AES Trojan

Team Yellow: Casey Avila, Ben Kunkle, Noah Masten

We implemented an AES hardware trojan for the Basys board. The board accepts data over UART, encrypts it using AES ECB, and then sends it back. Our trojan allows for extraction of the key.

Protocol

After programming the board, it waits to receive bytes serially over UART. AES operates on blocks of 16 bytes. The first 16 bytes received are saved as the key, and every next block of 16 bytes is encrypted with the key, and sent back over UART. One quirk is that sending the first 16 bytes (the key) will send back the key encrypted with itself, so these can be ignored.

If you want to program a new key, the center button can be pressed and the next block will be assigned as the key. The leftmost LED serves as an indicator; if it is on, the key is set and all sent blocks are encrypted with the key currently on the board. If it is off, the key on the board will be assigned to the next block.

Encrypting with the board looks something like this: first send 1 block to serve as the key, ignoring the response. Now that the key is set, send 1 block, then receive 1 block. Repeat this, keeping track of the received blocks, which is the encrypted data.

Using The Board

We made a Python script that you can use to interact with the board on the command line. First you'll want to install the dependencies using `pip install -r requirements.txt`.

```
usage: binary.py [-b | -l] hex_key file
        -b: Use board
        -l: Use AES library (no trojan)
        hex_key: 16 byte key for AES in hexadecimal
        file: Input file
```

Before using it, edit the `SERIAL_PORT` variable in the script to be the serial device that your board is connected to. We used the `pyserial` package, so reference their documentation for examples on your OS.

This script uses the protocol above to read in files on your filesystem, and print out the hexadecimal encryption of the file. By passing the `-b` flag, all the encryption is done on the board. `-l` can be used as a reference AES encryption implementation, with **no trojan**. The key and filename are also taken as command line arguments. The script uses PKCS#7 padding to ensure the blocks are sized correctly for AES.

Example

```
$ cat hi
hi there! good luck!
// Make sure to click the button here if using a different key than
// before
$ python3 binary.py -b FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF hi
d775a1b212741115839b854f79ecb539
8b500c8f9ce4a7c8492f2df4ae450766
```