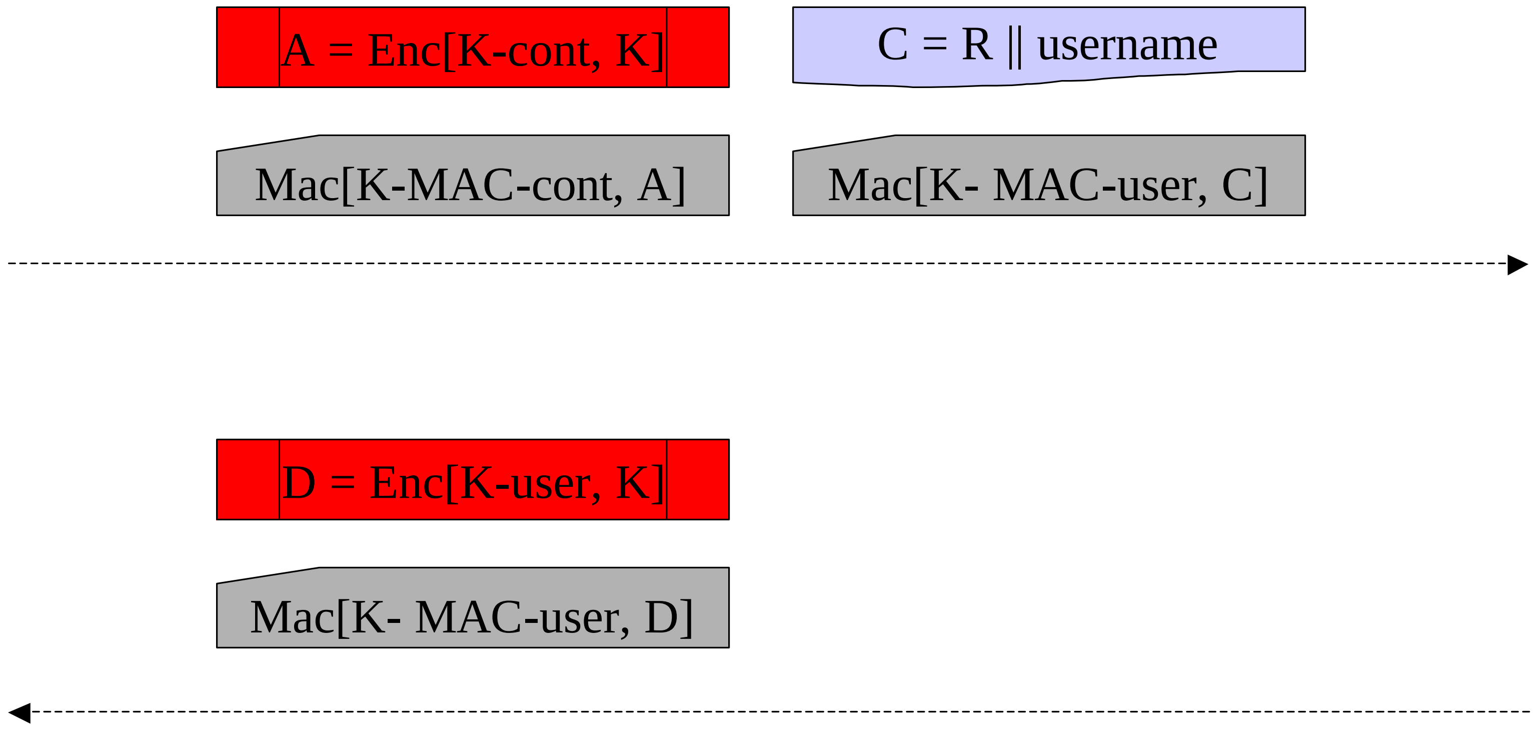
Components: Client

1. Generates key pair:
   – K-user, K-MAC-user (from userPwd)
2. Reads the header from the protected content file
3. Sends the authentication token to the server
4. Verifies and decrypts the content key
5. Verifies and decrypts the content

Components: Authority Server

1. Generates key pairs:
   - K-cont, K-MAC-cont (from masterPwd)
   - K-pass, K-MAC-pass (from masterPwd)
2. Verifies and decrypts the password file
3. For every client that connects
   1. Generates key pair from users password
   2. Verifies the authentication token
   3. Decrypts and sends the content key

Authentication Protocol



Generating Keys From Passwords

- You choose the design
- What NOT to do:
  - Use passwords as keys directly (weak keys)
  - Split passwords in half (easier to guess the password)
- Goal: Finding the key should be as hard as guessing the password
  - Even if related keys are compromised
- Tools available:
  - Block cipher (PRP), PRG, MAC, Cryptographic hash

Java Cryptography Extension

- Implementations of crypto primitives

Cipher	**`Cipher`**
Pseudo-random Generator	**`SecureRandom`**
Message Authentication Code	**`Mac`**
Cryptographic Hash	**`MessageDigest`**

JCE: Using Ciphers

1. Select the algorithm
2. Initialize with desired mode and key
3. Encrypt/Decrypt

```
// Create and initialize the cipher
Cipher cipher = Cipher.getInstance("AES/ECB/NoPadding");
cipher.init(Cipher.ENCRYPT_MODE, enckey);

// Encrypt the message
byte[] msg = "Content is here.".getBytes();
byte[] enc = cipher.doFinal(msg);
```

JCE: Generating Random Keys

1. Start the PRG (random seed set by default)
2. Initialize KeyGenerator with the PRG
3. Generate the key

```
// Generate a random encryption key
SecureRandom prng = SecureRandom.getInstance("SHA1PRNG");
KeyGenerator enckeygen = KeyGenerator.getInstance("AES");
enckeygen.init(prng);
SecretKey enckey = enckeygen.generateKey();
```

Counter Mode

- Not supported in JCE, must implement it yourself
- To get a “plain” cipher use ECB mode with no padding
  - Warning! CBC mode used by default
  - Need to specify “…/ECB/NoPadding”
- You can use any available block cipher

Networking

- Starter code communicates text, you need to send data
- Can use data streams

```
// Setup data streams
toServer = new DataOutputStream(clientSocket.getOutputStream());
fromServer = new DataInputStream(clientSocket.getInputStream());
```

- Can use for files as well
- Alternative: convert bytes to text

Networking: Example

- Send username and ciphertext to the server

```
// Send to server
toServer.writeUTF(username);
toServer.writeInt(enc.length);
toServer.write(enc);
toServer.flush();
```

- Receive username and ciphertext from the client

```
// Receive from Client
String username = fromClient.readUTF();
int enclength = fromClient.readInt();
byte[] enc = new byte[enclength];
fromClient.readFully(enc);
```

Implementation Issues

- Counter for CRT mode (try BigNum)
- Replay attacks (try HashMap)
- Minor issues
  - Message size not a multiple of cipher block size
  - Format of the plaintext password file
  - Exact format of files and network traffic

Starter Code

- Four Java source files

Provider code	**`ProviderGUI.java`**
Client code	**`ClientGUI.java`**
Global server code	**`AuthorityServer.java`**
Per-client server code	**`AuthorityServerThread.java`**

Submitting

- README file
  - Names, student IDs
  - Describe your design choices
- Sample plaintext content and password files
- Your sources

Grading

- Security comes first
  - Design choices
  - Correctness of the implementation
- Did you implement all required parts?
- We do not care about:
  - Cosmetics
  - Coding style
  - Efficiency

Stuck?

- Use the newsgroup (su.class.cs255)
  - Best way to have your questions answered quickly
- TAs cannot:
  - Debug your code
  - Troubleshoot your local Java installation