Weekly Blog 4

As part of, I undertook the development of GAEN API from scratch. Following are the learnings/outcomes:

1. I created a foreground process called: Exposure Service to generate TEKs and RPIs in the background even when the app was not in use/when killed.
    a. For this, a class that extends a Service class in JAVA was created.
    b. Implemented two objects utilizing ScheduledExecutorService in JAVA.
    c. The two objects were used to call the TEKGenerator thread, which generated the TEKs and RPIgenerated threads which created the RPIs.
    d. As part of the TEKGenerator thread, I used java.security.SecureRandom library for generating a cryptographic pseudorandom number generator.
        i. GAEN mentions the function CRNG or cryptographic number generator. Verified my implementation with a project found on GitHub and also using Stack Overflow.
    e. The secure random number generated was used for deriving the RPI Key.
        i. Used the standalone HKDF library from https://github.com/patrickfav/hkdf to make use of the HKDF function.
        ii. GAEN Documentation (https://covid19-static.cdn-apple.com/applications/covid19/current/static/contact-tracing/pdf/ExposureNotification-CryptographySpecificationv1.2.pdf?1) explains about using the HKDF function as HKDF(Key, Salt, Info, OutputLength).
        iii. Implemented the same function with the key parameter as TEK, salt as NULL, info as the byte array of the string “EN-RPIK” in UTF-8 format and OutputLength as 16 bytes.
    f. Successfully generated the RPI Key.
    g. I started with the implementation of RPIs following the GAEN documentation.
        i. RPI ← AES128(RPIK, PaddedDataj)
        ii. Used the RPI Key generated from the earlier step
        iii. Padded data comprised of:
            1. PaddedDataj[0...5] = UTF8("EN-RPI")
            2. PaddedDataj[6...11] = 0x000000000000
            3. PaddedDataj[12...15] = ENINj

   iv. Implemented the 0 to 5 bytes as a byte array of the string “EN-RPI” in UTF-8 format.
   v. Added the padded zeros for 6 to 11 bytes.
   vi. In progress: generation of ENIN or ENIntervalNumber
2. Tested the key generation by hardcoding the TEK and deriving the RPI key.
3. Confirmed that the same keys give the same RPI Key and RPIs
4. This could be used for later downloading the TEKs of infected users and generating to see if it generates any RPIs that matches with locally stored RPIs (scanned during the encounter)
5. TO-DO:
   a. Generate RPIs with ENIN
   b. Create the Bluetooth payload
   c. Test the entire process using the same key generation
   d. Proceed with the next steps.

HMAC-based Extract-and-Expand Key Derivation Function

What research question does the paper answer?

The paper comprehensively explains about HMAC-HKDF, which is an HMAC based key derivation function. HKDF is primarily used for deriving keys from existing secure random numbers for added security. Following are the main points:

1. HKDF uses an extract and expands methodology.
    a. Extract
        i. By extract, the HKDF takes a number that would not be truly random and can have some of its bytes repeated. For instance, “0b0b0b0b0b”.
        ii. In such a scenario, HKDF takes the above input keying material and uses a salt (which is yet another random data) and produces a pseudorandom key for usage in the expand phase of the algorithm.
    b. Expand:
        i. In the expand phase, HKDF takes the pseudorandom key, a parameter called info, and the length of the key to be produced and produces a secure random key.
        ii. More specifically, HKDF(key, info, outputKeyLength).
            1. Here the key is the pseudorandom key received from the extract part and info is something used to meaningfully tag the extraction process.
            2. For example, in GAEN, RPI Key is derived using HKDF(key, info, outputKeyLegnth).
                a. Here the key is supplied by us using a cryptographically pseudorandom number generator.
                b. Here info is the byte array of the string “EN-RPIK” in UTF-8 format.
2. In some situations like the scenario in our project, we are already having a pseudorandom number generated using java.util.SecureRandom library. In such situations, we directly use the expand() function of the HKDF library to generate the RPI Keys. Here the key is TEK or the temporary exposure key, info is the byte array of the string “EN-RPIK” in UTF-8 format, and outputKeyLength is 16 bytes.
3. Salt can or can not be used with HKDF. Authors recommend the use of salt to add more entropy (level of randomness) to the output keying material. The salt can be any random number and can be re-used. GAEN API does not use salt.

4. The info parameter is again optional. It provides contextual information about the key in the form of a byte array.

What is the conclusion of the research?

The conclusion of the research is an in-depth explanation of how HKDF works. Methods such as extract, expand are explained in reason to when and where they need to be used. Parameters such as salt, info and outpukeylengths are also explained.

How can you apply this to your project?

With the knowledge, I have successfully implemented the derivations of RPI keys from TEKs using HKDF. This is part of the GAEN protocol we are trying to implement from scratch.

Trust and Transparency in Contact Tracing Applications

In this paper, the authors propose fact sheets that contain questions concerning data, privacy, and requirements for various applications. The authors claim that the use of factsheets could be potentially used for analyzing the trust and transparency of applications used by the general public/public health authorities.

We analyze the factsheets corresponding to the current model of the project that we have/is in progress.

GENERAL

1. What is the scope of use of the application?

   The scope and use of the application is to enable contact tracing digitally for tracking users infected with COVID-19

2. What policies or laws applicable to the development, deployment or usage of this application? How do you ensure compliance with these regulations?

   We ensure that any data collected is stored in the device of the user and never leaves the device. When a person is indeed infected, only anonymous identifiers are uploaded to the server to identify potential contacts

3. Does this application connect to any other applications or IT systems (for example, public health, clinical laboratory, or hospital systems)? Identify the technique used for establishing contact (Bluetooth, location tracking via GPS, etc.)

   The application uses BLE as the technology for transmitting BLE payloads comprising of anonymous identifies. The application does not use GPS data.

4. What are the specific requirements for the efficacy of tracking and contact Identification? Distance the span of space that is used to identify a contact event. Time amount of time individuals are within the required distance to meet the threshold for exposure risk

   The tracking is done when the two users are in close proximity with each other (around 6 feet). The GPS data is not collected. After infection, a COVID positive

user will upload his anonymous identifiers to the server from which other users download and checks for a match. The API acts merely as an exposure notification indicator. Following which, the user can voluntarily go to a PH and submit his diagnosis

5. What concerns (positive and negative) might the beneficiaries have in how does the service work? How are these concerns addressed?

   The one concern of using this application for contact tracing is the creation of social graph is not easy. Specifically, the contacts are only notified of the exposure. It does not use any GPS data to pinpoint their location and submission to PH. However, from a privacy perspective, this is much more advantageous for the average user.

DATA

1. What data is collected by the application? Include data collected directly by the app, from the user, and data accessed from other applications/system.

   The only data collected from the user is temporary exposure keys. However, the user is provided with the consent to upload this data to the diagnosis key server. The data is completely anonymised and not at the disposal of other users/PH.

2. Is this data combined with any additional details about an individual, community, locale, or environment?

   No, the data is not combined with any of the above.

3. Identify any data collected that is of a sensitive nature (for example, health conditions, symptoms, etc.)

   Only when the user provides his/her consent to upload the data, they are uploaded to the key server. This does not reveal the identity of the user.

4. How is the collected data used?

   The collected data is downloaded periodically by other users to check if they have come in contact with a COVID positive contact user. They are anonymised and inbuilt mechanisms are used for identifying exposure.

5. Who has rights to access the data (explicitly define people, agencies, and/or organizations) What is the policy on data retention and deletion?

   The data is accessible to no one. The only available data is in the diagnosis key server which is anonymised. The data of an infected user is deleted every 14 days.

6. What mechanisms are used to keep the data secure?

   The anonymised identifiers are derived from cryptographically random numbers whose randomness prevents identical keys being produced by two users. Further encryption algorithms like AES and HKDF are used for deriving the identifiers used for broadcasting.

PRIVACY

1. Did you implement the right for a user to 1) withdraw consent, 2) object and 3) be forgotten in the application?

   Yes, currently the user has the consent to upload his/her data to the diagnosis key server. The data once uploaded will periodically be deleted every 14 days.

2. Does the application allow people to learn any personal information about others?

   No, the app itself does not collect any personal information such as name, mobile number. It is completely dependent on random unique identifiers for matchmaking purposes.

3. Are privacy-preserving techniques incorporated in the application (e.g. data anonymization, encryption, aggregation)? If so, provide details on the techniques used.

   Yes, the matchmaking process is done using unique identifiers which are derived from cryptographically secure random numbers followed by encryption algorithms like HKDF and AES.

4. What additional measures are used to protect the data and identity of infected and exposed individuals?

5. Could this application be used in a way that identifies people who are infected or at risk to 1) the developers, 2) people within an individual's social circles, 3) to those the app is warning about contact and potential exposure, or 4) to the government, employer, or managing organization?

   No, social graphing is not possible using the app. The app uses anonymised identifiers for generating a subset of those identifiers using cryptographic algorithms. Only users who have been in contact with a positive user are able to get those anonymised identifiers from which they can derive the subset of the former. As such, a match is found based on the creation of identifiers which are already present in a user's device (from the encounter with a COVID positive case). It is an exposure indicator and no personalised data is transmitted or used.

6. If the app connects to public health or hospital systems, how do you ensure that personal information isn't accessible during data sharing points?

   Same as point 5,

REFERENCES

1. Hobson, S., Hind, M., Mojsilovic, A., and Varshney, K.R., 2020. Trust and Transparency in Contact Tracing Applications. arXiv preprint arXiv:2006.11356.
2. Krawczyk, H. and Eronen, P., HMAC-based Extract-and-Expand Key Derivation Function (HKDF), 2010.