Nathaniel Fredricks

Assignment 1 Report

In this assignment, a 128-bit AES was implemented as firmware for the Chipwhisper Nano. This involved translating the AES to C and then connecting it to the serial libraries.

At a high level, the AES implementation has two main functions: encryption and decryption. Encryption involves taking in a 16-byte plaintext and a 16-byte key. First the key is expanded into 11 keys. There are then ten rounds where the state is modified. In each of the main rounds, the constant sbox array is used to substitute values in the state with new ones. Next the state's rows are shifted and columns mixed. At the end of each main round, the state is XORed with the corresponding round key. For the last round, the mixing of columns is skipped. Through this, a 16-byte ciphertext is created.

The implementation of decryption involved a sort of reverse process. An inverse of most of the encryption's functions were created to translate the provided ciphertext back into plaintext.

Encryption and decryption were tested by comparing C running on a desktop computer with a Python implementation of an AES. A key and plaintext would be run on both implementations to check that the outputs matched.

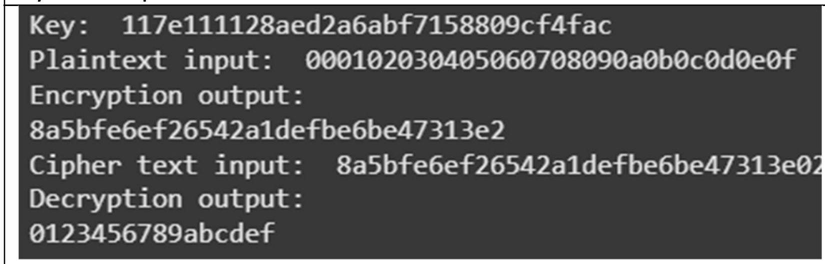
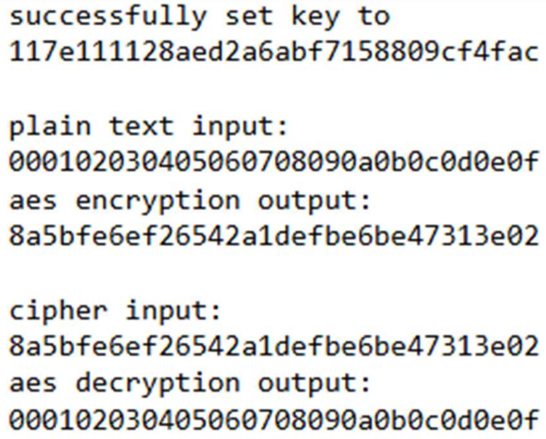
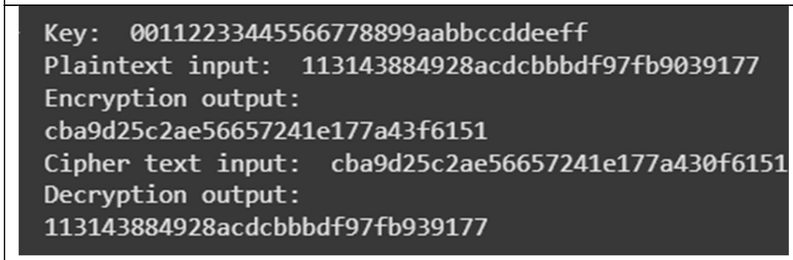
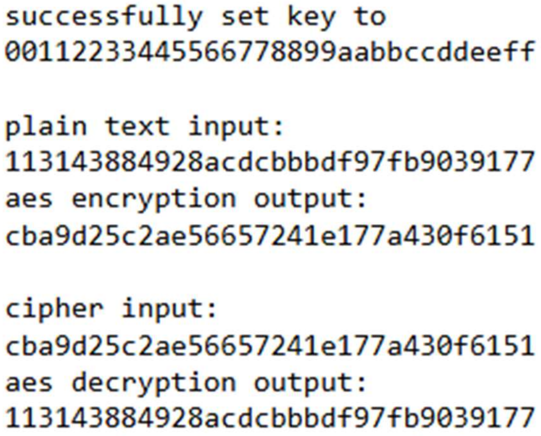
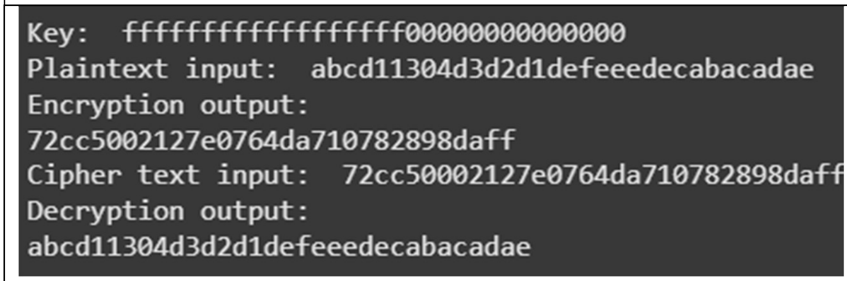
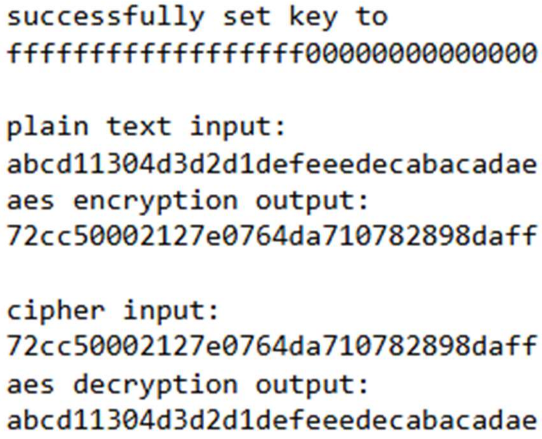
After checking that the initial encryption and decryption were working, the code was ported to the Chipwhisperer library. Previous "printf" statements would no longer work. Both encryption and decryption required a 16-byte key and a 16-byte input. However, the key would have to be given separately with the AES working in ECB mode. A function was created that would take in a pointer, k, and length, len. If the length was not 16-bytes, an error would be sent over the simple serial (0xFF). If the length matched, then the contents of k would be stored in a static array for the master key. If successful, a 0x00 value is sent to serial. To have this function work with the simple serial library, a command "k'" was added along with a reference to the key store function. When a serial message with the command "k" is sent to the ChipWhisperer, the key store function runs. The content of the message (the key) is automatically fed to the function.

Similar functions for encrypt and decrypt were created along with simple serial commands "p" and "c" respectively. Command "p" would receive a 16-byte plaintext. This would be fed into the AES encryption along with the stored master key. The resulting ciphertext would be sent out over serial. Similarly, the "c" command would decrypt a ciphertext and send the plaintext over serial.

The main function contained function calls to setup the serial communication. An infinite loop was used to continually read for sent serial messages.

A jupyter notebook was created to compile and run the firmware on the ChipWhisperer. It would compile the C code and then program it onto the ChipWhisperer. Methods for setting the key, encryption, and decryption were created. They would send the appropriate data using simpleserial library. The results from decryption and encryption would be both returned by the function and printed out (if the print setting was enabled).

Below is a set of tests to check the firmware implementation against the python implementation.

Python Implementation	Firmware Implementation
 Key: 117e111128aed2a6abf7158809cf4fac Plaintext input: 000102030405060708090a0b0c0d0e0f Encryption output: 8a5bfe6ef26542a1defbe6be47313e2 Cipher text input: 8a5bfe6ef26542a1defbe6be47313e02 Decryption output: 0123456789abcdef	 successfully set key to 117e111128aed2a6abf7158809cf4fac plain text input: 000102030405060708090a0b0c0d0e0f aes encryption output: 8a5bfe6ef26542a1defbe6be47313e02 cipher input: 8a5bfe6ef26542a1defbe6be47313e02 aes decryption output: 000102030405060708090a0b0c0d0e0f
 Key: 00112233445566778899aabbccddeeff Plaintext input: 113143884928acdcbbbdf97fb9039177 Encryption output: cba9d25c2ae56657241e177a43f6151 Cipher text input: cba9d25c2ae56657241e177a430f6151 Decryption output: 113143884928acdcbbbdf97fb939177	 successfully set key to 00112233445566778899aabbccddeeff plain text input: 113143884928acdcbbbdf97fb9039177 aes encryption output: cba9d25c2ae56657241e177a430f6151 cipher input: cba9d25c2ae56657241e177a430f6151 aes decryption output: 113143884928acdcbbbdf97fb9039177
 Key: ffffffffffffffffff00000000000000 Plaintext input: abcd11304d3d2d1defeeedecabacadae Encryption output: 72cc5002127e0764da710782898daff Cipher text input: 72cc50002127e0764da710782898daff Decryption output: abcd11304d3d2d1defeeedecabacadae	 successfully set key to ffffffffffffffffff00000000000000 plain text input: abcd11304d3d2d1defeeedecabacadae aes encryption output: 72cc50002127e0764da710782898daff cipher input: 72cc50002127e0764da710782898daff aes decryption output: abcd11304d3d2d1defeeedecabacadae