Trường Đại Học Kỹ Thuật TP. Hồ Chí Minh
Khoa Công nghệ Thông tin
---0---

BÁO CÁO KỸ THUẬT

Thử nghiệm Hệ thống Quản Lý & Điều khiển truy xuất dùng Fingerprint

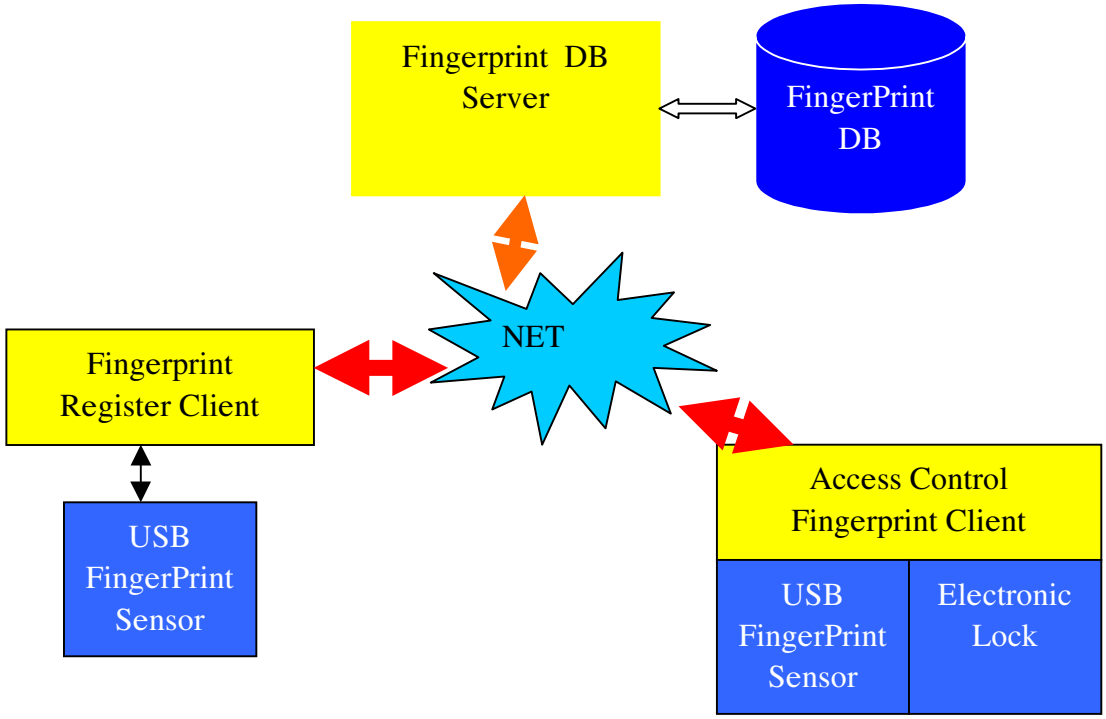
Trương Hồng Lĩnh
thlinh@dit.hcmut.edu.vn

08/2000

Mục lục

1. Mô hình tổng quát của hệ thống

- Mô hình hệ thống được mô tả sau đây :



- Trong hệ thống trên có 3 thành phần chính :
  - ♦ Fingerprint database server : lưu trữ thông tin người dùng (bao gồm cả feature của registration fingerprint)
  - ♦ Fingerprint Register Client: dùng đăng ký fingprint cho người dùng
  - ♦ Access Control Fingerprint Client : dùng điều kiển vào ra
- Trong hệ thống trên có hai thành phần khác đó là :
  - ♦ USB Fingerprint sensor : sensor dùng lấy fingerprint
  - ♦ Electronic lock : khoá điện tử dùng khoá mở hệ thống cửa.
- Toàn bộ hệ thống trên được kết nối qua mạng đo vậy có thể điều khiển & truy xuất từ xa.

2. Card điều khiển khoá điện tử

2. 1. Giới thiệu

- Card điều khiển khoá điện tử là thiết bị giao tiếp với cổng parallel và điều khiển đóng mở rờ le điện tử có nguồn là 12 vôn. Card này được thiết kế bởi nhóm làm việc

của Mr. Nguyễn Cao Trí (caotri@dit.hcmut.edu.vn). Card này có thể điều khiển nhiều khoá điện tử trong một thời điểm.

2.2. Thư viện điều khiển đơn giản

- Thư viện này chỉ dùng để điều khiển 2 cổng. Đường line thứ nhất dùng chỉ tính hiệu thông báo và đường line thứ 2 dùng điều khiển khóa mở cửa. Khi tín hiệu xuất ra là 1 thì cửa sẽ mở và 0 thì cửa sẽ đóng.

```
#define OPENNINGDOORTIME 5000
#define WAITINGINPUTTIMER 20000
 void ControlDoorProcedure(int Value, int port)
{
          _outp(port,Value);
}
void ControlOpenDoorProcedure()
{
     ControlDoorProcedure(0x03,0x378);
     Sleep(OPENNINGDOORTIME);
     ControlDoorProcedure(0x00,0x378);
}
void ControlErrorProcedure1()
{
     ControlDoorProcedure(0x00,0x378);
     Sleep(300);
     ControlDoorProcedure(0x01,0x378);
     Sleep(300);
     ControlDoorProcedure(0x00,0x378);
}
void ControlErrorProcedure2()
{
     for (int i=0 ; i < 3;i++)
     {
          ControlDoorProcedure(0x01,0x378);
          Sleep(300);
          ControlDoorProcedure(0x00,0x378);
          if ( i < 2)
               Sleep(300);
     }
}
```

- Thời gian đóng mở một khoá điện tử là tùy theo ứng dụng. Thông thường khoảng thời gian đóng mở cửa là 3-5 giây.

3. Sơ lược về fingerprint sensor & Lập trình U.A.U fingerprint toolkit

3.1. Fingerprint sensor & fingerprint toolkit

- Fingerprint sensor là thiết dùng nhận dạng fingerprint. Thiết bị này kết nối qua chuẩn USB. Đi kèm theo thiết bị là bộ tookit (SDK) dùng để lập trình phát triển ứng dụng.

Toolkit này hiện thực hầu hết các chức năng như lấy fingerprint, lưu trữ vào database, so sánh. Fingerprint toolkit (SDK) bao gồm các module sau :

- Feature Extraction Module : các chức năng lấy feature của fingerprint
- Matching Module : dùng so sánh các feature của fingerprint với nhau
- Database Module: dùng lưu trữ các feature
- High Level interface module: các functions ở level cao có registration & verfication
- Sensor Server
- Security Measures

- Quá trình thử nghiệm tại khoa CNTT chỉ sử dụng 2 chức năng cơ bản của toolkit là lấy fingerprint và so sánh các fingerprint với nhau. Các API này được định nghĩa trong:
  - dpFpFns.h : thiết lập, kết thúc, quản lý toolkit. Lấy thông tin hình ảnh fingerprint, đăng ký và kiểm chứng.
  - dpMatch.h : các hàm so sánh các feature fingerprint

các chức năng khác có thể tham khảo tài liệu đi kèm của fingerprint toolkit.

- Fingerprint thông qua sensor sẽ được quét và đưa vào xử lý. Hình ảnh của fingerprint có thể được dùng hiển thị cho người dùng. Tuy nhiên để xác định một fingerprint người ta không dùng hình ảnh của fingerprint mà sử dụng các đặc tính của fingerprint đó. Đặc tính của fingerprint được trích (extracting) từ fingerprint và gọi là feature. Các feature này tùy theo loại mà có thể dùng lưu trữ lại hay dùng để so sánh với feature khác. Khi hai feature được so trùng và có sự giống nhau theo một mức nào đó (tùy theo ứng dụng qui định) thì xem như hai feature đó được trích ra từ một fingerprint.

3.2 Lập trình U.A.U Fingerprint toolkit

3.2.1. Khởi tạo hệ thống toolkit U.A.U

a) Khởi tạo & kết thúc module lấy fingerprint

- Trước khi sử dụng hệ thống toolkit U.A.U thì chúng ta phải thực hiện thao tác khởi tạo. Quá trình khởi tạo thực hiện theo những bước sau đây :
  - Khởi tạo thư viện FT (FT_Init)
  - Tạo context (FT_createContext)
  - Lấy danh sách thiết bị (FT_getDeviceList)
  - Kết nối tới thiết bị (FT_connectDevice)

- Mỗi sensor kết nối vào máy tính sẽ có chỉ số từ 1 đến n. Để thuận tiện cho việc thử nghiệm toàn bộ quá trình này được khai báo lại trong hàm connectDevice() (FTObject.h). Đoạn code sau đây diễn tả việc khởi tạo cho 1 U.A.U fingerprint sensor :

```
FT_RETCODE rc ;
rc=connectDevice(1);
if (rc !=FT_OK) {
     AfxMessageBox("Error    in    connected    to
device",MB_OK);
     return FALSE ;
```

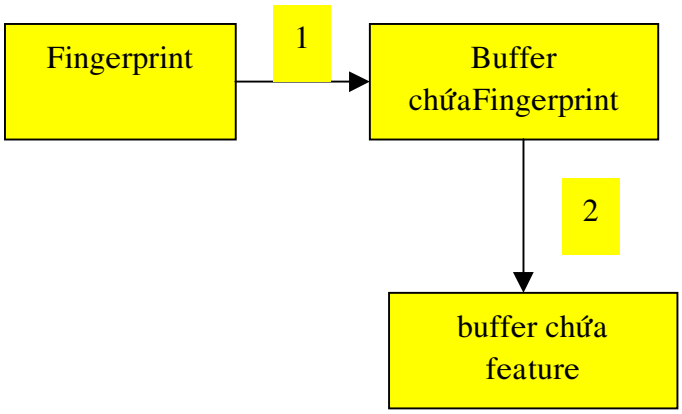
```
}
```

connectDevice() sẽ nhận vào thông số là số Id của sensor và thực hiện khởi tạo. Bản chất của hàm này là thực hiện khởi tạo dùng các hàm cơ bản của FingerPrint toolkit cung cấp.

- Để có thể lấy được thông tin về hình ảnh, feature của fingerprint thì phải thiết lập quá trình cho phép nhận được những thông đó tới user. SDK cung cấp phương cách trong đó các UI (User Interface) có thể nhận được thông tin từ device. Các quá trình UI thuộc sự điều khiển của ứng dụng. Quá trình tương tác giữa tookit và ứng dụng thông qua các callback functions. Để thiết lập cơ chế này chúng ta dùng hàm FT_setupUIlink(). Trong thử nghiệm quá trình này được viết lại trong hàm setupUILink() với một số thông số mặc định cơ bản. Thực hiện setup UI như sau :

```
rc=setupUILink();
if (rc !=FT_OK) {
     AfxMessageBox("Error in setup UI",MB_OK);
     return FALSE ;
}
```

setupUILink() thiết lập các thông số để giao tiếp với user interface. Để có thể lấy feature của một fingerprint chúng ta lưu ý quá trình lấy sau :



(Xem thêm FTObject.h)

- Trong quá trình (1) thì fingerprint sẽ được sensor nhận dạng và đưa vào buffer chứa fingerprint. Thông tin buffer này có thể được hiển thị dưới dạng một ảnh (xem class CGray8Dib). Quá trình (2) sẽ diễn ra việc lấy feature của một fingerprint. Feature được lấy dựa vào đặc tính là registration feature hay là verification feature. Để có thể lấy được feature thì yêu cầu thêm 2 thông số là kích thước của feature (feature len) và vùng đệm chứa feature. Các feature sau khi có được có thể dùng hay lưu trữ lại. Bên cạnh các kết quả chứa trong buffer thì các event sẽ được tạo ra để báo trạng thái hoạt

động của quá trình như : lấy hình ảnh tốt, lấy được feature hay không (xem cấu trúc FT_UI_LINK).
- Sử dụnt FT_terminate() để kết thúc và giải phóng tài nguyên đã cấp cho module này.

b) Khởi tạo & kết thúc Matching module

- Module này cho phép sử dụng các hàm để so sánh các fingerprint với nhau. Các API này được khai báo trong dpMatch.h. Quá trình khởi tạo bao gồm hai bước:
  - Khởi tạo thư viện (MC_Init)
  - Khởi tạo context (MC_createContext)

- Sau khi khởi tạo thì có thể sử dụng các API trong module này. Khi kết thúc thì chúng ta sẽ dụng API MC_terminate().

3.2.2 Ví dụ xây dựng một ứng dụng dùng fingerprint

- Trong ứng dụng này chúng ta thực hiện các quá trình sau :
  - Lấy registration feature của fingerprint lưu vào database
  - Lấy verification feature của một fingerprint và so sách với registration fingerprint đã lưu trước đó.

a) Lấy registration feature:

- Trước khi lấy thì phải khởi tạo hệ thống toolkit.
- Tạo 1 thread để lấy fingerprint từ thiết bị

```
UINT DoRegRegister(LPVOID Param)
{
     FT_RETCODE rc ;
          rc=GetFeaturesFT();
          if (rc !=FT_OK)
                return 0L ;
          return 1L ;
}
AfxBeginThread(DoRegRegister,GetSafeHwnd(),THREAD_PRIORITY_
NORMAL);
```

Hàm GetFeatturesFT() được khai báo trong module FTObject.h như sau :

```
FT_RETCODE GetFeaturesFT()
{
     FT_RESULT result ;
     FT_RETCODE rc ;
     rc = FT_register(ftContext,
```

```
                FT_FALSE,
                g_recommendedFtrLen,
                g_Features,
                NULL,
                &result);
     if (result ==FT_SUCCESS)
     {
     SendMessage(dlgAddUser.GetSafeHwnd(),MESSAGE_FT_SUCCESS
, 0,0);
     }
     else
     SendMessage(dlgAddUser.GetSafeHwnd(),MESSAGE_FT_NOTSUCC
ESS, 0,0);
     return rc ;
}
```

hàm này sẽ blocking và thoát ra khi đã nhận xong một registration fingerprint. Quá trình lấy một registration fingerprint đòi hỏi sensor phải lặp lại một số lần quá trình lấy fingerprint (điều này yêu cầu user phải đưa một finger vào sensor nhiều lần). Trong trường hợp thành công (FT_SUCCESS) thì kết quả thành công sẽ chứa trong g_Features . Khi nhận được sự kiện MESSAGE_FT_SUCCESS(do ứng dụng tạo ra) có nghĩa là hàm lấy registration feature đã thành công và khi đó ta có thể lưu trữ lại feature :

```
     memcpy(m_FIRSTFINGER,g_Features,g_recommendedFtrLen);
     m_FirstFingerFlag=TRUE ;
```

Trong quá trình lấy thông tin thì fingerprint toolkit sẽ báo lên một số các event(hàm getRegisterAction trong FTObject.h là một ví dụ) và chúng ta có thể dùng các event để biết được tình trạng :

```
FT_ACTION getRegisterAction (FT_STATUS_PT pStatus, void
*pParam)
{
     switch (pStatus->code) {
      ...
     case FT_FEATURES_INFO:

     SendMessage(dlgAddUser.GetSafeHwnd(),MESSAGE_FT_FEATURE
S_INFO, 0, (LPARAM)pStatus->param1);
                break ;
      ....
     }
    return FT_ID_CONTINUE;
}
```

tuỳ theo người phát triển ứng dụng mà thông tin đó sẽ xử lý tương ứng ví dụ như :

```
case FT_GOOD_FTR:
        GetDlgItem(IDC_EDIT_STATUS)->SetWindowText("Good
          feature obtained ! Pls. waiting for a short time
          and put the same finger to the sensor if you
          don't receive other instruction");
        m_pGray8Dib->UsePalette(pDC, true);
        m_pGray8Dib->Draw(pDC, 0, 0, g_FPBuffer);
        break;
```

(xem thêm source code và document đi kèm)

b) Lấy một verification feature từ một fingerprint.

- Quá trình lấy cũng thực hiện tương tự như trên tuy nhiên chúng ta sử dụng API khác :

```
rc = FT_acquireFeatures(ftContext, FT_VER_FTR,
     g_recommendedFtrLen,         g_VerFeatures,&qualityImg,
     &qualityFtr,&result);
```

c) Thực hiện so sánh:

- Sử dụng các hàm MC_*() để thực hiện việc so sánh. Trước hết chúng ta lấy registration feature từ database (hay đã được lấy trước bằng phương các trên). Sau đó thực hiện so sánh :

```
     pVoid =::GlobalLock(m_pSet->m_THIRDFINGER.m_hData);
                                break ;
                           }

     CurrentFinger++ ;
     ::memcpy(g_Features,pVoid ,g_recommendedFtrLen);
     ::GlobalUnlock(pVoid);
/*
 *  Call the verify procedure to performance verification
 */
     rc =
     MC_verifyFeatures(mcContext,g_recommendedFtrLen,g_Feat
ures,g_recommendedFtrLen,g_VerFeatures,FT_FALSE,
                          &b,featuresKey,&score,&verified);
/*
 * If the verification is success with the high score, we
assume that
 * the authentication is successful and permittion is
allowed
 */
```

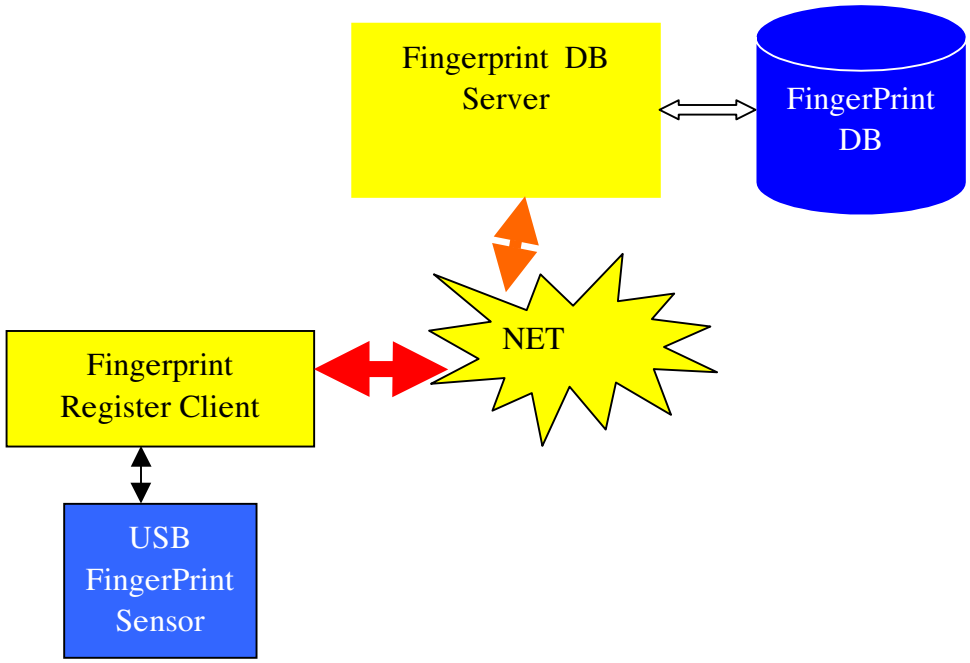
```
     if((rc==FT_OK)&&(verified==FT_TRUE)&&(score ==
FT_HIGH_VER_SCORE)) {
```

tuỳ theo kết quả mà chúng ta sẽ thực hiện tác vụ so sánh. Quá trình so sánh cũng có thể learing để lấy được feature tốt hơn. Ở đây score diễn tả cho độ chính xác khi so trùng.

4. Chương trình quản lý Fingerprint

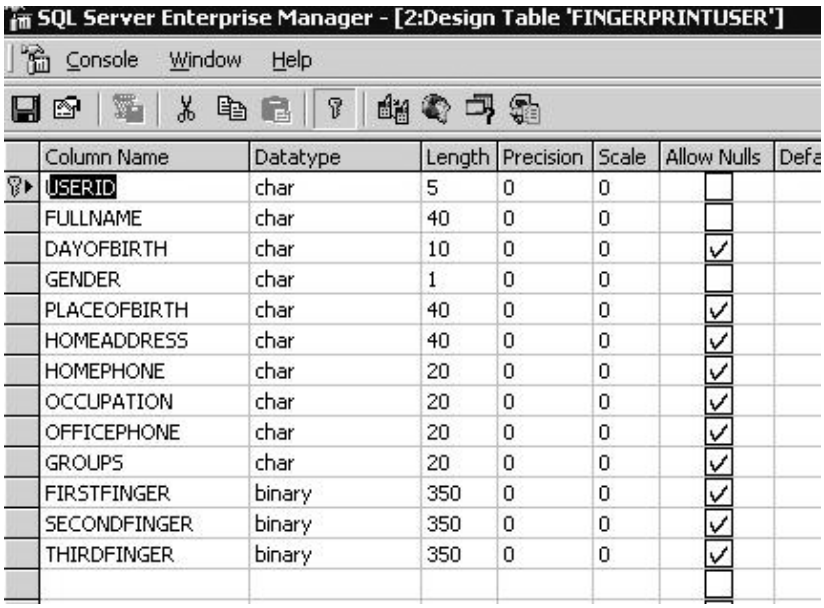
4.1. Mô tả hệ thống đăng ký fingerprint

- Mục tiêu là thấy các thông tin của người dùng như thông tin cá nhân, thông tin công việc , và các registration features của các fingerprint lưu trữ vào databases.
- Mô hình :



4.2.Cấu trúc cơ sở dữ liệu FINGERPRINTUSER:

- Hệ cơ sở dữ liệu fingerprint dùng là SQL server 7.0 trên môi trường Windows. Cấu trúc bản lưu trữ thông tin như sau :

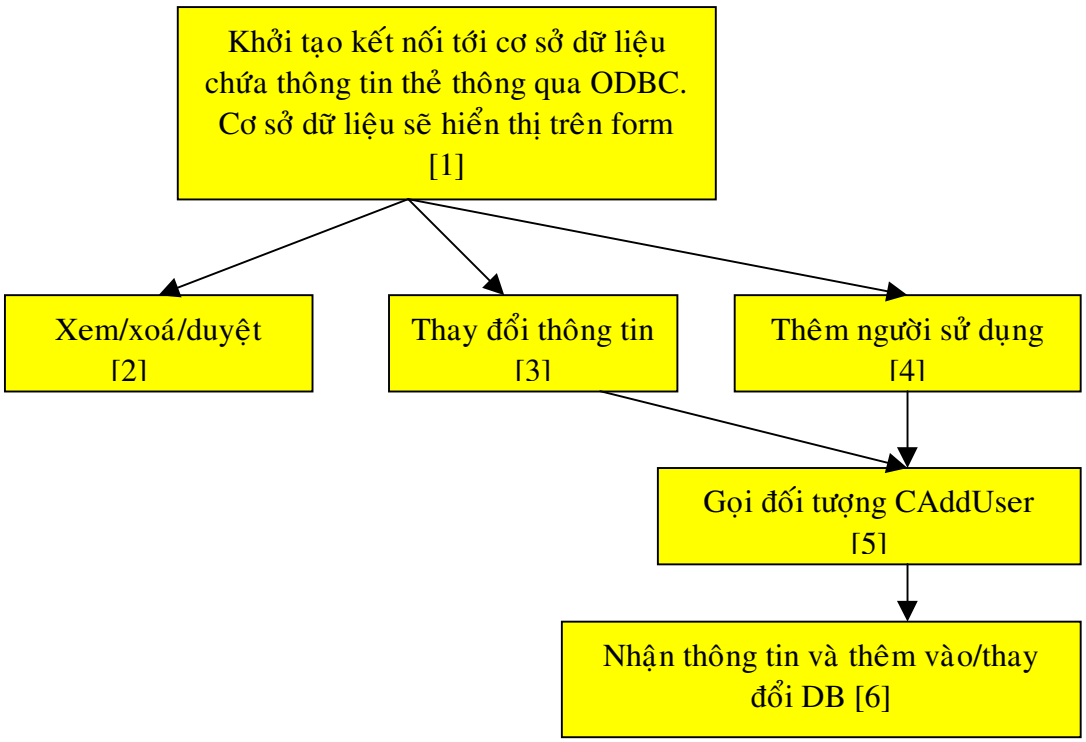


Column Name	Datatype	Length	Precision	Scale	Allow Nulls	Defa
USERID	char	5	0	0	☐	
FULLNAME	char	40	0	0	☐	
DAYOFBIRTH	char	10	0	0	☑	
GENDER	char	1	0	0	☐	
PLACEOFBIRTH	char	40	0	0	☑	
HOMEADDRESS	char	40	0	0	☑	
HOMEPHONE	char	20	0	0	☑	
OCCUPATION	char	20	0	0	☑	
OFFICEPHONE	char	20	0	0	☑	
GROUPS	char	20	0	0	☑	
FIRSTFINGER	binary	350	0	0	☑	
SECONDFINGER	binary	350	0	0	☑	
THIRDFINGER	binary	350	0	0	☑	
					☐	

- Trong đó trường FIRSTFINGER , SECONDFINGER và THIRDFINGER dùng lưu trữ 3 fingerprint của một nhân viên. Hệ cơ sở dữ liệu này được phép thay đổi, bổ xung thông qua một account có quyền đọc ghi.

4.3 Chương trình đăng ký

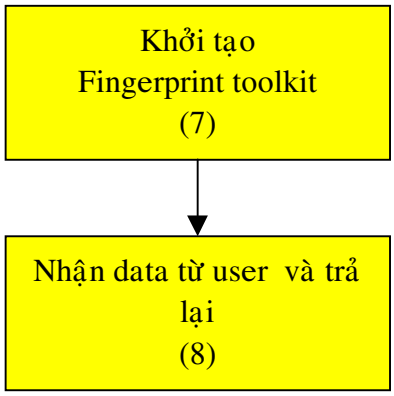
- Mô hình hoạt động của chương trình đăng ký



♦ Xem chi tiết source phần giải thuật :

[1] CFingerManagerView::OnInitialUpdate() , CFingerManagerODBCSet
[2]CFingerManagerView::OnButtonDelete(), :OnEditDelete() , ::OnRecord[First, Next, Prev, Last]()
[3]CFingerManagerView::OnEditUpdate() , ::OnButtonUpdate()
[4]CFingerManagerView::OnEditAdd(),::OnButtonAdd()
[5][6]CAddUser , CFingerManagerView::OnEditAdd()

- Đối tượng CAddUser() là đối tượng sẽ nhận thông tin đăng ký bao gồm các thông tin về fingerprint. Đối tượng này là một global object và hoạt động như sau:

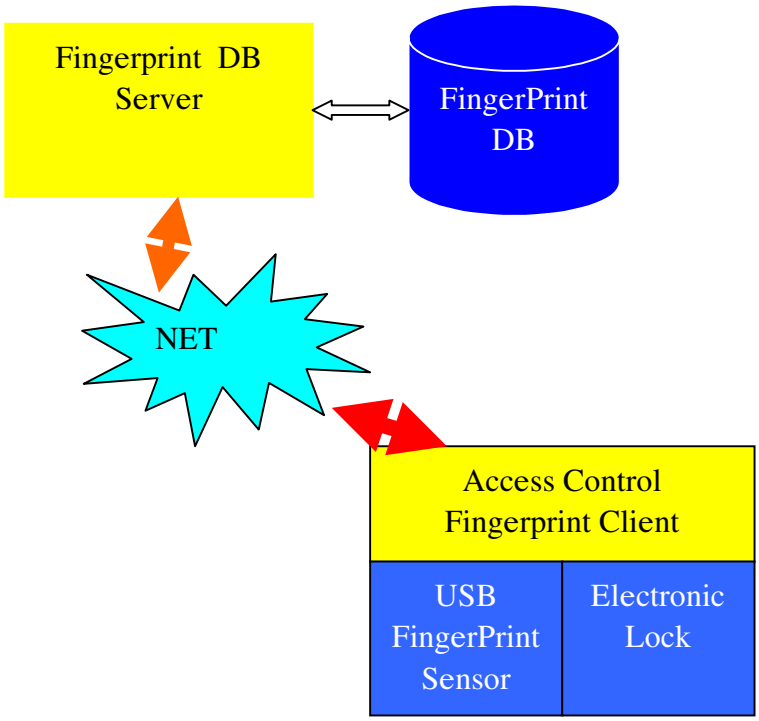


quá trình lấy fingerprint trong đối tượng CAddUser nói riêng và nói chung trong hệ chương trình access control dùng fingerprint được thực hiện như trong phần 3.2.

5 Mô tả hệ thống access control

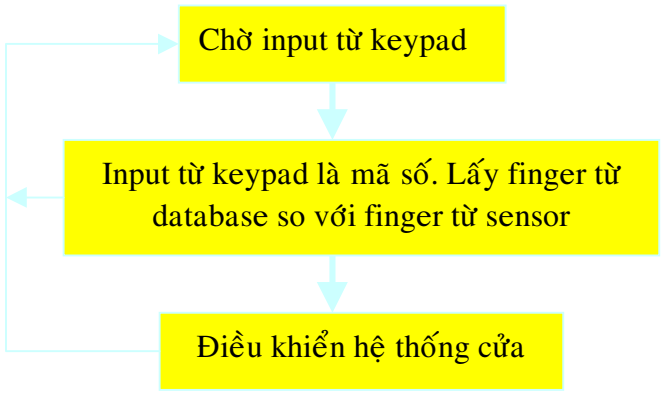
5.1 Mục tiêu và mô hình

- Nhận thông tin người dùng từ keypad và fingerprint sensor sau đó thực hiện kiểm tra xem có hợp lệ hay không. Trong trường hợp người dùng là hợp lệ thì sẽ điều khiển thiết bị mở cửa.



5.2 Chương trình

- Trong mô hình hoạt động ở 5.1 thì cơ sở dữ liệu fingerprint là cơ sở dữ liệu đã được thiết lập như trong đề cập ở mục 4. Chương trình Access Control là chương trình chạy nền. Hoạt động tổng quát như sau :



- Chương trình access control là một chương trình chạy nền (background) đó vậy input từ keyboard không thể sử dụng phương pháp thông thường. Trong chương trình sử dụng kỹ thuật keyboard hook để bắt phím nhập từ key board. Như vậy bất kỳ lúc nào cũng có thể nhận biết có thông tin input từ keyboard. Kỹ thuật này được trình bày trong phần mục lục. Ngoài ra để đảm bảo chương trình hoạt động 24/24 thì sử dụng chương trình monitor giám sát hoạt động của chương trình access control. Khi chương trình

access control không hoạt động thì monitor sẽ gọi chương trình access control hoạt động.

6. Các vấn đề đặt ra để giải quyết trong tương lai

- Một số vấn đề đề nghị nên tiếp tục giải quyết trong quá trình thử nghiệm:
    - Portable các ứng dụng trên Windows NT/2000
    - Hoàn thiện mạch khoá điện tử & thư viện điều khiển như là một module độc lập để ứng dụng vào nhiều trường hợp.
    - Mở rộng đường cable của USB
    - Chương trình Access Control dạng Services
    - Các thông tin cần thiết lưu trữ trong registry hoặc mã hoá (Vd : thông tin user, password để đọc fingerprint từ database)
    - Checking trên nhiều database
    - Chiến lược điều khiển truy xuất
    - Hệ thống logfile và thống kê trên logfile
    - Hệ thống fault-tolerance cho Access Control program

7. Tham khảo

[1] Visual C++ 6.0 Enterprise Edition
[2] U.A.U Toolkit Software and SDK
[3] Source Codes

- List sources :
    - ATLib.zip (Thư viện keyboard hook)
    - FingerprintManager.zip (Quản lý đăng ký fingerprint)
    - AccessControl2.zip (chương trình access control)
    - MonitorApp.zip (chương trình Monitor chương trình access control)

Phụ lục:

AT LIB : KEYBROAD HOOK LIB FOR ACCESS CONTROL PROGRAM

1. Giới thiệu

- Chương trình Access Control là một chương trình thực thi background. Khi người sử dụng nhập vào mã số thì trên cơ sở mã số đó chương trình sẽ tìm kiếm feature của fingerprint đã đăng ký ở trong database để so sánh với feature nhập vào. Yêu cầu đặt ra là làm thế nào để có thể lấy được mã số của người dùng khi họ dùng keyboard trong khi chương trình chạy background. Sử dụng kỹ thuật keyboard hook sẽ cho phép chúng ta thực hiện được điều đó.

2. Thư viện ATLib.DLL

- Thư viện ATLib.dll hiện thực cơ chế keyboard hook và bắt tất cả những thông tin người dùng nhập vào từ keyboard. Sau đó filter lại và trong trường hợp người dùng nhập vào là một chuỗi số thì dll sẽ trả về cho ứng dụng (ứng dụng gọi dll) chuỗi trên. Như vậy trong trường hợp input từ keyboard là một chuỗi số thì có thể xem là mã số người dùng và chương trình access control sẽ thực thi viện lấy các feature đăng ký và nhập vào từ sensor để so sánh.

3. Source codes

- Chương trình nguồn ATLib.zip
- Thư viện DLL : ATLib.dll
- Header file : ATLib.h

MONITOR : Giám sát chương trình access control

Chương trình này là một chương trình monitor đơn gọi chương trình access control và chờ cho đến khi chương trình access control kết thúc thực thi. Khi chương trình access control kết thúc thực thi thì monitor sẽ cho rằng đó là kết thúc ngoài mong muốn và gọi chương trình access control thực thi trở lại. Xem source code trong MonitorApp.zip