Multifactor Security System

CS152B Digital Design Project Laboratory

TA: Babak Moatamed

Students: David Chen, Luke Moore, Matt Dragotto

I. ABSTRACT

Following UCLA's attempt to improve online security with their implementation of a multi-factor authentication system, many were discontent with the addition. It was unwieldy to use at best, and some of it's proposed features were in fact non-functional. Thus, we set out to create a security system in Microblaze that could surpass the UCLA Duo Mobile App; in short, we wished to make a multi-factored security system.

While ultimately our design ran askew of our original intent -- such as as make our product a physical security device for preventing local access rather that network authentication -- we ultimately were left with a satisfactory product, which works as listed.

1) The user turns the device, loads the processor and waits for the camera to initialize. The camera initialization takes roughly 10 minutes. The camera can be recognized as ready when a red box appears roughly center on the screen.
2) A keycard (a 3x3 grid of red or green squares) should be placed as to appear centered in the red box on screen, with the box matching the edges of the card.
3) Flip the switch to scan the keycard. The switch that we implemented is the one farthest to the right when looking at the FPGA. Following this, if the keycard was valid, the security system will generate a random PIN, which is sent to a paired bluetooth device. The system will also prompt you for the code by indicating as such on the console.
4) Once the PIN has been corrected entered using the provided keypad, the system will log you in a welcome the user corresponding to the keycard.

This system provides local security based on 2 things you have and one thing you know, being the keycard, your phone or bluetooth device, and the PIN needed to pair with the security system bluetooth, respectively.

II. PROJECT BACKGROUND

The basis of this project was to create a multifactor authentication system. Most industry grade authentication systems contain some verification using a user's identity, as well as a code sent to their mobile phone. Since most people own some sort of mobile device, this provides an adequate layer of security, as typically the only person with access to that device is the user themselves.

In modelling an industry grade authentication system, we decided to use a similar approach involving mobile verification. The three

main components of our system for implementing security is a camera, a bluetooth device, and a keypad.

The camera's main objective is to run an image processing algorithm on a keycard that is unique to the user. In doing so, it will be able to identify the user, and send a code to their mobile device for verification. This aspect is not meant to provide a complex layer of security; instead, it is a intuitive way to interface with the user that is approachable and easy to use.

Some difficulties that we faced in implementing the image processing was coming up with a design for the key cards that would be easily distinguishable in the software. We at first tried setting a threshold for color and using that threshold to determine how light or dark an image was, but the camera itself turned out to be too low of quality to work with effectively. Instead, we realized that the software was very capable of detecting RGB values of pixels on the screen, so we went with an implementation that found out which value (R, G, or B) was most present in a section, and used that aggregate to classify the whole section as either red, green, or blue.

The Bluetooth module was used as the main layer of security, since it was the part of our system that actually interfaced with the user's mobile device. As mentioned earlier, a mobile device is unique per user and usually accessible by only the user themself, so most of our security came through the implementation of this part of the project.

Some difficulties that we faced when incorporating the Bluetooth module was that the module used the same interface as our hardware responsible for printing out messages to console. Whenever we would add in a UART-Lite module to our project to handle the Bluetooth RX and TX pins, we would no longer be able to print to console. This was probably attributed to some sort of naming convention in the xparameters.h file, where the device ID was similar between the two UART-Lite modules. We fixed this by adding in a different type of UART interface, the UART-RS16550. After this, we were able to print and send messages over Bluetooth.

The keypad was used as an input device for entering the 4 digit code sent to the user's phone into the system. It was a fairly simple keypad to begin with, consisting of keys corresponding to 0-9 and also A-F, but we further simplified it by limiting the inputs to only 1-9. In a more robust system, a full keyboard could be used, as well as an alphanumeric passcode.

Some difficulties we faced when implementing the keypad was in getting it to capture four consecutive key presses from the user. Basically with every press of a key, the keypad code would be called multiple times to return which key was being pressed. For this reason, we had to add a series of constraints in the software to provide sort of a "pseudo-debouncer", where if the system detected the same key press multiple times in succession it would know to register it as only one key press. Another challenge we faced was determining which key was being pressed, as in our previous experience working with the keypad, we only detected keys 1-3. To solve this, we ended up

finding an API online that contained a few simple functions for getting the correct char that corresponded with the key being pressed.

III. TUTORIAL

Our final product was based on Microblaze on the Xilinx Virtex 5 FPGA board. The modules used in additon were a Pmod Keypad (Gpio), a Switch found on the FPGA board, a camera (IIC), a RN42 Bluetooth chip. The project can be recreated as follows:

1) Physically connect the modules to appropriate locals on the board. Keep these in mind because they will need to be added to the hardware later.

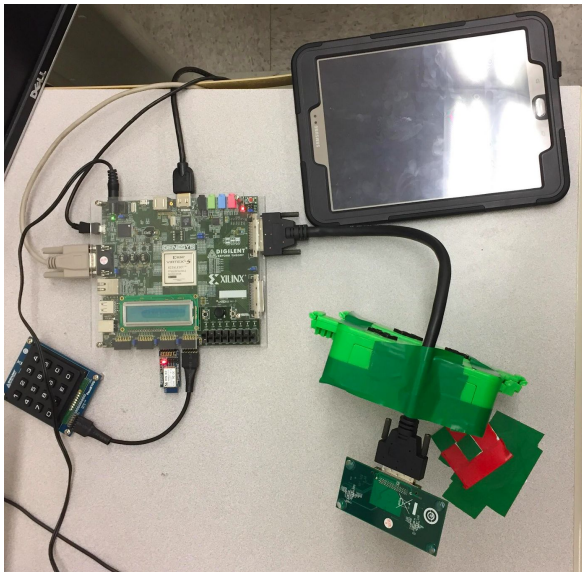

Fig. 1: Hardware Setup for both Modules and Cables

2) Connect all the cables for the board, being an HDMI cable to show the camera's display, a micro- USB from computer to allow the upload of code to the FPGA, the serial communication cable to be able to print to console via the UART interface, and last but not least the DC power cable.

3) Download and unzip the VMOD Camera Tutorial (provide on CCLE) and follow the tutorial provide to get the camera up and running.

4) Following this, open the system.xmp file in Xilinx XPS. This will open the initial hardware portion of the project in the Xilinx EDK.

5) Add the following modules to the EDK project:
   - GPIO for the switch functionality
   - GPIO for the keypad
   - UART 16550 for the bluetooth module

   Don't forget to make the pins external in the ports tab and to map the addresses.

6) Open the UCF file and add in the modules making sure that the pins are matched correctly with the physical pins of the FPGA. The corresponding pins that you use can be found in the master ucf file for the board on the Xilinx website.

7) Export the design to SDK. Note that this will take a significant amount of time, largely due to the camera.

8) In the SDK, go into the ld script and modify the stack and heap to be of size 1000 and

ascertain that that the memory mappings are set to `ilmb cntlr dlmb cntlr` instead of DDR2.

9) Next, go into run configurations and edit the STDIO connection, checking “Connect STDIO to console” and selection the right port and baud rate. The port can be found by checking the COM ports open on your machine’s device manager and the baud rate should be set to 9600.

10) Program the FPGA (after the first time running this code, the process can be sped up by going into Run Configurations and adjusting it to only re-run the processor. We want to re-run the processor only after the first run otherwise it will have to reconfigure the camera each run which takes roughly ten minutes. There are two chunks of camera initialization code that can be commented out after the first run and after the camera was properly initialized.

11) Following this, feel free to import the project files for the security system into the SDK. Assuming every last part was completed correctly, the code security system should run as described.

IV. DESIGN & IMPLEMENTATION

1) Camera Initialization

The first thing we do in our code is run the camera initialization code. Then, in a while loop, we continuously draw a red box onto the screen. This is the same red box previously mentioned that is used for the user to place his or her keycard into. Then, in the loop, we check if the user has set the switch. If the switch is set, then we call our process function which does the image processing.

2) Image Processing

The image processing works by scanning all the pixels in a cell (on e the segments in 3x3 grid). For each pixel, the RGB values are extracted and added to an accumulator. For each cell, these RGB values are measured against each other to find the predominant color of the block, labeling it as either red or green. If the configuration of cells found is equal to one of the predetermined keycards, then the user can move forward to the next step. Right now, the predetermined “hardcoded” keycards are L, D, and M which are the first letters of each of our names. You can simply create any keycard you want inside the 3x3 grid and write a single if statement to allow that specific user access. If a user is found, the next step is to send that user a 4 digit PIN to his or her smartphone.

3) PIN Generation

The pin generation uses selected pixels from the camera input as a seed to create pseudo randomness. Base of the RGB values of the selected pixel, a single digit integer is generated. This process is repeated 4 times to create the 4 digit pin. Once the 4 pseudo random digits are created, numbers between 0 and 9, the four character array is sent over bluetooth to the users phone which is running an android application that receives and print whatever message is sent.

4) Bluetooth & App

Our bluetooth implementation uses UART 16550 which has a relatively straightforward API. We first initialize the UART bluetooth device, then send over a packet specifying the number of bytes sent to the BTSend() function. This function sends the bytes over the bluetooth interface. Additionally, another form of security is necessary to gain access to our bluetooth module. Users must present a password to connect to the bluetooth module and receive the PIN to enter on the keypad. Without knowledge of this password, there is no way a user or an attacker could gain access to the system.

5) Keypad

We are using a PMODkypd API that is called and returns the number pressed by the user. We call our keypad function four times to get the four different digits that the user presses. If those four digits are equal to the PIN that we randomly generated and sent to their app, then the user has access, else, access is denied.

6) Console

We print to the console throughout each step of the application processing to allow easy user interaction and clarity. Once a user is found from the camera and keycard, the program prints to console letting the user know to check his or her phone to find the PIN to enter. Once the PIN is entered, the console displays whether access is granted or denied.

V. CONCLUSION

Despite the elementary nature of our image processing and non-robust keycards, our design could be implemented and used in real world systems. The multiple instances of authentication in our program allows for great security because it would be difficult for an attacker to present all of the necessary pieces of information to get access to the system. Being able to brute force the bluetooth module's password, having the right keycard to gain access to a valid user, and having the application on a smartphone is a lot of walls to get through for an attacker. For these reasons, our design is strong enough to compete with competition in the market.