Weekly Blog 9

This week I mainly worked on the completion of the regeneration code and its integration with the matchmaking process as well. Also, I have developed a second screen UI for displaying important metrics like ENIN, TEK and RPI to avoid the use of LogCat and improve the transparency of various metrics involved in Exposure Notification. Some of the relevant commits are as follows:

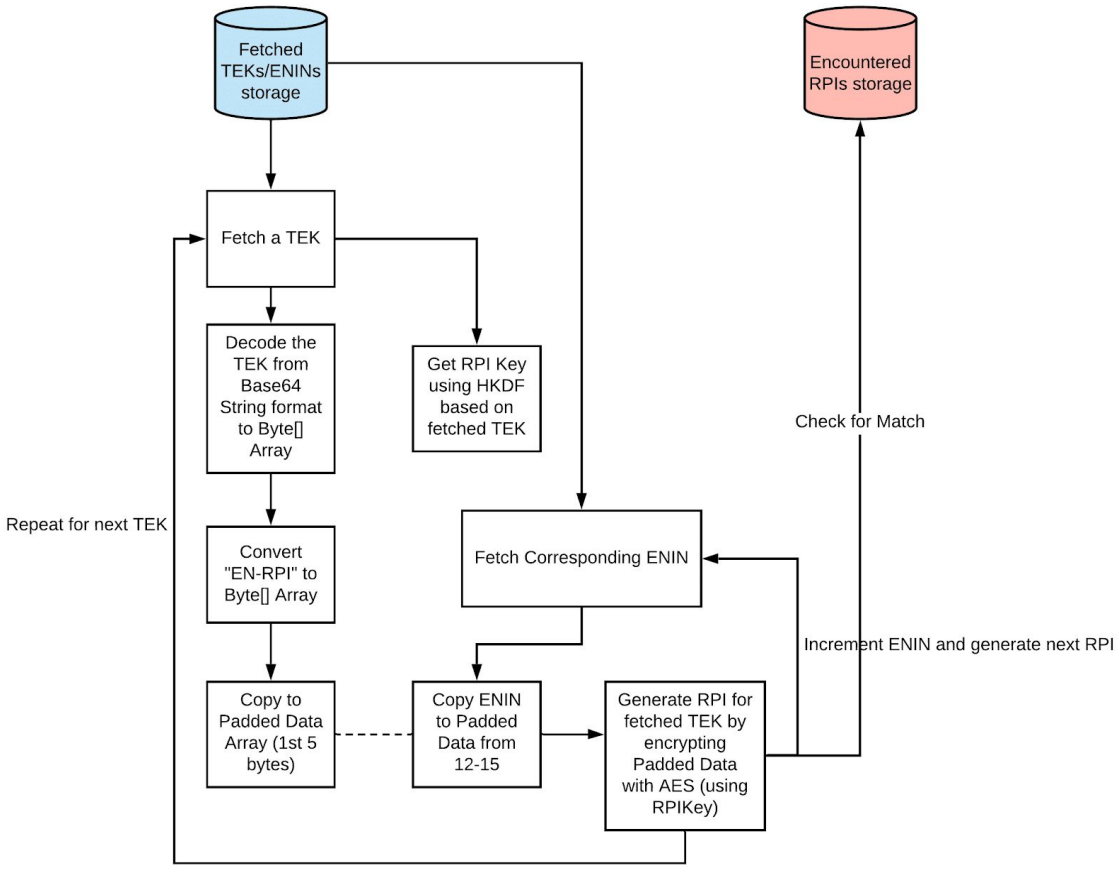
1. Second screen UI and developer metrics commit

   The following are the screenshots which display the current RPIs being received from a neighbouring phone, the current TEKs that are used to derive the RPI. The current UNIX epoch timestamp: the ENIN at which the TEK is derived.

   This screen was mainly included to explicitly show the working of the app without the need of LogCat. As seen in the third screenshot above, “No RPI is currently being received” will be displayed when no phone is nearby. When a phone with the app installed is nearby, the neighbour’s RPI will be received and displayed here.

2. Regeneration functions

   As part of the regeneration function, I had described my algorithm in Weekly Blog 8: MWS. The algorithm workflow can be seen below:



Following the previous logic conforming to GAEN's official documentation, I have clubbed and created the code which regenerates the RPIs from TEKs. For this, my I have fetched the TEKs and the corresponding ENIN (which was uploaded to the server and then downloaded for storing locally) and passed onto the function which regenerated around RPIs starting from the fetched ENIN. The entire process was repeated for all the TEKs and all the generated RPIs were passed as an ArrayList data argument to the matchmaking functions.

Sample output:

TEK: [-42, -103, -22, -10, 69, -70, 95, -67, 71, 2, 125, -3, -86, 68, -30, -58] ENIN:**128312983;** RPI: [86, 83, 32, 60, 134, 88, 51, 23, 11, 16, 18, 18, 42, 93, 178, 11]

TEK: [-42, -103, -22, -10, 69, -70, 95, -67, 71, 2, 125, -3, -86, 68, -30, -58] ENIN:**128312984;** RPI: [95, -44, -72, 36, 36, 52, 25, -95, 91, 92, 92, 83, -5, -119, -78, -34]

TEK: [-42, -103, -22, -10, 69, -70, 95, -67, 71, 2, 125, -3, -86, 68, -30, -58]

ENIN:**128312985;** RPI: [125, 35, 58, 133, 53, 23, 49, 2, 243, 167, 132, 232, 122, 229, 123, 245]

As seen in the above output, taking the TEK and its first ENIN, all the subsequent RPIs were derived. These are stored in an ArrayList for matchmaking as mentioned before.

3. Matchmaking logic commit

    a. Fetch the stored encountered RPI in rpiArrayList
    b. Fetch the first TEK and regenerate its RPIs
    c. Iterate and compare the RPIs regenerated with the RPIs from step a.
    d. Repeat step b until all TEKs are used.

Notification toast upon exposure: Testing procedure.

1. Let there be two phones A and B
2. Phones A and B come in proximity with each other for some time.
3. During the encounter, each phone's RPIs are exchanged and stored by the respective phones.
4. Phones A and B then part ways.
5. After a while phone A submits itself as a COVID positive case.
6. When phone B clicks for "*Check For Positive Contact*" button, phone B gets the notification: **You have been in contact with a COVID positive case. Seek medical attention immediately!**

Sample workflow screenshots depicting the scenario is shown below: