Comp Sci 7092 - Week 5 Blog

Sree Ram Boyapati	a1775690@student.adelaide.edu.au

What have I done this week, and what are the related outcomes?

Taking the professor's suggestions, I have started working on the android client work.

I have tried to set up an HTTP request and response objects and set up the package structure for the project. I have learned about android wireless ADB debugging to test my application on the phone. As part of my learnings, I have explored how to make network calls and decided on to use okhttp based networking stack by Square. I have found Google Tink[1] and was exploring if the API provided by it can be used to generate RPIs. The present HKDF library has only 38 stars on Github and google tink is internally used in android.

I spent a good chunk of time learning about android 10+ based development. These are the following things I have done to ramp up with my teammates as I primarily concentrating on implementing server features in the last sprint.

Outcomes:

1. Removed hardcoded or embedded dependencies and completely using remote dependencies for hkdf and alt beacon.
2. Refactor permissions to show the permission requirements when the app restarts. See Figure. 2
3. Replace UI components with material design[2] components for all the UI. We can later extend this using animation for good UX experience
4. Setup a web-service structure to make calls with the server. Also, add TLS 1.3 support and the relevant dependencies. Figure 1 shows the structure of the interactive service.

Challenges

1. Make Network calls in the non-UI thread and give a framework.
2. AsyncTasks have been deprecated in API 30 and need to explore something more primitive using callables.
3. I am planning to set up controllers which make multiple service requests in a Callable Thread and pass the response as an Object.
4. How to use encrypted SharedPreferences[3] to store registration token and value.

[1] "google/tink: Tink is a multi-language, cross-platform ... - GitHub." https://github.com/google/tink. Accessed 31 Aug. 2020.
[2] "Components - Material Design." https://material.io/components/. Accessed 31 Aug. 2020.
[3] "androidx.security.crypto | Android Developers." 10 Jun. 2020, https://developer.android.com/reference/androidx/security/crypto/package-summary. Accessed 31 Aug. 2020.

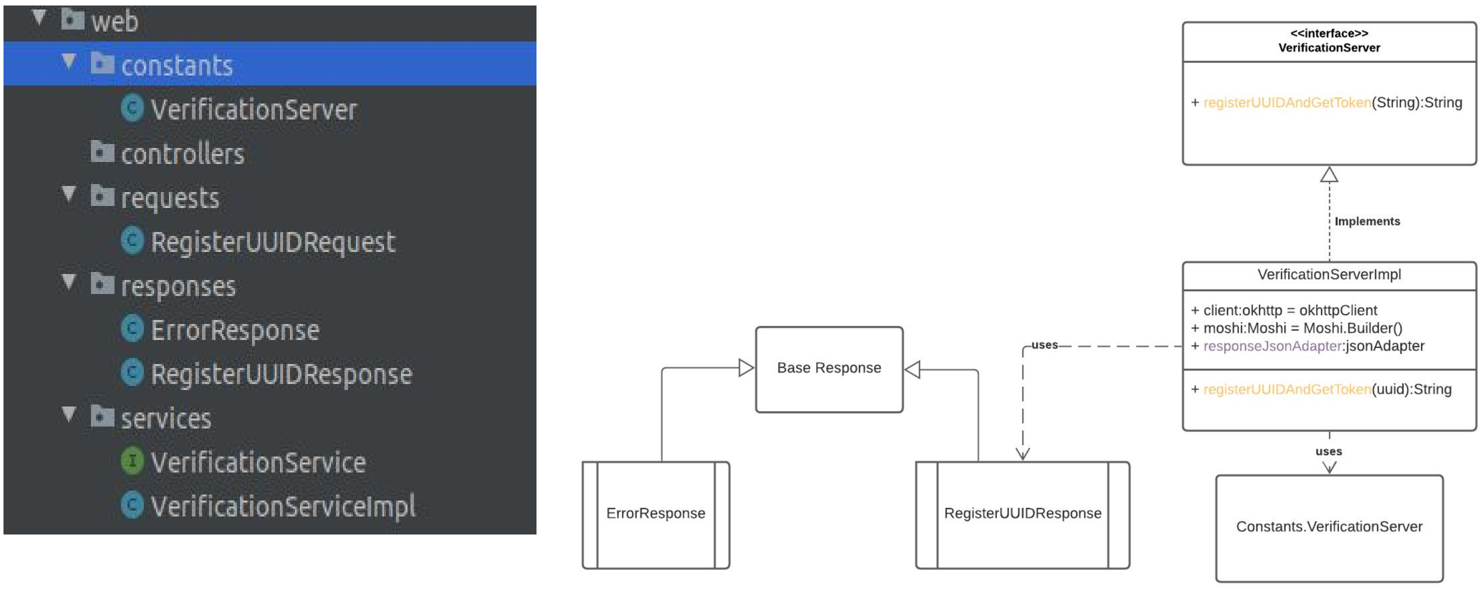
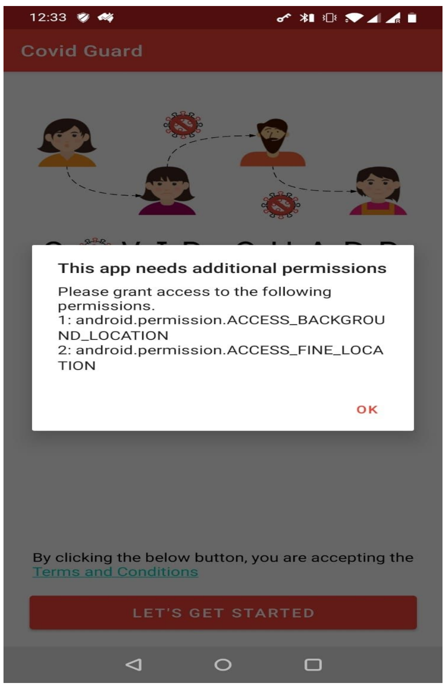


Figure 1: Project Structure and UML Diagram of the Server Interaction with Client



This app needs additional permissions

Please grant access to the following permissions.
1: android.permission.ACCESS_BACKGROUND_LOCATION
2: android.permission.ACCESS_FINE_LOCATION

By clicking the below button, you are accepting the Terms and Conditions

Figure 2: Android permission requirements

Paper Review #1

Chen, Bo-Rong and Y. Hu. "BlindSignedID: Mitigating Denial-of-Service Attacks on Digital Contact Tracing." ArXiv abs/2008.09351 (2020): n. pag.

Why have I chosen this paper?

From a security perspective, Denial-Of-Service (DoS) Attacks on the client have not been studied a lot. DoS attacks on the server have standard mitigation techniques. DoS attacks are possible on the client. In Vaudenay et al. 2020 [1], Militia or lazy-student attacks have been described in which a group of attackers attack certain phones to declare them as positive. Storage on phones might be limited.

A DoS attack on the phone can be initialized by sending too many chirps to a phone in a single scan period. We do not know the validity of the ephemeral id's or if they are similar to the ones sent earlier. We store all the ephemeral id's in the device.

According to the paper, BLE has a bandwidth of 1-2 Mbps and can store 1-2GB per hour of chirps data. With compression around 7 GB per day of 8-hour working day.

Is it possible in GAEN?

1. User can generate n number of RPIKeys using random salts to generate distinct RPIs and modify the application to generate more chirps.
2. If we are not rate-limiting by Bluetooth MAC address, n RPIKeys will generate n RPIs with same EN Interval Number.
3. Verifying co-related RPIs can be hard.
4. If we increase the number of scans to collect more genuine samples for better detection of close contact, The attacker can overwhelm the internal storage of the user.

The paper explored an approach by modifying the EphiIDs by using a blind signature[4]. By using verification, the paper tries to limit the number of Ephi IDs created in an interval.

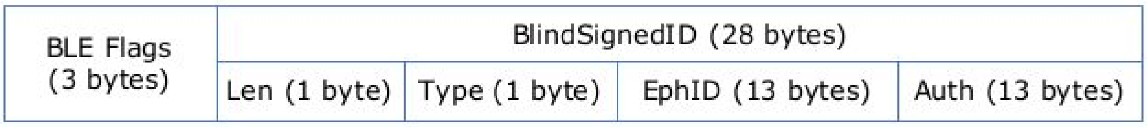
Pre-requisite Knowledge:

1. DP-3T protocol - The author suggests a modification based on DP-3T. Since GAEN is related to DP-3T, We can leverage the information here.
2. Pseudorandom functions[5] (PRFs) are different from pseudorandom generators (PRGs). AES-CTR is a PRG and HMAC-SHA256 is a PRF.

[4] "Blind signature - Wikipedia." https://en.wikipedia.org/wiki/Blind_signature. Accessed 30 Aug. 2020.
[5] "Pseudorandom function family - Wikipedia." https://en.wikipedia.org/wiki/Pseudorandom_function_family. Accessed 30 Aug. 2020.

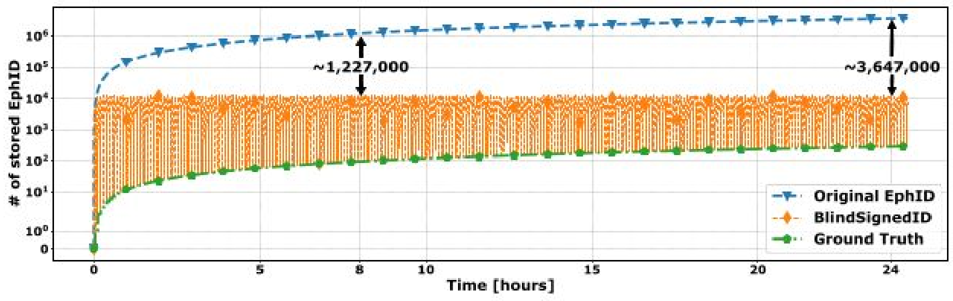
Conclusion of the Research

1. In Section 3.3, The paper established that introducing jamming based approaches at the MAC layer and Physical layer is a constraint to adaptability as the Application needs to run on commodity hardware which may lack compatibility. This is very relevant as we are looking for approaches within the protocol.
2. Uses TESLA, broadcast authentication protocol which is based on loose time synchronization (clock drift between sender and receiver does not go beyond a threshold). The is useful to check if the signed RPI is actually from the sender as the key is known at a later time.
3. The user lets the signing authority know the RPIs being generated 2 days earlier and gets the signed RPIs from the server. Before a day, Auth is generated for time interval by the server which is sent along with signed RPI. When the receiver gets an RPI and auth. If the auth does not match for two consecutive, chirps, the receiver will know that the RPI is not authenticated. Paper proposes in-place authentication to prevent DoS attacks on storage.
4. To send RPIs on day t, User maps a nonce with encrypted Auth key which user decrypts as the user knows the encryption key. Sends the auth and signed RPI to other users.

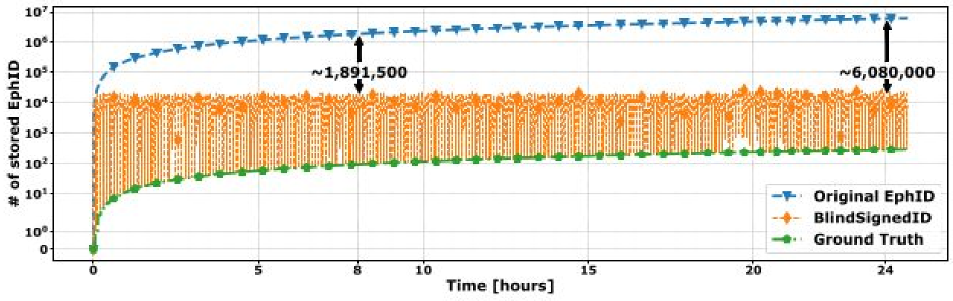


BLE Flags (3 bytes)	BlindSignedID (28 bytes)			
	Len (1 byte)	Type (1 byte)	EphID (13 bytes)	Auth (13 bytes)

Figure 3: Fig3 of the paper detailing BLE Beacon Format



(a) 2 Attackers.



(b) 4 Attackers.

Figure 4: Attack Experiments

Usefulness in the project

1. TESLA [2] based broadcast authentication protocol to verify if the RPI is indeed received from a certain individual seems very useful. Further research might be needed here.
2. Blinded RPIs using blind signatures based on RSA seems to be vulnerable to RSA Blinding attack. The server might sign an attacker's RPIs on the behalf of the user which will make the entire algorithm irrelevant.
3. The approach does not seem to follow a decentralized architecture with server playing a huge role. The server generates a key-value pair every day and has to blindly sign the RPIs and send the blind signatures to the user. The user gets the real signature by using inverted blinded seed used for signing the RPI.
4. The server knows the identity of the user to determine the user is unique. The DoS attacks can still take place as the attacker will send signed RPIs of the bogus users than the real users.
5. RSA blinding attacks [6] is possible and the attacker may trick the server. However, The authors are adding a PREFIX chosen for that day as padding. There is a third party mix server to not leak the user's identity to the central server which is the signing authority when it generates authenticator code.
6. There are too many moving parts and huge download cost to download the nonces and encrypted auth to use for the next day. There is a lot of processing power needed to generate auth for each time interval by verifying the RPI. GAEN uses 10-minute intervals. There is a higher window for the attacker to attack and guess the auth code for the interval reducing the performance.
7. If the Signing authority is down for 2 days, Applications may not work for the next two days. In such scenarios, the attacker will just DDoS the mixer or signing authority.

[6] "Blind signature - Wikipedia." https://en.wikipedia.org/wiki/Blind_signature. Accessed 30 Aug. 2020.

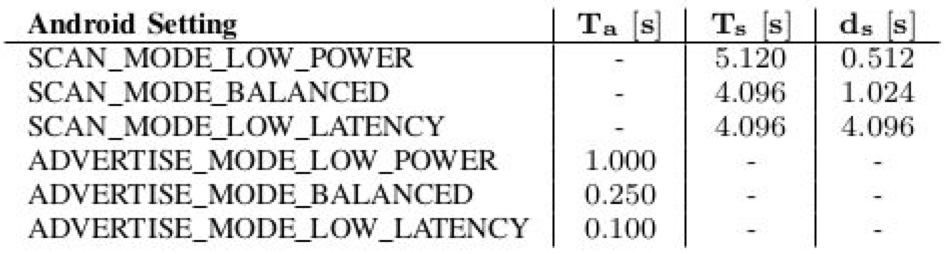
Paper Review #2

Gentner, C. et al. "Identifying the BLE Advertising Channel for Reliable Distance Estimation on Smartphones." ArXiv abs/2006.09099 (2020): n. pag. -

What have I chosen this paper?
In the previous week, I have read about using M-of-N Detector to find out about TFCTL (Too-Close-For-Too-Long) cases and set M according to RSSI strength [3].

In this paper, RSSI strength at the transmission stage is estimated by finding the Advertising Channel in which the beacon was transmitted. This will help tune the RSSI vs M parameters in the binary classifier.

Pre-Requisite:
Bluetooth has a scan mode to receive beacons and advertise mode for sending beacons. D(s) is the scan time. Ts is the scan interval. Ta, is the advertising interval.



Android Setting	T_a [s]	T_s [s]	d_s [s]
SCAN_MODE_LOW_POWER	-	5.120	0.512
SCAN_MODE_BALANCED	-	4.096	1.024
SCAN_MODE_LOW_LATENCY	-	4.096	4.096
ADVERTISE_MODE_LOW_POWER	1.000	-	-
ADVERTISE_MODE_BALANCED	0.250	-	-
ADVERTISE_MODE_LOW_LATENCY	0.100	-	-

TABLE I: BLE parameterizations in Android.

Conclusion of the Research

1. Indeed, the BLE host control interface, which is used for data exchange between radio and smartphone, specifies that incoming advertising packets are reported to the host without any channel information. However, channel information can be deduced based on the interval of beacons. Scanning is toggled in a periodic fashion from channel 37, 38, 39.

2. Based on the below formula, the authors propose a classification of scanning interval where Ts and ds relate to the scan mode. The time interval $I_{c(k)} = [T_l, T_r]$ and t_g is the guard time to account for minor clock drifts

$$\begin{aligned} \hat{t}_{l,c}(k) &= 3 \cdot (k-1) \cdot T_s + (c-37) \cdot T_s + {}^{t_g}/_2 \\ \hat{t}_{r,c}(k) &= t + 3 \cdot (k-1) \cdot T_s + (c-36) \cdot T_s - {}^{t_g}/_2 \,. \end{aligned} \quad (3)$$

3. The Bluetooth is first started with scan mode low power to detect BLE signals from far distances and record the time. After getting the initial time t, It restarts the scan again in SCAN_MODE_LOW_LATENCY mode which takes more power.

Algorithm 1: Android BLE Channel-Detector

```
doInit = true;
while 1 do
    if doInit then
        Scan-Mode = SCAN_MODE_LOW_POWER;
        Re-Start-Scan();
        doInit = false;
        Channel-Detection = false;
    if BLE signals detected then
        Channel-Detection = true;
        Scan-Mode = SCAN_MODE_LOW_LATENCY;
        Re-Start-Scan();
        t = GetSystemTime();            // see Eq. (3)
    while Channel-Detection do
        if BLE signal received then
            /* classify BLE signal into $\hat{I}_{37}$, $\hat{I}_{38}$ or $\hat{I}_{39}$ using Eq. (3) */
            ClassifyChannel(t, GetSystemTime());
        if GetSystemTime() - t > Max-Scan-Time then
            Re-Start-Scan();
            t = GetSystemTime() ;       // see Eq. (3)
        if No signals detected then
            doInit = true;
            break;
```

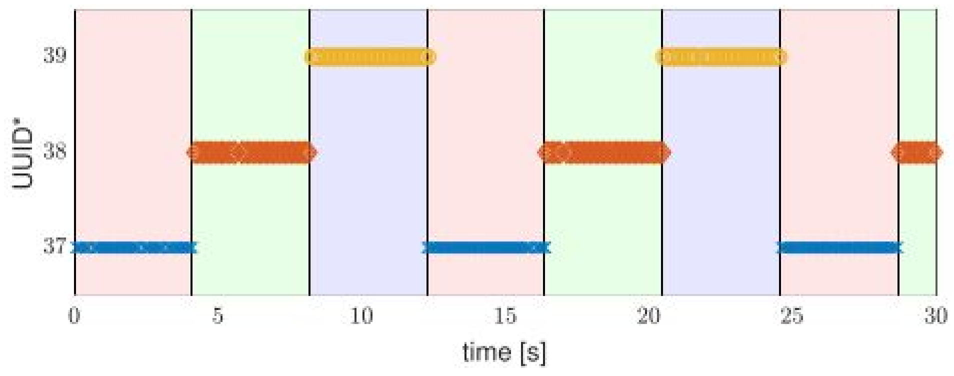


Figure 5: Payloads of same UUID received in different channels across 30 seconds

Usefulness in the project

1. The authors are attempting a stateful scanner to classify BLE payload based on the interval from the last scan. Based on the time difference and the fact that scan channels rotate from 2.402 GHz, 2.426 GHz and 2.480 GHz
2. Using SCAN_MODE_LOW_LATENCY in a while loop takes away 5-20% of the battery. However, detection of the channel without an offset by the SCAN_MODE_BALANCED needs to be explored. Requirements for a COVID-19 contact tracing application that it should at least run for 8 hour working days.
3. The approach is very phone-dependent. The authors have mentioned that Huawei P20 Lite and Samsung Galaxy M20 do not comply with the BLE standard of scanning.
4. There is a bit of ambiguity on how the restart of the scan to switch from BLUETOOTH_LOW_POWER to BLUETOOTH_LOW_LATENCY. According to StackOverflow[7], There is a 50-second delay for two immediate scan jobs. Also, alt beacon only runs the Bluetooth scanning in low power mode.

If the approach really works, We can adjust received RSSI based on the scan power and the channel we received it. For higher power, we can add an offset to the distance. This will help reduce false-positive cases in our detector.

[7] "LowPower AltBeacon Scanning Clarifications - Stack Overflow." 7 May. 2020, https://stackoverflow.com/questions/61632067/lowpower-altbeacon-scanning-clarifications. Accessed 31 Aug. 2020.

References

1. Vaudenay, S. “Centralized or Decentralized? The Contact Tracing Dilemma.” IACR Cryptol. ePrint Arch. 2020 (2020): 531.
2. Perrig, Adrian, et al. "The TESLA broadcast authentication protocol." Rsa Cryptobytes 5.2 (2002): 2-13.
3. Hatke, Gary F. et al. “Using Bluetooth Low Energy (BLE) Signal Strength Estimation to Facilitate Contact Tracing for COVID-19.” MIT Lincoln Laboratory.