Secure Communications

TPRG 2131

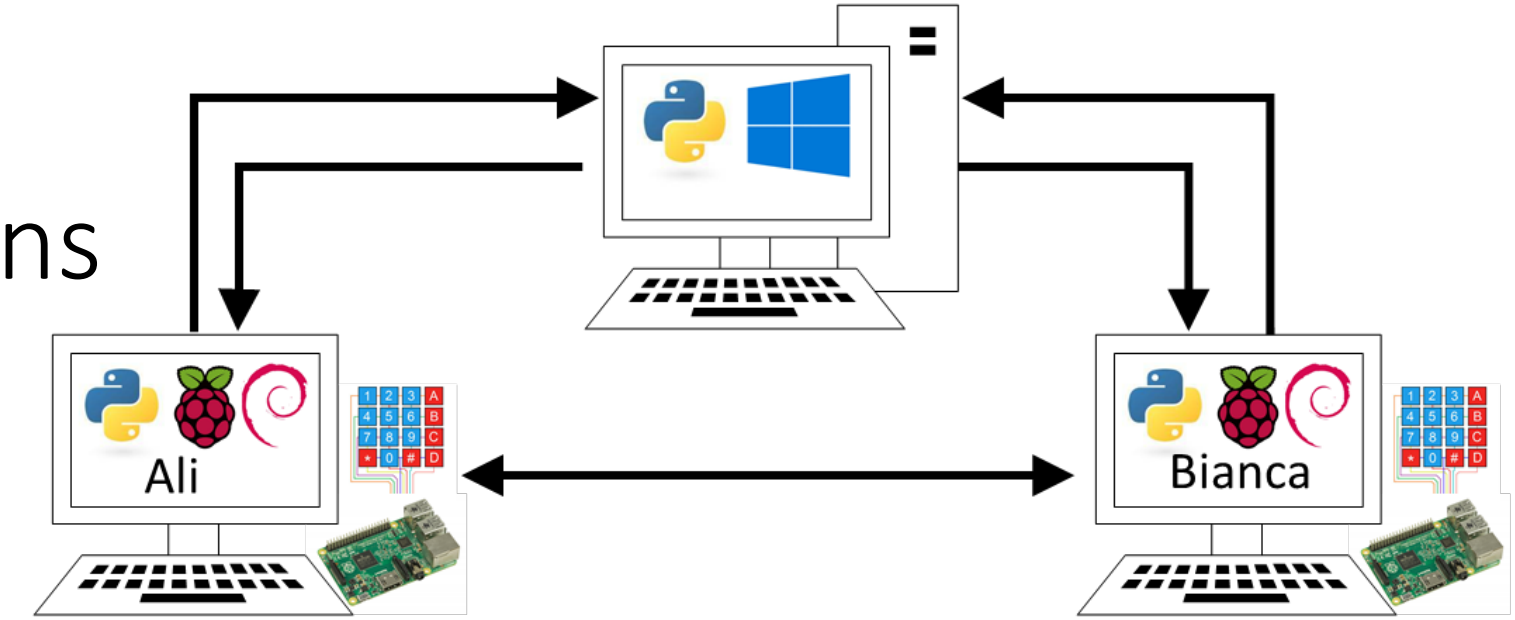
Programming

for technology II

Fall 2019

Project 2 (pairs)

Secure communications system

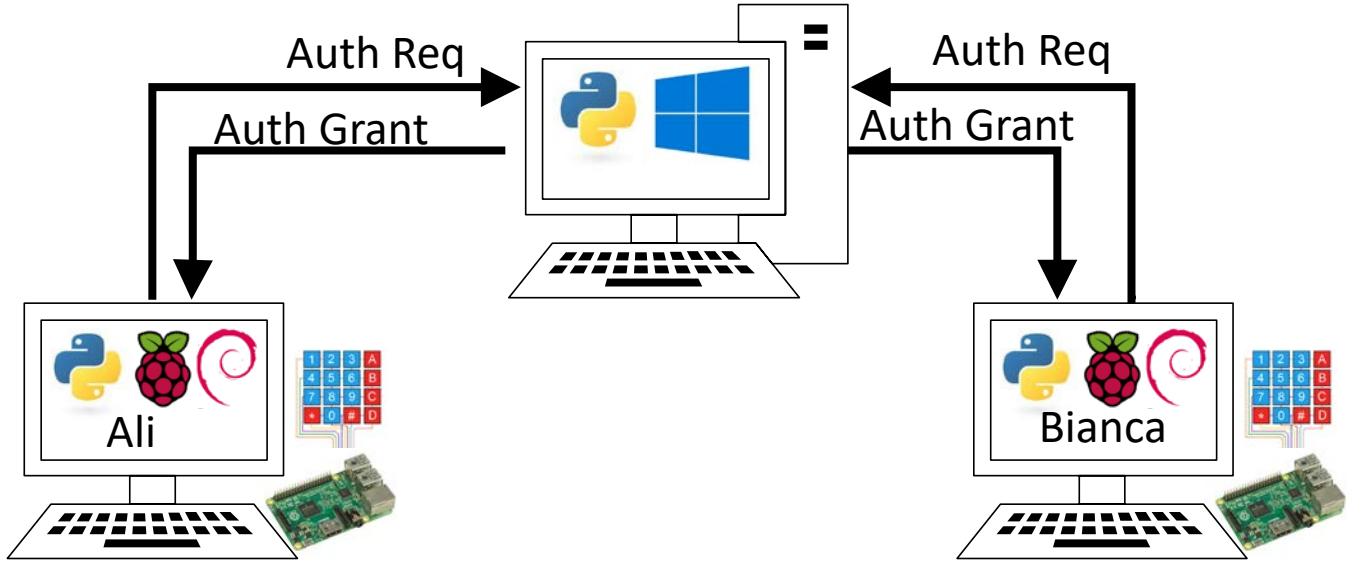


- Ali and Bianca want to communicate directly
- First they must sign in to the system
- The server knows their user ID and password
- Once logged in, the server sends them the other's IP address
- They can now communicate directly

System components

- Windows PC
  - Server application
  - Database of User IDs and hashed passwords
  - Database of logged-in users and their IP addresses
- Raspberry Pi (×2)
  - Chat application
  - 4×4 keypad (for password)

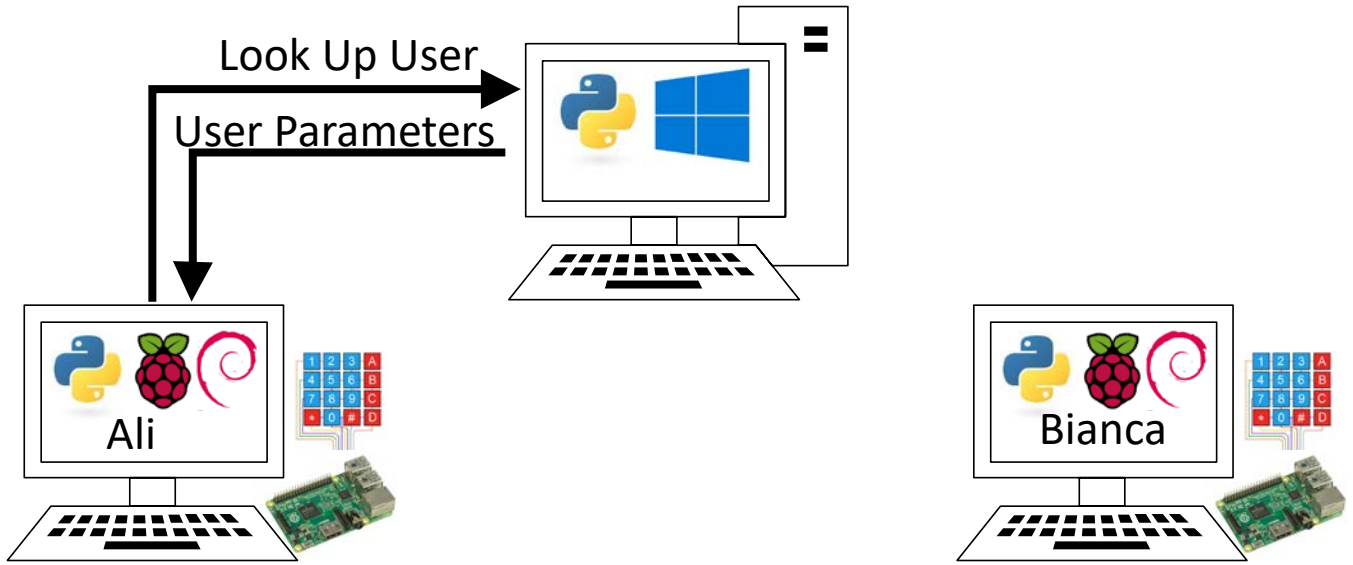
Auth phase: Ali and Bianca each sign in



Authentication phase

1. User enters user ID on Pi keyboard
2. User enters passcode on 4×4 keypad
3. Application hashes the passcode
4. Application encrypts user ID and passcode
5. Application sends authentication request with user ID & passcode to server
6. Server decrypts the user ID and passcode hash
7. Server compares hashed passcode with stored hash passcode
8. If they match
   - Server sends authentication grant message
   - Server stores user ID and user's IP address
9. Else, server sends authentication denied message

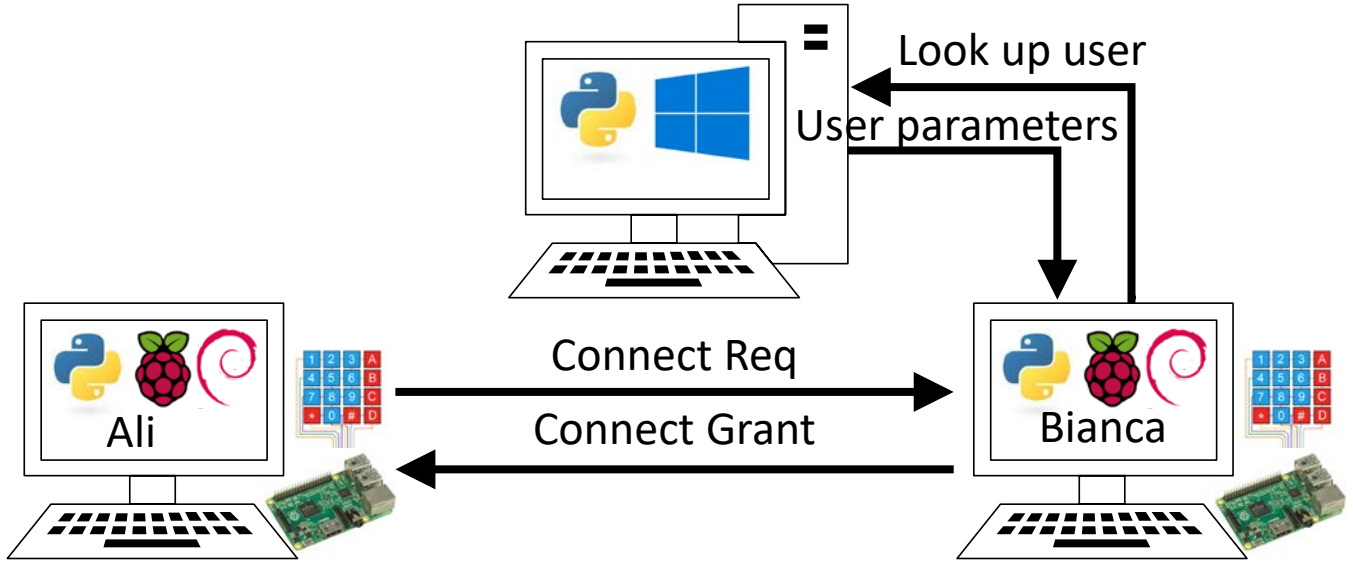
Lookup phase: Ali → Bianca



Lookup phase: Ali → Bianca

1. Ali chooses “begin chat” and enters Bianca’s ID on keyboard
2. Application sends lookup request with Bianca’s user ID to server
3. Server looks up IP address for that user
4. If user is found,
   - Server sends user IP address and encryption key
5. Else, server sends Not-Found message

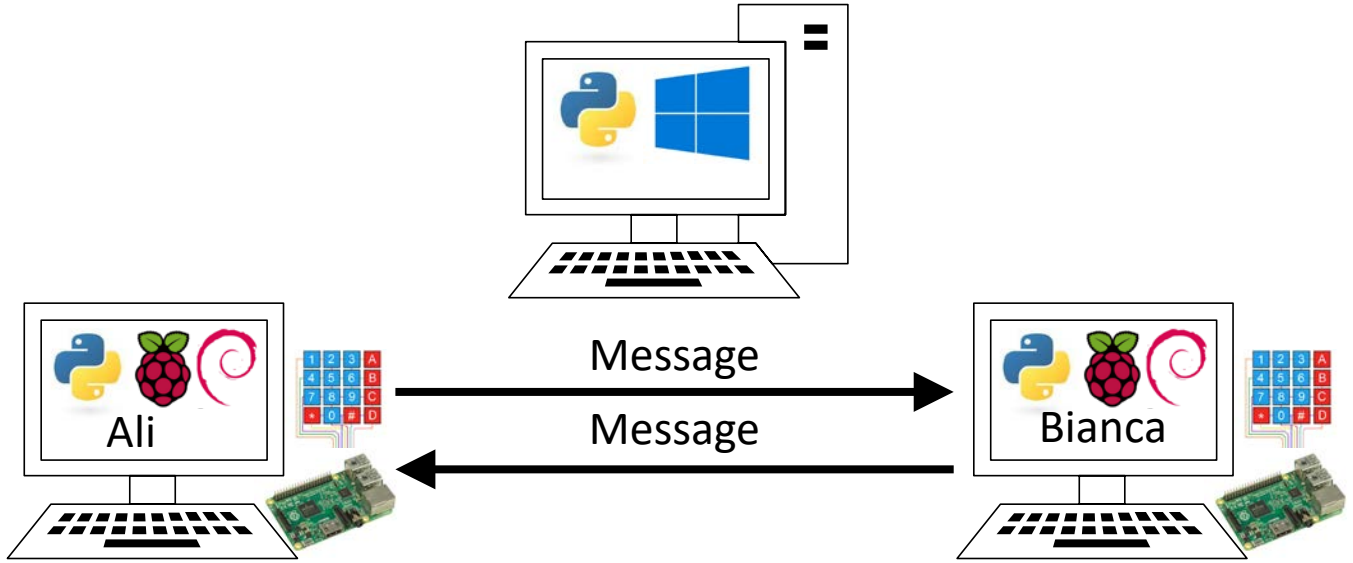
Connect phase: Ali → Bianca



Connect phase: Ali → Bianca

1. Ali's application sends connect request to the application at Bianca's IP address
2. Bianca's application sends lookup request with Ali's user ID to server
3. Server looks up Ali's user data
4. If user is found (Ali is logged in),
   - Server sends Ali's IP address and encryption key
5. Else, server sends Not-Found message
6. Bianca's application sends connect grant message to Ali's application

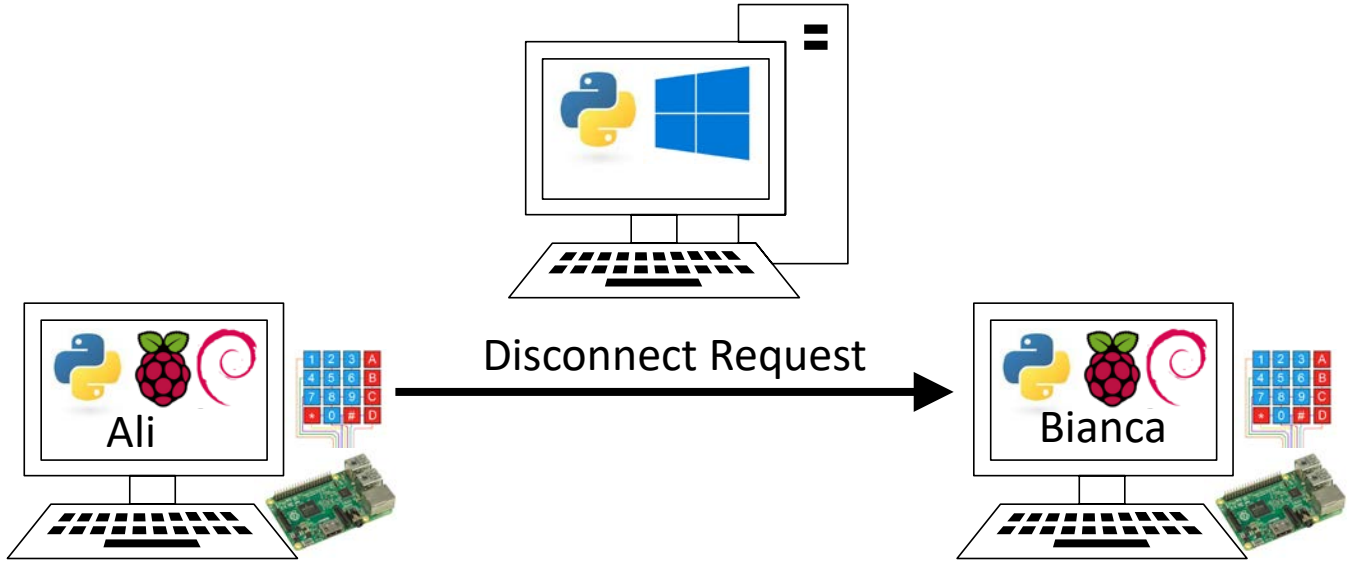
Chat phase: Ali ←→ Bianca



Chat phase: Ali ←→ Bianca

1. Ali and Bianca type lines at the Pi keyboard
2. Each application encrypts the line just entered
3. Each application sends the encrypted message to the other application
4. Each application decrypts messages received and displays them to the screen

Hangup phase: Ali | Bianca



Hangup phase: Ali | Bianca

1. Either Ali or Bianca types a CTRL-C KeyboardInterrupt
2. Application sends a disconnect request to the other application
3. Each application displays “Hang up” to the user
   (a disconnect request always results in the end of the chat session)

State machines to handle complexity

- Client Application is in one of several possible states
  - Idle – waiting for user to log in
  - Logged in – waiting for user to request a chat or for connect request
  - Requesting chat – User made a request, waiting for other side
  - Chatting – sending and receiving messages

Implementation levels

- Basic – no encryption (80%)
  - All messages including passwords (yikes!) are sent in plaintext
- Low security – simple encryption (100%)
  - All messages are encrypted
  - Passwords are hashed
  - No SSL/TLS between nodes
- High security – serious crypto (120%)
  - Python SSL module for TLS connection between nodes
  - Strong hashing algorithm with salt (Unix password hashing)

Development environment: Python + Fossil

- Shared Fossil repository on server:
  groupNN or team name (you choose the name if you wish)
- Sign up for groups on DC Connect
  - Deadline Thursday PM (or I pick pairs at random!)
- Next week: in-class exercise on shared Fossil server

Evaluation: Functional & Code quality

- Functional is in-class live test during week 13 (15% of final grade)
  - Set up both RPi clients + Windows server
  - Successful login from both clients
  - Conduct a chat session (connect – chat – hang up) from each end
- Code quality (10% of final grade)
  - State machine implementation
  - Conforms to style guide
  - Intelligent use of Fossil repository
  - All code will be evaluated from Fossil (no code on DC Connect)