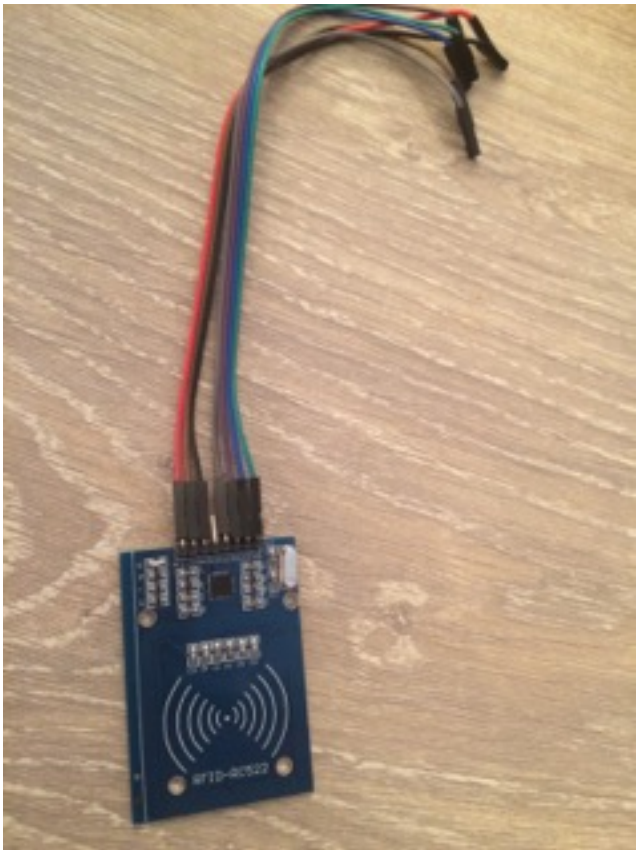
Project Information Security

RFID Stealer



Stefan Blommaert
Thomas Van Havere
Lorenz Put

10 januari 2016

Voorwoord

Wij Stefan Blommaert, Thomas Van Havere en Lorenz Put hebben het project RFID stealer toegewezen gekregen. Hierbij moesten we proberen aantonen hoe we informatie van een RFID kaart konden halen en dit proberen te klonen op een nieuwe kaart. Dit hebben we gedaan aan de hand van een raspberry pi en een Arduino uno in combinatie met een RFID reader (genaamd RC522).

Doelstelling

Dit project gebruikt de RC522 om RFID kaarten en tags uit te lezen. De uiteindelijke bedoeling van het project is om door middel van de RC522 reader de gegevens van een bestaande RFID kaart te kopiëren op een nieuwe kaart. Hiervoor hebben we eerst gebruik gemaakt van een raspberry pi. Toen dat we merkten dat dit niet echt lukte zijn we verder gegaan met een Arduino.

Taakverdeling

Thomas

- De nodige bibliotheken op raspberry pi geïnstalleerd
- Basis RFID code

Stefan

- Geavanceerde RFID code met Arduino en raspberry pi
- Documentatie raspberry pi

Lorenz

- Geavanceerde RFID code met Arduino
- Documentatie Arduino

Voorbereiding

In het begin is er geopteerd geweest om met een raspberry pi te werken, omdat we voor ons ander project reeds op een raspberry pi aan het werken waren. De bedoeling was om deze RFID reader te integreren in het andere project. Om deze reden was research naar de RC522 reader en raspberry pi noodzakelijk.

Na verloop van tijd hebben we ons gerealiseerd dat het werken met een Arduino simpeler was omdat we er meer ervaring mee hebben. Daarom hebben we dus besloten om over te schakelen naar Arduino. Om deze was dus ook research naar de werking tussen de Arduino en de RC522 reader noodzakelijk.

Benodigdheden

Voor het afgeleverde project hebben we volgende items/componenten gebruikt:

- Arduino uno
- Raspberry pi
- RC522 reader
- RFID tags/cards

Prijskaartje

De school beschikte over een RFID reader, maar deze werkte niet op de juiste frequentie. Dus hebben we onze eigen RFID reader aangekocht. Zie hieronder een tabel met de gebruikte componenten.

Wat	Hoeveelheid	Kost
RC522 reader	1	8 €
RPI B+	1	40 €
Arduino uno	1	0 €
RFID tags/cards	4	5 €
Totaal	7	53 €

De totale prijs voor dit project is €53. Onze componenten zijn afkomstig van hobbyelectronica.nl.

Research

Active en passive

Er bestaan actieve en passieve RFID kaarten. Passieve kaarten worden het meest gebruikt. Een actieve kaart heeft een eigen interne batterij, waardoor hij geen stroom nodig heeft van de lezer. Een passieve kaart heeft geen interne batterij en zijn meestal kleiner dan actieve kaarten. De stroom om een passieve kaart uit te lezen komt van het magnetisch veld van de kaartlezer. Om deze reden heeft een actieve kaart een groter bereik dan een passieve kaart.

Frequenties

RFID is een technologie die op vier verschillende frequentiebanden kan werken. De eerste band is de 125kHz of low frequency , de tweede is de 13,65 MHz of high frequency, 860/950 MHz of de Ultra high frequency en de 2,45 GHz of de microwave frequency. Al die verschillende banden hebben hun eigen voor - en nadelen. Zo hebben de hogere frequenties meestal een grotere range en een grotere overdracht snelheid.
Het soort kaarten dat wij proberen te kraken, zoals studentenkaart, sorteerpasje, Antwerpen - stad pasje, bevinden zich allemaal op de 13,56 MHz band. Om deze reden werken wij met de RC522, deze werkt enkel op de13,56 MHz band.

Structuur MiFare Card

Er bestaan 2 verschillende van versies van deze kaarten, namelijk de 1K versie en de 4K. Deze getallen bepalen de hoeveelheid data de kaarten op kunnen slagen. Het spreekt dus voor zich dat een 4K kaart niet gekopieerd kan worden op een 1K kaart. Omgekeerd gaat wel vermits het eerste deel van de 4K kaart over de structuur beschikt als de 1K kaart. Verder beschikken deze elk over een verschillende chipset.

Deze kaarten beschikken elk over een 4-byte UID. Deze UID wordt gebruikt om de kaart uniek te identificeren binnen de numerieke grenzen van de 4 bytes.

EEPROM geheugen

De structuur van de kaarten bestaan uit 16 sectoren, die nog eens onderverdeeld zijn in 4 blokken, waarvan iedere blok dan weer uit 16 bytes bestaan. In totaal is dit dus 1024 bytes of 1K. Elke sector blok kan geconfigureerd worden met verschillende toegangsvoorwaarden. De laatste blok van iedere sector, die ook de sector trailer genoemd wordt, bevat de authenticatie van deze sector. De authenticatie bestaat uit 2 sleutels namelijk A en B. Sleutel A wordt gebruik om te lezen, terwijl sleutel B gebruikt wordt om te kunnen schrijven. Hierdoor zijn er eigenlijk maar 3 blokken per sector beschikbaar om data op te slagen. De bruikbare opslagcapaciteit bedraagt dus eigenlijk maar 768 bytes in plaats van 1024 bytes.

Manufacture block

Block 0 is speciaal omdat het de fabricage blok is. In dit blok kan je de data van de fabrikant terugvinden. Deze data kan niet aangepast worden en zou ontweken moeten worden tenzij je weet wat je doet.

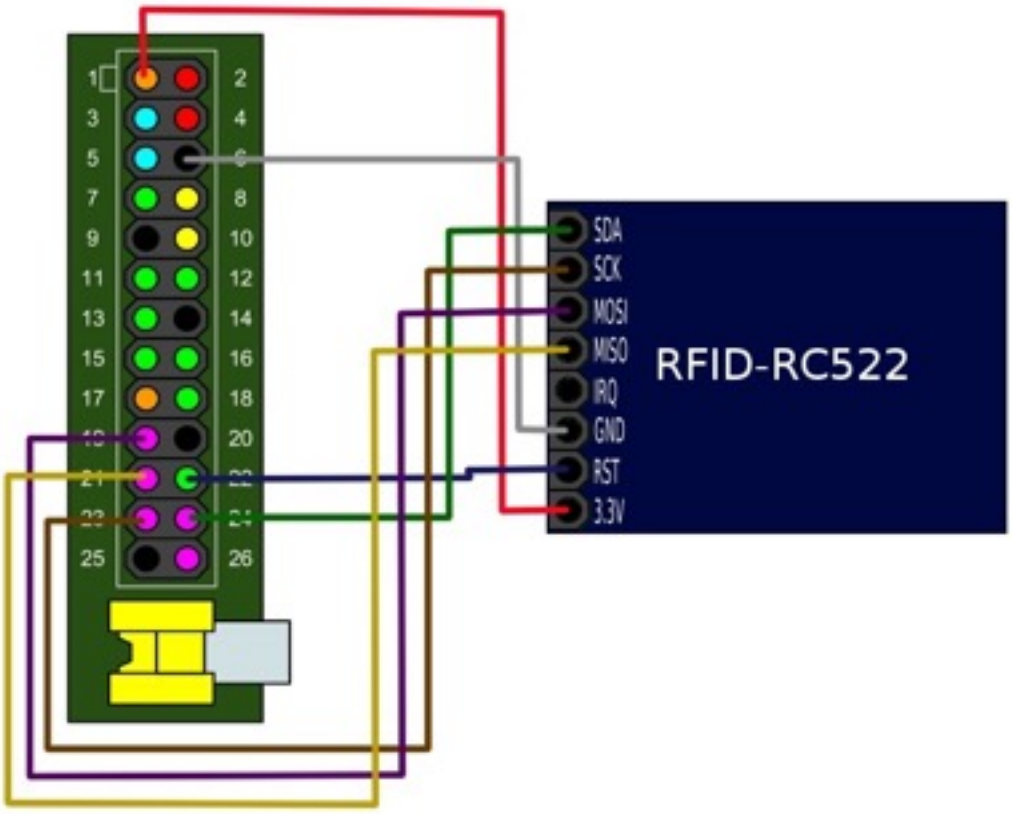
Datablocks

In de datablokken of ook value blokken genoemd, wordt de effectieve data weg geschreven. Naast data kunnen deze blokken ook elektronische functies zoals read, write, decrement, restore en transfer uitvoeren.

Elk value blok bevat een integer met teken die 32 bits lang is. Om de data integriteit te garanderen en voor beveiligingsredenen wordt elke waarde in de value blok drie keer opgeslagen. Van deze drie keer dat de waarden opgeslagen wordt, is er één geïnverteerd. De laatste 4 bytes die overschieten worden gebruikt om één adres vier keer in te bewaren. Wederom 2 keer geïnverteerd en 2 keer niet.

Raspberry pi

Hardware



Op bovenstaande afbeelding is te zien hoe we de RFID reader hebben aangesloten op de raspberry pi. De layout van de pinnen van de raspberry pi kan verschillen naargelang de versie. Merk op dat de IRQ pin niet aangesloten is.

Software

Vooraleer er geprogrammeerd kon worden, waren er verschillende acties nodig om de Raspberry Pi compatibel te maken met de RC522 software.
Om dit tot een goed einde te brengen zijn volgende tutorial
https://www.youtube.com/playlist?list=PLP7qPet500dcE-zP_-EVEisi7N1Lh4Ekk
en volgende github repository nodig https://github.com/mxgxw/MFRC522-python.

Read file

De 'Read.py' file zal proberen een kaart te detecteren, wanneer we een werkende en nieuwe (zal sebiet duidelijker worden bij authenticatie over de kaart) kaart op de RFID reader houden zal het programma de kaart detecteren en UID van de kaart geven (UID is een unieke 4-cijferige nummer van een kaart).

Het programma geeft een standaard authenticatie sleutel mee '0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF' (dit is de standaard authenticatie sleutel bij nieuwe kaarten op elke sector)

Deze code: 'status = MIFAREReader.MFRC522_Auth(MIFAREReader.PICC_AUTHENT1A, 3, key, uid)'
zal proberen authentiseren op de kaart, waar nu het cijfer '3' staat, dit wil zeggen dat hij in block 3 van sector 0 zal proberen authentiseren (er zijn 4 blocken per sector en de laatste block van een sector zit de authenticatie ook wel sector trailer genoemd)

maar door deze code: 'MIFAREReader.MFRC522_Read(0)' zal hij block 0 van sector 0 uitlezen als de authenticatie gelukt is. (Block 0 van sector 0 is de manufacturer block => dus blijf hier af zolang je niet weet waar je mee bezig bent !)

Wanneer je de sector trailer zou uitlezen zal je 16 bits op het scherm zien, waarvan je onder de key A bits een nul zal zien ook al is deze authenticatie sleutel helemaal anders, maar voor de veiligheid zie je op deze manier nooit je authenticatie key A.

Write file

De 'Write.py' file zal de aangeduide block van een sector vullen met 16 bits (hexadecimaal van 0 tot 255), dit zal alleen lukken als de authenticatie weer gelukt is zoals bij de 'Read.py' file.

Het programma zal wachten totdat hij een kaart detecteert en hiervan de UID op het scherm weergeven en proberen authentiseren met de standaard key.

Deze code: 'status = MIFAREReader.MFRC522_Auth(MIFAREReader.PICC_AUTHENT1A, 3, key, uid)'

Dit stukje code zal weer van op block 3 van sector 0 de authenticatie key gebruiken om te authentiseren.

```
# Variable for the data to write

        data = []

# Fill the data with 0xA

        for x in range(0,16):

        data.append(0xA)
```

Dit stukje code zal 16 bits met 0xA vullen in de block die verder in de code gekozen wordt.

Het volgende dat zal gebeuren bij deze code:

```
# Read block 2

MIFAREReader.MFRC522_Read(2)

print "\n"
```

Hij zal block 2 van sector uitlezen wat er al in staat, daarna zal hij de bits die in de data staat in block 2 zetten en nu zal hij opnieuw block 2 uitlezen met de nieuwe ingevulde bits dan, wanneer dit allemaal gebeurd is zal het programma stoppen.

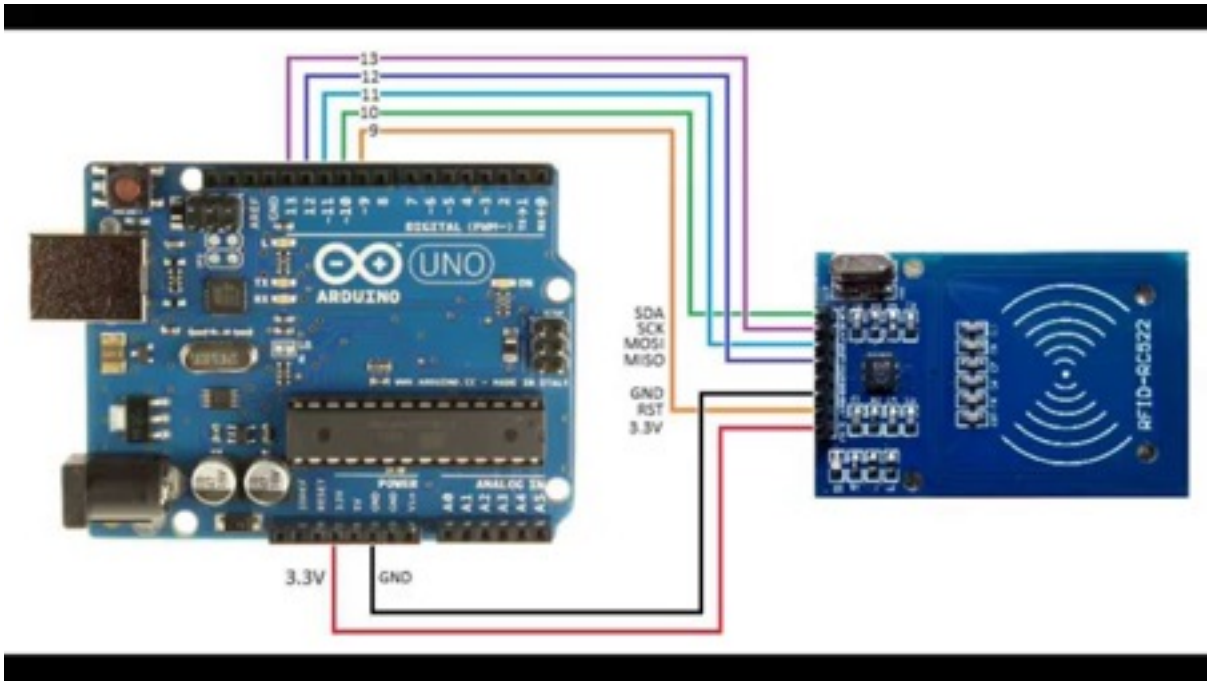
Dump file

De ‘Dump.py’ file zal proberen alle blokken van de sectoren uit te lezen en uit te schrijven op het scherm, zodat we een visueel beeld krijgen wat er allemaal op de kaart geschreven staat.

Eerst zal het programma detecteren of er een kaart gescand wordt, wanneer er een kaart gedecteerd is zal hij de UID naar het scherm wegschrijven. Met de default authenticatie key zal hij terug proberen authenticeren op de kaart, wanneer dit gelukt is zal hij heel de kaart doorlopen en alle gegevens van alle blokken en sectoren uitschrijven op het scherm. Je moet wachten tot hij volledig alle blokken doorgelopen is voordat je de kaart wegneemt of op CTRL-C drukt om het programma te stoppen.

Arduino

Hardware



Op bovenstaande afbeelding is te zien hoe we de RFID reader hebben aangesloten op de Arduino uno. De layout van de pinnen van de Arduino pi kan verschillen naargelang de versie. Merk op dat ook hier de IRQ pin niet aangesloten is.

Software

Voor Arduino hebben we code van deze github gebruikt en verwerkt naar ons project.

https://github.com/miguelbalboa/rfid.git

De algemene bedoeling dat er moet gebeuren in de Arduino code is dat hij een RFID card/ tag gaat uitlezen en al deze data gaat opslagen in variabelen in het programma. Dan een nieuwe kaart op de reader gaat leggen en al de data gaat kopiëren op de nieuwe kaart, behalve de data uit de authenticatie blokken. De reden hiervoor is dat als men deze blokken uitleest, men voor de eerste 6 bits (authenticatie sleutel A) altijd 6 nullen leest ook al heeft de authenticatie sleutel een heel andere waarde, dit is een veiligheid voor het kopiëren van kaarten. Dus wanneer we de uitgelezen authenticatie sleutel A uitlezen en deze willen kopiëren, zullen we dus altijd 6 nullen kopiëren op de nieuwe kaart voor de authenticatie sleutel A. Hierdoor slaag ik deze blokken over bij het kopiëren.

Code

```
for (byte i = 0; i < 6; i++) {
    key.keyByte[i] = 0xFF;
}
```

De code hierboven gebruiken we voor authenticatie keys te ontsleutelen met de standaard sleutel met 6 bits 0xFF of 255 (hex of ASCI).

```
// Look for new cards
  if ( ! mfrc522.PICC_IsNewCardPresent())
      return;

  // Select one of the cards
  if ( ! mfrc522.PICC_ReadCardSerial())
      return;
```

Deze code zal wachten tot dat hij een kaart detecteert en deze selecteren om er iets mee te doen.

```
// Show some details of the PICC (that is: the tag/card)
Serial.print(F("Card UID:"));
dump_byte_array(mfrc522.uid.uidByte, mfrc522.uid.size);
Serial.println();
Serial.print(F("PICC type: "));
byte piccType = mfrc522.PICC_GetType(mfrc522.uid.sak);
Serial.println(mfrc522.PICC_GetTypeName(piccType));
```

Voor dat we de data van de kaart proberen te lezen of te kopiëren gaan we eerst UID en de PICC type proberen uitlezen van de kaart.

```
//Via seriele monitor de bytes uitlezen in hexadecimaal

void dump_byte_array(byte *buffer, byte bufferSize) {
    for (byte i = 0; i < bufferSize; i++) {
        Serial.print(buffer[i] < 0x10 ? " 0" : " ");
        Serial.print(buffer[i], HEX);
    }
}
//Via seriele monitor de bytes uitlezen in ASCI

void dump_byte_array1(byte *buffer, byte bufferSize) {
  for (byte i = 0; i < bufferSize; i++) {
    Serial.print(buffer[i] < 0x10 ? " 0" : " ");
    Serial.write(buffer[i]);
  }
}
```

Deze 2 methodes worden gebruikt om de bytes uit te lezen uit de blokken en deze om te zetten naar hexadecimaal of ASCI waarden.

```
void loop() {
  choice = Serial.read();

  if(choice == '1')
  {
    Serial.println("Read the card");
    keuze1();

    }
    else if(choice == '2')
    {
      Serial.println("See what is in the variables");
      keuze2();
    }
    else if(choice == '3')
    {
      Serial.println("Copying the data on to the new card");
      keuze3();
    }

}
```

Wanneer het programma opstart en we de seriële monitor openen krijgen we eerst 3 keuzes wat we willen doen met het programma. Maar bij eerste opstart kunnen we niet veel anders dan 1ste keuze nemen om een kaart eerst uit te lezen.

```
// Try the known default keys
MFRC522::MIFARE_Key key;
for (byte k = 0; k < NR_KNOWN_KEYS; k++) {
    // Copy the known key into the MIFARE_Key structure
    for (byte i = 0; i < MFRC522::MF_KEY_SIZE; i++) {
        key.keyByte[i] = knownKeys[k][i];
    }
    // Try the key
    if (try_key(&key)) {
        // Found and reported on the key and block,
        // no need to try other keys for this PICC
        break;
    }
}
```

Voordat we de kaart kunnen uitlezen moeten we ons eerst authenticeren door de ‘known keys’ te testen op de kaart en wanneer de authenticatie gelukt is, zal het programma breaken en verder gaan om de kaart uit te lezen.

```
boolean result = false;

for(byte block = 0; block < 64; block++){

// Serial.println(F("Authenticating using key A..."));
status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, block, key, &(mfrc522.uid));
if (status != MFRC522::STATUS_OK) {
    Serial.print(F("PCD_Authenticate() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return false;
}

// Read block
byte byteCount = sizeof(buffer);
status = mfrc522.MIFARE_Read(block, buffer, &byteCount);
if (status != MFRC522::STATUS_OK) {
    Serial.print(F("MIFARE_Read() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
}
else {
    // Successful read
    result = true;
    Serial.print(F("Success with key:"));
    dump_byte_array((*key).keyByte, MFRC522::MF_KEY_SIZE);
    Serial.println();

    // Dump block data
    Serial.print(F("Block ")); Serial.print(block); Serial.print(F(":"));
    dump_byte_array1(buffer, 16); //omzetten van hex naar ASCI
    Serial.println();

    for (int p = 0; p < 16; p++) //De 16 bits uit de block uitlezen
    {
      waarde [block][p] = buffer[p];
      Serial.print(waarde[block][p]);
      Serial.print(" ");
    }

    }
}
Serial.println();
```

Bij de bovenstaande code zal hij via de for lus de blokken 0 tot 63 proberen uitlezen. Hij zal per blok de standaard sleutel gebruiken om te authenticeren. Wanneer dit mislukt zal hij een foutbericht weergeven en stoppen met te proberen lezen. Dan zal hij de blok zelf controleren voor hij het begint uit te lezen. Als de read van de blok succesvol is zal het programma wegschrijven welke key er gebruikt is en dit wordt in hexadecimaal weggeschreven. Hierna zal hem alle bits van de blok uitlezen in ASCI waarden. Dit zal hem blijven doen tot dat hij de laatste blok heeft gelezen en daarna komt hij terug in het keuzemenu.

```
for(int i = 4; i <= 62; i++){ //De blocken 4 tot 62 kopieren, behalve al deze onderstaande blocken (omdat deze de authenticatie blokken zijn)
  if(i == 7 || i == 11 || i == 15 || i == 19 || i == 23 || i == 27 || i == 31 || i == 35 || i == 39 || i == 43 || i == 47 || i == 51 || i == 55 || i == 59){
    i++;
  }
  block = i;

    // Authenticate using key A
  Serial.println(F("Authenticating using key A..."));
  status = (MFRC522::StatusCode) mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, block, &key, &(mfrc522.uid));
  if (status != MFRC522::STATUS_OK) {
      Serial.print(F("PCD_Authenticate() failed: "));
      Serial.println(mfrc522.GetStatusCodeName(status));
      return;
  }

  // Authenticate using key B
  Serial.println(F("Authenticating again using key B..."));
  status = (MFRC522::StatusCode) mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_B, block, &key, &(mfrc522.uid));
  if (status != MFRC522::STATUS_OK) {
      Serial.print(F("PCD_Authenticate() failed: "));
      Serial.println(mfrc522.GetStatusCodeName(status));
      return;
  }
```

Bij keuze 3 zullen we de data op de nieuwe kaart proberen kopiëren. Hierbij zullen we weer eerst authenticeren met de sleutel A om de kaart in te mogen lezen en nu moeten we ook gebruik maken van de sleutel B om op de kaart te mogen schrijven (zelfde standaard sleutel met 6 bits 0xFF).

```
// Write data to the block
Serial.print(F("Writing data into block "));
Serial.print(block);
Serial.println("\n");

dump_byte_array(waarde[block], 16);


 status = (MFRC522::StatusCode) mfrc522.MIFARE_Write(block, waarde[block], 16);
  if (status != MFRC522::STATUS_OK) {
    Serial.print(F("MIFARE_Write() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
  }
```

Met deze code zullen we opgeslagen waardes van dezelfde blokken overschrijven naar deze blokken op de nieuwe kaart.

Besluit

We hebben in dit project niet helemaal bereikt wat we hadden gepland, maar we zijn toch ergens geraakt. In het begin heb ik geprobeerd alles te programmeren op de raspberry pi door in python te programmeren. Op een bepaald punt begon ik vast te geraken met het uitlezen van hexadecimaal naar ASCI waarden. Ook met het kopiëren had ik wat problemen met python.

Dus ben ik overgestapt naar Arduino en hier ben ik nog wel wat verder geraakt dan in python. Met Arduino heb ik een aantal programma's waarin ik persoonlijke data kan wegschrijven op de kaart, vaste waardes in het programma wegschrijven naar de kaart en een programma waarbij ik een kaart kan uitlezen en deze data kan opslagen in variabelen en ze daarna dan wegschrijven naar een nieuwe kaart.

Het probleem bij dit laatste programma is wel dat we wel data kunnen wegschrijven naar authenticatie blokken (sector trailers), maar omdat we altijd 6 bits van 0 uitlezen bij sleutel A, zal er dus 6 bits van 0 weggeschreven worden naar de nieuwe kaart. Dit is dus geen juiste kopie en kunnen we het ook niet met dezelfde sleutel A van de eerste kaart, de nieuwe kaart uitlezen.

Als we deze blokken overslaan (zoals in het programma) kunnen we data van de eerste kaart op de nieuwe kaart krijgen.