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. Create prototype application to generate user / finger print pairs for the server (Mike and Jyothish): The server must keep a record of username / finger print pairs in order to authenticate valid users. This will be accomplished by storing finger print data in a binary file named after the user whose finger print is represented. In order to accomplish this task – we created a prototype that contained the necessary functions for saving finger scan data in a file and then reading it back out of the file.

   To ensure we had this capability, our prototype performed the following:
   a. The user scans in a finger print. The data is stored in container_1.
   b. The system writes container_1 to file.
   c. The system reads the file and stores the replicated container_1 in container_2.
   d. The user scans their finger a second time. The data is stored in container_3.
   e. container_1 and container_3 are compared.
   f. container_2 and container_3 are compared.
   g. The results of step e and step f were ensured to match.

   When a finger print is scanned in using the BioAPI provided by HP, a BioAPI_BIR is returned (see figure 1). It consists of three data members: (1) the header contains various meta information including the length of the actual data portion (2) the data itself, and (3) a signature (in the HP BioAPI implementation this data member contains no information).

```
typedef struct bioapi_bir {
   BioAPI_BIR_HEADER Header;
   BioAPI_BIR_BIOMETRIC_DATA_PTR BiometricData; /* length header */
   BioAPI_DATA_PTR Signature; /* NULL if no signature; */
} BioAPI_BIR, *BioAPI_BIR_PTR;
```

Figure 1 – BioAPI_BIR is the return value of a finger scan on HP iPAQ

   A binary file (currently named "mytemp") is generated and the contents of the scanned BioAPI_BIR are stored in the file. There are eight integers in the header portion of the BioAPI_BIR, and these are stored space delimited at the beginning of the binary file. The actual binary data is stored right after the header.

To ensure that everything was functioning properly – two test sequences of input were required:

a. Matching scans
    i. User 1 scans finger for container_1 / container_2
    ii. User 1 scans finger for container_3
    iii. Comparing container_1 and container_3 results in a match
    iv. Comparing container_2 and container_3 results in a match
b. Mis-matching scans
    i. User 1 scans finger for container_1 / container_2
    ii. User 2 scans finger for container_3
    iii. Comparing container_1 and container_3 results in a mis-match
    iv. Comparing container_2 and container_3 results in a mis-match

2. Create prototype client application (Jyothish): The client requiring access to a server's resources must be able to send a user name and finger print scan. This prototype set out to prove that it was possible to send the user's information over the network to a server application.

   The first step in creating the client was to create a GUI containing two text boxes and a button. The first text box allows the user to specify the remote server by IP. The second text box allows the user to specify their ID. After specifying these values – the button is used to send the data over the network.

   The client application currently functions as follows:
    a. User inputs target IP address and ID in the appropriate text boxes
    b. User clicks button
    c. User is prompted to scan finger – click OK
    d. User scans finger
    e. Data is sent over the network

   Code from the original network client (toy ping pong application) was recycled to send three separate packets to the specified server. The first packet is of fixed size and contains the user's name. The second packet is also of fixed size and contains the header data. The final packet is of variable size (specified in the previous header packet) and contains the BioAPI_BIR data.

3. Create prototype server application (Mike): The server providing resources must be able to read a user ID / finger print scan over the network and reply with a response to clients.

   In order to create this prototype – another GUI was built for debugging purposes. The GUI continually updates what the system is doing (i.e. waiting for a connection, connected, authenticating, etc).

The server application currently functions as follows:

a. Server listens on pre-specified port
b. Upon a connection – the server reads three packets
    i. User name (fixed size)
    ii. Header data (fixed size)
    iii. BioAPI_BIR data (variable size based on data from header)
c. Server reads in appropriate binary file based on user name provided (code from prototype described in 1).
d. Server compares networked BioAPI_BIR with the local BioAPI_BIR from the file
e. Server responds with "success" or "failure" based on comparison in step d.

Remaining milestones:

1. Clean up "generate user" prototype: This application should allow the server user to easily add users. A GUI must be provided that will allow someone to specify a user name and a button to commence the scanning process. The scanning process should still contain three steps to ensure a good scan:

    a. The user scans in a finger print. The data is stored in container_1.
    b. The system writes container_1 to binary file using user name provided.
    c. The system reads the file and stores the replicated container_1 in container_2.
    d. The user scans their finger a second time. The data is stored in container_3.
    e. container_2 and container_3 are compared – on failure go to step a.

    This extra step ensures that a good scan is stored on the server for authentication purposes. The target date for this step is Tuesday, November 22, 2005.

2. Clean up "server" prototype: The server is required as per our original specification to issue tokens upon successful authentication. A system that stores tokens and can verify tokens must be implemented. The target date for this step is Tuesday, November 22, 2005.

3. As time provides – extra functionality is desired to be built into the system. This includes (1) the client remember the last IP address recorded, (2) the server records a history of finger prints and performs a memory comparison – if any matches occur we know that a man-in-the-middle attack has occurred and a malicaious user may be trying to replay a former authentication sequence, (3) the create user application should also allow for deleting users. These will be added in as time allows

4. Testing and verification of the system. The hard deadline for this step is to begin Tuesday, November 22, 2005 – Tuesday, November 29, 2005.