

**SE-518**

**Lab Report - 1**

**Introduction to Correlation Power Analysis**

**Authors :**

**HAMOD Karim ID:422030353**

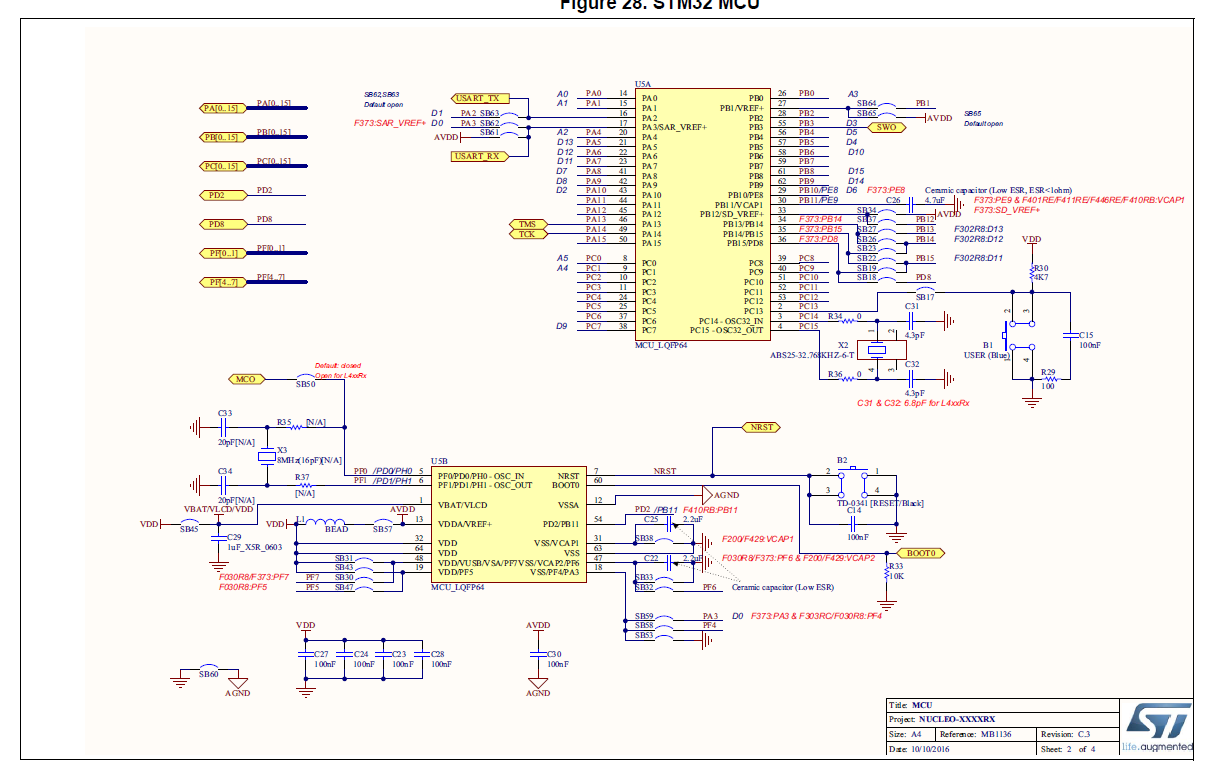
**AHSAN MD. Shahnauze ID: 42203082**

# 

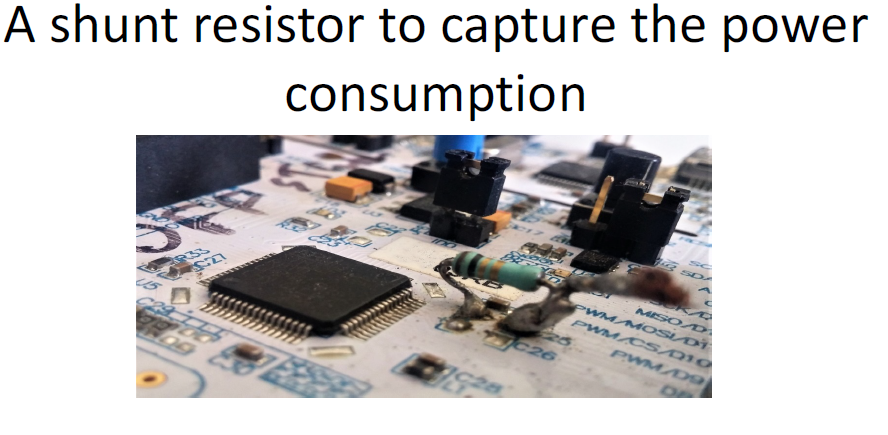
# 

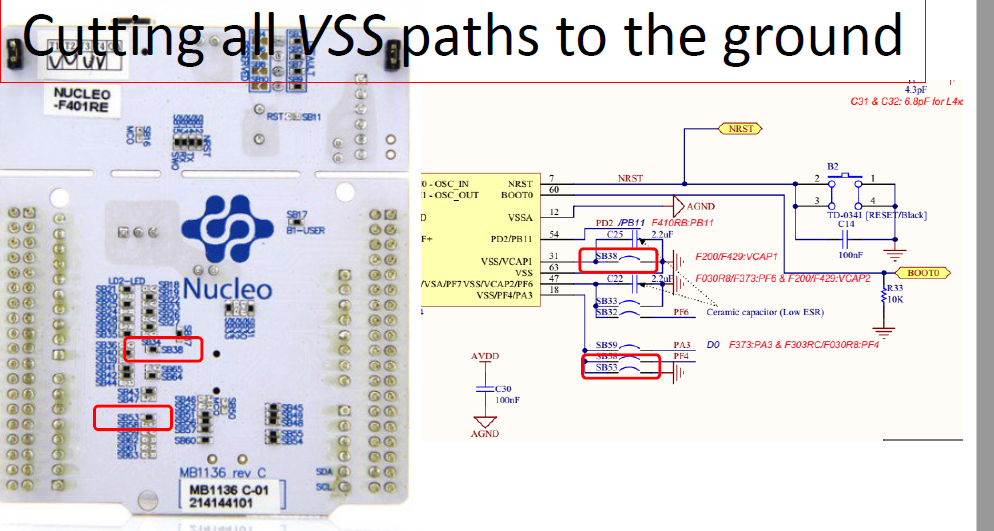
# Task 1:

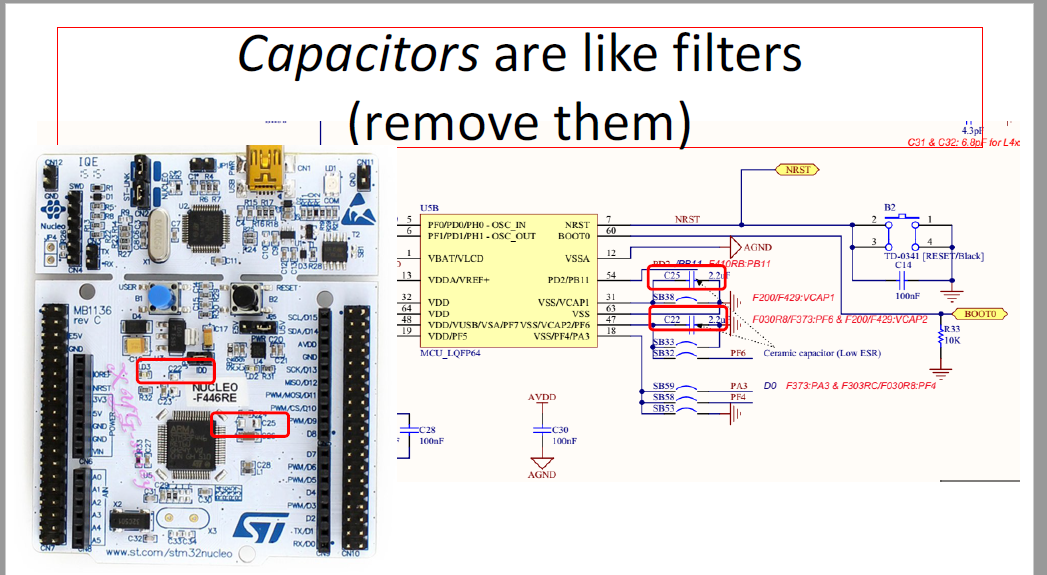
1.3



<<fig 1:Stm 32 Nucleo 64 board schematic >>

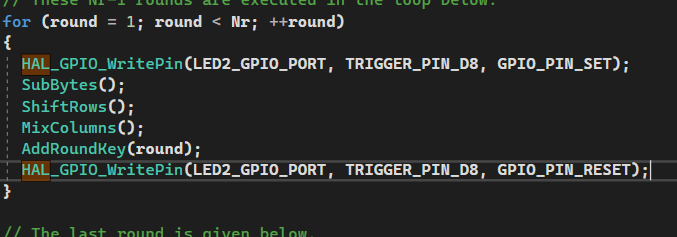






A 1-12 ohms shunt is added instead of the C25 capacitor. These steps are done to capture the power consumption .

# Task 2:



<< fig 2 addition of oscilloscope library commands >>

To trigger the oscilloscope we have used these 2 HAL library commands and we placed them in a for loop for the round choose this because here we will record the power consumption of the first round of the aes encryption thus we also get the power consumption for each round during add round key.

# Task 3:

**-HAL\_UART\_Receive(&UartHandle, (uint8\_t \*)&keyin, 16, 0xFFFF):**

This function is a blocking function which blocks the cpu from the main routine until a certain amount of uart data bytes are received where here it receives 16 bytes to keyin where the cpu will be kept blocked from main until it finishes.

**-HAL\_UART\_Receive(&UartHandle, (uint8\_t \*)&datain, 16, 0xFFFF):**

This function is a blocking function which blocks the cpu from the main routine until a certain amount of uart data bytes are received where here it receives 16 bytes to datain where the cpu will be kept blocked from main until it finishes.

**-AES\_ECB\_encrypt(datain, keyin, buffer, 16);**

This function is an AES encryption mode where the encryption operates on the datain as input and keyin as key and store the result as a 16 byte cipher text output in buffer.

-**HAL\_UART\_Transmit(&UartHandle, (uint8\_t \*) &buffer, 16, 0xFFFF):**

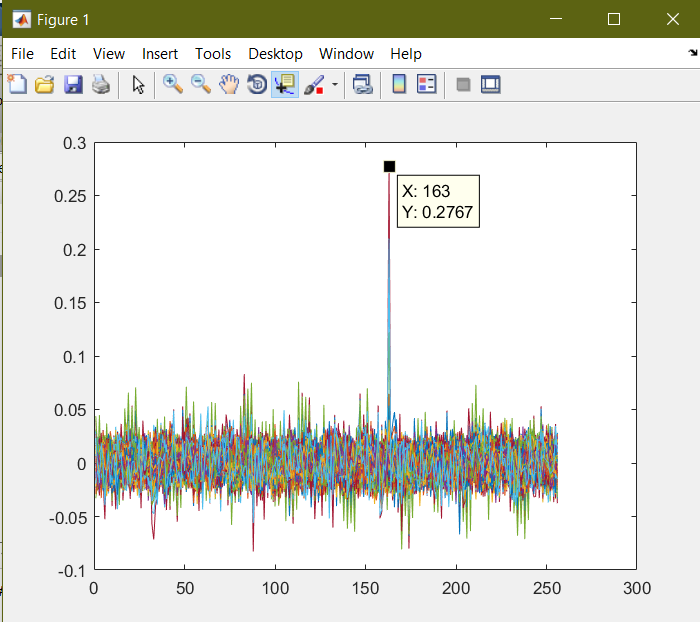
This function is a blocking function for sending data that prevents cpu from returning to main context until timeout. Where here we are sending the cipher text stored in buffer.

# Task 4:

4.2 : In CPA attack of any cryptographic device the array of **D** means different data blocks, **K** denotes the total number of possible choices for hypothetical intermediate values, **V** is a matrix of size DxK which contains the intermediate results. The array **H** denotes the hypothetical power consumption values that we have mapped from intermediate results. This is also known as Hamming distance. **R** is the result of comparison between hypothetical power consumption values of each keys with recorded traces.

4.7- The number of symbols on a string that are different from the alphabet's zero-symbol is known as the hamming weight. It is comparable to the Hamming distance from the string made entirely of zeros.The quality of the hypothetical model that is utilised to predict the outcomes of the actual device is a key factor in differential power-analysis attacks.A hacker with competence in the design of cryptographic hardware, for instance, who lacks insider information, can glean plausible details about the implementation. Understanding the statistical characteristics of the algorithm being attacked can also help to develop the fictitious model. This hamming weight is one of the major characteristics of the algorithm, and that's why we think Hamming weight can be used as a hypothetical power model.

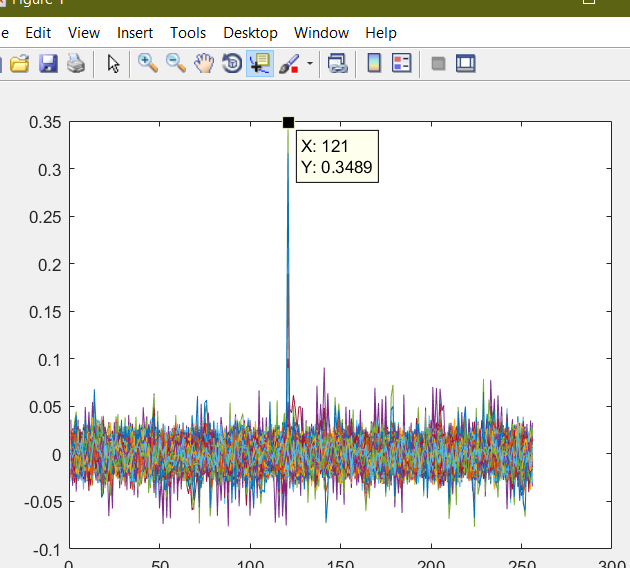
4.5- after completing the code and doing successful runs we get the key values:



<<for Key byte 1>>

Key= 162

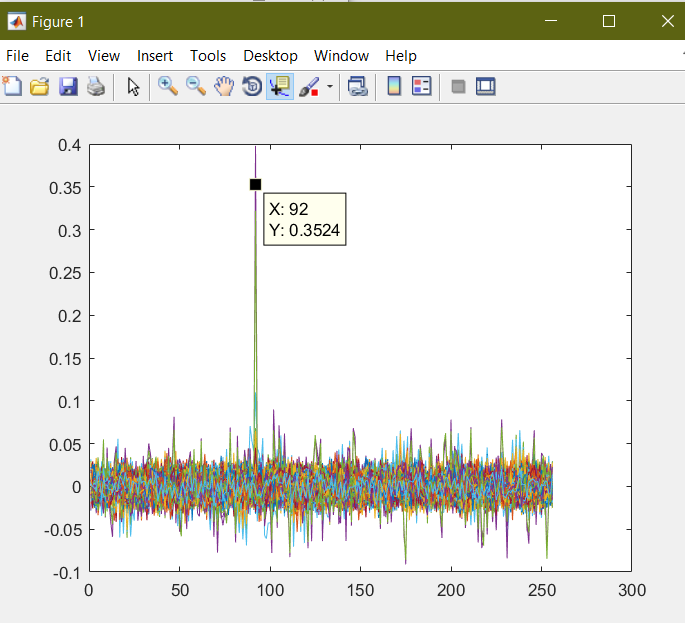
Maximum correlation=0.2757



<<for Key Byte2>>

Key=120

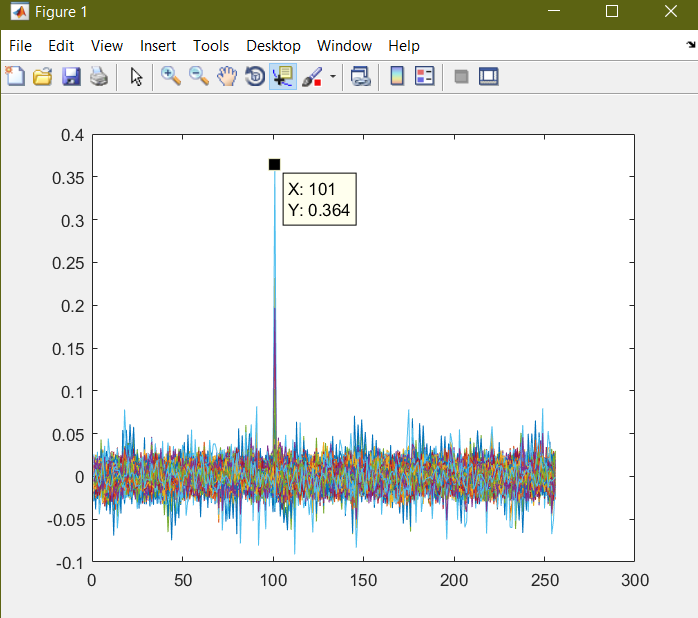
Maximum correlation=0.3489



<<Key Byte 3>>

Key=91

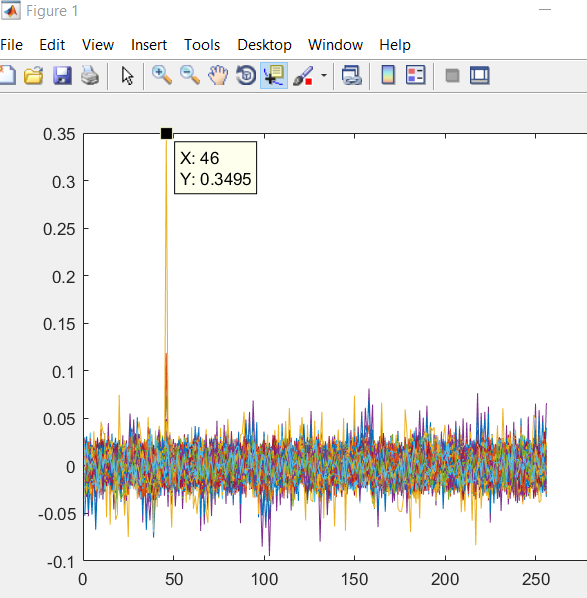
Maximum correlation=0.3524



<<Key Byte 4>>

Key=100

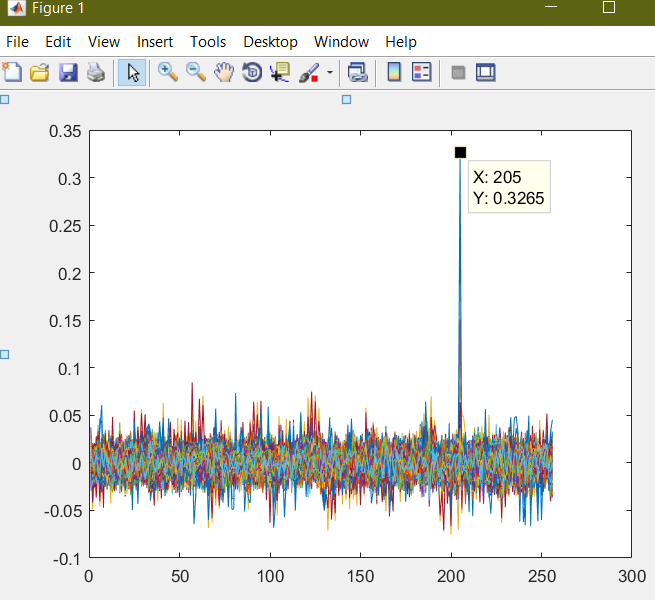
Maximum correlation=0.364



<<Key Byte 5>>

Key=46

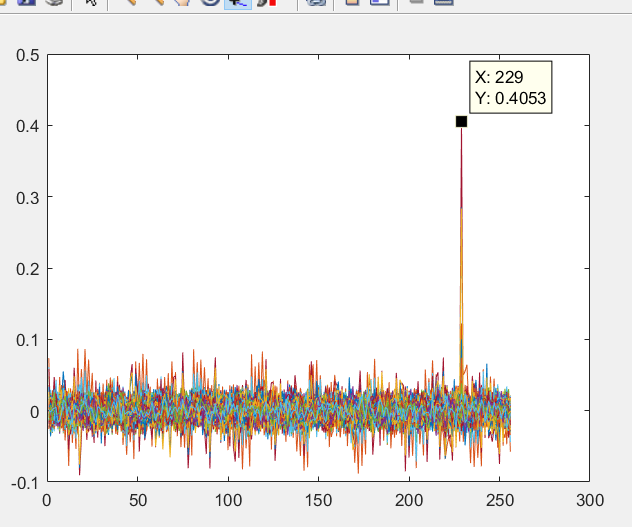
Maximum correlation=0.3495



<<Key Byte 6>>

Key = 204

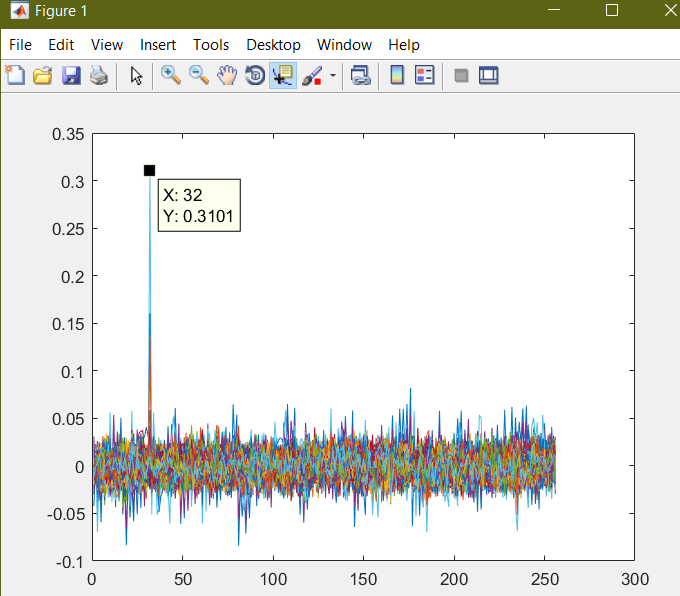
Max correlation =0.3265



<<Key Byte 7>>

Key=228

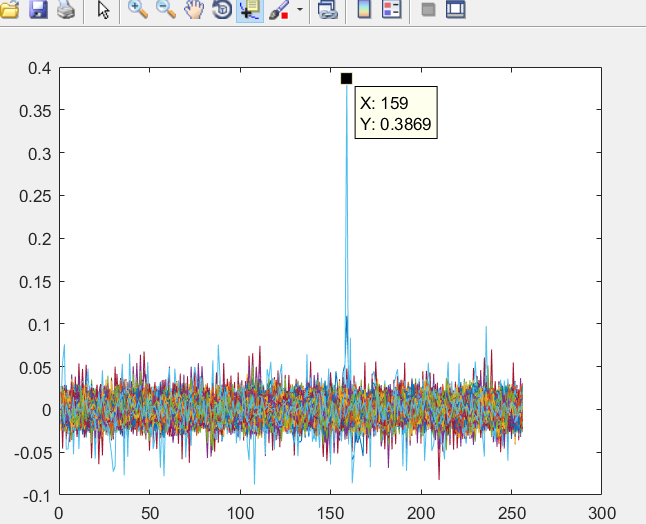
Max correlation =0.4053



<<Key Byte 8>>

Key=31

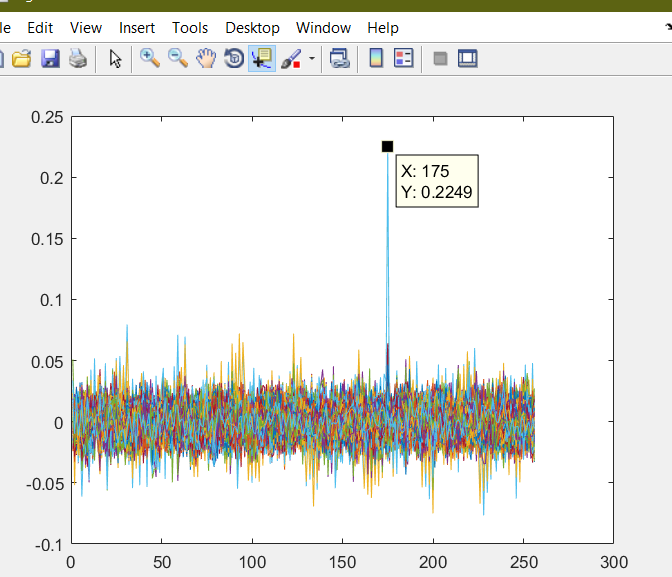
Max correlation=0.3101



<<Key Byte 9>>

Key=158

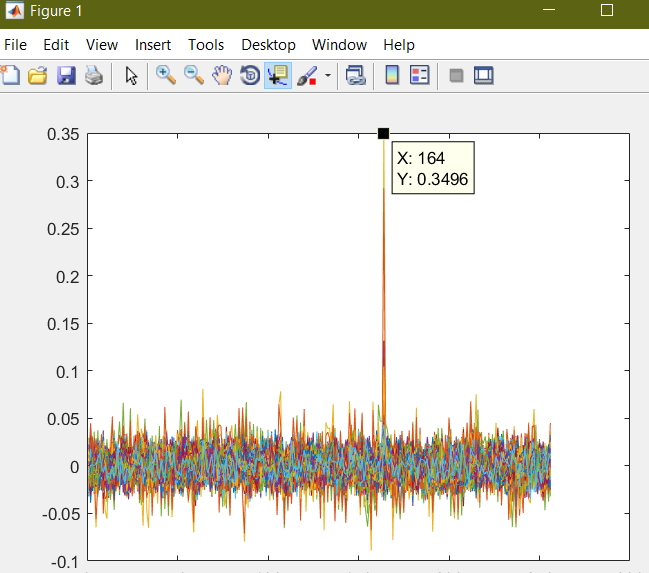
Max correlation=0.869



<<Key Byte 10>>

Key=174

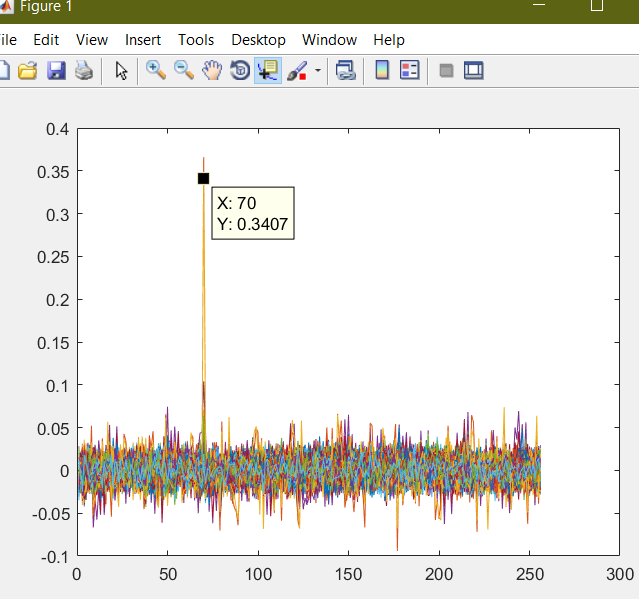
Max correlation=0.2249



<<Key Byte 11>>

Key=163

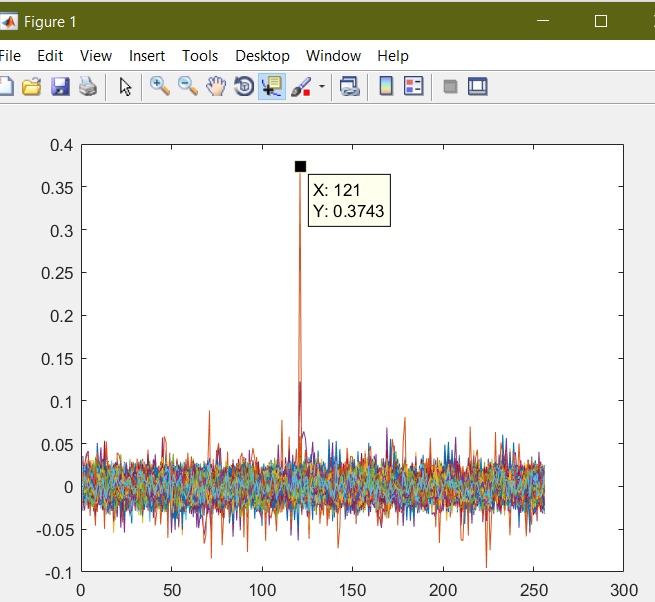
Max correlation=0.3496



<<Key Byte 12>>

Key=69

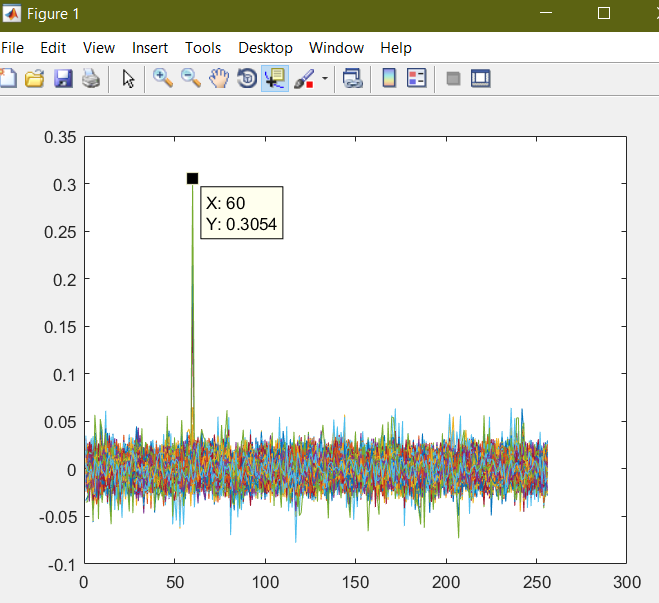
Max correlation=0.3407



<<key Byte 13>>

Key=120

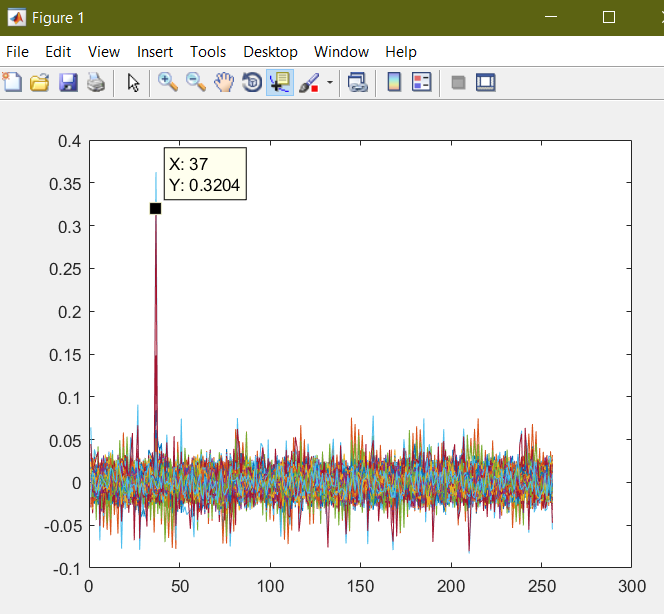
Max correlation=0.3743



<<key Byte 14>>

Key=59

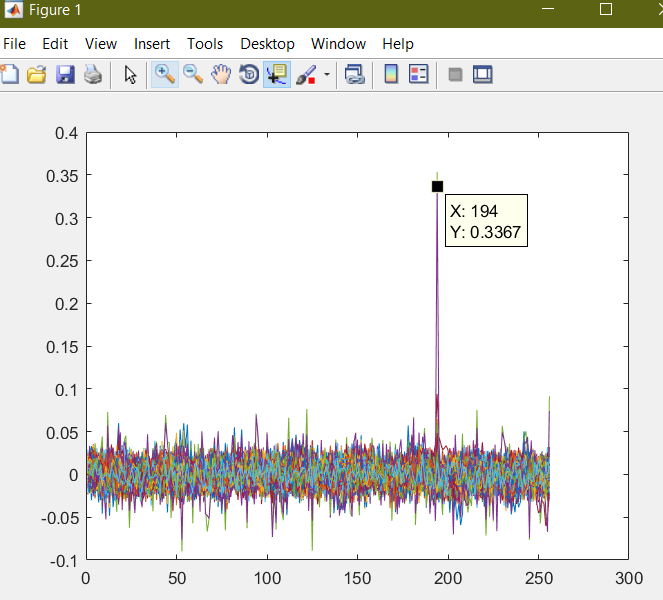
Max correlation=0.3054



<<key Byte 15>>

Key=36

Max correlation=0.3204



<<key Byte 16>>

Key=193

Max correlation=0.3367

Finally after all the runs we get the full Key

Full Key=162 120 91 100 45 204 228 31 158 174 163 69 120 59 36 193

# Task 6:

An AES implementation can be kept secure against first and second-order DPA by employing a variety of countermeasures or their combinations. When using the randomization and masking procedures, one should use both countermeasures for operations that are susceptible to second-order DPA and masking for operations that are susceptible to first-order DPA.

The initial two and a half rounds and the final three rounds of AES should be appropriately protected in order to render an AES implementation secure against first and second-order DPA. Apart from it, the Helion Tiny-AES core is best suited for applications in which efficient use of logic resources and/or minimal use of device resources and power are requirements. It provides fully functional AES encryption and decryption with integrated key expansion supporting all three AES key sizes, and if all of these features are not required for a particular application, it may be scaled down to save space.

So, we think that rather than general AES, Tiny-AES would be the best option for increasing the security against CPA attack.