Monday 21 March 2022

ELEC-C7420 Basic Principles in Networking Spring 2022

# Assignment V: Endpoint Authentication

PSALTAKIS GEORGIOS

Aalto University
School of Engineering

# Goals of the experiment

This experiment is quite similar to the first assigment but the main differences are that its implement as stand alone using the arduino microcontroller. The purpose of the expiriment is to actually use the capabilities of this specific arduino which includes an arduino antenna built in. So far we havent implemented anything with the Wireless capabilities of the microcontroller. This arduino sketch is made to print the mac adress of the device scan for wifi networks as well as their authentication protocol and signal and then connect to the preset wifi in the code after three searches.

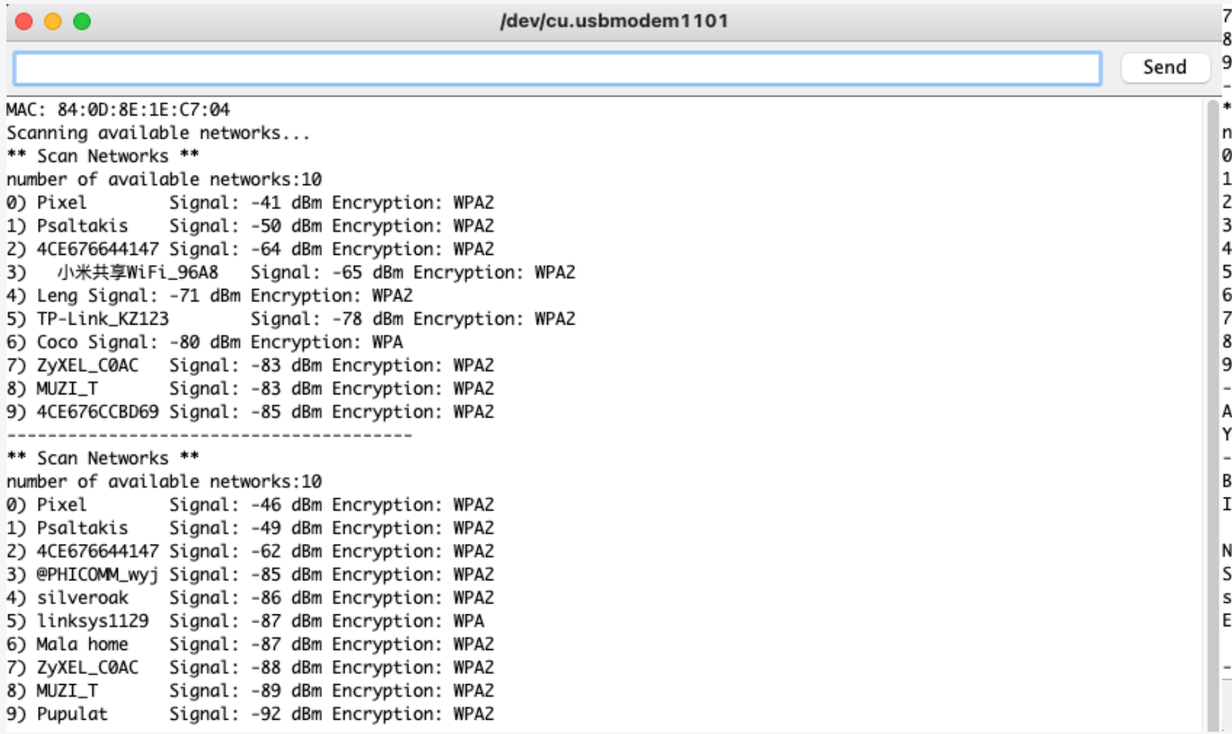
# Experimental Setup

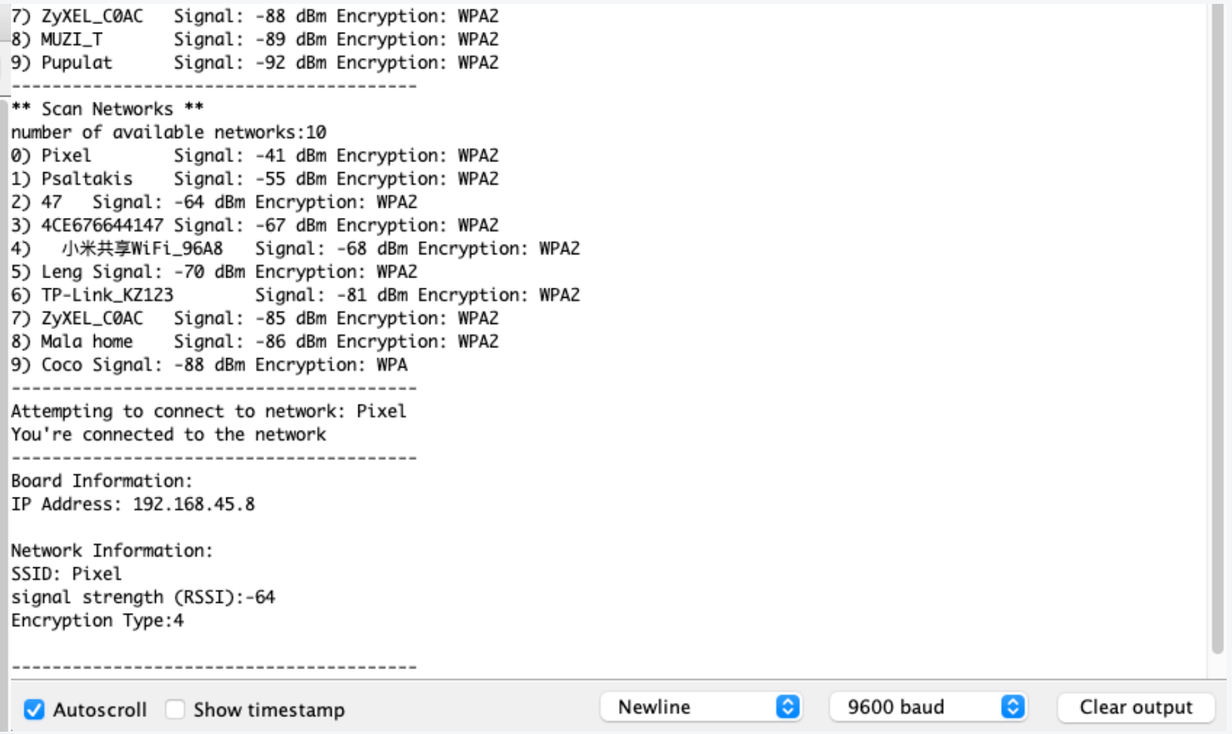
The setup of the experiment is simple. We have implemented the code that firstly prints out the mac adress. In continuance we have created in the void loop a code that searches and decodes the available wifi networks by assigning them the security type of authentication and signal stength. We have implemented that using a for loop running the search only for three times and then trying to connect to the ssid we have written in the beggining of the code alongside the password we gave. If this combination works it will login to the wifi network and then exit the loop terminating the programm completely. The implemented code need the WifiNINA library which we installed to our arduino programm and the code was heavilly influeanced by the arduino official documentation for the MKR WIFI 1010 and by combining two codes and adjusting them to work in one sketch and execute exactly what the assigment requested.

- https://docs.arduino.cc/tutorials/mkr-wifi-1010/connecting-to-wifi-network
- https://www.arduino.cc/reference/en/libraries/wifinina/wifi.scannetworks/

# Results & Conclusion

Ive splitted the screenshot to be more visible but its clearly visible that the code begins by printing the mac adress and then continiues by searching three times and then procceds by connecting to the selected wifi with password that has been set up in the beggining.



# Annex

Users > georgepsaltakis > Desktop > CODE > CODE.ino

```
#include <SPI.h>
#include <WiFiNINA.h>


char ssid[] = "Pixel";        // your network SSID (name)
char pass[] = "6972329644";    // your network password (use
int status = WL_IDLE_STATUS;


void printEncryptionType(int thisType) {
  // read the encryption type and print out the name:
  switch (thisType) {
    case ENC_TYPE_WEP:
      Serial.println("WEP");
      break;
    case ENC_TYPE_TKIP:
      Serial.println("WPA");
      break;
    case ENC_TYPE_CCMP:
      Serial.println("WPA2");
      break;
    case ENC_TYPE_NONE:
      Serial.println("None");
      break;
    case ENC_TYPE_AUTO:
      Serial.println("Auto");
      break;
    case ENC_TYPE_UNKNOWN:
    default:
      Serial.println("Unknown");
      break;
```

Users > georgepsaltakis > Desktop > CODE > CODE.ino

```
void printMacAddress(byte mac[]) {
  for (int i = 5; i >= 0; i--) {
    if (mac[i] < 16) {
      Serial.print("0");
    }
    Serial.print(mac[i], HEX);
    if (i > 0) {
      Serial.print(":");
    }
  }
  Serial.println();
}


void printData() {
  Serial.println("Board Information:");
  // print your board's IP address:
  IPAddress ip = WiFi.localIP();
  Serial.print("IP Address: ");
  Serial.println(ip);

  Serial.println();
  Serial.println("Network Information:");
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());

  // print the received signal strength:
  long rssi = WiFi.RSSI();
  Serial.print("signal strength (RSSI):");
  Serial.println(rssi);
```

CODE.ino

Users > georgepsaltakis > Desktop > CODE > CODE.ino

```
  byte encryption = WiFi.encryptionType();
  Serial.print("Encryption Type:");
  Serial.println(encryption, HEX);
  Serial.println();
}

void setup() {
  //Initialize serial and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }

  // check for the WiFi module:
  if (WiFi.status() == WL_NO_MODULE) {
    Serial.println("Communication with WiFi module failed!");
    // don't continue
    while (true);
  }

  // print your MAC address:
  byte mac[6];
  WiFi.macAddress(mac);
  Serial.print("MAC: ");
  printMacAddress(mac);
  delay(10000);
  // attempt to connect to Wifi network:
}

```

CODE.ino

Users > georgepsaltakis > Desktop > CODE > CODE.ino

```

void loop() {
  // scan for existing networks:
  Serial.println("Scanning available networks...");
  listNetworks();
  delay(10000);
  Serial.println("--------------------------------------");

}

void listNetworks() {

  for (int x = 0; x < 3; x++) {

  // scan for nearby networks:
  Serial.println("** Scan Networks **");
  int numSsid = WiFi.scanNetworks();
  if (numSsid == -1) {
    Serial.println("Couldn't get a WiFi connection");
    while (true);
  }

  // print the list of networks seen:
  Serial.print("number of available networks:");
  Serial.println(numSsid);

  // print the network number and name for each network found:
  for (int thisNet = 0; thisNet < numSsid; thisNet++) {
    Serial.print(thisNet);
    Serial.print(") ");
    Serial.print(WiFi.SSID(thisNet));
    Serial.print("\tSignal: ");
    Serial.print(WiFi.RSSI(thisNet));
    Serial.print(" dBm");
```

CODE.ino

Users > georgepsaltakis > Desktop > CODE > CODE.ino

```
      Serial.print(") ");
      Serial.print(WiFi.SSID(thisNet));
      Serial.print("\tSignal: ");
      Serial.print(WiFi.RSSI(thisNet));
      Serial.print(" dBm");
      Serial.print("\tEncryption: ");
      printEncryptionType(WiFi.encryptionType(thisNet));
    }
    Serial.println("----------------------------------------");
    }

      while (status != WL_CONNECTED) {
      Serial.print("Attempting to connect to network: ");
      Serial.println(ssid);
      // Connect to WPA/WPA2 network:
      status = WiFi.begin(ssid, pass);

      // wait 10 seconds for connection:
      delay(10000);
    }

    // you're connected now, so print out the data:
    Serial.println("You're connected to the network");

    Serial.println("----------------------------------------");
    printData();
    Serial.println("----------------------------------------");
    exit(0);
  }
```

The Code is actually pretty simple and its refferenced from the arduino documentation. We firstly have a list to decode and understand what type of encryption the network is using so we can assign it while we scan. Then we pretty much print the mac adress while creating the functions to print the data we request down at the list networks where all the major work is done. There we run three times the scanning and then once the connection and then exit the program itself. Above in setup we initialise the arduino for errors and port ans to print the MAC

# Answer of the given questions

- Which authentication methods did you find for 802.11?
  - Open Authentication
  - Wired Equivalent Privacy (WEP)
  - Wi-Fi Protected Access (WPA)
  - Wi-Fi Protected Access 2 (WPA2)

- Please describe three authentication methods in detail / Application Scenario

**Open Authentication**

This method is as the name implies with no need of knowing any preshared key or credential to connect with the network. After selection and auth the client is associated with the AP. This doesn't mean that it has to be completely open to using since there can be further authentication on the web browser before the traffic is unblocked.

**Application Scenario**

This authentication method is usually used for mass use APs that are usually not for personal use. For example, this could be a free city WIFI or airport wifi that will not ask for anything else or for example a hotel or conference wifi that will ask for your credentials afterwards to let you access any internet traffic. This could also be a university wifi network that requires the users to connect with their academic email after the connection for them to use the internet.

## WEP

This method has indeed encryption using the RC4 cipher to encrypt every frame so the contents are not able to be read. When users have the same WEP key they can decrypt each others messages. Its uses either open auth or shared key auth. With open auth anyone can enter the network but their data frames will be encrypted. With shared key their frames plus authentication will use the WEP encryption. So the user needs the correct key.

## Application Scenario

Unfortunately, the WEP has been broken since 2001 and its use has been minimized since 2004. The application scenario was for almost everything since it provided some sort of encryption of the data frames as well for personal APs since it can also have a key to be accessed. But now it's not recommended anymore for secure WLAN since the data can be decrypted easily and the key can be cracked

## WPA

This method has indeed encryption using the RC4 stream cipher to encrypt every frame so the contents are not able to be read. WPA is only with a pre shared key which must be configured in both the AP and client. It complies with the wireless security standard and it increases the level of data protection of a wifi. It enforces the IEEE 802.1x auth an key exchange and can be seen with different naming variation . (WPA2 is a bit enchanced by ensuring that the ap and client are using the the same wpa version and key )

## Application Scenario

WPA and WPA2 is the more widespread authentication method at this time for secure WLAN. Its used from small offices to houses to almost anywhere that requires better encryption and security of the network. It's not incredibly secure(Exploitations were found on 2021)  but with more updates to the version from wpa2 to wpa3 its the best available for widespread secure use.