

Private Contact Tracing based on Trajectory-data through Trusted Hardware

by

ASTHA KURKIYA
202011010

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of

MASTER OF TECHNOLOGY
in
INFORMATION AND COMMUNICATION TECHNOLOGY
to

DHIRUBHAI AMBANI INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGY



May, 2022

Declaration

I hereby declare that

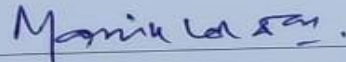
- i) the thesis comprises of my original work towards the degree of Master of Technology in Information and Communication Technology at Dhirubhai Ambani Institute of Information and Communication Technology and has not been submitted elsewhere for a degree,
- ii) due acknowledgment has been made in the text to all the reference material used.



ASTHA KURKIYA

Certificate

This is to certify that the thesis work entitled PRIVATE CONTACT TRACING BASED ON TRAJECTORY-DATA THROUGH TRUSTED HARDWARE has been carried out by ASTHA KURKIYA for the degree of Master of Technology in Information and Communication Technology at *Dhirubhai Ambani Institute of Information and Communication Technology* under my supervision.



Dr. MANIKLAL DAS
Thesis Supervisor

Acknowledgments

Dr. Maniklal Das, my supervisor, whose genuineness and encouragement I will never forget, deserves my heartfelt gratitude. He has been an inspiration to me as I worked through this Master's programme. He is the epitome of leadership and the ideal role model. This thesis would not have been done without his supervision, which allowed me to build a grasp of the subject from the beginning. I am grateful for the tremendous possibilities he provided for me to progress professionally and for the extraordinary experiences he planned for me. It is a privilege to study under Dr. Maniklal Das.

I am thankful for my parents' unwavering love and support, which keeps me inspired and confident. I am also thankful for my friends' constant support throughout the master's programme.

Contents

Abstract	iv
List of Principal Symbols and Acronyms	iv
List of Tables	v
List of Figures	vi
1 Introduction	1
1.1 Related Works	2
1.2 Intel SGX	3
2 Proposed Scheme	5
2.1 Problem Statement	5
2.2 System Overview	8
3 Security Analysis	12
4 Experimental Result	19
4.1 Theoretical Analysis	19
4.2 Performance Analysis	19
4.3 Computational Cost	20
5 Conclusion	22
References	23

Abstract

The COVID19 pandemic has triggered technical measures to halt the disease's spread. One of the most promising solutions for this is private contact tracing (PCT). However, the recently suggested Bluetooth-based PCT has some functionality and flexibility restrictions. Only direct contact can be detected by current methods, not indirect contact. Furthermore, dangerous contact regulations are not adaptable to changing environmental conditions or disease strains. They are also not flexible enough to adapt duration of contact into account while calculating PCT. We present a robust and reliable trajectory-based PCT system based on trusted hardware in this study. The conceptual model of trajectory-based PCT is a generalisation of private set intersection (PSI) which is well studied till now. This difficulty is handled by creating new algorithms that result in a safe, efficient, and flexible PCT system that uses trusted hardware like Intel SGX. Experiments have demonstrated that this system is capable of great performance and scalability.

Private Contact Tracing, Trusted Execution Environment (TEE), Intel SGX, Trajectory Compression.

List of Tables

- 2.1 Notations 7
- 4.1 Computation cost of the existing system 19
- 4.2 Computation cost of the Proposed system 20
- 4.3 Time taken by both the systems to complete whole process 20

List of Figures

- 1.1 Trajectory-based PCT overview 2
- 2.1 System Architecture 7
- 2.2 Encoding flowchart: Encoding trajectory-data into string..... 10
- 3.1 Masquerading attack on Remote Attestation..... 13
- 4.1 Graph between Number of data items (x axis) and time taken to perform old system (y axis) 21
- 4.2 Graph between Number of data items (x axis) and time taken to perform proposed system (y axis) 21

CHAPTER 1

Introduction

Currently, contact tracing is expected as an effective measure to handle the spread of infection. Hence, PCT is necessarily needed. Decentralized Privacy Reservation Proximity Trace (DP3T) [8], a PCT protocol with Bluetooth Low Energy beacon, is already in use in apps made in Europe. The basic mechanism of DP3T is that the application uses the smartphone's Bluetooth to send a random ID. Sensitive information such as the user's ID or location is not contained by this random ID. And the upcoming smartphone will only send data for a limited time.

However, Bluetooth-based private contract tracking has some constrains. First constraint is,that Bluetooth-based PCTs only recognize direct contact. Second, Bluetooth-based PCTs do not have the flexibility to determine risky contact rules. It's supposed that the health agency sends confirmed COVID-19 patients' location data to a server that customers don't trust. Clients submit inquiries and encrypted personal data to the server, which gives a Boolean value as a result indicating whether or not a dangerous contact exists.

The method of PCT allows two (or more) parties to work together to get the intersection of the private set and reveal nothing about the personal data to any party,only result of the intersection. However, existing methods of PSI, primarily based on elementary cryptography, not all of the prerequisites are met. The firstTEE based PSI by given by Tamrakar et al [1] .

The problem is demonstrated to be an extension of the well-studied Private Proximity Testing and Private Set Intersection problems (PSI). The formula is specified for both time and space and can be applied in a variety of situations. Intel SGX was used to build the system. It is demonstrated that a single machine running SGX can process thousands of queries on millions of records of trajectory data in a matter of seconds.

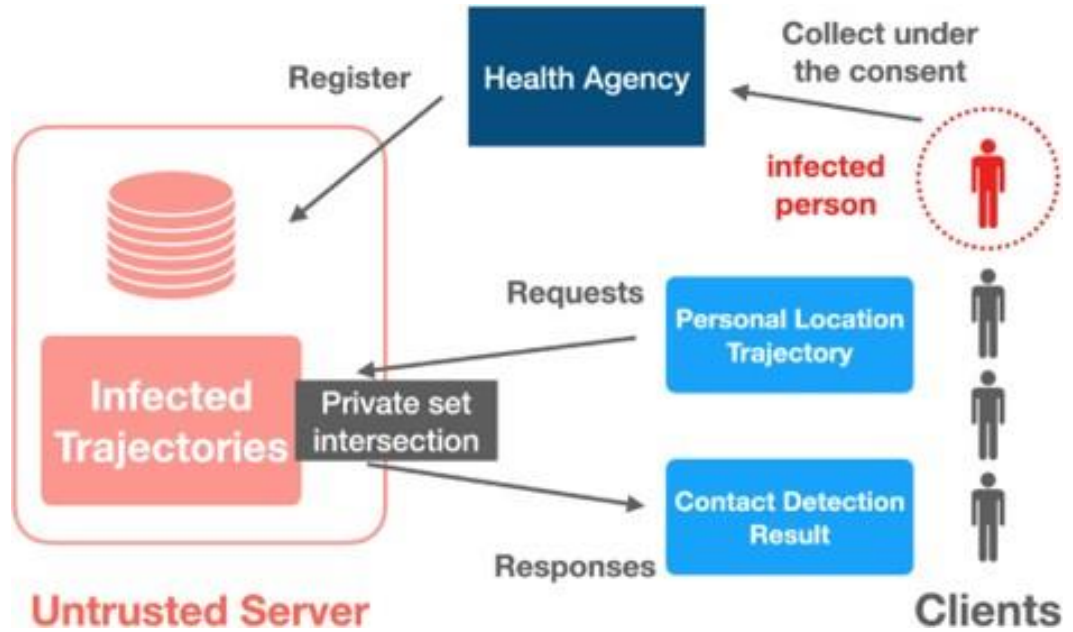


Figure 1.1: Trajectory-based PCT overview

1.1 Related Works

There are BLE-based decentralised architectures that employ a device’s wireless signals, such as DP3T and similar methods. To date, they are the most widely used implementation approaches. The main difference between our suggested system and ours is that here the trajectory data is maintained directly in a discreet way to detect implicit interactions, whereas they only manage contact information through random ID tracking. Hybrid architecture, also called as want architecture, shares many of the same qualities as decentralised architectures. Desire [2] users submit the server random ID-based contact information, but the ID cannot be used to identify a specific person. Unlike our system, AAROGYA SETU [6] collects actual trajectory data, which raises major privacy risks.

In comparison with these works, there are a few differences we found. First, the characteristics are substantially different from a security and privacy standpoint. It is critical in centralised designs to discuss the model of trust. When the server is untrusted, we must consider methods to preserve private data of the. There is no such security problem with the BLE-based technique because personal data is not transmitted. BLE-based methods, on the other hand, pose security issues that aren’t present in other methods. Continuous tracking of random IDs, for example, in a centralised architecture. In the majority of circumstances, to ensure

either a very expensive method such as secret computation or the security of a centralised architecture. It is necessary to make a major assumption, such as server trusting. Secondly, unlike the Bluetooth-based method, which computes the contact detection decision on each device, our method computes the contact detection decision on the server, that can be costly. One of the main advantage of the BLE-

based method is that it has a smaller computing cost than our method, which compares data from a large number of infected persons, because the computation is limited to those who have been directly infected. The BLE-based technique, on the other hand, for instance, cannot detect indirect contact. In terms of communication costs, the BLE-based technique essentially necessitates data broadcasting of the infected person. Despite the tiny size of the data, it is vital to alert all users.

A synopsis of this paper appeared in [5]. When compared to that, there are three major differences. First difference is, that an additional parameter with respect to duration of contact is added in this paper. Second difference is, that a key agreement protocol is used along with remote attestation in order to prevent the system from masquerading attacks [7]. And the third difference is, that a new method – Bing maps tile system is used for encoding the trajectory data more efficiently. In this paper, we use comprehensive experimental methods to measure the accuracy, that includes a comparative study to previous work on trajectory data PCT.

Douglas-Peucker, or route-wise compression, is a well-studied technique for trajectory data compression [3]. The main concept is to recreate a route by estimating it from a small number of sites. Instead of approximating the position information, the compression is done by approximating the route data. The commonality of trajectory points is the basis for our compression. As a result, it is incompatible with these compression techniques. However, in contact tracing, routewise compression may not function effectively because it is impossible to know whether or not points are in contact just by crossing. It is vital to provide some type of time data in the route information to identify touch.

1.2 Intel SGX

Intel SGX [4] is an advanced instruction set of Intel-x86 processors that allows you to create a separate and reliable execution environment called an enclave. The enclave is located in a protected memory part called the Enclave Page Cache (EPC). In this area, a memory encryption engine with a private key used only by the processor, allows all programs and data to be processed quickly and trans-

parently outside the CPU package without encryption. In this trusted area, the CPU prevents access from untrusted applications, including operating systems / hypervisors, thus protecting the integrity and confidentiality of the data in the system and enclave. This hidden process is software transparent and much faster than traditional advanced encryption-based methods.

Local and remote attestations are allowed by SGX. The focus of this section is on remote authentication (RA). You can acquire a report comprising measurements (such as MRENCLAVE or MRSIGNER) which is based on the hashed original state of the enclave by requesting a RA from an enclave. This allows you to find critical information about your application, as well as the memory design layout and the contact information of the builder. This measurement is signed by the Intel Enhanced Privacy ID [4], and Intel Attestation Services can validate the signature's accuracy as a trusted third party/client. Secure key transactions, like ECDSA, are done between the remote client and the enclave, who wants to attest to the server, within this RA protocol, along with the verifying of the SGX environment's Remote attestation. In existing systems, the procedure for remote attestation is vulnerable to masquerading attack. A new RA algorithm with key agreement protocol is proposed in this paper which is not vulnerable to masquerading attack.

The following section 1.1 describes some preliminary studies related to the trajectory based private contact tracing system. Chapter 2 represents the proposed algorithm. Chapter 3 includes the security analysis of the system. Performance analysis is included in the chapter 4. Finally, the last chapter 5 represents a conclusion to the whole report.

CHAPTER 2

Proposed Scheme

A centralised system is examined in this research, which keeps infected patients' trajectory data on a remote server and collects PCT queries from clients along with their own trajectory-data in form the of queues. While the system is operational, the server maintains past data records for infected people for the previous 14 days (or up to 21 days which is decided by the government). To add and remove data, update in batches on a regular schedule (once a day). The server has already pre-structured the trajectory data and is prepared to allow the PCT requests from the client. As a PCT request, the client additionally transmits his trajectory data of the last 14 days, and the server does contact tracing and produces the result and return it to the client.

2.1 Problem Statement

PCT depending on trajectory. Between the client and the server, it is an asymmetrical protocol. This approach gives, 1 or 0, to the client as a result and does not provide the client's data to the system's server even if the person desires to know the server's contact information. Every client has a collection of trajectory data for one person, while the server has numerous trajectory data for infected people in the instance of an infection.

A PCT technique based on Trajectory-data can be described as an expansion of such a concept in its most basic form. For contact tracing, contacts can be calculated using navigation data from user time series. Individual location information can be extended to time series trajectory data to accomplish this. we can represent the formula as: denoting trajectory data of user k as

$$D_n = (\sigma^k = (t_1^k, \iota_1^k) \dots \dots \dots, \sigma^k = (t_n^k, \iota_n^k)) \quad (2.1)$$

we can get the answer returned as k's contact with v,

$$\begin{aligned}
& \{1 \quad (\exists \square_{\ddot{}}^{\square} \in D_{\square}, \square^{\square} \in D_{\square}) \\
& \quad \text{such that} \quad (\square^{\square} - \square^{\square}) \leq \Theta_{geo} \\
& \quad \text{and} \quad (t^{\square} - t^{\square}) \leq \Theta_{doc} \\
& \quad 0 \quad (\text{others})\}
\end{aligned}$$

Where Θ_{geo} is spatial and Θ_{doc} is temporal duration of contact range threshold. However, the server doesn't at all learn anything else about X_{\square} , and with this protocol, u can only acquire a 1 or a 0 about X_{\square} . For our experiment, we considered Θ_{geo} and Θ_{doc} equal to 0.

Some important needs of trajectory-based PCT are described below: Efficiency. In many ways, the system necessitates efficiency. First, there will be response throughput. Since the server is constantly bombarded with requests from a huge number of clients, this is the case. The second consideration is bandwidth. Because this protocol affects a large number of people, bandwidth must be decreased to improve effectiveness of communication. To put it another way, protocols that transfer a lot of data to devices at the end, should be avoided. The final point to consider is scalability. Efficiency requirements completely depend on the factors that are: the amount of the PCT used by the total number of users, frequency of use of the system and amount of data entered in the system.

Security. Consider a malicious server attempting to steal data without adhering to protocol. Because an attacker can observe the processing of raw data, it's critical to safeguard user privacy, therefore PCT computation on the server and network provisioning must be cryptographically distinct. You can assume that a trustworthy and curious client is following the rules but attempting to gain personal information from the system's server. You can also employ brute force attacks to reveal some part of the trajectory data on the server side if the trajectory data in teh server side is not encoded. By outputting the value returned to the user as a binary value, each user is authenticated and the number of requests can be limited, reducing the amount of information leaked and increasing the cost of the attacker.

Flexibility. Flexibility is the requirement needed that is satisfied by parameters $(\Theta_{geo}, \Theta_{doc})$ in the system.

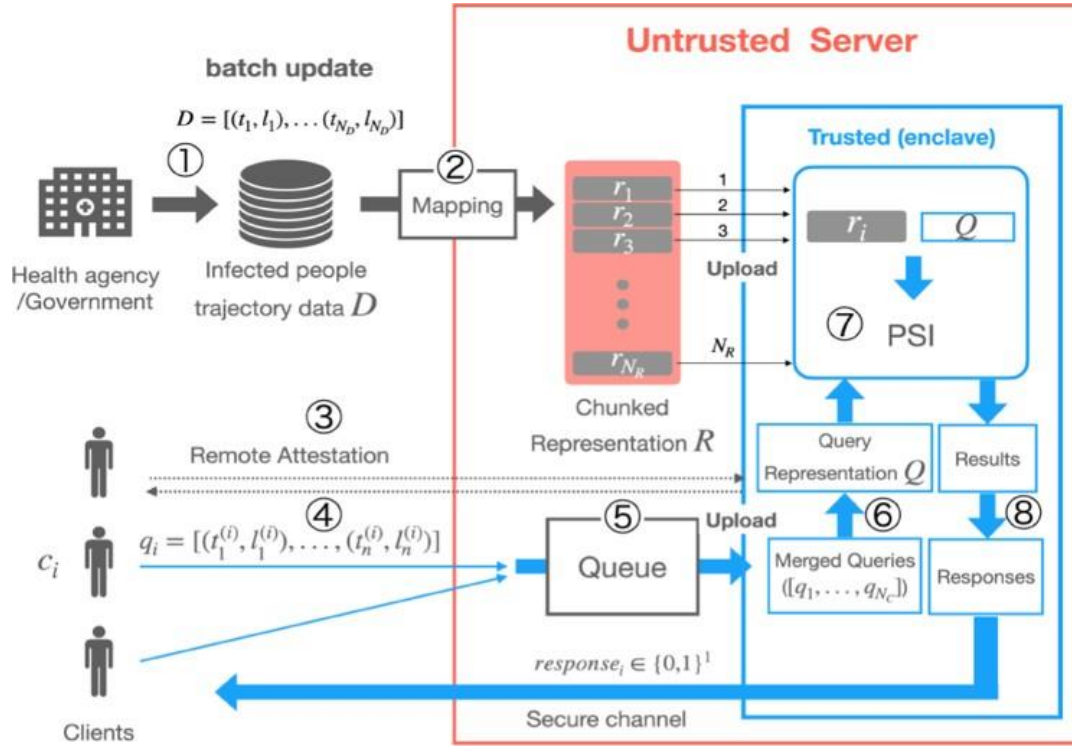


Figure 2.1: System Architecture

Table 2.1: Notations

Symbols	Definition
N_D	number of sick patient's trajectory-data
D	sick patient's trajectory-data
N_C	number of clients requesting PSI
$c_i \in \mathcal{C}$	a single client i ($i = (1, \dots, N_C)$) and set of all the clients
N_R	Total number of chunks of data
R	D after mapping (into arrays of similar chunks)
R_i	R_i ($i = 1, \dots, N_R$)
q_i	query data sent by client i
Q	N_C query data after merging
N_Q	size of Q
Θ	Parameters of PCT : $(\Theta_{geo}$ and Θ_{doc})
(t_i, l_i)	i^{th} row of trajectory-data as time and location tuple

2.2 System Overview

In figure 2.1: In step 1, in batch processing, the health agency uploads the compromised patient's data D . As there the need to differentiate trajectory data by infected users is not required, D is in the raw trajectory data form and does not need to include the users' IDs.

In step 2, with the function `mapToChunkedDictionary`, we map the raw data form D to the dictionary representation R as mapped array of chunks. Encoding process, chunking process, and translation process into the dictionary representation are all part of this mapping function. Each piece of trajectory data is encoded into 1-D bytes of strings. This step is carried out in SGX, and the R 's binary data is stored in the untrusted server's memory. While uploading, the health agency obtains the key used for encryption from SGX via RA.

In step 3, Remote attestation is happening. Before transmitting the RA request to the server, the client uses the new RA protocol to verify the remote enclave. The client can inspect to see whether the enclave is tampered before securely exchanging the key with it or not. The data is transmitted to the enclave via a secure channel after being encrypted using the shared key. It is to be noted that a secure channel is one in which just the trajectory query data in the request is kept private and encrypted. Client's personal data (like IP address) is not kept hidden on the server side and must be used in the response. The answer data, along with the request query data, are kept private and encrypted.

In step 4, PCT requests are sent to the server by a large number of clients. c_i sends q_i in form of a request parameter in the figure figure 2.1, which comprises her trajectory data of 14 days. Before encryption, Θ encodes trajectory data. As a result, in advance, the client and the server can share this parameter. After exiting the client environment, q_i is encrypted in all untrusted parts and can only be seen and accessed in the verified enclave.

In step 5, q_i is queued on the outside of the enclave's protected region until a specified amount (N_c) of requests have been accumulated, at which point it is uploaded to the enclave joined by the `loadToEnclave` function. This function's implementation is done by the `ECALL` function, which is used to call an SGX function. By batch processing, we want to improve the process of data loading for numerous users while reducing the overheads.

In step 6, The data is finally decrypted after being uploaded to the trustworthy enclave. All the q_i queues are collected inside the enclave and using `mapToArray`, mapped to query representation Q . The amount of the enclave memory is tightly

limited, despite the fact that these query data are private and cannot be handled outside the enclave. As a result, it's crucial to encode trajectory data into minimal bytes.

In step 7, the chunked data r_i is inserted one after another into the enclave, and in the enclave, the set-intersection of r_i and Q is computed. Only searches that provide the non empty result for a set intersection are returned as 1. If a query is deemed as 1, we can save time by breaking the loop for the PSI calculation for rest of the r_i .

In step 8, ConstructResponses method compute responses for all (N_C) clients from the results of the PSI and total query data $q_i (i = 1, \dots, N_C)$ inside the trustworthy enclave after all the iterations for all chunks are done. Simply encryption of the results of PSI as positive or negative for each client inside the enclave can do this. After this, the operation sends each client the encrypted result via the secure channel.

The encoding module converts the trajectory data into a string representation that is then used to produce a compact dictionary representation with the help of a finite state automaton (FSA). FSAs are circular, deterministic finite acceptors that can share prefixes and suffixes between nodes (see later sections). As a result, comparable strings can be efficiently stored. Trajectory-based coding. The first requirement for encoding is that the discrete - time independent trajectory data and the unique string must have an injective function. Due to FSA [4], there are numerous similarities between string suffixes and prefixes, which is a desirable quality.

Trajectory data D contains arrays of time-location such as date-time format and geographic data such as longitude and latitude tuples:

$$D_n = ((t_1, l_1), \dots, (t_n, l_n)) \quad (2.2)$$

Where, $t_i \in$ time in date-time format and $l_i \in$ co-ordinate (e.g. (latitude, longitude))

Encoding

Here, t and l are encoded distinctly and merged in series. For location, a method of mapping 2D coordinate system data to a string, Bing Maps Tile System [?] is used. In Bing Maps Tile System, the strings that are obtained after encoding of the trajectory data are going to be similar in suffix. For t , UNIX epoch encoding scheme is used. This coding will generate similar strings for time. And a simple

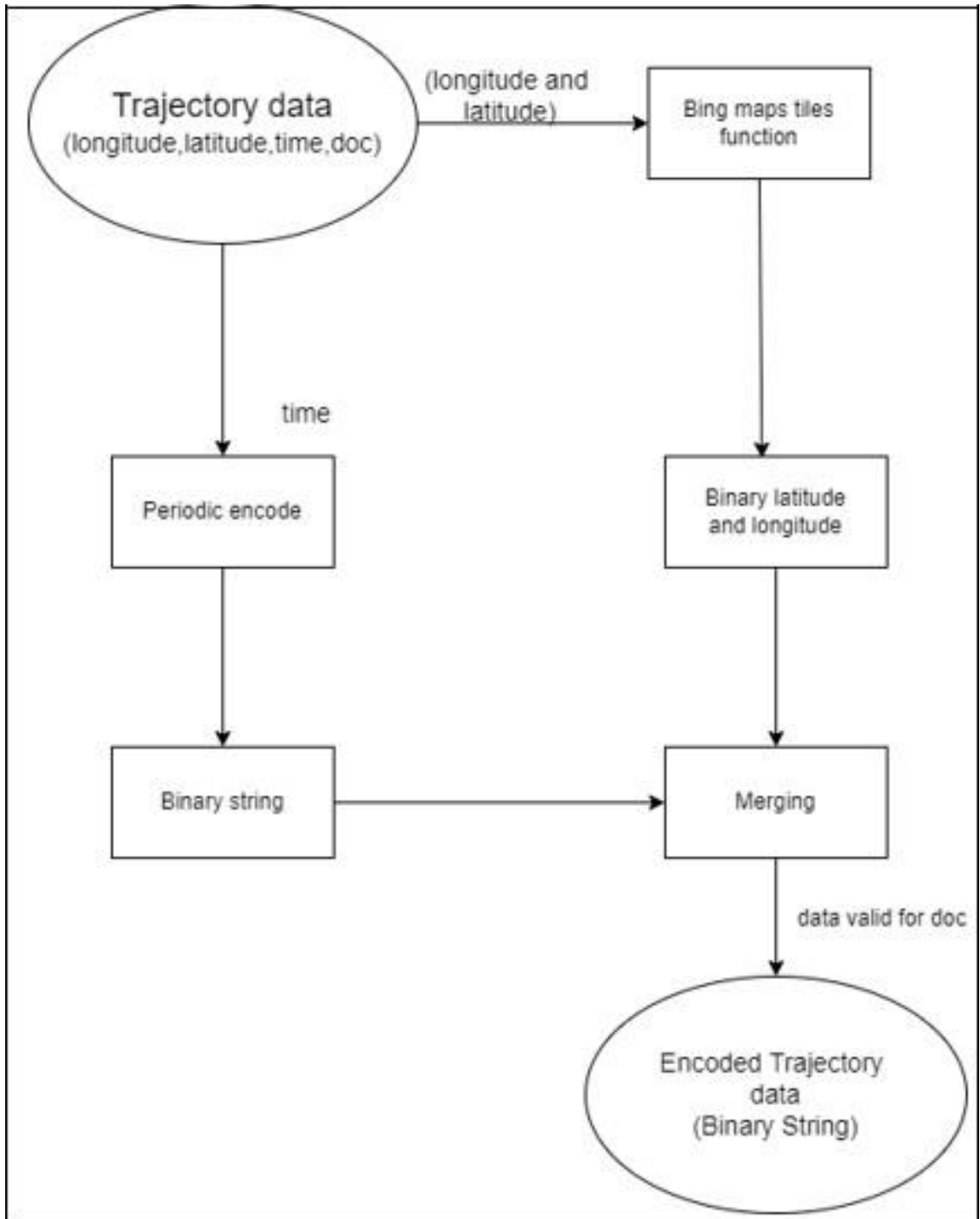


Figure 2.2: Encoding flowchart: Encoding trajectory-data into string.

merge will show you the prefix and suffix-like structures that the FSA can compress more efficiently. Also, the encoded string will be different for each trajectory data.

Remote attestation with key agreement protocol

In this, an improved method for remote attestation is used, which includes a key agreement protocol. This is done in order to improve the performance of RA and make it more secure. How it secures the system is explained in Security Analysis Section.

CHAPTER 3

Security Analysis

Since we have considered that the enclave is trusted and secure, we can only analyse attacks that can happen outside the enclave. Here are some attacks that may be possible:

Denial-of-Service attacks

To avoid DoS attacks on SGX, the client must be authenticated before it can be processed. Because the server-side data changes just once a day, each client should be limited to one request per day. Because the information of the client requesting the query does not need to be disguised, this may be accomplished with a simple plugin.

Query-abusing attacks

To reduce the impact of attacks, verifying the client can also be used. Only 0 or 1 information is available to the client. Even if the request is only made once a day, there is a chance that some information will be released due to the use of several infected clients. Other protection techniques are required to prevent this; like, assuming the ability to link the discovered information with other data, differential privacy must be taken into account.

Side-channel attacks

Data cannot be leaked from the network since it is encrypted with a key shared between SGX and the client.

False answer attack

False answers are prevented thanks to RA's verification of the enclave's internal blogs.

Fake data injection

Suppose that the health agency is a trusted institution, and that when a third party deposits data, the data in the memory protected by SGX must be modified. As a result, it is cryptographically secure.

Replay attack

Even if we consider that a secure channel is in place, it is possible that a replay attack may occur. However, it can be totally avoided by authenticating the user, limiting the number of requests each day, and by changing the key every other day.

Masquerading attack

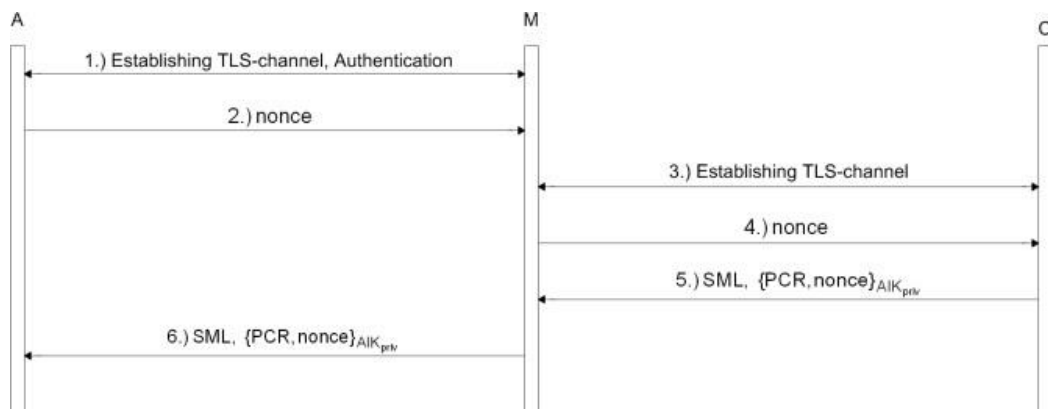


Figure 3.1: Masquerading attack on Remote Attestation.

In this attack strategy, the attacker targets the platform settings of the client which is honest to vouch to the client which is malicious and operating on M, circumventing M's remote attestation. It is an attempt to get illegal access to personal computer information by impersonating identity of someone else, such as

a network identity. A masquerading attack can make an authorisation process exceedingly vulnerable if it is not adequately safeguarded.

Masquerading Attack in Remote attestation

Algorithm 1 Masquerading Attack in Remote attestation

- 1: A : create a non-predictable 160 bit nonce
 - 2: $A \rightarrow M$: send challenge Request(nonce)
 - 3: $M \rightarrow C$: send challenge Request(nonce)
 - 4: C: loadkey($A^i K_{pr^i}$)
 - 5: C : Retrieve the Quote : sig(PCR, nonce) $A^i K_{pr^i}$
 - 6: C : get stored measurement log : SML
 - 7: $C \rightarrow M$: challengeResponse(Quote, SML) and certificate $A^i K_{p^i b}$
 - 8: $M \rightarrow A$: challengeResponse(Quote, SML) and certificate $A^i K_{p^i b}$
 - 9: A : validate the certificate($A^i K_{p^i b}$)
 - 10: A : validate the sig(PCR, SHA1(nonce)) $A^i K_{pr^i}$
 - 11: A : using PCR, validate nonce and SML
-

Step 1

A non-predictable nonce is created by A.

Step 2

The nonce is sent to the M by A.

Step 3

M forwards this nonce to C.

Step 4

The storage root key is used, and the attester, that is C, uploads the Attestation Identity Key($A^i K_{pr^i}$) from the TPM's protected storage (SRK).

Step 5 and 6

the attester C issues a TPM Quote instruction, all the selected PCRs and the given nonce n are signed by the private key $A^i K_{pr^i}$. In addition to this, the attester C retrieves the measurement log that has been saved (SML).

Step 7

M receives the attester's response, which includes the signed nonce, signed Quote, and SML. C also hands over the AIK credential, which is an $A^0K_{p^0b}$ signed by a CA.

Step 8

M send this received information to C.

Step 9

Whether the AIK credentials that were signed by a reputable CA and hence belongs to a legitimate TPM, is determined by A. Along with it, A also checks the trusted issuing party's certificate revocation list to see if $A^0K_{p^0b}$ is still valid.

Step 10 and 11

A validates the Quote's signature and then A checks for freshness.

Since, AIKs can't be utilised to create secure channels or verify communication partners directly, therefore, the challenger has no way of knowing whether the received message is from the attesting system M or not. He can just know that he received a message form a legitimate TPM. Because the attesting system M uses all the integrity values with a valid AIK signature from platform C to pretend that it has the matching $A^0K_{p^0a}$, masquerading attacks cannot be stopped.

Remote Attestation with key agreement protocol

Both the parties(attester and attestee) must agree on a common generator- g and a common group- m for the protocol to work.

Step 1

A non-predictable nonce is created by A.

Step 2

$K_{p^0b}^A = g^a \text{ mod } m$ is generated by A, where a is the private component of K^A .

Algorithm 2 Remote Attestation with key agreement protocol

- 1: A : create a non-predictable 160 bit nonce
 - 2: A : Generate the Keys(K_{pub}^A, K_{priv}^A)
 - 3: $A \rightarrow C$: send Challenge Request(nonce, K_{pub}^A)
 - 4: C: Generate the Keys(K_{pub}^C, K_{priv}^C)
 - 5: C: loadkey($A:K_{priv}$)
 - 6: C: Retrieve the Quote = sig(PCR, SHA1 (nonce, K_{pub}^C)) $A:K_{priv}$
 - 7: C : get stored measurement log (SML)
 - 8: C : compute the Session Key (K^{AC})
 - 9: A: validate the certificate($A:K_{pub}$)
 - 10: A : validate the sign(PCR, SHA1(nonce, K_{pub}^C)) $A:K_{priv}$
 - 11: A : using PCR, validate nonce and SML
 - 12: A : compute the session Key (K^{AC})
 - 13: A : create a non-predictable 160 bit *nonce*₁
 - 14: $A \rightarrow C$: send the challenge Request (*nonce*₁)
 - 15: C : compute Response = enc(*nonce*₁, K^{AC})
 - 16: A : validate the *nonce*₁
-

Step 4

The primary part is that, the attester A generates a public key K_{pub}^C , before signing the Quote with the AIK, it puts the produced key in the Quote message along with the PCRs and the nonce.

Step 5-11

The client's CPU A, is used in order to generate the public key K_{pub}^C , which is then fed as an external data into the procedure of TPM Quote. Since the TPM Quote command can only allows for some bits (160 bits) of data, the message must be compressed using SHA1 to get under the 160 bit limit. As C is running a trustworthy OS with a trusted platform configuration the key's, private part is unavailable to a possible malicious client.

Step 12

The challenging party uses his private part and the public part of C to generate the shared session key K^{AC} .

Step 13-16

Following the computation of the session key, A performs a second challenge-response authentication to see if C also possesses this shared key.

We will prove that the proposed system is safe against masquerading attacks by contradiction.

Proof by Contradiction

Let $E \square p_{C,M}^{MA}$ be an experiment where the challenger A, wanted to authenticate the client M, before any secret message is sent in a collaborative masquerading attack. All messages are transferred from A to C by the malicious system M. Then, the probability of a successful attack

$$Pr[E \square p_{C,M}^{MA}] = 1 \quad (3.1)$$

1. Let i and j be two plaintexts ($1 \leq i, j \leq N$)
2. The ciphertext resulting from the experiment upto polynomial times. The outputs are the ciphertexts from the algorithm 2.
3. For this to happen, after step 1, algorithm 2, these step must happen.

$A \rightarrow M : \text{ChallengeRequest}(\text{nonce}, K_{p \square b}^A)$

$M \rightarrow C : \text{ChallengeRequest}(\text{nonce}, K_{p \square b}^M)$

But as we know that,

1. M can not generate his own session key between him and A, since his software is corrupted and malicious platform details are transmitted to A by his TPM.
2. The platform owner cannot transmit the session key K^{AC} , which is generated by A, to the malicious host since extraction (through memory dump or system software modification) would result in a compromised system state that would be identified during the attestation phase.
3. As a result, neither M nor C have the right K^{AC} and so are unable to encrypt the given nonce_1 , the second challenge response authentication (Step 5d to Step 10) discovers this substitution.

Hence,

$$\Pr[E^{\square} p_{C,M}^{MA}] = 0 \quad (3.2)$$

Comparing equation (3,1) and equation (3,2). This contradict our assumption and proves that our system is safe against masquerading attack.

CHAPTER 4

Experimental Result

4.1 Theoretical Analysis

For calculating the complexity of the whole system, all the steps are evaluated step by step and then the final complexity of the system is calculated as c. Table below shows the complexity analysis of the proposed system.

Table 4.1 summarizes the costs to perform computation in existing system. And Table 4.2 summarizes the costs to perform computation in proposed system. To represent the complexity measurements we use notations table Table 2.1.

We can see from tables : Table 4.1 and Table 4.2, that our proposed system is more reliable than the existing system in terms of computational cost.

4.2 Performance Analysis

To conduct the experiments, we use Python 3.8.10 and pyQuadkey2 library for quadkey generation, encryption and decryption methods. Hardware platform has OS as Windows 11 64-bit with CPU Intel core i5,10th generation, and 8 GB memory, which supports the SGX instruction set. For experimental purpose, we

Table 4.1: Computation cost of the existing system

Step description	Complexity
1, N_D trajectory data will be uploaded	$O(N_D)$
2, N_R chunks are encoded and mapped into the chunked dictionary.	$O(N_D^2)$
3, N_R each client will perform RA	$O(N_C)$
4, let n be the size of each request.	$O(n)$
5, clients' requests are uploaded in the Q. N_C is client requests at a time	$O(N_C * n)$
6, these queues are merged into a single unique queue.	$O(N_C * n)$
7, set intersection is performed for all N_R chunks and $N_C * n$ queue data	$O(N_C * N_R * n)$
8, The intersection is stored in a queue of size N_C And client get the result	$O(N_C)$
Total	$O(N_C * N_D^2 * n)$

Table 4.2: Computation cost of the Proposed system

Step description	Complexity
1, N_D trajectory data will be uploaded	$O(N_D)$
2, N_R chunks are encoded and mapped into the chunked dictionary.	$O(N_D)$
3, N_R each client will perform RA	$O(N_C)$
4, let n be the size of each request.	$O(n)$
5, clients' requests are uploaded in the Q. N_C is client requests at a time	$O(N_C * n)$
6, these queues are merged into a single unique queue.	$O(N_C * n)$
7, set intersection is performed for all N_R chunks and $N_C * n$ queue data	$O(N_C * N_R * n)$
8, The intersection is stored in a queue of size N_C And client get the result	$O(N_C)$
Total	$O(N_C * N_D * n)$

Number of Data	Time taken by old system	Time taken by our system
10	1.5878918170928955	1.0629105567932130
20	1.6581721305847168	1.0428571701049805
50	1.5846397876739502	1.0560708045959473
70	1.5680992603302002	1.0538170337677002
100	1.7373738288879395	1.0621485710144043
120	1.7136499881744385	1.0761454105377197
150	1.7136499881744385	1.0870766639709473
200	1.6356916427612305	1.1187500953674316
250	1.8569145202636719	1.0985748767852783
300	1.6048930644989014	1.1755142211914062

Table 4.3: Time taken by both the systems to complete whole process

have created a set of random data. Also, N_D is updated in fixed time regularly in form of batches. Let us assume that 10000000 trajectory data is uploaded in one batch. Hence, N_D will be 26 bits. Also, let number of clients N_C be 100. Then N_C will be of 9 bits. And for each client, trajectory data sent be 1000 for 1 day. Then n will be of 12 bits. As a result, the total computation will take 2808 bits (if calculated for worst case.) Also, Θ_{doc} is considered 2 hours or 7200 seconds for this experiment. It can be changed and decided by the government according to disease or virus.

4.3 Computational Cost

Figure 4.1 shows the time taken to perform the existing system. And Figure 4.2 shows the time taken to perform the proposed system.

The experiment includes all phases running once. From Table 4.1 and Table 4.2, it can be derived that the performance of the proposed system is efficient in terms of computation.

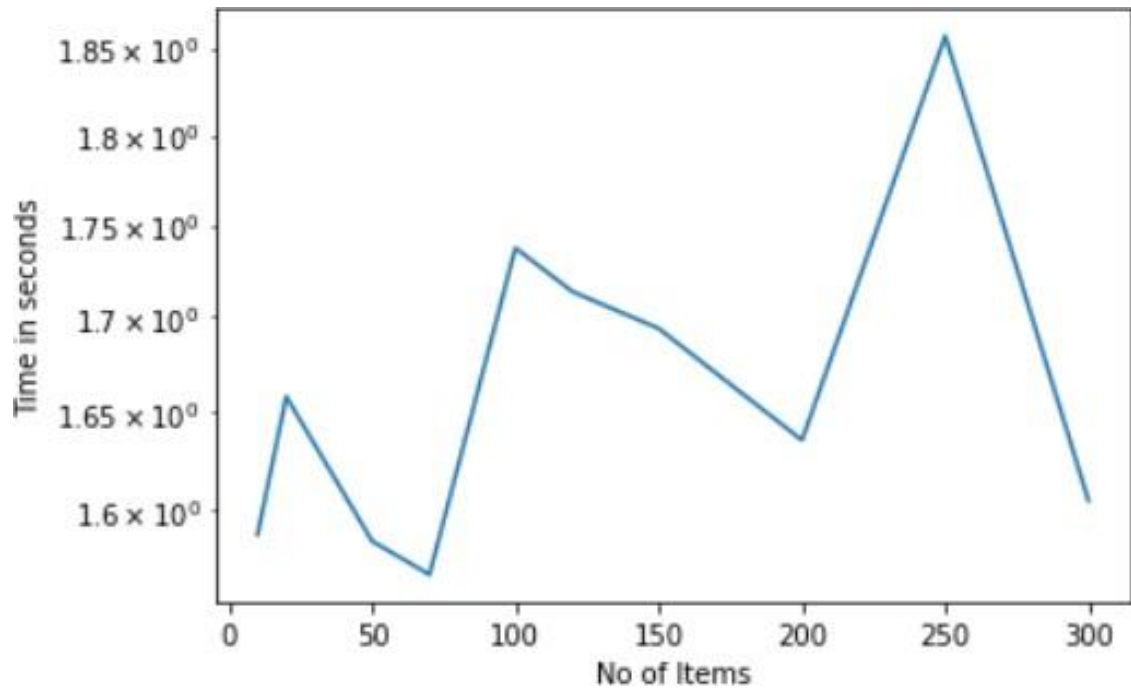


Figure 4.1: Graph between Number of data items (x axis) and time taken to perform old system (y axis)

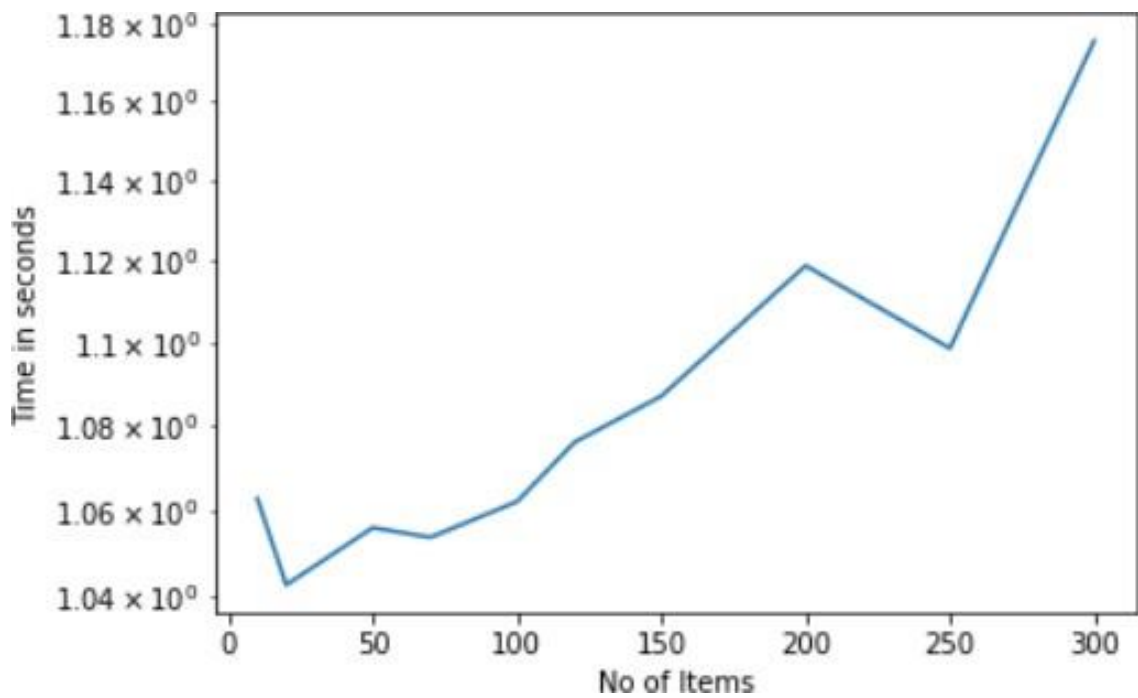


Figure 4.2: Graph between Number of data items (x axis) and time taken to perform proposed system (y axis)

CHAPTER 5

Conclusion

To restrict the spread of contagious diseases, this report provides a private contact tracing system based on trajectory data of patients that employs trusted technology/hardware. The criteria for the same has been specified, and a architecture based on TEE for safe, efficient, and flexible contact tracing has been proposed. The results of the experiments indicate that this technology can work on a broad scale. We can make the system more flexible according to our requirements in future as flexibility requirement is satisfied in our system. We can also try to improve the proposed system since it can also detects the contact even if the patient is in a protected space, like a car, near the infected patient.

References

- [1] The circle game: Scalable private membership test using trusted hardware <https://dl.acm.org/doi/abs/10.1145/3052973.3053006>, author=Tamrakar, Sandeep and Liu, Jian and Paverd, Andrew and Ekberg, Jan-Erik and Pinkas, Benny and Asokan, N, booktitle=Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, pages=31–44, year=2017.
- [2] C. Castelluccia, N. Bielova, A. Boutet, M. Cunche, C. Lauradoux, D. L. Mé-tayer, and V. Roca. Desire: A third way for a european exposure notification system leveraging the best of centralized and decentralized systems <https://arxiv.org/abs/2008.01621>. *arXiv preprint arXiv:2008.01621*, 2020.
- [3] C. Chen, Y. Ding, X. Xie, S. Zhang, Z. Wang, and L. Feng. Trajcompressor: An online map-matching-based trajectory compression framework leveraging vehicle heading direction and change <https://ieeexplore.ieee.org/abstract/document/8697124/>. *IEEE Transactions on Intelligent Transportation Systems*, 21(5):2012–2028, 2019.
- [4] V. Costan and S. Devadas. Intel sgx explained <https://www.intel.com/content/dam/develop/external/us/en/documents/intel-sgx-support-for-third-party-attestation-801017.pdf>. *Cryptography ePrint Archive*, 2016.
- [5] F. Kato, Y. Cao, and M. Yoshikawa. Secure and efficient trajectory-based contact tracing using trusted hardware <https://arxiv.org/pdf/2010.13381.pdf>. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 4016–4025. IEEE, 2020.
- [6] D. R. Sehgal. Unpacking the privacy concerns of aarogya setu app-<https://blog.ipleaders.in/unpackingprivacy-concerns-aarogya-setu-app/>. 2020.

- [7] F. Stumpf, O. Tafreschi, P. Röder, C. Eckert, et al. A robust integrity reporting protocol for remote attestation https://www.researchgate.net/profile/Claudia-Eckert-4/publication/228958139_A_robust_integrity_reporting_protocol_for_remote_attestation/links/5452ccc30cf2bccc49094c47/A-robust-integrity-reporting-protocol-for-remote-attestation.pdf. In *Second workshop on advances in trusted computing (WATC'06 Fall)*, pages 25–36, 2006.
- [8] C. Troncoso, M. Payer, J.-P. Hubaux, M. Salathé, J. Larus, E. Bugnion, W. Lueks, T. Stadler, A. Pyrgelis, D. Antonioli, et al. Decentralized privacy-preserving proximity tracing <https://arxiv.org/abs/2005.12273>. *arXiv preprint arXiv:2005.12273*, 2020.