Biometric Fingerprint Checking With Ipaq SDK

Group members: Michael Noeth and Jyothish Varma.

Website: www4.ncsu.edu/~jsvarma/csc714

Proposed project: We propose creating a client / server authentication protocol based on the Ipaq H5550's biometric API. Instead of using the traditional user name and password pair to authenticate users, a user name and finger print pair will be used in its place.

Completed tasks:

1. Specified the behavior of the new biometric authentication protocol (Mike and Jyothish):

Roles

The server shall be a system that has a protected resource. It will allow access to the protected resource to any system that presents a valid authentication token. The server is responsible for (1) distributing authentication tokens, (2) allowing access to a resource, and (3) adding new users into a list of "authorized" users.

The client shall be a system that requests access to a protected resource from the server. The client will be able to (1) obtain an authentication token and (2) trade in that token for access to a protected resource.

Data Structures (both reside on the server)

user_lookup: A list of unique user names and corresponding biometric finger print data (stored in BioAPI_BIR data structure).

token_lookup: A list of unique token numbers paired with a valid token. The actual token will be based on the system clock. Although this is somewhat insecure, we are only creating a prototype. To ensure better security, it is recommended that the system time be combined with a random number to better avoid attacks.

Processes

Obtaining a token:

Step 1: Client sends ASCII user name.

Step 2: Server accepts user name.
Step 3: Client sends BioAPI_BIR data structure containing finger print data.
Step 4: Server accepts BioAPI_BIR data structure.
Step 5: Server looks uses its local user_lookup data structure to find if the user name / BioAPI_BIR data structure match.
Step 6a: If they match – the server generates a token based on system time and a unique token number. The token is hashed via the md5 hash function and sent to the client along with the token number.
Step 6b: If they do not match – the server sends an error code: "Invalid user name / password".
Step 7: Client accepts response.

Using a resource:
Step 1: Client sends token / token number pair
Step 2: Server accepts the token number and looks up the corresponding token number in its token_lookup data structure.
Step 3a: If the md5 hash of the stored token match the sent token – access is granted to the resource.
Step 3b: If the md5 hash of the stored token match the sent token – access is restricted.
Step 4: Server sends a newly generated token / token number pair to the client.

2. Download, install, and practice with Embedded Visual C++ (Mike and Jyothish)
    a. We each attempted to install Embedded Visual C++ on our own personal machines. We both failed and decided to simply use the OS lab facilities.
    b. Each of us went through various tutorials on creating applications for the Pocket PC. We have a few sample "hello world" type programs available on request.
3. Research communication options for the iPAQ H5550 (Jyothish): There were three communication mediums available for use; 802.11 Wireless Connection, Bluetooth, and Infra red. Socket communication via 802.11 wireless card satisfy all of our communication needs.
4. Communication prototype (Mike and Jyothish)
    a. Description: We developed a simple client / server application that would transmit a virtual ball back and forth. The server setup a TCP socket and began listening on a pre-set port. Clients were able to connect to the server and transmit a payload. The payload gets sent back and forth between the client and server five times before the client terminates. The server continues to listen for additional connections. Both the server and the client output messages to indicate at what stage they have reached in the simple communication pattern described above.
    b. Problems encountered: We encountered major difficulties setting up the iPAQs' wireless capabilities. We have documented the procedure required to setup the 802.11 to ensure we don't run into this problem again.
    c. The client and server programs are available upon request.

5. Research iPAQ's biometric API (Jyothish): We needed to obtain a copy of the biometric API available for the model of Pocket PC that we are working on. After a lot of searching we found the API available: http://devresource.hp.com/drc/downloads/index.jsp#a095b8d84802485d2
6. Biometric prototype (Mike)
    a. Description: To get a feel for the biometric API provided by HP, we created a simple application utilizing it. The application allows a user to scan two finger prints in and then compares them. A message displays whether the prints match or not.
    b. The prototype is available upon request.

Remaining milestones:

1. Create simple prototype using both communications and biometrics. This prototype will have a client and server. The client will request a finger print and send the data to the server. The server will compare the finger print to a stored finger print and report back to the client whether the finger print matched or not. We hope to complete this milestone by Friday, November 4, 2005.
2. Create a prototype of the authentication prototype in which the server has a database of users and finger prints. Multiple clients should be able to connect to this database by specifying a user name and a finger print. We hope to complete this milestone by Monday, November 7, 2005.
3. Build the biometric authentication into an existing client / server application for the Pocket PC. Time transactions with biometric authentication versus the same application without biometric authentication. This will give an idea of the overhead the biometric authentication will add to the program. We hope to complete this milestone by Monday, November 14, 2005.
4. Testing and verification of the system.