University of Regina

Research Project

Security and Encryption in the Government of Canada's COVID Alert Application

Date/Version: 2021.12.07

Author: Bernadette Veninata (200259971)

Course: CS 435 Cryptography and Network Security, Professor Qian Yu

University of Regina

Table of Contents

University of Regina

Terms used in this document

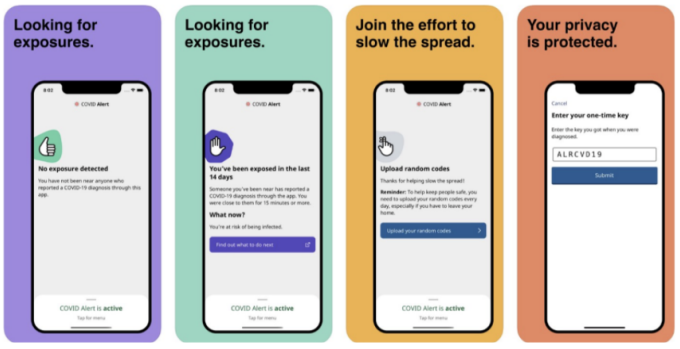
Term	Description
DDoS	Denial of Service attacks
RPI	Rolling Proximity Identifier
AWS	Amazon Web Services

1. Introduction

1.1 Project Topic

The global coronavirus pandemic has resulted in rapid transmission of the COVID-19 disease in Canada. In response, Health Canada, the national health policy department of the Government of Canada, in combination with Shopify, created a contact tracing application, COVID Alert, in order to slow the spread of the virus. The application utilizes the Google-Apple Exposure Notification framework and is available through the Google Play store and the Apple App Store. Approximately 6.6 million Canadians have since downloaded the app, representing about one in five Canadians. (Government of Canada, 2021)

The COVID Alert application will send a notification to users when they've come in contact with another user who tested positive for COVID-19. Bluetooth technology is utilized to determine a user's proximity to other users of the app, and the length of exposure, based on the strength of the Bluetooth signal. When a user receives a positive test for COVID-19 they will enter a one-time key into the COVID Alert application which will send their data and codes from the last 14 days to the government's key server which operates on the Amazon Web Service Cloud solution. If a user has been in contact with another user who has tested positive and registered their codes in the server, the user will then be notified that they were exposed.



Chevreau, C. (2020, August 1). *COVID Alert: What's Inside? - Chris Chevreau*. Medium. https://medium.com/@chris.chevreau/covid-alert-whats-inside-764d609c24fa

1.2 Research Question and Project Motivation

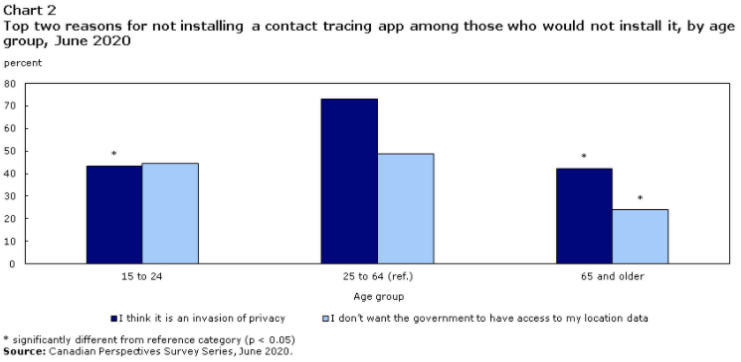
The research question that this paper will address is: What are the potential risks to users associated with the COVID Alert application and how may we seek to remedy them?

This research was prompted by concerns that the COVID Alert contact tracing application may potentially leave users vulnerable to a breach of their information. Security and privacy flaws in

the COVID Alert app can be of potential risk to users. The major concerns of the contact tracing application are it's data storage and management, architecture, privacy, and security.

Several different disciplines are concerned about the use of contact tracing applications. "Industry stakeholders like the American Civil Liberties Union and about 200 scientists, among others, responded that these apps have the potential to overreach, and developers must ensure proper technical measures are taken to ensure the app is secure." (Davis, 2020) The Electronic Frontier Foundation voiced similar concerns, focusing on cybersecurity risks wherein "hackers can target the data sent from the app and undermine the system." (Davis, 2020)

Statistics Canada conducted a survey by which Canadians have also shown reservations in downloading the application, as seen in the below chart. Mass data collection is a concern, as the application has the potential to lead to an increased breach of privacy. Therefore, the security and encryption of the application should be assessed. Furthermore, additional and more secure encryption techniques for the application should be analyzed, as well as new ideas to improve the security of the application.



1.3 Objectives

The objectives of this research paper are: To provide an analysis of the encryption techniques currently being used by the Government of Canada in its COVID exposure application, COVID Alert. This analysis will include existing security and privacy flaws of the app and potential risks to users. This paper will also put forth encryption techniques that are currently available and more secure, but not utilized by the government in its application. Along with new and innovative ideas for encryption techniques that can alleviate some of the security risks to users of the COVID Alert app.

2. Methods

This research project was completed utilizing a combined body of tools including publications, qualitative data, government websites, existing applications, books and online websites. Research compiled for the CS 435 Research project was collected and analyzed from the following sources; Publications issued by the Government of Canada, detailing the framework of the existing Covid Alert application; Health Canada's, February 9, 2021, COVID Alert: *COVID-19 Exposure Notification Application Privacy Assessment;* The Office of the Privacy Commissioner of Canada's, July 31 2020, *Privacy review of the COVID Alert exposure notification application*. Technical reports issued by Apple and Google (which published the application to their

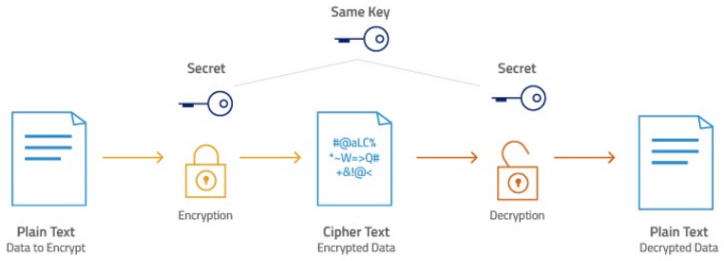
stores and provided the Exposure Notification framework portion of the application) were assessed in order to determine the privacy level of the application; Report by Apple and Google, *Privacy-Preserving Contact Tracing* (2020). The Amazon Web Service's Cloud solution was also assessed (as it hosts the government's key server which will store data of the application)*;* Amazon Web Service's technical report, January 8, 2021, *Keeping Canadians safe while protecting their privacy: COVID Alert app*. Various educational reports and case studies were researched in order to ascertain perspectives of researchers and technical experts (noted in the "References" section of paper).

3. Results (Pt. I)

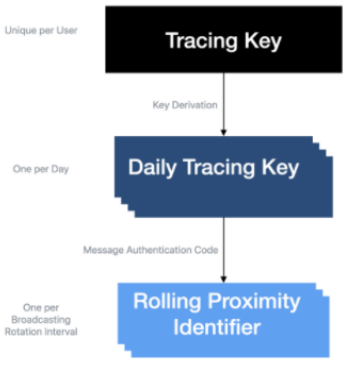
3.1 The COVID Alert Application's Security and Encryption Techniques

3.1.1 AES Encryption

The COVID Alert application is encrypted using AES (Advanced Encryption Standard) encryption. This is when plain text is converted to cipher text, and can only be decrypted by those with the private key. The same private key is used to both encrypt and decrypt the data. A key schedule is used when encrypting and decrypting data with AES, to expand an initial key (in this case the Tracing Key) into several separate round keys. This is done through the application's use of Daily Tracing Keys and Rolling Proximity Identifiers.



Secure your data with AES-256 encryption. (2020). Atpinc. https://www.atpinc.com/blog/what-is-aes-256-encryption



Key Schedule

Apple & Google. (2020, April). *Contact Tracing Cryptography Specification*. Blog Google. https://www.blog.google/documents/56/Contact_Tracing_-_Cryptography_Specification.pdf/

The Tracing key is generated when a user enables contact tracing on their device. Daily Tracing Keys are generated every 24-hours and consist of random numbers that are SHA-256 hashed and truncated to 16 bytes. This Daily Tracing key then generates random codes every five to twenty minutes, called Rolling Proximity Identifiers which are a concatenation of the Daily Tracing Key and a timestamp value. This concatenation produces a SHA-256 hash and is truncated to 16 bytes. Rolling Proximity Identifiers are locally-stored, on the user's devices and are used to determine who a user has been in contact with and their movements. When two users reach a certain proximity (within 2 metres) the app will exchange their Rolling Proximity Identifiers using Bluetooth on their devices.

When a user has a positive test for COVID-19 their Daily Tracing Keys and consequent Rolling Proximity Identifiers for the previous 14 days are collected from the Tracing key. This set of keys is called the Diagnosis key. This Diagnosis key is encrypted using a crypto public key and a crypto private key. The public key is given to the application so that the user can send their encrypted Diagnosis Key (and consequent Daily Tracing Keys and Rolling Proximity identifiers) to the key server.

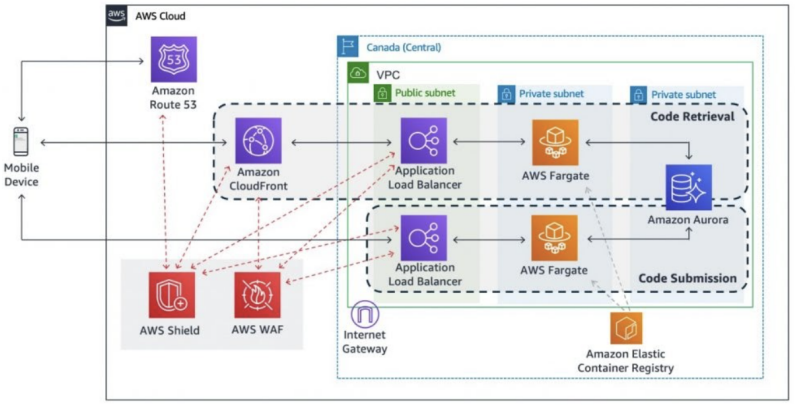
3.1.2 Bluetooth Technology

As described in the previous section, data is collected through Bluetooth technology, via an exchange of Rolling Proximity identifiers (device level) when two users of the application are within a certain distance to one another. When a user tests positive for COVID-19 their Daily Tracing Keys and Rolling Proximity identifiers for the previous 14 days, otherwise known as their Diagnosis Key, is retrieved from the Tracing Key and sent to the Government of Canada's key server within Canada (federal level). The key server stores this data, and operates on a cloud solution.

Matching is then carried out between local and server-side Rolling Proximity identifiers, at the local device level. The Rolling Proximity identifiers are checked against the exposure notification risk criteria which considers the length of exposure time and the strength of the Bluetooth signal. Based on this criteria a positive signal is sent via the Google/Apple API exposure notification layer to the app and an exposure notification is sent to the user.

3.1.3 Amazon Web Services

The Government of Canada selected Amazon Web Services, specifically the AWS Cloud, to host COVID Alert.



Keeping Canadians safe while protecting their privacy: COVID Alert app. (2021, January 8). Amazon Web Services. https://aws.amazon.com/blogs/publicsector/keeping-canadians-safe-protecting-privacy-covid-alert-app/

All mobile devices are routed to the Amazon CloudFront content delivery network (CDN), the cloud domain name system service, through Amazon Route 53. This is done through converting urls to numeric IP addresses that the system uses to communicate. The AWS Shield is used to protect against Distributed Denial of Service (DDoS) attacks in order to shield COVID Alert's DNS service. The AWS Web Application Firewall safeguards COVID Alert's Content delivery network against web exploits and attack patterns through its custom security rules.

There are two functions of the AWS service, code retrieval and code submission. In terms of code retrieval; whenever a user has an internet connection, a list of the Diagnosis Keys of users who have contracted COVID-19 is retrieved from the government's key server. This is matched to the user's locally stored keys in the past 14 days and, if there is a match, the application notifies the user that they have been exposed to the virus and provides further steps. This is seen in the above AWS diagram through the use of the Amazon CloudFront content delivery network (CDN) to obtain the list of Diagnosis Keys of the users who have contracted COVID-19 diagnoses. The CDN obtains these keys through Amazon Aurora (otherwise known as the Government of Canada's key server), a relational database of Diagnosis Keys retrieved from mobile devices. AWS Fargate works as a serverless container compute engine by which clusters are used to fetch and cache Diagnosis Keys and send user's Diagnosis keys to the Amazon Aurora database. These keys are encrypted at rest and in transit between Amazon Fargate and Amazon Aurora. Lastly, the Application Load Balancer sends the list of Diagnosis Keys, retrieved from AWS Fargate to the Amazon CloudFront content delivery network, to be sent to mobile devices.

For code submission; user's who have tested positive for COVID-19 may choose to send their Diagnosis Keys to the Government of Canada's key server (hosted on AWS). To send their Diagnosis Keys to the server, users will receive a one-time key to enter into the application, the keys are then stored in the database. This aspect of code submission utilizes Amazon Aurora (the key server) to store the keys in the database and AWS Fargate to send user's Diagnosis Keys to the Amazon Aurora database in the same way as above, in order to submit a user's codes. However, the Amazon CloudFront content delivery network (CDN) is not needed as a user is directly submitting their Diagnosis Key from their mobile device to the Application Load Balancer.

3.2 Security/Privacy Flaws and Potential Risks

3.2.1 Flaws of AES Encryption

As explained in the previous sections, the application works by generating Daily Tracing Keys and consequent Rolling Proximity Identifiers. These keys are widely distributed between many user's devices and the key server. This leaves open the possibility of re-identifying infected users through these keys. Researchers are concerned that "any proximity tracking system that checks a public database of Diagnosis Keys against rolling proximity identifiers on a user's device—as the Apple-Google proposal does—leaves open the possibility that the contacts of an infected person will figure out which of the people they encountered is infected" (Davis, 2020). "The Electronic Frontier Foundation also warned Apple and Google's proposal raises inherent risk of cyberattack, as there is currently no way to verify the device sending a Rolling Proximity Identifier. As a result, bad actors could collect Rolling Proximity Identifiers from other devices and rebroadcast them from their own device." (Davis, 2020) In order to solve this, a system in which users can send and match against other user's Rolling Proximity Identifiers without leaking the data of any users must be analyzed.

The current application protocol creates new Rolling Proximity Identifiers every five to twenty minutes regardless of any other factors. This raises the chance of a bad actor passively tracing the paths traveled by users through these Rolling Proximity Identifiers. This could be carried out through the use of sensors placed in public locations for the purpose of scanning Rolling Proximity Identifiers on user's mobile devices, tracking users for a 24 hour period and using this data to identify them. Researchers have raised concerns over "Bluetooth beacons being set up on busy street corners that rebroadcast all the Rolling Proximity Identifiers they observe. Anyone who passes by a 'bad' beacon would log the Rolling Proximity Identifiers of everyone else who was near any one of the beacons. This would lead to a lot of false positives, which might undermine public trust in proximity tracing apps—or worse, in the public health system as a whole. Taken to an extreme, bad actors could collect Rolling Proximity Identifiers on mass, connect them to identities using face recognition or other tech, and create a database of who's infected." (Davis, 2020)

3.2.2 Mac Address Tracking

The COVID Alert application uses Bluetooth technology to broadcast information about the user's device, such as it's MAC address. A MAC address is set in the Bluetooth chip in your phone and provides a unique networking identifier for your device. When Bluetooth is turned on for the COVID Alert application, the device automatically broadcasts its MAC address stored in the Bluetooth payload. Since MAC addresses are unique they can be recorded and tracked. This runs the risk of identifying a user as MAC addresses' have a small search space. Therefore, a few iterations of simple hashing utilized and deanonymization can be very easily carried out.

It is possible for a device's MAC addresses to be randomized in order to avoid device tracking. However, a carryover attack is still possible as bad actors may track the device if the randomization time of the MAC address and the time that the Rolling Proximity Identifiers update have not coincided. For example, if Rolling Proximity Identifiers are generated every 20 minutes, but the MAC address is randomized every 10 minutes, a bad actor may track all MAC addresses generated within the lifetime of the same Rolling Proximity Identifier. And, vice versa, all Rolling Proximity Identifier's created within the lifetime of the same MAC address can be tracked and connected.

3.2.3 Linkage Attacks

A linkage attack is "an attempt to re-identify individuals in an anonymized dataset by combining that data with background information". (Privitar, 2021) Researchers believe that linkage attacks are an example of a potential threat to users of the application, due to users publicly sharing their Daily Tracing Keys once per day, rather than every few minutes. (Davis, 2020) A linkage attack can occur when a user becomes infected and uploads their Diagnosis Keys (and consequent Daily Tracing Keys and Rolling Proximity Identifiers) to the key server. A bad actor may link together all Rolling Proximity Identifiers from a single day, exposing the user's daily routine. A linkage attack may also occur when the person managing the data tries to re-identify the contact history of users by correlating the Rolling Proximity Identifiers received by different users. Linkage attacks may also take the form of a user keeping track of their contacts. For example, if a user left their house once within the 14 days to visit a friend, and then the application confirms that they have been in contact with someone who has COVID-19 within that week, the user can deduce that the infected person might be their friend. This can be the case in rural areas where there are not very many people to be in contact with.

University of Regina

3.2.4 Security Issues Associated with the Cloud

The COVID Alert application's key server operates on a Cloud solution via the AWS Cloud. The AWS Cloud is not without its own security threats. This was seen on October 22, 2019 when AWS underwent a distributed denial of service attack on their Amazon Route 53 Domain Name System web service which consequently affected other AWS services for eight hours. "Amazon's own mitigation service, Shield Advanced, reportedly could not fully mitigate the attack." (Newman, 2019) Due to AWS's use of legacy DDoS mitigation tools, the Shield Advanced service's mitigations blocked legitimate customer queries in order to stop the malicious ones getting through. This left users unable to connect to the AWS Cloud for eight hours.

Bad actors can now use "multi-vector attacks that are automated and can morph every few seconds or minutes. They also typically launch low-threshold, sub-saturating, attacks which further increases the challenge for legacy DDoS mitigation tools to distinguish them from regular traffic. In these cases, organizations either end up with attacks still getting through, or taking their own services offline to keep the attacks at bay." (Newman, 2019). In this way, attackers of the COVID Alert application could utilize a denial of service attack to block service of the application's data, stored on the AWS key server, to legitimate users. If attackers are able to get through to the AWS key server through these modern denial of service attacks, it would leave user's vulnerable to having their identifiers traced, putting their identities at risk.

4. Improvements and New Solutions (Pt. II)

4.1 More Secure Encryption Techniques that Could be Utilized

4.1.1 Homomorphic Encryption

Homomorphic encryption can protect user's privacy by utilizing a homomorphic encryption scheme where the person managing the data encrypts it and sends it to the decentralized server. Then, through homomorphic encryption, computations are performed on the data without decrypting it. This further encrypts the data at rest and in transit without ever exposing it's decrypted state. Fully homomorphic encryption is a type of homomorphic encryption whereby strong security guarantees, called semantic security, are utilized. Semantic security provides strong security whereby minimal, if any, information about the plaintext can feasibly be derived from the ciphertext. Therefore, bad actors who have obtained the public key and a cipher-text, will not be able to learn relevant information about the underlying plaintext as the data is continuously encrypted. In this way, only the person who manages the data will be able to decode the data. Contacts of a user who has tested positive for COVID-19 can be identified through search queries to the server of the homomorphic encrypted data. Therefore, only the contacts of that user may be decrypted by the government and identifiers can be retrieved from the server and matched locally at the device level to assess whether a user has been exposed to the virus without revealing any personal data of individuals.

The cons of this type of encryption are such that, with homomorphic encryption requiring massive amounts of computational power, it slows down the ability to perform basic processes - since computer power will be utilized to process encrypted data while simultaneously keeping it

encrypted. Homomorphic encryption, in other words, may not prove efficient enough to run applications with real time requirements.

4.1.2 Functional Encryption

In functional encryption the person managing the data can encrypt it with a public key. Then, with the secret key, can provide keys for functions which will further encrypt the data at rest and in transit. This is different to homomorphic encryption, where the server will receive encrypted data based on computations of the function, but does not know the function used. Therefore, since the function being used to encrypt the data is known, functional encryption is not as secure as the semantic security used in homomorphic encryption. If the function is chosen with specificity, however, information relating to the input can be kept secure. Functional encryption security ensures that bad actors learn nothing about the input - the function in plaintext form may be the only extracted information. Some schemes hide the function as well. Functional encryption can provide more control in terms of decryption allowed by third parties, more so than standard encryption schemes. It is beneficial in this manner as it provides the ability to 'leak' information about encrypted data to select users without actually leaking plaintexts. Thus, Diagnosis Keys can be shared, but exposure of further users data will not occur.

The cons of a functional encryption scheme are that it requires a large processing time, especially during the decryption step. However, the cryptographic overhead could be made to be very small. Functional encryption is more practical than Homomorphic encryption in terms of the COVID Alert applications real time requirements as it is closer to practice than homomorphic encryption.

4.2 New Solutions to Remedy Security Risks

4.2.1 Contact Duration Threshold

Diagnosis Keys of the application (which consequently contain user's Daily Tracing Keys and Rolling Proximity Identifiers) are widely distributed between user's devices and the key server. This leaves open the possibility of re-identifying infected users through these keys. Therefore, minimizing the amount of contacts of a user to only the essential ones would require less sensitive data to be retrieved from and stored in the key server.

Currently, the application's contact duration algorithms are set through it's software configuration, wherein a user must be a certain distance away from another user, established through Bluetooth range. In order to minimize sensitive data, the application may add a cryptographically guaranteed rule whereby a user has to be in contact with another user over a minimum duration of time before they are registered as a contact. This way whole-day tracking will be less easily carried out because only key interactions, not all interactions, will be logged and sent to the server when the user is infected. This will cryptographically guarantee that data is limited before the threshold is met.

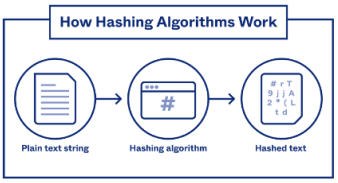
Cryptographically guaranteeing this contact duration threshold can be carried out through the Rolling Proximity Identifiers. These identifiers will be generated at the same time frequency as the contact duration threshold. This is done so that the Rolling Proximity Identifiers generated within the lifetime of the contact duration threshold may not be tracked and vice versa. Then, encryption could be carried out by hashing the current generated Rolling Proximity Identifier and the previously

generated Rolling Proximity Identifier hash, to create a key. The key contacts of the user can then be securely retrieved from and stored to the remote server.

4.2.2 Hashing

Hashing can be used to encrypt data by providing an irreversible one-way function to compute data into a unique number to be stored locally and on the key server. Therefore, when matching of data (Rolling Proximity Identifiers) takes place, in order to determine whether or not a user has come in contact with COVID-19, hashed local identifiers would be matched to the hashed identifiers on the server.

For further security, two servers could be utilized; a hashing server and a storage server, controlled/managed by different organizations. The person managing the hashing server would have a private key in order to perform the hashing function. The hashing server, which focuses on encrypted keys, would not, therefore, store sensitive data. Then, the data would be sent to the storage server. The storage server would not contain this key and therefore would not be able to decrypt the hashes of the Daily Tracing Keys (and consequent Rolling Proximity Identifiers).



Hashing Algorithm Overview: Types, Methodologies & Usage | Okta Singapore. (2021). Okta, Inc. https://www.okta.com/sg/identity-101/hashing-algorithms/

4.2.3 Mix Network

A mixed network has a minimum of three servers. The servers are controlled, however, by individual entities. Each server functions to mix hashed data of users. After this has been done, the hashed data is sent to the next of the (three) servers. At the end point, when the data arrives at the government key server storing the hashes, the final server would not have the capacity to locate or associate the previous hashed data with any user. Intermediate servers would only be able to ascertain, or determine, who a specific user is if there was improper collaboration. Ultimately, the final government server would retain all the hashed data needed to determine if a user was infected.

4.2.4 TPM chips

Hardware encryption in the form of TPM chips can also be utilized to protect the application's encryption keys once it has been sent in an unencrypted form. A TPM chip is a crypto-processor and when it reflects the system's integrity is compromised by, for example malware, then it can restart up in a quarantine mode. The purpose of this is to ensure that bad actors are unable to passively gain access to Rolling Proximity Identifiers, thereby giving them the ability to target a user if the Diagnosis Key is divulged. The chip may be placed in the motherboard or CPU of a device and, therefore, if the device containing the COVID Alert application's data has been compromised by a bad actor the system will restart up in a quarantine mode.

University of Regina

4.2.5 Universally Unique Identifier

As discussed above, if the regeneration time of the Rolling Proximity Identifiers and the Bluetooth random MAC address is not synchronized, the lifetime of all contacts within that time period may be tracked. To avoid device tracking via Bluetooth technology, the frequency of the identifiers and the Bluetooth random MAC address must be completely synchronized.

Another safeguard of the application can be carried out by the application anonymising the identity of the device. This can be done by storing a Universally Unique Identifier, only, rather than a MAC address. Universally unique identifiers can be regularly regenerated and broadcasted, providing the safety measure of not issuing or duplicating the same/identical identifier but a unique one every time. Thus, reducing the risk of device tracking.

4.2.6 Modern DDoS Mitigation Solution

As seen in the mentioned attack of the Amazon Route 53 web service, Denial-of-service attack mitigation must be analyzed. In the mentioned attack, in an effort to prevent the attack the AWS shield blocked legitimate traffic to the service for 8 hours. The Amazon Route 53 web service is responsible for handling data of the COVID Alert application, and so solutions to DDos attacks must be assessed.

One solution for the application is to use what is called Modern DDoS Protection. Modern DDoS Protection works against today's sophisticated Denial-of-service attacks. It ensures that only malicious traffic is blocked, and allows legitimate traffic through. This is done by analyzing each packet to determine if it is malicious through granular detection. Malicious traffic will then be blocked through blocking filters.

5. Discussion

In the climate of the current COVID-19 global pandemic, focusing on recent contact tracing technology, such as the COVID Alert application, and ensuring it's safety for users is an important task. This research paper has successfully answered the question of: What are the potential risks to users associated with the COVID Alert application and how may we seek to remedy them? It has been concluded throughout the paper that the major concerns of the application are it's data storage and management, architecture, privacy, and security, potentially leaving users vulnerable to a breach of their information. However, through the analysis of "Pt. I" of the paper, it has been noted that the application is secure at a baseline given the time constraints caused by the fast execution of the app in order to slow the spread of the virus. The flaws and security risks associated with the passive scanning/tracking and reidentification of users, however, are still an issue that needs to be addressed. As noted in the paper, further rules for the application can be cryptographically guaranteed in order to only retain the most key data (in terms of storing contacts for the application) to provide more security to the application. Using the pre-existing framework of the application in conjunction with further encryption techniques, such as hashing and mixed networks, would further the security of the application. The server which holds the data of the application, operating through the AWS Cloud solution, should also provide a means to further protect against DDoS attacks by implementing Modern DDos Protection and keeping on top of the ever changing, sophisticated DDoS attacks carried out by bad actors.

University of Regina

6. References

Bluetooth tracking and COVID-19: A tech primer. (2020). Privacy International. https://privacyinternational.org/explainer/3536/bluetooth-tracking-and-covid-19-tech-primer

Dabrowski, K. (2020, October 19). Cloud Security in AWS: The Most Common Issues | PGS Software. PGS Software | Application Development, Outsourcing Offshore Software Development Company, Outsourcing .NET, Java, Nearshoring. https://www.pgs-soft.com/blog/cloud-security-in-aws-the-most-common-issues/

Davis, J. (2020, April 29). EFF Warns COVID-19 Tracing Apps Pose Cybersecurity, Privacy Risks. HealthITSecurity. https://healthitsecurity.com/news/eff-warns-covid-19-tracing-apps-pose-cybersecurity-privacy-risks

Keeping Canadians safe while protecting their privacy: COVID Alert app. (2021, January 8). Amazon Web Services. https://aws.amazon.com/blogs/publicsector/keeping-canadians-safe-protecting-privacy-covid-alert-app/

Health Canada. (2021, February 9). COVID Alert: COVID-19 Exposure Notification Application Privacy Assessment - Canada.ca. Government of Canada. https://www.canada.ca/en/public-health/services/diseases/coronavirus-disease-covid-19/covid-alert/privacy-policy/assessment.html

Khader, D. H. S. (2020, December 22). How can homomorphic encryption address privacy in COVID-19 apps? International Association of Privacy Professionals. https://iapp.org/news/a/how-can-homomorphic-encryption-address-privacy-in-covid-19-apps/

Newman, S. (2019, December 18). Learning from the Amazon Web Services (AWS) DDoS Attack. Corero. https://www.corero.com/blog/learning-from-the-amazon-web-services-aws-ddos-attack/

Privacy review of the COVID Alert exposure notification application - Office of the Privacy Commissioner of Canada. (2020, July 31). Office of the Privacy Commissioner of Canada. https://www.priv.gc.ca/en/privacy-topics/health-genetic-and-other-body-information/health-emergencies/rev_covid-app/

Privacy-Preserving Contact Tracing - Apple and Google. (2020). Apple. https://covid19.apple.com/contacttracing

Privitar. (2021, May 3). Linkage Attack. https://www.privitar.com/glossary/linkage-attack/#:%7E:text=A%20linkage%20attack%20is%20an,sets%20to%20establish%20identifying%20connections.

Romailler, Y. (2019, November 25). Forget Homomorphic Encryption, Here Comes Functional Encryption. Kudelski Security Research. https://research.kudelskisecurity.com/2019/11/25/forget-homomorphic-encryption-here-comes-functional-encryption/

Sowmiya, B., Abhijith, V., Sudersan, S., Sakthi Jaya Sundar, R., Thangavel, M., & Varalakshmi, P. (2021). A Survey on Security and Privacy Issues in Contact Tracing Application of Covid-19. SN Computer Science, 2(3). https://doi.org/10.1007/s42979-021-00520-z

Yi, X., Paulet, R., & Bertino, E. (2014). Homomorphic Encryption and Applications (SpringerBriefs in Computer Science) (2014th ed.). Springer. https://doi.org/10.1007/978-3-319-12229-8