EC 544 Networking the Physical World, Spring 2020
Electrical and Computer Engineering Department, Boston University

AR Communication:
Secured Communication in AR Mobile Game

Authors:

Alex Jeffrey Lin (ajlin@bu.edu)
Haotian Cheng (hcheng3@bu.edu)

Project Advisor:

Prof. Babak Kia (bkia@bu.edu)

4/29/2020

System Diagram

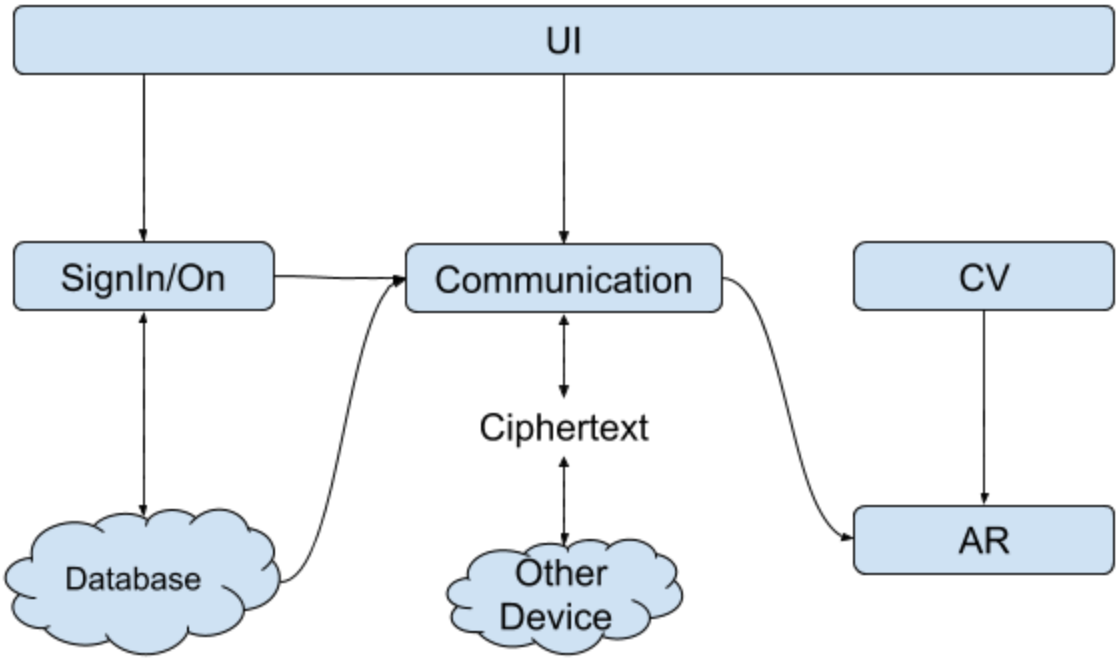

Figure 1: System Diagram

Architecture breakdown:

1. UI:
    a. Provides username and password to SignIn/On Module.
    b. Provides text messages entered by the user to the Communication Module to communicate with the other mobile device.

2. SignIn/On:
    a. Create an account to the cloud if the username entered by the user has not been taken.
    b. Hash the password entered by the user.
    c. Fetch the hashed password link to the username in the cloud database.
    d. Authenticate the user by comparing the two hashes.

3. Database:
    a. Stores users information in the cloud.

4. Communication:
    a. Communicates with the other mobile device by sending and receiving each other's ciphertext.
    b. Verifies the identity of the owner of the ciphertext.
    c. Verified results and successfully decrypted plaintext will be sent to the AR module to display the result.

5. CV:
    a. Detects hands' gesture and depth from the camera of the device.
    b. Send the hands' gesture and depth information to the AR module.

6. AR:
    a. Spawns a character figure on hand when the user has his hand opened in front of the device camera.
    b. Displays the text message sent by the other device or a warning message of being hacked depends on the result provided by the Communication Module.

Implementation

Modules breakdown:

[1] User Interface Module:

This module is successfully implemented as proposed originally.

Breakdown:

a. In our app, there are three main scenes and they are all designed using Unity Editor:
    i. The first scene is used for users to SignIn/On.
    ii. The second scene is a lobby for users to create a chatroom and find each other.
    iii. The final scene is the AR scene where users can interact with the virtual character using their hands and send messages to the other user in the same chatroom securely.

b. In different scenes, there are different buttons to interact with:
    i. [Create] (in the first scene) for users to sign on.
    ii. [Login] for users to log in.
    iii. [Create] (in the second scene) for users to create a chatroom.
    iv. [Join] for users to join a chatroom
    v. [Start Chat] for the user who creates room to start AR chat
    vi. [Send] for users to send messages
    vii. [X] for users to get back to the previous scene

c. An Error Log is designed at the bottom of the screen to inform users of the states or the problems they might encounter:
    i. Wrong password when signing in.
    ii. Duplicate name when creating a new account.
    iii. Empty room name.
    iv. Successfully logged in.

[2] Augmented Reality Module:

This module is successfully implemented as proposed originally.

Breakdown:

a. Unity AR Foundation is the API we used to achieve AR chat
b. As soon as Unity AR Foundation has the permission of the use of the device camera, it can overlay a virtual world in real life once it detects a vertical or horizontal plane in the surroundings. In addition, it can keep track of the camera position through the virtual world.
c. Once we can touch into the virtual world we can simply use Unity itself to spawn and despawn game objects and design how these objects behalve in our program.
d. In our APP, we use it to spawn a character figure on our hand and display a textbox on top of its head in order to display messages including the warning message if system detects Man In the Middle Attack(MIMA).

[3] User Login Module:

This module is successfully implemented as proposed originally.

Breakdown:

a. Designed a function to create an account with desired username and password.
b. Designed a function to hash passwords and store them into a database to corresponding username once the account is created.
    i. In order to hash the password for security purpose, two hashing methods were considered which are MD5 & SHA1:
        1. MD5 and SHA1 hashing algorithms both provide one-way encryption which is ideal for storing passwords as you will not need to see the encrypted version.
        2. The speed of MD5 is fast in comparison to SHA1's speed. However, SHA1 provides more security than MD5.
        3. MD5 output length is 128 bits and SHA1 output length is 160 bits. Thus, SHA1 has much higher security strength than MD5.

4. While both hashing methods are announced as broken or successful collision attacks against them. But we decided to use SHA1 over MD5. Because the amount of time to generate a collision to break SHA1 hash is far more difficult than MD5 due to the output length. As we tested the speed of both hash functions, even though SHA1 is slower, this factor is negligible for our application.

c. Designed functions to match username and password from database for the user to log in. The implementation also authenticates the user by verifying the hashed output of the inserted password with the hashed password stored in the database.
d. Implemented some error detecting methods:
    i. Unique username is required to retrieve or search user's information. Therefore an unused username is required to create an account, else error will be reported.
    ii. Users are allowed to create the same password but under a different username.
    iii. Incorrect username or password will be reported as an error.
    iv. Any invalid or empty inputs will be detected and reported as an error.

[4] Database Module:

This module is successfully implemented but different than the original proposal, the detailed breakdown of approaches and difficulties will be explained as follows.

Difficulty:

As the project proposed originally, designing a functional storage by implementing HashMap data structure was the initial plan. However, having an HashMap as a dictionary can only save data locally and users with different devices cannot access the data from this storage. Here comes an important concept to use cloud databases in order to achieve database synchronization.

After conducting research, we decided to use Google's Firebase. Firebase is a cloud-hosted NoSQL database that lets you store and sync data between your users in real-time. Results came out perfectly not only solved database synchronization but also took a big role in our cryptographic key distribution and authentication purposes.

Breakdown:

a. Implemented Firebase APIs and functions to efficiently store and retrieve user's information in real-time.
b. Information stored in the database:
    i. Username: An unique username will mainly be used to access the data from the database.
    ii. Password: a hashed password will be stored under the corresponding username.

iii. Publickey: a publickey will be stored under the corresponding username. It's use for encryption and authentication purposes, details will be explained in the cryptography module.

[5] Computer Vision Module:

This module is successfully implemented as proposed originally.

Difficulty:

The Manomotion API used in the CV Module has less documentation for user to easily apply the methods. Instead, it provides an example that uses its API to achieve most of its features. Therefore, we spent a lot of time grinding existing code samples and eventually extracted the methods we need to complete this module.

Breakdown:

a. Implemented functions using Manomotion API mainly to detect hand gestures and hand depth from the mobile device with camera.
b. The detection speed of Manomotion API is considerably good at about 20ms while FPS stays at about 30.

[6] Cryptography Module(Communication):

This module is successfully implemented but different than the original proposal, the detailed breakdown of approaches and difficulties will be explained as follows. There were many decision changes during this module design.

Difficulty:

a. At the time we presented the original proposal, we were simply trying to use AES to encrypt and decrypt messages. However, there are two problems:
    i. How do users exchange their public keys without getting replaced by others?
        1. When we use the key pair for confidentiality purposes, we typically want everyone to be able to send confidential messages to a receiver, so that only such receivers can interpret them. Therefore, AES as a symmetric key algorithm cannot perform such security as the cipher text could be switched by someone else in the middle. Therefore, we decided to use the Asymmetric key algorithm, RSA, instead of AES. RSA has a key pair, public key for encryption and private key for decryption. Therefore, user A can use B's public key to encrypt the message and only B's private key can decrypt the message in order to confirm the correct decrypted data.

ii. How do users identify the identity of the message that is truly sent by the user A to user B and vice versa, and not the middle man who has the public key of the user A or B?
    1. On the other hand, when we use the key pair for authentication purposes, we typically want everyone to be able to establish a signature to a message that the message really comes from a certain originator, and not from anybody else. We decided to use the RSA's sign and verify methods. In such a case, the originator would use his own private key to sign the message, and the receiver could retrieve the originator's public key to verify it. If the verification pass, the receiver would know the message comes from a certain originator.

b. To solve the first problem, we came up with the solution of using the cloud database (Google Firebase).
    i. Only authenticated users can upload their public keys to the cloud.
    ii. Users can retrieve the public key of the other user from the cloud.

c. To solve the second problem, we have come up with two solutions.
    i. User A first use his own private key to **encrypt** and then use user B's public key from cloud to **encrypt** the message so when user B receives the ciphertext, first use his own private key to **decrypt** the ciphertext and finally use user A's public key from cloud to **decrypt** the message. Therefore, the key distribution and identities authentication are both satisfied and perform a secure information exchange between the users in our AR game.
    ii. Instead of double encryption and decryption, we finally chose this method that is digital signature. By signing the original text using our private key, we can get a hash which can be verified by our public key. Initially C# RSA only provides encryption with public key, decryption with private key and not vice versa. After conducting research, we learned that asymmetric sign/verification and encryption/decryption are entirely distinct concepts. Digital signature is not encrypting or decrypting the data, but a hash which does not leak original information or pattern.

Breakdown:

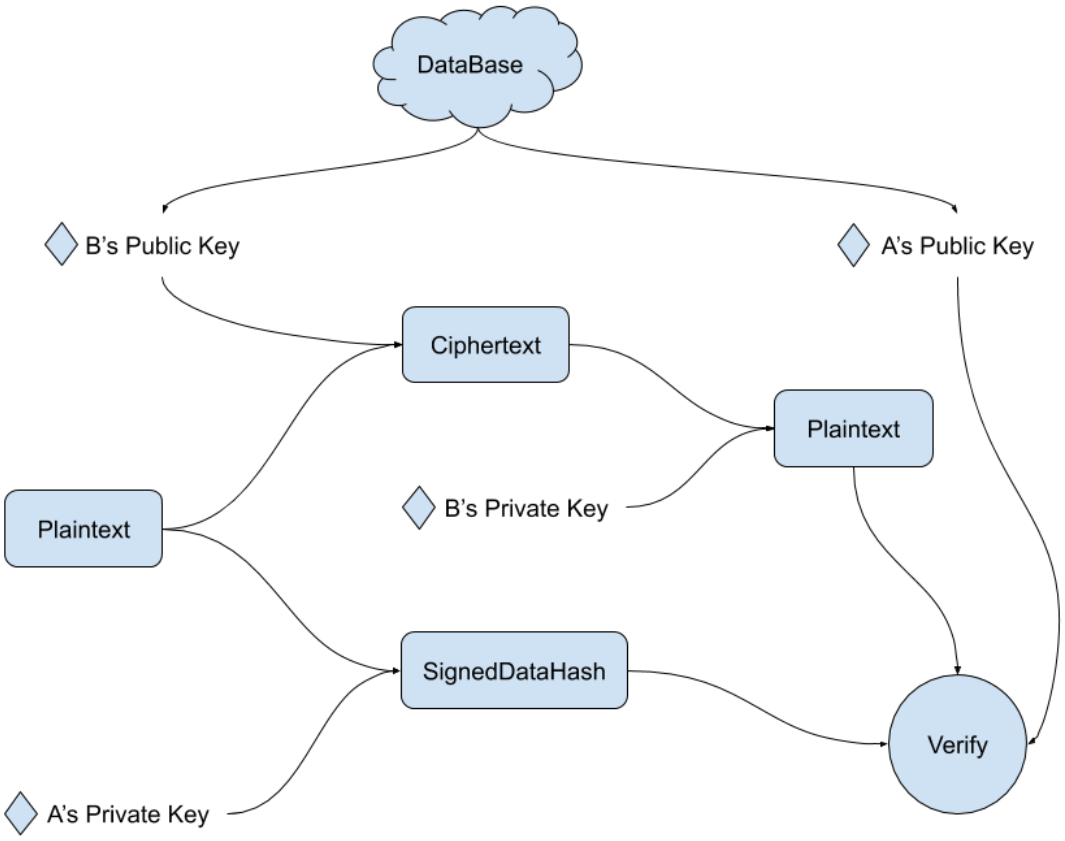

Figure 2: Communication workflow of Device A sending messages to Device B to prevent Man In the Middle Attack

a. Photon API is used here to send and receive Ciphertext and SignedDataHash between two communicating devices.
b. As soon as Photon API pairs the devices, both of them create a new RSA key pair and upload their public key to the cloud for communication this time to prevent Dictionary Attack.
c. Implemented functions to encrypt, decrypt, sign, verify using RSA.
    i. *Encrypt:*
        1. Fetch the public key of the other device from the cloud.
        2. Cipher plaintext with the public key we fetched.
    ii. *Decrypt:*
        1. Decrypt the ciphertext we received with the private key we generated previously.
    iii. *Sign:*

1. Sign the plaintext we are going to send to with our private key.

iv. Verify:

1. Verify the plaintext by using the other device's public key and the signature that comes with the ciphertext which should be the signature of the other device. If not, it means we get hacked that there is another device that is talking to me.

Technical Skills & Knowledge Takeaway

In this section, we will highlight key concepts and technologies learned from this project as a conclusion.

[1] Augmented Reality(AR):

We have learned how to create a user interface for user account creation and log in system in Unity, utilized AR knowledge on spawning and despawning game objects with various functionalities. We also learned how to create an interface of chatting rooms with virtual characters to perform communication through input text box and chatting box.

[2] Cloud Database:

We have not only learned how to sync data and access user information in real time in Firebase, but also store users' information securely in the database.

[3] Hashing:

We have learned how to choose and use suitable hashing methods for security purposes such as password storing. Comparison of different hashing functions such as MD5 and SHA.

[4] Computer vision:

We have learned how to implement Manomotion API to detect hand gestures and depth from the mobile device with the camera in real time.

[5] Asymmetric key algorithm and Digital Signature:

We have learned that Asymmetric encryption/decryption: everyone can encrypt using a public key, only the host can decrypt using a private key. Asymmetric signature/verification: only the host can sign using a private key, everyone can verify using a public key.

[6] Key distribution and Authentication:

We have learned to combine RSA encryption/decryption with RSA sign/verify methods to secure validation of the message and also authenticate the identity of the sender and receiver to prevent Man in the middle attack.