UiT The Arctic University of Norway

Faculty of Science and Technology
Department of Computer Science

Privacy-preserving smart nudging system: resistant to traffic analysis and data breach

—

G M A Mehedi Hussain

INF-3990 Master's Thesis in Computer Science - October 2020

This thesis document was typeset using the *UiT Thesis LaTeX Template*.
© 2020 – `http://github.com/egraff/uit-thesis`

For my friends and family, thank you for the support.

“Fully secure systems don’t exist today and they won’t exist in the future.”
–Adi Shamir

Abstract

A solution like Green Transportation Choices with IoT and Smart Nudging (SN) is aiming to resolve urban challenges (e.g., increased traffic, congestion, air pollution, and noise pollution) by influencing people towards environment-friendly decisions in their daily life. The essential aspect of this system is to construct personalized suggestion and positive reinforcement for people to achieve environmentally preferable outcomes. However, the process of tailoring a nudge for a specific person requires a significant amount of personal data (e.g., user's location data, health data, activity and more) analysis.

People are willingly giving up their private data for the greater good of society and making SN system a target for adversaries to get people's data and misuse them. Yet, preserving user privacy is subtly discussed and often overlooked in the SN system. Meanwhile, the European union's General data protection regulation (GDPR) tightens European Unions's (EU) already stricter privacy policy. Thus, preserving user privacy is inevitable for a system like SN.

Privacy-preserving smart nudging (PPSN) is a new middleware that gives privacy guarantee for both the users and the SN system and additionally offers GDPR compliance. In the PPSN system, users have the full autonomy of their data, and users data is well protected and inaccessible without the participation of the data owner. In addition to that, PPSN system gives protection against adversaries that control all the server but one, observe network traffics and control malicious users. PPSN system's primary insight is to encrypt as much as observable variables if not all and hide the remainder by adding noise. A prototype implementation of the PPSN system achieves a throughput of 105 messages per second with 24 seconds end-to-end latency for 125k users on a quadcore machine and scales linearly with the number of users.

Acknowledgements

I want to take this opportunity to thank my Professor Anders Andersen and Professor Randi Karlsen of the ODS research group in the Faculty of Science and Technology at UiT for letting me extend their brilliant idea of Green Transportation Choices with IoT and Smart Nudging (SN). I am obliged to them for making me be a part of enhancing the privacy and security of such an exciting and significant project. I am also thankful for their advice and feedback to prepare the manuscript of this thesis and all the valuable discussions about the Privacy-preserving smart nudging system (PPSN)—one of the contributions of this thesis.

I especially would also like to thank my best friend, Jan Helge Brask Pedersen, for all the encouragement and support.

Finally, I would like to thank my family and friends for believing in me and supporting me throughout this whole journey.

Contents

Abstract iii

Acknowledgements v

List of Figures xi

List of Tables xiii

List of Abbreviations xv

Glossary xvii

1 Introduction 1

1.1 Motivation 2

1.2 Privacy in smart nudging 3

1.3 Research problem 3

1.4 Contribution 4

1.5 Thesis outline 5

2 Theoretical background 7

2.1 Smart nudging 7

2.1.1 Data collection to provide nudge 8

2.1.2 Green Transportation Choices with IoT and Smart Nudging (SN) [9] 8

2.1.3 SN architecture 12

2.2 Defining privacy and security 15

2.2.1 Importance of privacy and security 15

2.2.2 General Data Protection Regulation (GDPR) [6, 26] . 16

2.3 Cryptography and methods to achieve privacy 19

2.3.1 Cryptography 19

2.3.2 Encryption 22

2.3.3 The RSA algorithm [36] 23

2.3.4 Advanced Encryption Standard (AES) [40] 27

2.3.5 Digital envelope 28

2.3.6 Differential privacy [41] 30
2.3.7 Mix-net 32
2.4 Implementation technologies 37
2.4.1 Programming languages and frameworks 37
2.4.2 Protocol buffer 38
2.5 Related work 42

3 Privacy in smart nudging 43
3.1 Privacy issues in personalization 43
3.2 Situational awareness and its privacy concerns 44
3.3 Privacy concerns in SN architecture 44
3.4 The dark side of nudging 46

4 Design 47
4.1 Design goals 48
4.2 Adversary model 49
4.3 Proposed system architecture 50
4.3.1 Protocol schematics 51
4.3.2 SN backend endpoints 53
4.3.3 Key-exchange 54
4.3.4 One round of communication 55
4.3.5 Content hiding 56
4.3.6 Untraceable end-to-end communication 57
4.3.7 Hiding meta-data 57
4.3.8 PPSN system as middleware 58
4.3.9 Data storage and processing techniques 58

5 Implementation 61

6 Evaluation 65
6.1 Method of evaluation 65
6.1.1 System latency 65
6.1.2 System load average 67
6.1.3 Noise performance 69
6.2 Experimental setup 70
6.3 Privacy evaluation of the PPSN system 71
6.4 Performance of the PPSN system 74
6.4.1 End-to-end latency of the PPSN system 75
6.4.2 System load of the PPSN system 77

7 Discussion 81

8 Future work 85

9 Conclusion 87

Bibliography 89

List of Figures

2.1 Visualizing environment friendliness (EF) vs. different vehicle types[9]. 9

2.2 Smart Nudging publish-subscribe architecture with the modules Sense, Analyse, and Inform and Nudge[9]. 14

2.3 GDPR defines personal data is any information that is related to an identified or identifiable natural person. 16

2.4 Encryption and Decryption process in an RSA algorithm. . . 24

2.5 Overview of the RSA algorithm shows that the RSA algorithm is an algorithm which takes numbers and returns numbers (i.e, All data is treated as some kind of number representation of that specific data in order to work with the RSA algorithm) [35]. 25

2.6 Secure data transmission through insecure channel by using digital envelope. 29

2.7 Probability distribution of message-box access after adding noise shows us that adversary is having a hard time predicting a situation (equation 2.2) (This plot uses Laplace's double exponential probability distribution (equation 2.1). 30

2.8 Probability distribution of message-box access before adding noise—makes it easy for an adversary to tell which situation a user/variable is currently in. 32

2.9 An overview of a mix network where the links between senders and addressees are broken. 33

2.10 Simple decryption process in mix-net. Clients encrypt the messages with the sequence of public-keys. Each node in a mix-net removes a layer of encryption with its own private-key and shuffles the message order and passes the results to the next destination. 34

2.11 Protobuf serialization and deserialization flow diagram. . . . 38

2.12 Sample serialized Protobuf file. 39

2.13 Encoded data sizes of different data serialization techniques. 40

4.1 High-level overview of the Privacy-preserving Smart Nudging system (PPSN). 50

4.2 A High-level overview of SN backend's virtual endpoints to obfuscate and maintain constant message size of requests and response from SN Backend. 54

6.1 An adversary can only see the incoming and outgoing encrypted messages if the first server (1^{st} server in the PPSN server array figure 4.1, from top to bottom) in the PPSN server array gets compromised. If at least one server in the PPSN server array is honest, an adversary should not be able to tell about the content of the messages and original destination of the messages. 71

6.2 Identifiability of a real user in the PPSN system for a different amount of constant noise when varying the number of active users in the system. 72

6.3 An adversary can only see the message-box access counts if the last (3^{rd} server from figure 4.1, from top to bottom) server in the PPSN server array gets compromised. If at least one server in the PPSN server array is honest, an adversary should not be able to tell about the order of the messages, a correlation between accessing the message-box, the origin of the messages and the content of the messages. 73

6.4 Performance of the PPSN system's end-to-end communication while varying the number of users with three servers in the PPSN server array and 10k noise. 75

6.5 Performance of the PPSN system's end-to-end communication when varying the number of servers in the PPSN server array with 50k users and 10k noise. 76

6.6 PPSN system's overall system load average when varying the number of users with three servers in the PPSN server array and 10k noise. 78

6.7 PPSN system's overall system load average when varying the number of servers in the PPSN server array with 50k users and 10k noise. 79

List of Tables

2.1 User profile dimensions in Smart Nudging[9] 12

2.2 Examples of historical data, current data, plans and predictions[9] 13

List of Abbreviations

GDPR General Data Protection Regulation

HMAC Hash-based Message Authentication Code

JSON JavaScript Object Notation

LGPL GNU LESSER GENERAL PUBLIC LICENSE

ODS Open Distributed Systems

PPSN Privacy-preserving smart nudging

SMC Secure Multi-party computation

SN Green Transportation Choices with IoT and Smart Nudging

SN Smart Nudging

TCP Transmission Control Protocol

UDP User Datagram Protocol

UiT University of Tromsø

XML Extensible Markup Language

Glossary

anonymized The process of altering individual data in such a way that it is no longer be related or identifiable for that specific individual.

cyphertext The other term of encrypted information or encoded information is 'Cyphertext'. It holds the original information in a way that is unreadable either by a human or a machine without the proper cypher algorithm to decrypt it. Cyphertext is also referred to as 'Gibberish' text. Cypher texts can be classified as weak cyphertext and strong cyphertext based on the two properties. One is a repeated pattern in the shifts, and the other one is a uniform frequency distribution in an encrypted message. [1]

data custodian An entity (e.g., organizations, hospital, or laboratory) that collects data about individuals and is responsible for protecting the data.

data subject An entity (e.g., individual person) to which the data refer.

proxy server A proxy server is a server/node in a computer network which acts as a mediator in a server-client communication. Proxy servers take requests from clients and seek resources from servers on behalf of the client—hiding the true identity of the request to the resource server. [2]

sandbox Sandboxing is a technique to address malware threats by containing their malicious behavior within a safe domain inside a system. [3]

threat model In a secure software system, a "Threat model" is a process of identifying, quantifying and analyzing probable security threats in computer-based systems. When a threat model is referred to a secure and private system, it means that the specific system can be resilient under potential attacks described in the specified threat model [4] [5].

1

Introduction

Privacy has been an issue since the inception of the data-science and computing. Whether it is a single person or a large group of people (e.g., citizens of a country, citizens of a continent), many of them prefer not to give up their personal data even if it is for the greater good. Besides in some places data regulations are inherently tighter, for example, in Europe data regulations for user's personal data became stricter after the introduction of General Data Protection Regulation (GDPR) on 25 May 2018 [6]. On the other side, for the greater good and the big picture, we also need a massive amount of user data to process and provide services accordingly to the users by which they—the people, the country, the whole world—can be benefited. Green Transportation Choices with IoT and Smart Nudging (SN) is one of the services which aims to resolve urban challenges like increased traffic, congestion, air, and noise pollution by influencing people towards environment-friendly decisions in their daily life [7]. In order to make the nudge successful, this kind of system depends on a hefty amount of personal user data (e.g., location data, age, gender, health data and many more) and environmental sensor data (traffic data, weather data, bus schedule data and many more) for personalization. Even with this dichotomy between personalization and privacy, most people are happy to give up personal information as long as their perceived advantage of the services surpasses the perceived cost of giving up the required information [7]. Everything usually works out in the basis of trust for the service provider and the user until the appearance of sneaky adversaries who unlawfully compromises the service providers and users to intervene in their system and steal the data thereby [8]. Privacy-preserving Smart Nudging (PPSN) system aims to address these challenges and nudges users most effectively while maintaining their privacy in the face of a strong adversary. That is, an adversary should not be able to tell which user is in which scenario even after interfering with the system. For example, whether the user is in location A or the user is in location B. We are focusing on privacy in smart nudging for SN system. However, this system can be used for any type of nudge services which requires personalization, mining of sensitive user data and aggregating environmental sensor data as long as our core system architecture is compatible with the targetted services. Following sections outline the motivation for the Privacy-preserving Smart Nudging (PPSN) system, PPSN system overview, its research problem, contributions and thesis outline.

1.1 Motivation

In this decade of machine learning and AI, data is the primary fuel to run these SN systems with desired precision. Without substantial data sets, personalization systems are unthinkable. Just like all other personalization systems, SN system also needs a lot of personal user data and sensor data to tailor a successful nudge and improve the whole nudging system by following up user activity [9]. It is accepted that it is hard to nudge people successfully without the presence of personalization—no wonder, the task of user profiling requires a substantial amount of user data. These user data are harnessed either explicitly, through direct user participation or implicitly by monitoring users [7]. To make the nudge effective, we also need environmental and ambience data, which are collected from different types of IoT sensors and servers [9]. Even if the data collected from external servers are encrypted, it is difficult to hide the metadata about when a user is receiving data and from where. It is sufficient enough to create a story just by using metadata. Officials at NSA have even affirmed that "if you have enough metadata you don't really need the content" and that "we kill people based on metadata" [10].

The issue with user privacy does not end there after getting the consent of collecting user data and begin the process of personalization. The real challenge is to protect the user data from the external adversaries when the system gets compromised or tempered. Privacy-preserving smart nudging is not only vital to protect the user from this type of data catastrophes, but also essential to increase the user base of such kind of systems. For both the sake of user's privacy and as system provider to avoid hefty fines from lawmakers [6]. The reason why it's so hard to achieve privacy under our threat model (§4.2) can be illustrated with an example. Let's assume Alice is going to be nudged about her daily office commute. Collecting local bus schedule, local weather forecast, and traffic data is enough to reveal her home and office location. By looking up locations, it is possible to identify Alice. Then the adversary can learn about Alice's behaviour and target her for ads/scams. It is even worse as some of the public APIs for public transport data and weather data are not encrypted at all, making correlation easier for an adversary. We are aiming to resolve the privacy concerns of the user in the face of strong adversaries who can temper with the network and take over the servers but one. There has been a lot of work done to achieve privacy in IoT, messaging system, statistical data analysis and so on but privacy in Green Transportation Choices with IoT and Smart Nudging (SN) (§2.1.2) remained derelict.

This thesis's security goals protect the users from not only the network adversaries, but also the adversaries who compromise all the backend servers but one. The focus of this thesis is to hide and protect the metadata when client-side app communicates with backend servers. Client-side applications are usually sandboxed. Meaning, they are run on individual security domain, and privacy issues in client-side applications are beyond the scope of this thesis [11]. The primary focus of this thesis is to protect identity information of a user when the data leaves the client-side application and when the client-side application receives the data.

1.2 Privacy in smart nudging

The concept of privacy-preserving smart nudging is nothing contemporary. It has been discussed but rather broadly and partially by stating probable privacy-preserving processing, client-side processing, obfuscation and anonymization, and Secure Multi-party computation (SMC) [12] [7]. It tells us some of the ways how we can achieve personalization while maintaining privacy. Encryption alone cannot hide the metadata, and it is essential to protect metadata leakage as we have already discussed how dangerous metadata leakage can be. Nevertheless, we lack a concrete nudging system architecture which addresses all these mentioned privacy issues. As a result, privacy in the smart nudging system has not been practiced yet. Thus, a more practical and efficient concrete system architecture is needed to achieve privacy in smart nudging.

1.3 Research problem

The main objective of this thesis is to ensure user privacy while tailoring user-specific nudge for Green Transportation Smart Nudging (SN) system using user data which most of the time involves using private and sensitive user data (i.e., location, health and so on). This thesis addresses this objective by answering the following research questions.

Q1 – *What are the privacy issues and research gaps in the current solution for a privacy-preserving smart nudging system for Green Transportation Choices with IoT and Smart Nudging (SN)? (Chapter 3)*

This question is directed to find privacy issues, research gaps in user privacy and identifies sensitive user-data in Green Transportation Choices with IoT and Smart Nudging (SN). Latter also points out the user-privacy vulnerabilities that the SN system has.

Q2 – *What are the technical requirements of a privacy-preserving smart nudging system for Green Transportation Choices with IoT and Smart Nudging (SN)? (Chapter 4)*

Piling up a ton of privacy measures can easily make the system impractical since most of the measures are heavily dependent on computationally expensive cryptography and obfuscation, which requires a lot of computing power and bandwidth. This question aims to answer what are the technical requirements to make the system private, yet keeping it practical and usable in real-life.

Q3 – *How to implement and evaluate the privacy-preserving smart nudging (PPSN) system that satisfies the identified requirements for Green Transportation Choices with IoT and Smart Nudging (SN)? (Chapter 5 and Chapter 6)*

This thesis provides with PPSN system to fulfil the privacy need of the SN system based on the identified requirements. As part of the requirement, a concrete design and prototype implementation for the PPSN system is also produced to answer this question. Finally, this thesis illustrates the answer by evaluating and discussing the PPSN system.

1.4 Contribution

The contribution of this thesis is the Privacy-preserving smart nudging (PPSN) system (§4.3) which is a middleware that improves user privacy and overall system privacy for Smart Nudging system in the context of GDPR. The communication, data storage technique and data processing technique of the SN System are rethought to make it compatible with the proposed middleware PPSN system producing SN backend. SN backend (§4.3.2) is typically the SN system (§2.1.2), but with all the features that require to make it compatible with the PPSN system.

This thesis extensively studies GDPR and identifies the current privacy issue in the SN system and addresses them in terms of GDPR. This thesis produces a whole nudging system architecture that has GDPR compliance.

A working prototype of the PPSN system has been implemented using Node.js, protocol buffers, c++ and ZeroMQ. The prototype includes the minimum viable features to complete one round of secure end-to-end communication that have been proposed in the PPSN system design.

Performance of the PPSN system has been evaluated quantitively in terms of privacy, system resource utilization, and end-to-end latency. Subsequently, this thesis reflects how it improves the privacy of the SN system (§2.1.2). Consequently, this thesis produces a system that can hide user's sensitive information, hide end-to-end communication between users and the SN backend, and provides with safer data storage techniques in the SN backend that significantly improves the privacy of the SN system.

1.5 Thesis outline

This section outlines the remainder of the thesis chapter by chapter.

Chapter 2 reflects the theoretical background of smart nudging, then discusses the importance of privacy in the context of GDPR. It also presents the knowledge base of cryptography and the methods to achieve privacy. Then the chapter outlines the implementation technologies that are used to develop the prototype. Finally, the chapter concludes by discussing the related works.

Chapter 3 addresses the privacy issues in Smart Nudging system in the context of GDPR. This chapter also criticizes the legacy Smart nudging architecture in terms of user privacy. Lastly, it ends by discussing the dark side of smart nudging.

Chapter 4, presents the design of the Privacy-preserving Smart Nudging (PPSN) system—the main contribution of this thesis. It also discusses the presumed adversary model and layouts the proposed PPSN system design.

Chapter 5 illustrates the implementation of the Privacy-preserving Smart Nudging (PPSN) system—a barebone prototype of the PPSN system.

Chapter 6, evaluates the Privacy-preserving Smart Nudging (PPSN) system and it’s barebone implementation. It outlines the methods of evaluation and the experimental setup that is used throughout the tests. Then it evaluates the PPSN system qualitatively and quantitively in term of user privacy. This chapter then concludes by outlining the performance evaluation of the PPSN system.

Chapter 7 contains the discussion of PPSN system performance, design choices, prototype trade-offs, scalability and fault tolerance of the PPSN system. It also argues the privacy achieved by the PPSN system. It also discusses the ideal experimentation. Finally, this chapter concludes by asserting how this thesis settles all the stated research problems.

Chapter 8 outlines the future research directions of the PPSN system.

Chapter 9 eventually concludes the thesis.

2

Theoretical background

This chapter illustrates the terminology, the important concepts and the theoretical backgrounds that has been used and have significance throughout this thesis.

2.1 Smart nudging

We consider smart nudging as digital nudging matching the current situation of the user. The whole point of a digital nudge is to inform and motivate the user to choose the recommended activity or thing. The essence of digital nudging is:

> "subtle form of using design, information, and interactive elements to guide user behaviour in digital environments, without restricting the individual's freedom of choice" [8].

Smart nudging and digital nudging are interchangable, where the guidance of each user behavior is tailored to be relevant to the particular circumstances. Through personalization and context-awareness, tailoring of smart nudges is done where knowledge about the user and her situation is central. Before a personalized nudge is designed, knowledge is collected from a wide range of information, creating a user profile and analyzing information in the context of the user. A tailored nudge is more likely to be successful than a non-tailored nudge as it has a higher probability of being accepted and followed by the user [8]. Nudging people to a behavioral change is an illustration of moving target. According to a nudging goal, the intention is to nudge people to behave better. To tailor nudges, the user's normal behavior and the behavior that may change over time must be monitored to determine the difference in the behavior. Nudging should adjust to target the next level of desired behavior as the behavior changes. When nudging does not affect user behavior, it is detected through monitoring, and the nudges need to be adapted. For instance, changes in timing, aimed behavior, information supporting the nudge and/or the presentation of the nudge are revised.

2.1.1 Data collection to provide nudge

In a complex environment, the recommendations are provided by combining data from a wide range of sources, restoring the user profile to reflect the user's responses to nudging in a uniform manner, and tuning the nudges to improve the likelihood of a positive user response. Back-end processing tasks are known to perform simple data integration and analysis or more complex data mining or machine learning based analysis. While some back-end processing tasks have outcomes that are ready to be used to inform and nudge the user directly, others create a pre-processed result that needs further processing at the edge (for example at the user's mobile device). Normally, it is evident that the final processing of a nudge normally happens at the edge. It is because the pre-processed data are combined with local, fresh, and possibly sensitive and private data on the user's smartphone, which include the user's calendar, current location, user profile, and recent preferences.

2.1.2 Green Transportation Choices with IoT and Smart Nudging (SN) [9]

Green Transportation with Choices with IoT and Smart Nudging (SN) is a brilliant idea to solve urban traffic challenges by available traffic resource utilization which helps to avoid expensive infrastructure and unpopular traffic regulations and lesser usage of fossil fuel. The core of Green Transportation Choices with IoT and Smart Nudging (SN) is dependent on nudging people towards environment-friendly decisions. "Nudging" is used to reach a long-term goal for the greater good of our society and the environment in a more subtle way. Here the greater good is when we aim to solve a problem on a global scale (e.g., climate changes, global warming, health problem) by addressing and patching local problems (e.g., increased traffic, congestion, air and noise pollution) [13]. Extrapolating environment friendliness is a challenging task; to give an idea about vehicle vs environment-friendliness, we can plot environment-

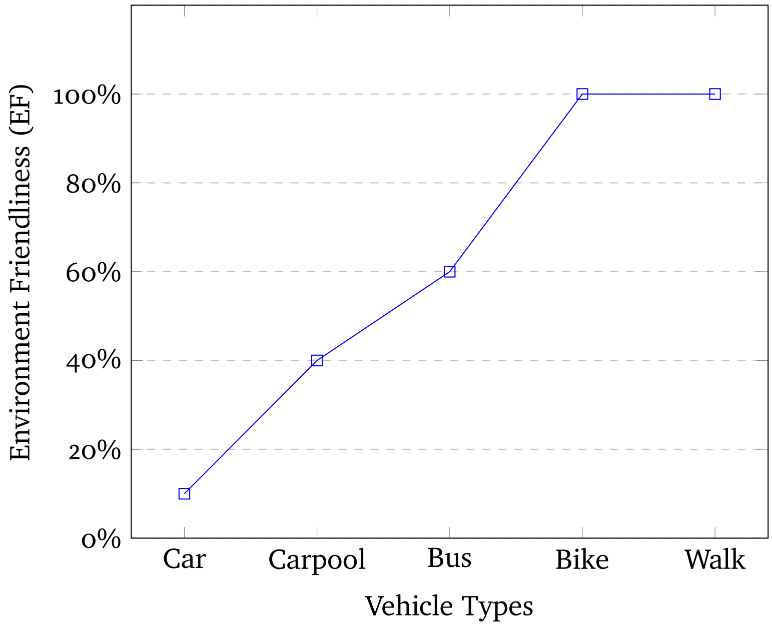


Figure 2.1: Visualizing environment friendliness (EF) vs. different vehicle types[9].

friendliness on the y-axis and vehicle types on the x-axis—gives us a rough visualization of EF by vehicle types listed on figure: 2.1 [14].

One could presume, when people are convinced and free to make their own decisions, only the greater good is achievable for society. Employing coercion and force will, on the contrary, only create chaos and disharmony. Hence, the term nudging comes to dominate the core of SN to push people toward socially desirable outcomes. The EF hierarchy of vehicle or the preferred transportation choices is from walking, biking, taking public transportation, carpool or ride-sharing, to our last choice in the hierarchy, which is the car. Thus, the goal of SN is to find convincing and compelling nudges to persuade people to take the transportations from the top of the EF hierarchy.

Constituting compelling and successful nudges is a difficult task. Nevertheless, it is possible if we have enough data points by which we can tailor and personalize nudges for a specific group of people or an individual. We are gathering the data points primarily from information about transportation choices made by users, current situation of traffic, weather, road conditions and more. The recent expansion of IoT has been seen as an essential building block to facilitate smart nudging for green transportation choices. [9]

In SN, smart nudging is presenting people with relevant transportation information to their decision making (e.g., public transport routes and schedule, real-time traffic and air pollution condition). The point of presenting them with concise contextual information is to make them choose more environmentally friendly transportation ways. Smart nudging is not possible without extensive knowledge of the user. In other words, smart nudging is a type of nudge that has to match with the current situation of the user. Therefore, the more nitty-gritty we know about the user, the more precisely we can nudge them which most likely to be successful and encouraging them. The knowledge base of the users is a continuous process which involves a wide range of data collected from diverse sources (e.g., user's device activity, weather info, the current condition of biking trails, and more), then analysing the context of the user and personalising the nudge. This thesis will take a close look at the user knowledge base and data acquisition in SN architecture in term of user privacy.

Smart nudging is a superset of the term "nudging" that has been first coined in [15] defined as:

> "... any aspect of the choice architecture that alters people's behaviour in a predictable way without forbidding any options or significantly changing their economic incentives"

Here, the choice of architecture is an environment in which individuals make decisions. Besides, according to the author:

> "...to count as a mere nudge, the intervention must be easy and cheap to avoid. Nudges are not mandates. Putting the fruit at eye level counts as a nudge. Banning junk food does not"

Nudges are not only beneficial for society but also beneficial for an individual's long term goals [15]. For SN it is encouraging people towards more environmentally friendlier transportation decision. Smart nudging has all these properties of a nudge and added contextual parameters in the system which most likely makes it more successful. Both the smart nudging and digital nudging are interchangeable, and it is referred to:

> "... the use of user-interface design elements to guide people's behaviour in digital choice environments" [16]

The eminence of smart nudging is irrefutable when we take into account the fact that the decisions people make are not only influenced by the mere fact of the number of choices, but also how it is presented. Thus, the right set of information for a given context comes to play in digital nudging. Collecting user information and making a continuous knowledge base for a specific user is collecting all the usage data of a digital entity that the user is using and interacting with (e.g., smartphones, wearables and more) [9]. Among many other tools, four types of nudging tools have been identified (i.e., 1. Simplification and framing of information 2. Changes to the physical environment 3. Changes to the default policy, 4. Use of social norms), which is also relevant for smart nudging and in the context of SN system [17]. Collecting too much information and information overloading will not be helpful for decision making arguments. Smart nudging will most likely be successful if we have clear, concise and relevant information, which is directly related to the nudging tools "Simplification and framing of information". Also, the smart nudging can have a significant role in implementing the other tools of nudging "changes to the default policy" and "use of social norms". This thesis does not provide any further explanations on nudging tools. However, the idea was to give an insight into how the smart nudging is futile without personalization and situational awareness.

Personalization

Personalization is one of the essential building blocks of smart nudging. Presentation with tailored content and services to a specific individual based on knowledge about their preferences and behaviour, is known as "personalization" [18]. In SN, personalization plays a vital role in influencing people's transportation choices. This personalization or tailoring is extrapolated from various sources of data including user behavioural data, user preference data, traffic, public transportation, road conditions, environmental conditions, and also information about transportation pattern of each user. The process of personalization is also closely related to the user's current transportation need. As we know smart nudging is to convince people towards a greater good for society, a generalisation of data and information overload will not be helpful to convince an individual because different people have a different way of thinking and different preference. Personalization also assists people to make up their mind easily with less effort—has become a valuable tool in searching, filtering, and selecting information of interest.

Situational awareness

Knowing the current ambience and act on it accordingly for a specific user, is "situational awareness". It is the second important building blocks of smart nudging. Situational awareness is the knowledge of the ambience where the user is and will be. Situational awareness plays an important role to make a nudge more likely to be successful. Situational awareness comes from a wide range of data points from the context of the user. Most of this information is publicly available and loosely related to a specific user. Meaning, most of

Dimension	Description
Personal data	Gender, age, nationality and preferred language
Cognitive style	The way in which the user process information
Device information	May be used to personalise presentation of information
Context	The physical environment where the user processes information
History	The user's past interactions
Behaviour	The user's behaviour pattern
Interests	Topics the user is interested in
Intention/ Goal	Intention, goals or purposes of the user
Interaction experience	The user's knowledge on interacting with the system
Domain knowledge	The user's knowledge of a particular topic

Table 2.1: User profile dimensions in Smart Nudging[9]

this data is accounted for a substantial group of people. Information that is related to situational awareness can only be harnessed by sensing, analysing, monitoring, aggregating and predicting from the context of the user.

This kind of data is more available than ever before and easier to monitor, aggregate and predict, regardless of their format. According to SN, among many other data points of situational awareness, we can also monitor, predict and aggregate:

- car traffic in cities and on highways, relevant to services offering traffic-routing advice,
- the flow of vehicular traffic, including average speed and numbers of cars,
- levels of air pollution including carbon monoxide, nitrogen oxides, particulate,
- matter and hydrocarbons,
- used capacity of public transportation,
- the status of footpaths and bicycle paths, and
- weather information that influences transportation choices.

2.1.3 SN architecture

In this section, the legacy SN system architecture will be outlined.

Data	Examples
Historical	Historical sensor data (weather / pollution); congestion history and traffic flow; successful and unsuccessful user travel experiences; past events
Current	Current sensor data (e.g. weather / pollution); road, footpath and ski-track conditions; current pollution levels; current traffic conditions; current location of expected bus
Plans	User's calendar events; bus and train schedules; planned infrastructure maintenance; holidays / recreation days; festivals / events influencing transportation infrastructure
Predictions	The weather the rest of the day; predicted pollution levels during the day; the deviation from the schedule of a bus leaving a nearby bus stop

Table 2.2: Examples of historical data, current data, plans and predictions[9]

SN proposes a somewhat flexible system architecture with a handful of options. However, SN outlined some fundamental building blocks of the SN architecture. According to SN, IoT based smart nudging architecture has three major components: Sense, Analyse, and Inform and Nudge (Figure: 2.2). Finally, these three components will be responsible for the below services:

1. data collection using sensors, crowdsensing, third-party data sources, and crowdsourcing,
2. an analysis that transforms raw data into information, and
3. outreach to the public/user through information and nudging services. [9]

In [9], the SN architecture is outlined and reviewed based on a set of requirements for IoT architecture discussed in [19]. In [20], an architectural approach suggested taking advantage of both the static and dynamic data and support for the actuator interface. The combination of historical data (static data), current data (dynamic data), plans (static data), and predictions (dynamic data) are analysed and used to encourage to change user behaviour through an actuator interface.

SN system also outlines how fog-computing can resolve latency, network bandwidth and access to sensitive private data issues. However, due to weak global knowledge, weak edge nodes and unpredictability of mobile and dynamic environments, fog-computing cannot be the only solution [21]. SN addressed this issue by proposing distributed publish-subscribe, which can be regarded as unifying cloud and fog-computing which supports partial or a hybrid approach of edge and back-end processing. Data collection and global knowledge build

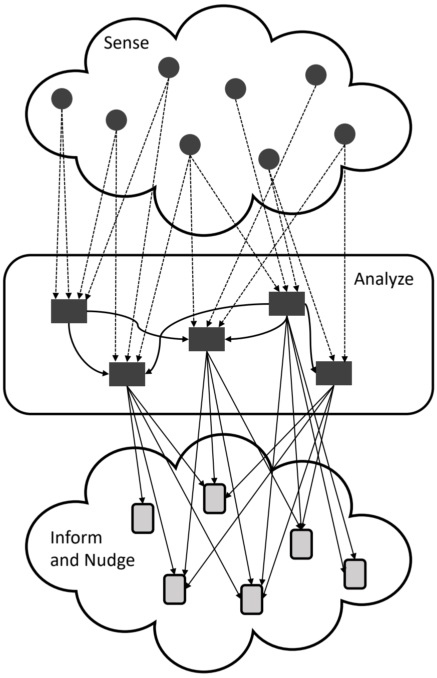


Figure 2.2: Smart Nudging publish-subscribe architecture with the modules Sense, Analyse, and Inform and Nudge[9].

up happens in both back-end and on the edge.

To combine different services from different end-points, IoT based smart nudging architecture uses a publish-subscribe server-client architecture. Thus, the data collection service in the "sense" components can be subscribed to and used by multiple services in the "Analyse component". Similarly, Inform and Nudge component will be able to subscribe and use multiple services provided by Analyse component's filtering services. Depending on the situation, two or more Inform and Nudge component can combine and process raw data differently with the help of different data analysis tools. All types of real-time data, data from user's smartphones, data from external services (e.g., weather data, public transport API), and other types of data from heterogeneous IoT sources are considered in sense component.

Inform and nudge basically sends the nudge directly to the user's smartphones. Inform and nudge can also be sent to a public display to nudge a group of people in a bigger area [9].

2.2 Defining privacy and security

This section outlines the definition of privacy and security for different data stakeholders (i.e., users, applications) and methods to achieve privacy and security in a digital application. This section also elaborates on the relationship between privacy concerns and an individual's decision to release or not to release personal information, and illustrates the GDPR and how it shapes today's information privacy.

2.2.1 Importance of privacy and security

Privacy is a broad concept, and there is a lack of consensus about the definition of privacy. Fundamentally, it refers to an individual or a group's ability to seclude themselves or information about themselves and express themselves selectively. Security is the concept of the appropriate use of data and the protection of information from being misused. Thus, the domain of privacy partially overlaps with security [22, 23].

For personal benefit, a person may want to provide information voluntarily by trusting the data custodian—expecting that there will not be any harm and misuse of the provided data. Despite the good intentions of the data custodian, data might get stolen or misused, leading to identity theft. "Nothing to hide argument" does not hold up quite well when the data gets into the wrong hands and being misused thereby [24].

In a nutshell, privacy is the freedom from unauthorized intrusion and protection from the use of someone's information, which can negatively impact someone's life. Most people are happy to give up their data for the greater good for themselves and society. It is not considered as privacy violation as long as the data is not misused. However, once information is released, it may be impossible to prevent misuse without proper measures [25]. Repeated information leaks may cause distrust among information systems users, and they will highly unlikely to provide information since data security is vulnerable. Therefore, it is clear that privacy is essential for several reasons [22].

This thesis focuses on the technical aspects of user information privacy and ways to disguise the data and metadata from adversaries and protect the user information thereby. In essence, this thesis will discuss the techniques of protecting user's metadata and content while using the data for the Green Transportation Smart Nudging (SN) System.

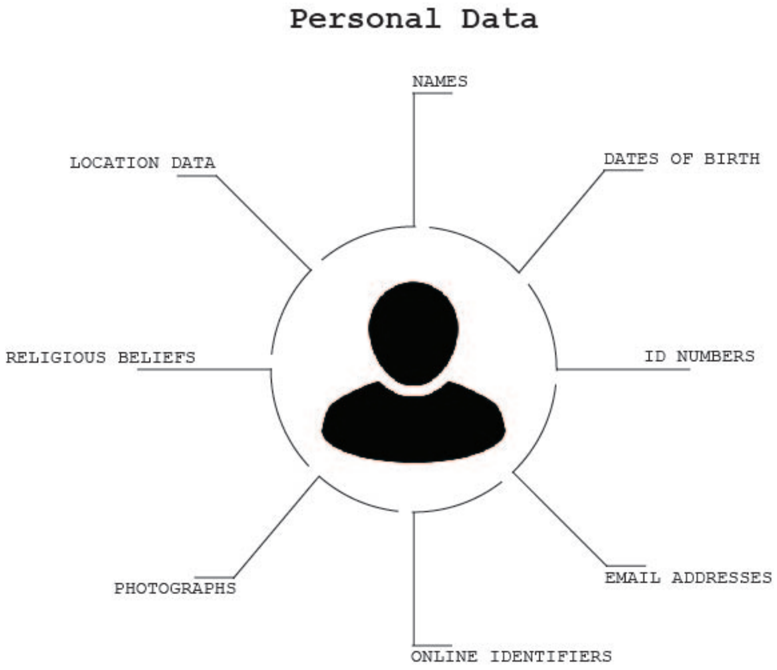


Figure 2.3: GDPR defines personal data is any information that is related to an identified or identifiable natural person.

2.2.2 General Data Protection Regulation (GDPR) [6, 26]

GDPR is an important part of the legal framework for the protection of personal data. Article 1 (2) points out that GDPR protects fundamental rights and freedoms of natural persons and in particular their right to the protection of personal data. Furthermore, the preamble recital 1 highlights that the protection of natural persons in relation to the processing of personal data is a fundamental right. Article 8 (1) of the charter of fundamental rights of the European Union (the "Charter") and article 16 (1) of the treaty on the functioning of the European Union ("TFEU") provide that everyone has the right to the protection of personal data concerning him or her. Unlike the Charter and TFEU, which are general in form, GDPR lays down specific rights for EU citizens in relation to the processing of personal data and also implements enforceable obligations for data custodians.

In order to run operations on EU resident's data, data custodians must comply with GDPR. GDPR shook the data custodians both locally and internationally since this data protection law covers all the EU citizens. Failure to comply can result in significant amount of penalties up to €20 million or 4% of annual global turnover, which ever is greater [6]. The advent of this law enabled EU citizens to have more control over how their information is being collected and processed. GDPR harmonizes the personal data protection in EU, which means that data custodians who are operating in several countries within the EU, only have to comply with one regulation. GDPR tells us what needs to be protected and how to protect it. It does not cover the technicality of how to achieve the level of privacy for the data custodian.

Data that considered as personal under GDPR

The material scope of GDPR is laid down in article 2 (1)-(4), cf. article 4 (1). According to article 2 (1), GDPR applies to "the processing of personal data wholly or partly by automated means and to the processing other than by automated means of personal data which form part of a filing system or are intended to form part of a filing system". In other words, GDPR applies to both automated and manual data processing.

It is also vital to know which information are considered "personal data". According to article 4 (1), "personal data" means "any information relating to an identified or identifiable natural person [...] in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person". Figure 2.3 depicts some of the type of data that are considered as personal under GDPR. Proper identification of personal data not only makes the process of preserving privacy relatively easier, but also makes it easy to identify appropriate technical methods to achieve privacy.

GDPR's six data processing principles

Article 5 (1) lays down six general principles related to the processing of personal data. Data processors perform all the data processing on behalf of data controllers, determining the purpose and means of the data processing. According to article 5 (2), the controller is responsible for demonstrating compliance with these six data processing principles. Personal data must be:

1. Processed legally, fairly, and transparently.
2. Collected for explicit and legitimate purposes.
3. Relevant, and limited to what is necessary.
4. Accurate and up to date where necessary.
5. Retained as long as it is necessary.
6. Processed appropriately to maintain security.

The six lawful grounds for processing the information

According to article 6 (1), there are six lawful grounds for processing personal information. Personal data can only be processed if at least one of the following applies:

1. If the necessary consent from the data subject is received.
2. If it is necessary to meet contractual obligations.
3. To comply with the legal requirements.
4. To protect the data subject's interests.
5. For tasks in the public interest.
6. For the legitimate interests of the data custodian.

Consent is arguably the weakest lawful ground for data processing since it can be withdrawn at any time. The consent can be withdrawn via any medium. Upon the withdrawal of the approval, the individual's data must be erased from the data custodian's system, unless there is a lawful ground to retain it. Legitimate interest is the most flexible of the six lawful grounds of processing data. It could theoretically apply to any processing carried out for any reasonable purpose, sustaining data subject's rights and freedoms. An data custodian must record the processing activity regardless of the lawful ground for processing the data. It is also necessary to give the data subject privacy notices as part of their right to be notified when their personal data is acquired actively or passively.

Data subjects rights

Along with the right to be informed, data subjects have other rights that the data controller needs to facilitate. Data subject's rights:

- The right to be informed (article 13 and 14)
- The right of access (article 15)
- The right to rectification (article 16)
- The right of erasure (article 17)
- The right to restrict processing (article 18)
- The right to data portability (article 20)
- The right to object (article 21)
- The right in relation to automated decision making and profiling (article 22)

Technical and organisational measures to protect personal data

As already mentioned in §2.2.2, article 5 (1)(f) demands that data is processed in a manner that ensures appropriate security of the personal data, including protection against unauthorised or unlawful processing and against accidental loss, destruction or damage, using appropriate technical or organisational measures.

According to article 33 (1), data processors must report all breaches and leakage of personal data to the data controllers. Data controllers are required to report to the Information Commissioner's Office within 72 hours after their discovery.

Any potential risk that can hamper data subject's rights and freedoms, should be informed directly to the data subject without undue delay, according to article 33 (2). However, as outlined in article 34 (3)(a), if the data is anonymized or encrypted to the degree that it is no longer possible to identify the data subject, communication to the data subject is not required.

SN (§2.1.2) is a data-driven system for positively pushing people's behavior for the greater good of the environment in a non-coercive fashion; compliance with GDPR is thus a prerequisite. This thesis will not detail GDPR any further; rather, this thesis has discussed the relevant laws and regulations of GDPR for SN that needs to be complied with and will in the continuation focus on the technical methods to achieve that compliance.

2.3 Cryptography and methods to achieve privacy

This section is all about cryptography and the methods to achieve information privacy. These methods are useful for hiding content, concealing metadata, data masking, and data mining without revealing specific data to third parties called adversaries.

2.3.1 Cryptography

Cryptography involves the practice and study of secure communication techniques (e.g., encryption, decryption) in the presence of adversaries [27]. In other words, cryptography is the construction and analysis of protocols and rules that prevent third parties or the public from reading sensitive information.

Modern cryptography focuses on the numerous aspects of information security, such as data confidentiality, data integrity, authentication, and non-repudiation. The junction of mathematics, computer science, electrical engineering, communication science, and physics forms the state-of-the-art cryptography. In today's cyber world, cryptography applications are everywhere. Some of the applications are e-commerce, chip-based payment cards, digital currencies, computer passwords, and military communications, or anywhere the privacy is crucial.

Unlike modern cryptography, previously, the term encryption and cryptography were interchangeable, making information obscure and hard to read for the unintended recipients. It is necessary to share a decoding technique with the intended recipients, to make them read the original message/information. The use of cryptography ramped up with the advent of rotor cipher machines in World War I, and it got unarguably complex when it met computers in World War II. With the increase of computing power, cryptographic applications have become more ubiquitous, and its techniques become significantly complicated. Today it is unthinkable to have an application in production that does not use any cryptography. Mathematical theory and computer science are the bases of state-of-the-art cryptography. Cryptographic algorithms are outlined around computational hardness assumptions, making such algorithms hard to break in any adversary practice. Theoretically, it is possible to break such a system, but it infeasible to crack by any known practical means. Therefore, in cryptography, "secure" means it is "computationally secure", meaning these schemes provably cannot break with current computing technology. However, some schemes and algorithms are difficult to be broken even with quantum computing power but these schemes and algorithms are mostly impractical to use in practice due to their complexity and computational cost (e.g., one-time-pad [28]).

How cryptography works

First of all, it starts with secrets. Secrets are an essential part of practical cryptography. Cryptography without a secret is pointless or overkill of computing power. In ancient cryptography, they used secret methods, which means knowing how to go from cipher text to plain text and vice versa. For this scheme, we need to know the method that is the secret part. Cipher text (i.e., the text with maximum entropy which makes no sense to adversaries) back to the plain text, which is the secret. Entropy is a lack of predictability or a gradual decline into disorder where real-world data has a predictable pattern. Removing pattern and achieving maximum entropy is the primary goal of a cryptographic scheme.

Secrets in modern cryptography are done in the form of keys. Cryptographic algorithms require those keys to convert cypher-text (i.e., apparent nonsense information with higher entropy) into plain text (i.e., readable information). Whoever has the key or a set of keys with the specific algorithm's exposure can read the original information. Not to mention, generating and exchanging keys is tricky business and vulnerable. In modern cryptography, the most challenging part is the key-exchange between intended parties. One of the popular and widely used key-exchange techniques is the Diffi-Hellman key exchange.

Potential attacks and work around

Cryptography is impenetrable when it obscures data so that it is hard and computationally expensive to duplicate or reverse. Entropy and computation together are the key concepts to achieve theoretically good (i.e., requires massive computing power to break within a reasonable time) cryptography.

Two common ways to break the encryption is Pattern analysis and Brute-force technique. Crypt analysis is the term taking some information about the raw subject, which is useful for reducing the time of brute-force attacks. If we can determine patterns, then it is just plain old hit and tries. In a bruit-force attack for a known pattern, an attacker first learns the pattern and generates a set of all possible values. Then keep trying to decrypt by key in the values in the algorithm until it succeeds or runs out of possible values. For this simple attack cipher-text, the pattern of the key, and the cryptographic scheme is known.

Iterations and adding salts are very useful for encrypting and hiding the same data over the system, which is commonly used in password hashing. For example, the MD5 hashing algorithm is widely used to store sensitive information, which takes a string of any length and encodes it into a 128-bit fingerprint [29]. It is a one-way transaction meaning it is almost impossible to reverse engineer to its original value. However, encoding the same string using the MD5 algorithm will always result in the same 128-bit hash output. The 128-bit hash output of the string ”12345” is ”$827ccb0eea8a706c4c34a16891f84e7b$”. If an adversary gains access in a password database, he will be able to identify the common passwords using known hashes. Moreover, an adversary can also tell which passwords in the systems are similar.

To work around this vulnerability, we can use a random number of iterations to hash over hashing output repeatedly or adding a random string (i.e., salt) to the original string and then hash it to achieve entropy throughout the system. A simple algorithm that adds salts and random iteration while hashing the original string makes it hard for an adversary to steal passwords that are hashed with the MD5 hashing algorithm, as all the hashing output will be unique. Similar hashes can occur, yet it will not be as useful as before, as the hash string points to different original strings.

Based on Mor's law [30], the hammer is getting stronger with computing power and parallel algorithms. With distributed computing, it is getting easier to perform brute-force attacks than ever before. GPUs are very good at math to crack encryptions as well. There is another term called quantum secure which makes encryption safe in terms of current quantum computing.

There are ways to fight a substantial power. Algorithm complexity plays a vital role here. Cryptography and encryption is a continuously evolving process, and keeping up to date with new crypto techniques are vital. For example, ten years ago, Triple DES was safe. Now it is entirely terrible, and AES is the new standard. Large public/private keys are also beneficial. Specifically large prime numbers for asymmetric encryption.

2.3.2 Encryption

Encryption is a process of converting original representation of the information into an alternative form known as ciphertext [31]. This encrypted information is only readable for authorized users with the specific key or method that used to encrypt the original data [31]. Although encryption cannot prevent inference, it makes the content incomprehensible to an adversary. Generally, an algorithm generates a secret in the form of a pseudo-random[1] encryption key for an encryption scheme to encrypt and decrypt. Theoretically, it is possible to decrypt the data without the key. However, for a well-designed scheme and a reasonable length key, the required time to decrypt without the key is impractical as it may take hundreds of years to decrypt with current computing power. Modern state-of-the-art encryption schemes utilize the concept of public-key encryption and symmetric-key encryption [32]. These encryption methods ensure security as the computing power is insufficient to crack the encryption within a reasonable amount of time.

[1] For specific given information, a pseudo-random process produces predictable outcomes, which is typically difficult to acquire without that piece of information.

Symmetric encryption

Symmetric-key algorithms[2] are encryptions algorithms that use the same key for encryption (i.e., generating ciphertext) and decryption (i.e., converting ciphertext to plaintext) [33]. Here the key is the identical secret that is shared between the authorized parties. With the presence of adversaries in an insecure channel of communication, it is quite challenging to exchange keys between the authorized parties in comparison to public-key encryption [34]. However, with the help of a key-exchanging algorithm (i.e., Diffie–Hellman key exchange), it is possible to exchange keys securely and take advantage of symmetric-key algorithms' faster operation comparative to public-key encryption [35].

Asymmetric encryption

It is no surprise that asymmetric cryptography is the opposite of symmetric-key cryptography. In this scheme, the keys come with a pair; public-key and private-key. Public-keys can be disseminated widely, and private-key is kept secret for owners only to decipher the encrypted information. A one-way mathematical function (i.e., multiplication of even prime factors) is constructed that is not straightforward to solve without knowing a part of the solution. This scheme is only effective when the private-key for a specific entity is reserved, meaning keeping the private-key private; the public-key, on the other hand, can be distributed without compromising the security. Any person who has the receiver's public-key can encrypt information with the receiver's public-key. That encrypted message can only be decrypted using that specific receiver's private-key. Asymmetric encryption performance is almost similar to the symmetric-key schemes, except it takes longer to decrypt the encrypted information for the public-key scheme than a symmetric-key scheme. [35]

2.3.3 The RSA algorithm [36]

RSA is one of the widely used public-key cryptography scheme named after the authors' Rivest–Shamir–Adleman' of the paper "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems" [36]. This asymmetric-key cryptography uses a pair of a key to complete the whole process of information encryption and decryption. The key which is used to encrypt (i.e., public-key) will not be able to decrypt that information; instead, it is related to the

[2] Symmetric-key encryption algorithms are also known as secret-key, single-key, shared-key, one-key, and private-key encryption algorithms. Likewise, asymmetric-key cryptography (i.e., public-key cryptography) uses the term secret-key and private-key, which can cause ambiguity between the two different encryption schemes, symmetric-key encryption, and asymmetric-key encryption algorithms.

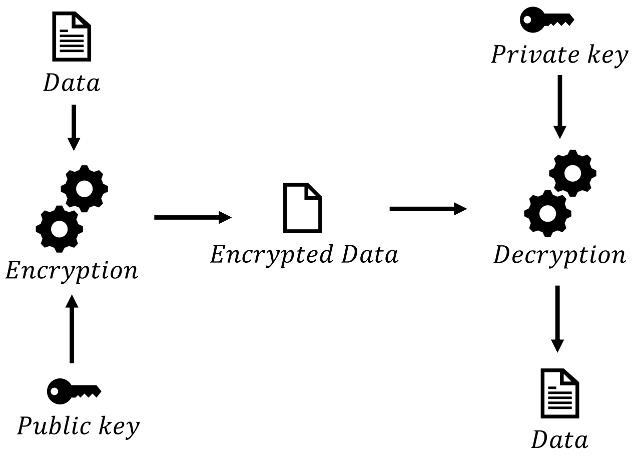


Figure 2.4: Encryption and Decryption process in an RSA algorithm.

decryption key (i.e., private-key) mathematically. Figure:2.4 shows a generic encryption and decryption process flow of an RSA algorithm. The key which is used to encrypt the data is publicly available, and anyone will be able to access it, hence the word "Public-key"—publicly available key. In this cryptographic algorithm, it is essential to keep private-key private to retain the integrity of the system. The RSA algorithm and all other public-key cryptosystem treat information like numbers and perform mathematical operations on them. [35] Figure:2.5 shows a superficial overview of how different type of data gets encrypted with this scheme. (e.g., encrypting a text "GOAL" will first convert into corresponding ASCII codes and then numbers. After the conversion, we plug the number in a one-way function which produces another vague number that converts into byte code and then lastly into ASCII). [35]

A modulus and a public exponent formes a public-key in RSA where the same modulus is also used with a private exponent to generate a private key—makes both the public and private key mathematically intertwined. The modulus is formed by the resultant multiplication of two considerably large prime numbers. In the cryptographic terminology, these numbers are often given romantic names n, e and d where n is the modulus, e is the public exponent and d is the private exponent. The prime numbers that formes the modulus, n is often denoted by p and q. [35]

Generating RSA key pair (i.e., private key and public key) requires a program to decide on a public exponent, e which then can be used to determine the two compatible large prime numbers, p and q for that public exponent, e. Subsequently, from compatible prime numbers p and q we can derive the modulus, n. Later, from that same p, q and public exponent, e, we can compute the private exponent d. Then the onetime function is achieved by destroying

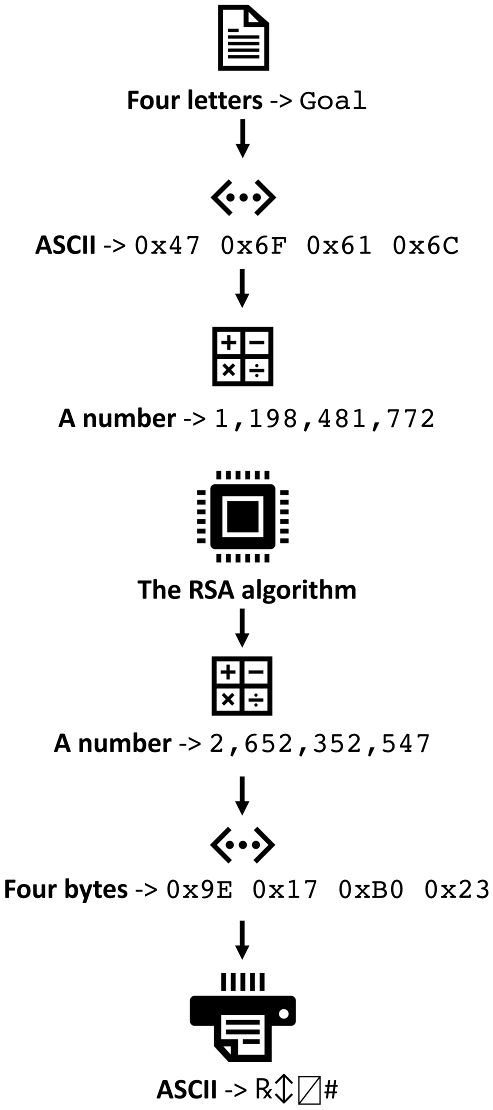

Four letters -> Goal
ASCII -> 0x47 0x6F 0x61 0x6C
A number -> 1,198,481,772
The RSA algorithm
A number -> 2,652,352,547
Four bytes -> 0x9E 0x17 0xB0 0x23
ASCII -> ℞↕⍁#

Figure 2.5: Overview of the RSA algorithm shows that the RSA algorithm is an algorithm which takes numbers and returns numbers (i.e, All data is treated as some kind of number representation of that specific data in order to work with the RSA algorithm)[35].

the compatible prime numbers p and q. [37, 35]

Fundamentally the critical components of the RSA algorithm are large prime numbers (i.e., p and q) and an onetime function (i.e., $n = p \times q$). As easy as it sounds, yet it is too complicated that the oneway function is a multiplication. Multiplication is itself not an oneway function unless we do not know what to divide it by. The intricacy comes from when an attacker is given solely with a result to trace back the original two numbers that participated in that multiplication—known as factoring problem. Mathematically, factoring problem is found to be difficult depending upon how enormous is the resultant of a multiplication. In other words, the difficulty of factoring problem increases with the length of the multiplication product (e.g., factoring $35 = 7 \times 5$ is way easier than factoring $893 = 19 \times 47$, assuming that we do not know the multiplicand and the multiplier, and the factoring difficulty continues with the increase of the length of the product). [35]

Breaking the RSA encryption

From figure 2.6, we see that Eve needs to derive Bob's private key in order to get the key, D_k for decrypting the gibberish data D' into some meaningful information, D. If Eve wants to derive Bob's private key, B_{priv} to get the symmetric key D_k to decypher the ciphertext D' then all she needs to derive is n and d. Since, the public key is public and she has access to it, she technically knows n as it is the part of the public key B_{pub}. To break the security, all she needs to do is to derive the private exponent, d. Mathematically, d is the inverse of $e \mod \phi(n)$. Considering, Eve already knows the public exponent, e; now the problem is narrowed down to finding $\phi(n)$ and perform a modular inverse function which made easier by "extended euclidean algorithm" [38]. Since, n is a product of two prime numbers, Euler's phi-function can be written as, $\phi(n) = (p-1)(q-1)$. As a result, the problem of deriving d is narrowed down to finding p and q. As $n = p \times q$, factoring n will lead Eve to derive the d. However, factoring itself a hard problem which is the foundation of the RSA algorithm. Generally, the length of RSA key is 1024bit or 2048bit long. Which means our p and q is respectively 512bit or 1024bit long. As of 2020 no one able to broke 1024bit or 2048bit RSA key within a reasonable time. Breaking 1024bit RSA key is bruit-forcing on values of 512bit since $n = p \times q$. Since p and q both are prime numbers which make p and q odd numbers. Therefore, the least significant bit is set. The most significant bit is also set since the number is 512 bit long. However, knowing 2 bits and bruit-forcing on 510bits is not any good. [35, 39]

2.3.4 Advanced Encryption Standard (AES) [40]

Advanced Encryption Standard (AES) is a symmetric encryption scheme that is selected and established in 2001 by the U.S. National Institute of Standards and Technology (NIST). AES encryption scheme is derived from Rijndael block cypher and submitted initially to NIST for AES selection process by Vincent Rijmen and Joan Daemen. The proposal contained a different size of blocks and keys later 128-bit block size and three different key sizes (i.e., 128, 192 and 256 bits) were standardized. AES now used worldwide as a standard of symmetric encryption scheme which supersedes the Data Encryption Standard (DES).

Advanced Encryption Standard (AES) is a symmetric encryption scheme that is selected and established in 2001 by the U.S. National Institute of Standards and Technology (NIST). AES encryption scheme is derived from Rijndael block cypher and submitted initially to NIST for AES selection process by Vincent Rijmen and Joan Daemen. The proposal contained a different size of blocks and keys later 128-bit block size and three different key sizes (i.e., 128, 192 and 256 bits) were standardized. AES now used worldwide as a standard of symmetric encryption scheme which supersedes the Data Encryption Standard (DES).

A design principle substitution–permutation network is the basis of AES which efficient for both software and hardware. AES uses a block size of 128-bits represented in a two-dimensional array, and calculations are performed in a particular finite field. The number of transformation rounds (e.g., 14 rounds for 256-bit keys) is determined by the key size that is used to convert the input, plaintext into the output, ciphertext.

Each round has certain processing steps to complete the encryption, and a set of reverse steps are performed to convert the cypher text back into the original meaningful text. The whole process of AES-256bit is as follows:

1. KeyExpansion
2. Round key addition
3. Rounds
4. Final round or 14th round

KeyExpansion. AES key scheduler derives a round key from the cypher key. As stated earlier, AES uses a 128-bit round key block to perform each round of operations.

Round key addition. Bitwise XOR is used to combine the state with a byte of the round key.

Rounds (e.g., 13 rounds for 256-bit AES). For each round, it goes through four steps, and it is as follows: SubBytes—replaces each byte with another according to a lookup table. ShiftRows—performs the transposition step to shift the last three rows cyclically with a certain number of steps. Mixcolumns—mixes four bytes in each column. Finally, it goes to the previous step, round key addition until the final round.

Final round (e.g., 14th round for 256-bit AES). Likewise, the final round goes through all the steps stated in the process rounds, but it does not mix the column

Breaking the AES encryption

AES is prone to several attacks like brute-force attack, XSL attack, side-channel attacks, key-recovery attacks and many more. However, at the time of writing this thesis, if implemented correctly, there are no practical attacks against AES without the knowledge of the key to reading the data encrypted by AES. For example, depending on the computing power, a conventional computer would need about 2,117.8 trillion years to break the AES encryption without the knowledge of the key—making it one of the most secure encryption algorithms yet practical and feasible.

2.3.5 Digital envelope

Digital envelopes are a hybrid version of both the public key encryption and the private key encryption. The concept of encrypting large dataset (i.e., Megabytes of content) with symmetric key encryption and encrypting that symmetric key with a public-key algorithm to transfer the data through an insecure channel is known as Digital Enveloping. Fig:2.6 shows how digital envelope works. Let us consider a scenario where Alice wants to send some data D (e.g., ~5 Megabytes) to Bob securely and Eve is trying to eavesdrop them. First, Alice will encrypt the data D (the content) with a symmetric-key D_k and produce cyphertext D'. Then, Alice will acquire Bob's public-key B_{pub} from Bob and encrypt the key D_k with B_{pub} and produce cyphertext D'_k. Now, Alice is technically ready to transfer the data through an unsafe channel to Bob. As Eve aims to read the content, cyphertext D' does not make any sense to Eve. In order to read the content as D, she needs to get the original key D_k to decrypt D'. Unfortunately for Eve the symmetric-key D_k is encrypted as D'_k with B_{pub} which is the public

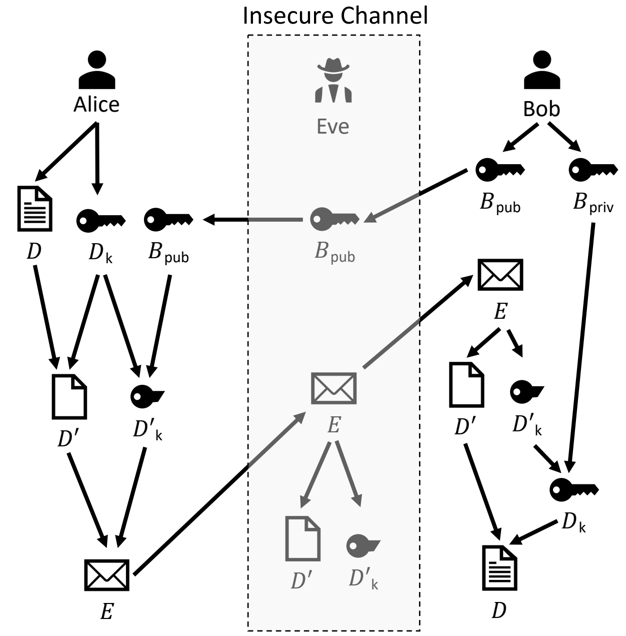


Figure 2.6: Secure data transmission through insecure channel by using digital envelope.

key of Bob. Now, Eve needs Bob's private key B_{priv} or Break the public-key encryption by guessing the other prime multiple. For a reasonable length key (i.e., RSA-2048bit key) it would take ~300 trillion years to get D_k for a classical computer [39]. Which technically gives the content D ~300 trillion years of security since the length of the symmetric D_k is 128 bit. For a Bruitforce attack on 128bit key, it would take 146 trillion years to try only 1% of the keyspace [35]. On average it takes 50% key trial to break a 128bits symmetric encryption which requires even more time to crack, ~8000 trillion years [35].

The digital envelope does not increase the level of security; instead, it is capped towards security level of public encryption. However, with the digital envelope, the goal is not to achieve a higher level of security but performance. The time complexity of encryption and decryption for public-key encryption algorithms increases drastically with the size of data. For example, RC4, one of the fastest symmetric algorithms, will encrypt data 700 times faster than the rate of RSA-1024bit (the most commonly used RSA). Depending on the platform, the throughput of the symmetric algorithms can reach the speeds of 10 Megabyte, 20 Megabyte, 50 Megabyte per second [35]. On the other hand, the public-key algorithms can reach up to the speed of only 20 kilobytes to 200 kilobytes per seconds depending on the platform [35]. Therefore, if we encrypt the ~5Megabytes of content data with 128 bit symmetric-key and then

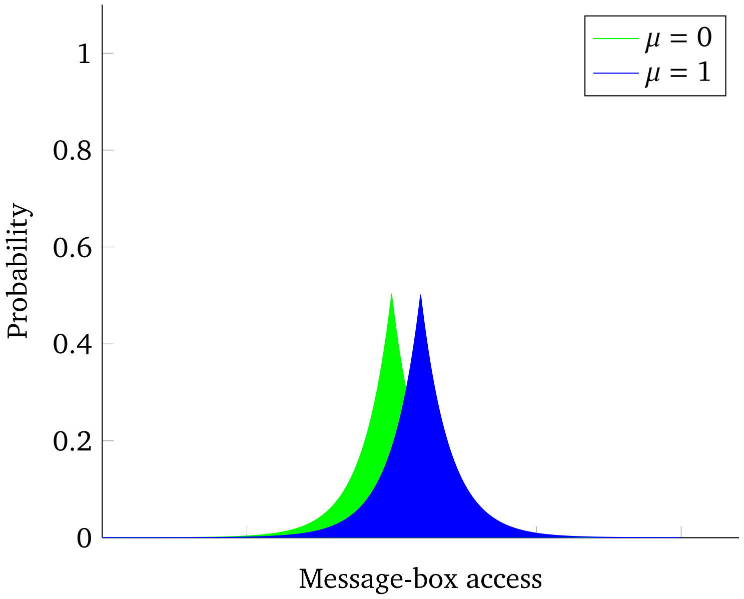


Figure 2.7: Probability distribution of message-box access after adding noise shows us that adversary is having a hard time predicting a situation (equation 2.2)(This plot uses Laplace's double exponential probability distribution (equation 2.1).

encrypt 128bit key with RSA-1024, the approximate total time would be ~100 milliseconds. In contrast, if we use RSA-1024 for ~5Megabytes of content data encryption then we would have ended up with the approximate total time ~25 seconds.

2.3.6 Differential privacy [41]

Differential privacy is a method of achieving privacy in publicly shared data—describing patterns within the group while hiding individual dataset. In other words, it is an algorithm which is used to publish aggregate information about statistical information that limits the disclosure of private individual data. Differential algorithms are widely used in government agencies to publish demographic data and keeping private information private. Since cryptographer also developed this technique of achieving privacy, it is no surprise that it borrows much of its language from cryptography. [42, 43]

A concept of ϵ-differential privacy —a mathematical definition for a privacy loss from a statistical database—was inaugurated by the 2006 Dwork, McSherry, Nissim and Smith article. In that article, the main inspiration was to achieve a level of privacy for data contributors (i.e., individual user) as such the statistical result remains unchanged even if their data is removed [44]. In other words, the statical function should not be overly reliant on specific individual data. To achieve such level of privacy noise—fake data— is added to the original data in such a way that if an individual is removed, it will not affect the aggregated result. The amount of noise needed to hide a number of variables is depends on how much an individual contributes to a result of the data set concerning how much people's data are involved in that result of the data set. The fewer the number of contributors makes up the query, the more noise is needed to achieve the same level of privacy. Therefore, the amount of noise is inversely proportional to a number of contributors in a query. The term "noise" in the differential privacy is based on the Laplace noise (Laplace distribution equation 2.1) [45].

$$f(x \mid \mu, b) = \frac{1}{2b} \exp\left(-\frac{|x-\mu|}{b}\right) \tag{2.1}$$

Here, the location parameter is μ and the diversity parameter is b where $b > 0$

In this thesis, a weaker form of differential privacy will be used—which does not involve any noise calibration that we have discussed earlier. Metadata that reveals who is communicating to a system (e.g., who is receiving nudges from SN and when) can be hidden by making the probability of an entity receives a message, and it does not receive a message roughly equal.

Let, i be a random observation and ϵ be a positive real number then the differential privacy can be formulated as follows:

$$\begin{gathered} Pr[i \,|\, \text{Alice received a nudge}] \leq \epsilon \times Pr[i \,|\, \text{Alice did } \textbf{not} \text{ receive a nudge}] \\ Pr[i \,|\, \text{Alice received a nudge}] \approx Pr[i \,|\, \text{Alice did } \textbf{not} \text{ receive a nudge}] \end{gathered} \tag{2.2}$$

Where, $0 \leq \epsilon \leq 1$

In this epsilon-delta differential privacy when the $\epsilon = 1$ the system has, its maximum differential security level. Likewise, the privacy degrades when ϵ reaches towards 0.

As stated above, according to differential privacy, probability of having made this observation i, given that Alice did receive a nudge through smart nudging system, should be roughly equal to the probability of having made the same observation given Alice did not receive the nudge from smart nudging system. In other words, the adversary can see this observation, and it should not be

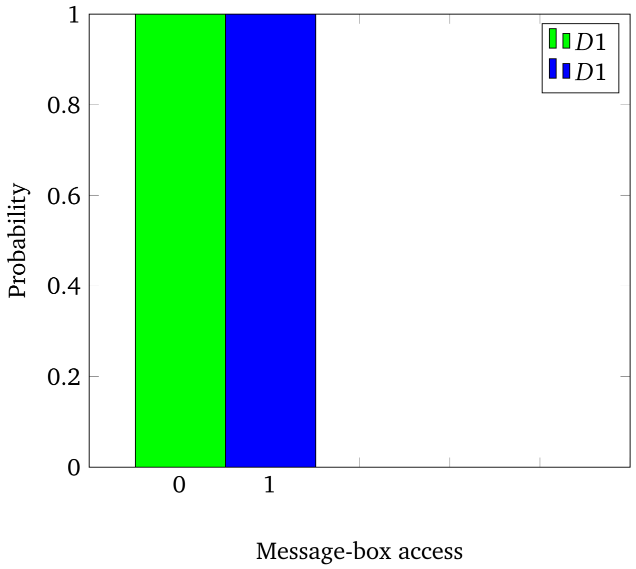


Figure 2.8: Probability distribution of message-box access before adding noise—makes it easy for an adversary to tell which situation a user/variable is currently in.

able to tell which world a user is currently in—whether Alice getting the nudge or not.

In figure 2.8, the probability distribution shows how the data is easily distinguishable without any noise. In the figure 2.7, the probability distribution shows how negligible is the probability distribution shift—providing better privacy by adding noise. Therefore, an adversary finds it more hard to predict what happened throughout the system.

2.3.7 Mix-net

The concept of "Mix network" (mix-net) first appeared in an article published in 1981 by David Chaum[3] [47]. The core idea of a mix-net is to make an end-to-end communication that is hard to trace. It uses a chain of "proxy servers" to mix and obfuscate the source and destination. Senders send messages to the mix (i.e., collection of proxy servers) then the mix shuffles the messages and sends back out in random order to another mix node until the message reaches the final destination. This mixing creates anonymity between the actual source and destination by breaking the link between the sender and the receiver. Thus, adversaries and eavesdroppers find it hard to trace end-to-end communications.

[3] David Lee Chaum (born 1955)—an American computer scientist and cryptographer [46]

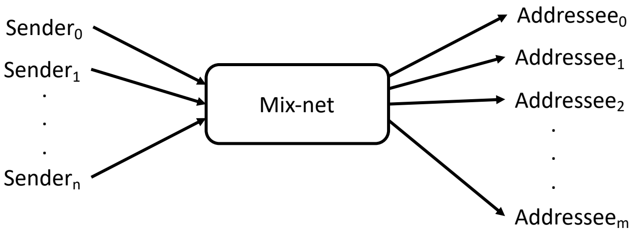


Figure 2.9: An overview of a mix network where the links between senders and addressees are broken.

Moreover, one of the core design principles of a mix-net is that it knows only about the previous node (i.e., from where it received the message from) and the immediate destination (i.e., the destination to send the shuffled messages to), which makes it immune to malicious mix nodes. Hence, limiting the overall knowledge for each node which is a crucial aspect of mix-net to preserve privacy. [48, 49]

Message encryption and decryption in mix-net are based on public-key cryptography discussed in §2.3.2. The final encrypted message in a mix-net is like an onion with the message in the innermost layer. Each node in the mix-net strips off its own layer of encryption and expose the next destination, and the journey of that message continues through the mix-net until it reaches its destination. Even if all the nodes are compromised, but one; can still provide untraceability of the message's original source and destination against weak adversaries. There are several applications based on this concept, including onion routing, garlic routing, and key-based routing (e.g., Tor, I2P, and Freenet) [50, 51, 52]. This thesis will take advantage of mix-net's streamlined architecture to achieve privacy preserved end-to-end communication in smart nudging system. The later sections will outline the schematics, message format and vulnerabilities of mix-net.

Mix-net schematics

Figure 2.9, depicts a simple overview of mix-net, where N number of senders send messages to M number of addressees via mix-network. Here, the mix network mediates and forward the messages until it reaches the originated destination—breaking the link between Senders and Addressees. Thus, the source and destination have no idea about who was the message originator. In the mix-net, message orders are usually shuffled and sometimes delayed (e.g., to lessen time attacks). Figure 2.10, outlines a simple decryption process of messages which are encrypted using the sequence of public-keys of the mix. The sequence of encryption and decryption is closely related to the message

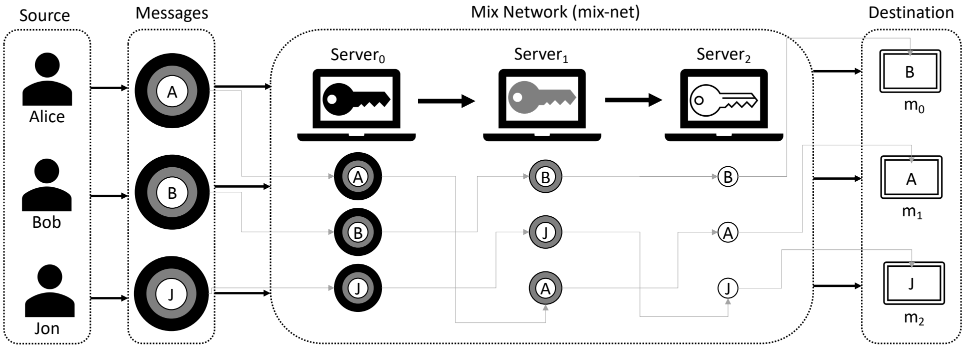


Figure 2.10: Simple decryption process in mix-net. Clients encrypt the messages with the sequence of public-keys. Each node in a mix-net removes a layer of encryption with its own private-key and shuffles the message order and passes the results to the next destination.

travel path through the chain of proxy servers in the mix-net, as the private-keys which are required to decrypt messages live in specific servers privately.

In figure 2.10, Alice, Bob and Jon prepares messages for original destination $m1$, $m0$ and $m2$, and encrypt it with each servers public-key, random salt and next destination then finally sends it to that server who can strip off the outer layer of the encryption and carry on the message transmission. Black, grey and white colours denote the corresponding keys and encryption layers. Message movements through the server have also been shown with directional grey arrows. Finally, the last server strips off the innermost layer of encryption and aware of the final destination $m0$, $m1$, $m2$ and sends the messages respectively.

Lets us assume a participant A wants to send a message to a participant B. Participant A prepares a message by appending random value R then encrypting with addresses public-key K_b, $K_b(message, R)$. After that participant A appends the B's address and encrypt the message with the mix's public-key K_m, $K_m(R1, K_b(message, R), B)$ and sends the constructed message to the mix, M. M decrypts the message with M's private-key and strips off $R1$ which exposes B's address. Then, M sends the message $K_b(message, R)$ to B. Lastly, B decrypts the message and strips away R and gets the original message. When B get the message, it is evident that B does not seem to aware of the message originator.

Message format

The standard message format for the mix-net is listed below.

$$K_m(R1, K_b(R0, message), B) \longrightarrow (K_b(R0, message), B) \tag{2.3}$$

To build this message format, the sender appends a random string $(R1)$ and encrypts the nested envelope containing the message with the mix's public-key (K_m). The nested envelope[4] contains a random string $(R0)$ and the original message which is encrypted with the recipients public key (K_b). When mix-net receives the encrypted message, it decrypts the most outer layer of the envelope by using the mix's private key, which then reveals the address of B. The random string $(R1)$ is also discarded in this stage.

It is presumed that the adversary is active and can monitor all the messages which are passing through the insecure channel. The salt (e.g., random string $R0$, $R1$) plays an important role here to prevent attackers from guessing the messages. Let us assume that the salt was not used and K_b(message) was sent to B. A good guess can lead the attacker to conclude that $message'$ was sent to B and he only needs to test if $K_b(message') = K_b(message)$ holds. However, adding salt (i.e., random string) changes the situation by raising the intricacy of guessing the actual message because learning $message' = message$ is true does not reveal the original message since the attacker does not know $R0$. Thus learning, the $message' = message$ does not help the attacker as he is left with the confusion that the message he derived might not be right.

Untraceable response

In mix-net, it is possible to make an anonymous response to the source (e.g., B responds to A while A's identity is secret). This possible because a source provides a onetime public key for encrypting the response and source's address encrypted with mix-net's public key. If we take the example that has been discussed earlier, A needs to form an untraceable return address for B. It is done by encrypting A's address with mix-net's public key K_m, and salt $S1$ is added to prevent the address from guessing. A public key is also provided by A, which will be used by B to encrypt the response. Thus, the response will only be able to be decrypted by A. The return address provided by A will be encrypted by K_m, mix-net's public key as $K_m(S1, A), K_x$. K_x is the public key which will be used by B to encrypt the response to A and $S1$ is a random salt. A can send the message anonymously along with the return address that has been discussed in §2.3.7.

[4] Layered mix-net's enveloping does not include symmetric encryption; thus, it should not be confused with the digital data enveloping discussed in the §2.3.5

When the message from A reaches to B, B transmits the message as $K_m(S1, A)$, $K_x(S0, response)$ to the mix-net, M. M transforms the message into $A, S1(K_x(S0, response))$. Since both the public key, K_x and random salt, $S1$ has been created by A; the mix cannot see the response created by B. The following indicates the untraceable request and response between the node A and B over a mix-net M.

The message from $A \longrightarrow B$:
$K_m(R1, K_b(R0, message, K_m(S1, A), K_x), B) \longrightarrow K_b(R0, message, K_m(S1, A), K_x)$

Reply message from $B \longrightarrow A$:
$K_m(S1, A), K_x(S0, response) \longrightarrow A, S1(K_x(S0, response))$

Where: K_b = B's public key, K_m = the mix's public key. Therefore, in mix-net, an addressee can reply to a sender anonymously and holds the anonymity between the sender and receiver.

Threat model

Although mix-network provides anonymity between source and receiver, even when an adversary is able to observe the entire path, it is not absolutely perfect in the face of strong attacks (e.g., long term correlation attacks—tracing sender and receiver packets) [53]. Given that fact, the later paragraphs will outline the threat model and some probable attacks of mix-net. The "threat model" of a mix-net is as follows. An adversary can:

- Monitor both the incoming and outgoing network traffic.
- Analyze time between multiple packets.
- Observe all the links of the network.
- Demystify the strategies and infrastructure of the mix-net.

Given that threat model, correlation of a packet on an input link and output link is not possible by what time the packet arrives, the size of the packet or the content of the packet. Packet timing-based correlation attacks are prevented by batching the requests and responses. In addition to that, encryption and packet padding prevents the correlation attack based on packet size and packet content.

Possible attacks against mix-net

Among many types of attacks, time analysis attacks, packet gap attacks, packet burst attacks, and sleeper attack are common for mix-net. Most of these attacks are subjected to temper the packets and then observe how the mix-net behaves. Another way of observing mix-net behaviour is to force re-transmission of Transmission Control Protocol (TCP) packets which can be done by corrupting packets. [54, 55]

2.4 Implementation technologies

The prototype implementation of the PPSN system requires several implementation technologies. This section layout the choice of implementation technologies and frameworks needed for implementing the prototype of the PPSN system.

2.4.1 Programming languages and frameworks

Several programming languages and frameworks have been used to implement, test and evaluate the PPSN system's prototype implementation, and some of them are as follows:

Javascript and Nodejs. Nodejs is a runtime environment for running the javascript—a high-level interpreted programming language with object-oriented capabilities—codes on the server-side that uses Chrome's V8 JavaScript engine—one of the fastest javascript engine available as of writing this thesis. It is platform-independent—runs on Windows, Linux, Unix, Mac OS X and more. Together with javascript and Nodejs, it gives an advantage of asynchronous and non-blocking programming. Threading is possible in Nodejs by spawning child processes. Clustering in Nodejs also made it easy for load balancing and share sockets between processes.[56][57]

C++. C++ is a statically typed, compiled, general-purpose, case-sensitive, free-form programming language. C++ supports procedural, object-oriented, and generic programming. It is considered as a middle-level language since it has a combination of both high-level and low-level language features. [58]

Sockets. Sockets enable communication between two different processes on both the homogeneous and heterogeneous computing environment. It uses the concept of standard Unix file descriptors as a method of communication. Every I/O operation in Unix is done by writing and reading a file descriptor—and integer associated with an open file. The file descriptor can be a network connection, a text file, a terminal, and more. There are mainly two types of sockets, namely TCP and UDP. Transmission Control Protocol, TCP is reliable

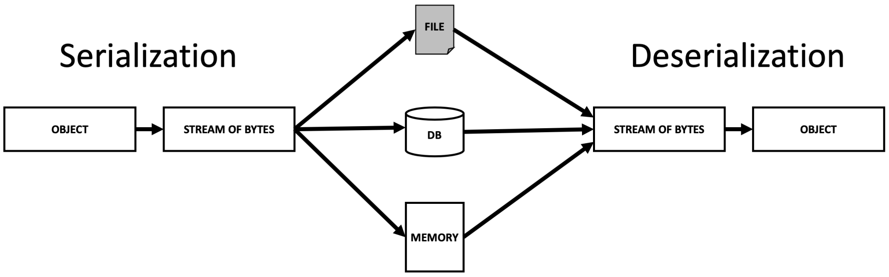


Figure 2.11: Protobuf serialization and deserialization flow diagram.

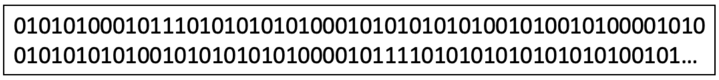
communication sends back a delivery report, and data transfer happens in a stream and maintains a sequence. On the other hand, User Datagram Protocol, UDP, is unreliable and does not have any order. Their implementation made UDP faster and more efficient than TCP with some cost of reliability. [59]

ZeroMQ. ZeroMQ (also known as ØMQ, 0MQ or ZMQ) is a high-performance asynchronous messaging library. It intended at use in distributed and concurrent applications. ZeroMQ system can run without a dedicated message broker even though it provides a message queue. It supports common messaging patterns, namely pub/sub, request/reply, client/server and many more. In addition to that, it also supports a variety of transports, namely TCP, in-process, inter-process, multicast, WebSocket and many more—making inter-process messaging simple as inter-thread messaging. Besides, It is relatively easy to scale and has support for a wide variety of languages. [60][61]

2.4.2 Protocol buffer

Protocol buffers are mechanisms for serializing structured data (i.e., Extensible Markup Language (XML), JavaScript Object Notation (JSON)) that significantly reduces the overall payload for data communication through wire compared to any other human-readable structured data scheme (i.e., XML, JSON) in an uncompressed environment as of now (July, 2020). This Google's method of serializing structured data is language independent. [62]

In order to work with protocol buffers having a schema (Listing 2.1 shows a protobuf schema, equivalent of JSON data in listing:2.2) is mandatory, which follows a set of rules defined by proto-language. "Protoc" command is responsible for compiling the schema file (i.e., .proto) to corresponding language source (e.g., python, java, C++ and many other languages). What protocol buffer compiler does is to produce a language-specific unique source code to read and write structured data from a variety of data streams using a vari-



```
01010100010111010101010100010101010101001010010100001010
0101010101001010101010100001011110101010101010100101...
```

Figure 2.12: Sample serialized Protobuf file.

ety of languages. Figure 2.11 shows a general control flow of protocol buffers serialization and deserialization. [62]

Listing 2.1: Sample Protobuf schema

```

syntax = "proto3";

message PersonalData {
  required string Name = 0
  required string Age = 1
  required string Location = 2
  repeated string Activities = 3
}
```

Listing 2.2: Sample JSON Data

```
{
        "PersonalData": {
                "Name": "Craig␣Federighi",
                "Age": "59",
                "Location": "California",
                "Activities": ["Running",
                        "Walking",
                        "Cycling",
                        "Skiing"]
        }
}
```

There are several reasons why we are interested in using protocol buffers in our proposed solution, privacy-preserving nudging system (PPSN). First, it reduces the request and response data size almost in half in comparison to JSON data (e.g., 106 bytes of JSON encoded data into 47 bytes of Protobuf encoded data). Encryption and decryption time significantly decreases if we can reduce the data size of the actual content.

Protocol buffers turn contents into serialized binary data (Figure: 2.12 depicts a serialized protobuf data file) which is suitable for transferring it through wires

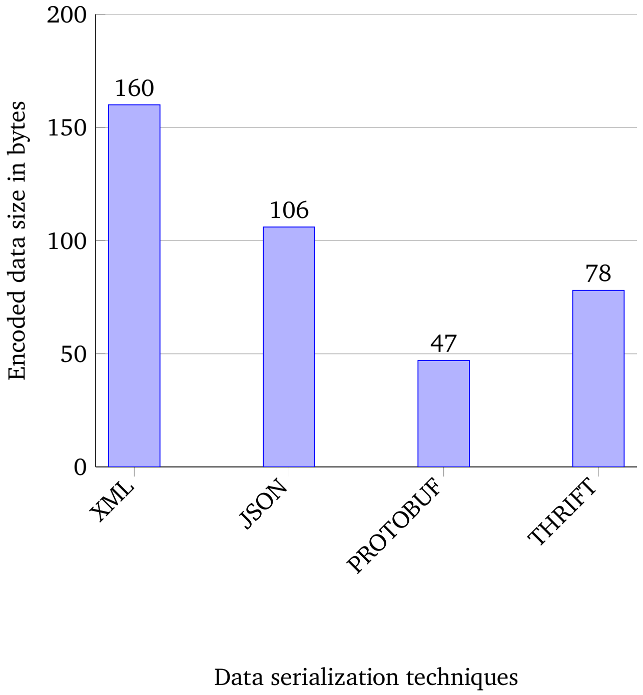


Figure 2.13: Encoded data sizes of different data serialization techniques.

among servers. Since we are also aiming for a differentially private system, hence reducing payload means reducing overall bandwidth cost.

Protocol buffers also guarantee type safety. It not only prevents schema violations, but also provides fast serialization and deserialization. Since we are required to use a schema; it gives us another layer of protection as the attacker has to know the schema to read the content. In other words, protocol buffers native format are not human-readable and human-editable (Figure: 2.12); without a specific schema, it does not make any sense of a protocol buffer.

Protocol buffers vs. popular data serialization choices

XML(Listing: 2.3) and JSON(Listing: 2.2) are the popular language-neutral data serialization methods that are widely adopted throughout many programming languages. XML is suitable for a heterogeneous environment [63]. However, the redundant use of tags and larger XML encoded file size made it inefficient. To overcome this disadvantages of XML, JSON emerged with relatively lower data size than an equivalent encoding in XML. JSON's straight forward data representation and better performance than XML is making JSON a popular choice of data representation [64]. Both the XML and JSON does not require any predefined schema, meaning they do not enforce any particular schema. Moreover, they are not suitable for transferring over the wire. To overcome these problems, binary-data-representations have appeared, and protocol buffers are one of them. Figure 2.13 shows the data size comparison of XML, JSON, protocol buffers and some other data-representation schemes. In protol buffers, data serialization and deserialization happens under 0.1 ms for ~200 bytes of equivalent JSON data [65]. From figure 2.13, we can clearly see that protocol buffer's data size is significantly smaller than XML and JSON[5][6]. Hence, using protocol buffer data serialization should give us some performance edge in terms of encryption and decryption time.

Listing 2.3: Sample XML Data

```
<?xml version="1.0" encoding="UTF–8"?>
<root>
   <PersonalData>
      <Activities>
         <Activity>Running</Activity>
         <Activity>Walking</Activity>
         <Activity>Cycling</Activity>
         <Activity>Skiing</Activity>
      </Activities>
      <Age>51</Age>
      <Location>California</Location>
      <Name>Craig Federighi</Name>
   </PersonalData>
</root>
```

There are some other comparable binary data serialization methods and techniques (e.g., Thrift, Flat Buffers, Avro). Some of them slightly outperform the protocol buffers in some scenarios, but they are mostly similar in performance. However, the more relaxed and less risky integration of protocol buffers outweighs the competitions.

[5] A unique number is given to every field in the message definition which use to identify the fields in the message binary format, and should not be changed while message type is in use. Field numbers (including the field number and the field's type) in the range 1 through 15 take one byte to encode. Field numbers in the range 16 through 2047 take two bytes. For better performance, field numbers 1 through 15 should be used for very frequently occurring message elements [66]

[6] Field numbers 19000 through 19999 (FieldDescriptor::kFirstReservedNumber through FieldDescriptor::kLastReservedNumber) are reserved for the protocol buffers implementation hence unusable [66]

2.5 Related work

Privacy in the smart nudging system is relatively new, especially when it comes to the context of GDPR. Hence, there is a lack of a complete privacy preserved nudging system which is practical to use. However, the privacy in smart nudging is addressed in "Privacy Preserving Personalization in Complex Ecosystems" [8], which addresses some of the privacy issues in smart nudging and provides a possible solution. Nevertheless, it does not argue the privacy of the SN system in terms of GDPR (§2.2.2). Neither it provides any concrete end-to-end privacy preserved system design, nor a proof of concept that preserves privacy for the SN System.

Privacy is addressed heavily in private messaging systems, secure client-server communications, untraceable end-to-end communication and more. Systems like Dissent [67], Riposte [68], Vuvuzela [69] and Stadium [70] provides privacy for a private messaging system. Tor [71] and mix-net (§2.3.7) also provide untraceable end-to-end communication. Needless to say, none of them is for smart nudging. However, some of the concepts—message-box, mix-net, communication in rounds, shuffling requests, and adding noises— are borrowed from Stadium [70], Vuvuzela [69] and Tor [71].

Time-lapse cryptography [72] for encrypting data for the future and secure multiparty computation (SMC) [12] for secure data aggregation has also been studied. However, these methods to achieve privacy have not been used in this thesis to avoid intricacy and performance bottlenecking.

3

Privacy in smart nudging

It is no surprise that the nudging can pose serious threats to citizens' privacy [73]. Therefore, as a nudging system SN (§2.1.2) needs to be addressed in terms of both the user privacy and system privacy. This chapter outlines the privacy issues of the SN (§2.1.2) by addressing them in terms of GDPR (§2.2.2) guideline and user biased privacy policy. Later in this chapter, there will be a discussion about the essential requirements for the Privacy-preserving smart nudging (PPSN) system—a contribution of this thesis, Privacy-preserving Smart Nudging system (PPSN)—and discuss how to make it GDPR compliant and resistant to traffic analysis.

3.1 Privacy issues in personalization

Personalization requires composing user profiles which can then be used to identify interests, behaviour and other characteristics of a specific user. In SN, the probable user profile and its dimension is listed in table 2.1. From Table 2.1, we can see we are interested in a whole range of private data including personal data, cognitive data, device information, context, history of past interaction, The user's behaviour data, users interests, intention/goal, interaction experience, domain knowledge. All these data points are subjected to identify a physical person—considered as personal data according to GDPR (§2.2.2). Some of these data points can directly identify an individual (e.g., gender, age, nationality, location), and some of the data (e.g., domain knowledge, interaction/goal) can require some processing to identify a user. Some of the sensitive user data cooperating with additional knowledgebase can cause stigma, and accidentally exposing them can cause serious privacy breach. As discussed, according to GDPR, SN is entirely responsible for processing and protecting such type of personal data (§2.2.2). No matter how SN gather user information, if the data is enough to identify an individual, it counts as personal data, and SN is obliged to protect that data 2.2.2. As personalization is inevitable to construct smart nudge, providing privacy for users also become inevitable according to GDPR—discussed in §2.2.2 [9].

3.2 Situational awareness and its privacy concerns

As we can see, these data are targetted to a situation, location and a specific moment. They are related to a large group of people, and without any further information, it hard to identify a single individual. These kinds of data are not classified as personal data. Thus, we do not need to take tighter privacy measure to protect it. After all, this kind of data is publicly available. Here the primary goal of PPSN is to isolate or disjoint any relations with the user data which can complete a story and identify a user. In table 2.2, we have discussed some of the historical data, current data, plans and prediction. From table 2.2, we can see some privacy vulnerability for some of the data which are revealing user-specific information. In plans, users calendar events, in historical data, user interactions information needs to be protected as they reveal critical user information which can be used to identify a specific user.

3.3 Privacy concerns in SN architecture

This section addresses the privacy vulnerabilities of the SN system (§2.1.3) in user perspective and GDPR (§2.2.2) compliance.

As outlined in §2.1.3, the primary components of SN system are sense, analyse, and inform and nudge (Figure: 2.2). All these components are somewhat sacrificing user privacy.

These three components are basically working data collection using sensors, crowdsensing, third-party data sources, and crowdsourcing, data collection using sensors, crowdsensing, third-party data sources, and crowdsourcing, an analysis that transforms raw data into information, and outreach to the public or user through information and nudging services. Later paragraphs will outline their probable privacy vulnerability.

From §2.1.3, among all other functions, SN services more or less related to mobile phone sensing [74]. Mobile phone sensing is, in a sense, tracking and monitoring an identifiable user. These type of data are closely related to individuals, and upon running some set of data operation, it is possible to identify a specific user. According to GDPR guidance discussed in §2.2.2, any data that can identify an individual is counted as personal data which needs to be protected.

In [9], the SN architecture is outlined and reviewed based on a set of requirements for IoT architecture discussed in [19]. In [20], an architectural approach suggested taking advantage of both the static and dynamic data and support for the actuator interface. Meaning, the combination of historical data (static data), current data (dynamic data), plans (static data), and predictions (dynamic data) are analysed and used to encourage to change user behaviour through an actuator interface. This architecture indicates a mixture of public and private data which opens up vulnerabilities for individual private users.

Even though SN intends to enable fog computing instead of back-end computation due to latency, network bandwidth and access to sensitive private data, ultimately SN is unable to go solely on fog computing [21]. One of the main reason due to its bleak prospect of global knowledge, weak edge nodes and unpredictability of mobile and dynamic environments. SN addressed this issue by proposing distributed publish-subscribe which can be regarded as unifying cloud and fog computing which supports partial or a hybrid approach of edge and backend processing. As a result, some data leaves the edge nodes to the designated backends for further information treatment, then again, it begs the question of user data privacy and metadata leakage thereby.

Sense component of SN system—discused in §2.1.3—blends all types of real-time data, data from user's smartphones, data from external services (e.g., weather data, public transport API), and other types of data from heterogeneous IoT sources. Thus, all the operations and acquisition of data is happening in the sense component, regardless of the privacy severity of the information. Therefore, the SN system architecture's "sense" component needs to be addressed in terms of privacy vulnerability.

The SN architecture component "inform and nudge" usually targets individual user's smartphones, hence leaks metadata. Adversaries can add bits and pieces to complete the whole story and analysing the type of nudge an individual is receiving can reveal a lot about their activity and identity. However, public displays and public information broadcast in "inform and nudge" contains nudging for a group of people or a location which does not reveal individual user data [75].

SN system architecture is somewhat event-driven by nature of the publish-subscriber model—the arrival of new data can trigger data analysis and inform and nudging. These type of event-driven systems are easier to attack as an adversary can listen for changes for particular events and predict the story thereby. IoT based sensing devices which produce data periodically are less-vulnerable due to their intermittent behaviour. However, IoT sensing devices which are closely related to a specific individual (e.g., wearables, health bands, smartwatch, power meter) needs to be protected.

3.4 The dark side of nudging

Just like how the strength of nudging can be used as influencing people towards the greater good for the society; similarly, it can also be used to manipulate a demographic once the power of nudging reaches the wrong hand. Unfair nudging can push people towards some particular situation (e.g., to buy specific consumer products, influencing to support a political party and many more) which can be exploited by an unfair-meaning party [76] [77]. Thus, fair nudging systems are prone to be compromised and manipulating people, thereby. A potential danger of manipulating people and criticism of nudging is outlined in [78]. Providing transparency about making aware of the people who are being nudged could potentially solve this issue [77]. However, an unfair-meaning party will not reveal their transparency or the true motive of nudging. People consciously trust well-meaning parties to get nudges to do greater good for society. Thus, leaving the nudging system of well-meaning parties (e.g., Smart Nudging for Green Transportation) in a critical situation of being targetted by adversaries [9]. Hence achieving privacy in fair nudging systems is inevitable to protect the users from being exploited by unfair nudges—one of the primary goals of this thesis.

4

Design

In the earlier chapter (Chapter 3), it has been discussed why SN system architecture (discussed in §2.1.3) needs to be addressed in the context of user privacy (discussed in §2.2.1) and GDPR (discussed in §2.2.2). §3.3 pointed out some of the critical vulnerabilities in SN system architecture with respect to user privacy and GDPR. This chapter outlines the design of PPSN—one of the contributions of this thesis— and it's approach to mitigate all, if not some of these critical privacy challenges to gain GDPR compliance.

PPSN aims toward a user biased privacy policy—meaning PPSN will try to give users as much as autonomy as possible throughout PPSN system design. Thus, users have more control over their data, and they will more likely be sharing their information for the greater good for the society. Moreover, the PPSN system design not only provide strong privacy but also decent scalability. This novel system design offers a decoupled middleware for SN, which can be implemented with legacy systems given that the legacy program runtimes are sandboxed or partially virtualized and no direct communication is happening between users and SN system back-end. PPSN is a middleware for handling communications that are happening on SN and it also provides data security. It is possible to use PPSN middleware without modifying an existing SN system. However, for full compatibility with PPSN middleware, some of the components of SN from the outer core needs to be modified—discused in §4.3.8. Thus, background processors, daemons and all other communication with public APIs can still be on the operation with the help of PPSN middleware. PPSN will isolate only those communications which are vulnerable to user privacy which has been discussed in §3.3. Therefore, public-APIs and data belong to larger groups will not be passed through PPSN middleware for PPSN system optimization point of view.

The primary objective of PPSN system is to mitigate three major vulnerabilities as follows, metadata leakage, content theft and end-to-end user tracing under pervasive network monitoring in the face of strong adversaries, given that at least one server is honest. PPSN's key insight is to reduce the number of observable variables concerning user privacy. Later sections will outline the proposed system design of the PPSN system—a privacy preserved super-set of the legacy SN system.

4.1 Design goals

Designing of the PPSN system started with some essential design goals in mind, which could potentially make the system robust and hard to crack under imminent privacy threat or attacks. These design goals are derived from a thorough study of GDPR (outlined in §2.2.2), privacy (discussed in §2.2.1) and meta-data leakage while transferring data through insecure channel in the context of SN—dicussed in §2.1.2. This section will outline the design goals below, and it will explain how these design goals are related and essential for the proposed PPSN system to make the legacy SN system private and provide GDPR compliance.

Proposed PPSN system design should be able to:

1. Hide user's sensitive information.
2. Hide end-to-end communication tracing under pervasive network monitoring.
3. Anonymize requests and responses between servers and clients.
4. Protect users from their meta-data leakage.
5. Dismantle user's data in an occasion of a severe data breach.
6. Opt-out users from PPSN services even when the users are offline.

These design goals are targeted to give users the maximum advantages and enable user autonomy thereby. The benefit of these design goals will potentially impact SN system positively since more and more users will likely to participate in strengthening the nudging system as their privacy has been preserved. Finally, these design goals will make the SN system (§2.1.2 GDPR compliant and will protect the user data thereby.

4.2 Adversary model

This section outlines the adversary assumption for PPSN system. It is assumed that the imaginary adversary controls all but one server in the PPSN server array. The imaginary adversary also controls an arbitrary number of clients and can monitor, block, delay, or inject traffic on any network link. At any given time, we are assuming that we have honest clients behaving bug-free and honestly and honest servers behaving honestly and bug-free. In addition to that, the PPSN servers are assumed to be free from data leakage through side channels. For honest clients who are using smart nudging and the communication between smart nudging servers and the honest clients is protected by PPSN system. The level of protection is outlined in Chapter 4. Communication happens through multiple rounds, and it is assumed that an adversary can temper and interfere with the PPSN system in multiple rounds.

It is also assumed that the cryptographic operations, key-exchange mechanisms and hashing are happening as intended. In any circumstances, private-keys will remain private for each entity (i.e., server, client). Public-keys among users and smart nudging server is known to have the communication. Users (i.e., clients) and PPSN system (i.e., servers) know each other public key prior to communication. Clients are basically mobile phones apps on a smartphone which are shipped with default endpoints and server public-keys. A separate mechanism is also present to handle the public-key discovery[1]. It is also assumed that all server handles request properly and do not bypass any request. Even though bypassing request will be protected by mix-net. Though, any server can employ a denial-of-service attack on PPSN as per the previous assumptions made in this section. However, one server will always remain honest to have the integrity of PPSN system. Any strong adversary beyond this description can cause serious attacks—running a modified version of PPSN server and stop encryption using clients public key when all servers are compromised will reveal the actual data. However, the data is safe as long as one server is honest and faithfully do what it meant to be doing. Smart nudging backend server can be compromised and will not impact the privacy of the system as long as the code base doing what it was intended to do (i.e., using message-box to process the data and encrypting the data accordingly). Finally, another assumption is the decryption only happens in-memory and decrypted data stream discarded as soon as the process finished or discarded the event of the process crash. It is worth mentioning that only encrypted data is stored in non-volatile storage forms. It is not hidden from the adversaries that the users are connected to the

[1] Public-key discovery is a vital part of the PPSN system and needs an extensive study to address potential pitfalls of public-key discovery. In this thesis, it is assumed that the application will be shipped with the necessary keys, and they will be updated in a timely manner with a valid signature

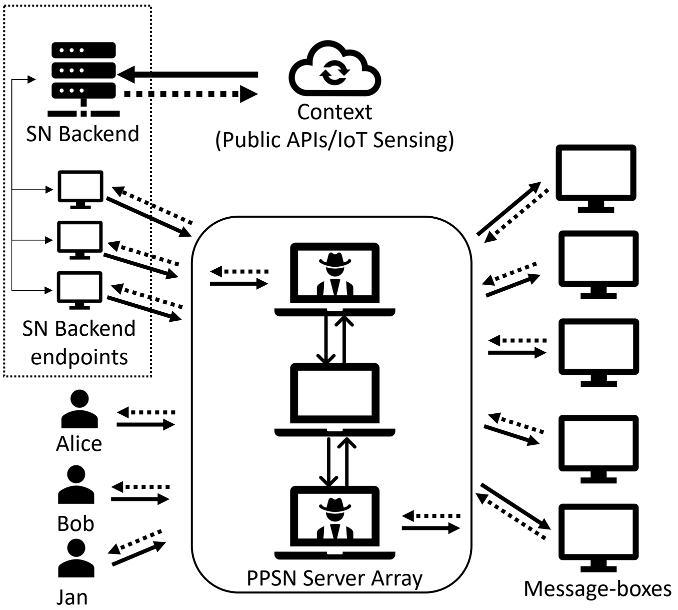


Figure 4.1: High-level overview of the Privacy-preserving Smart Nudging system (PPSN).

PPSN system. However, instead of intermittent connection (i.e., connection on demand), PPSN system uses a persistent connection between the users and the system. As a result, adversaries can not guess the whole story by speculating connection pattern. Nevertheless, it is assumed that the adversaries can see the initial connection request and the final termination of the connection, but nothing in between.

4.3 Proposed system architecture

In chapter 3, the privacy issues of SN architecture has been discussed, and it is found that there are two major information categories. One of them is public, which related to a broader audience and does not expose any particular user. The other one is private information which can expose a particular user. Contextual information such as IoT sensing for the environment, public APIs for transportation are considered as public information, and they are not mean to be protected by the PPSN system. On the other hand, user tracking data, user backend processing, and inform and nudge schematics are counted as private information and need to be protected. In figure 4.1, it is observed that the source of environmental context and public API is disjointed.

PPSN has a "Thick client" [79] and "Thin server" [79] architecture. Meaning sense, analyse, inform and nudge will be prioratised and sandboxed on the client for processing [3]. Whatever information is not processable on the client side will be handled and processed on smart nudging server with the help of PPSN system. Any process-heavy task and processing and constructing user profiling from substantial data will also be done in our PPSN servers since our clients are basically smartphones and have finite battery power. With that in mind, PPSN system consists of a single chain of servers. Clients (i.e., users) are connected to the first server on the chain. Then the first server is connected to the second server, the second server connected to the third server and $(N-1)th$ server connected to N^{th} server. All clients will use the same chain, and they will also know the server's public keys. Similarly, servers will know each other's public keys and the client's public keys beforehand. In PPSN, response and requests are performed in rounds, and message lengths and size are identical to reduce variable exposures for adversaries (§4.2). All the protocols of PPSN communicate through message-box—where clients deposits messages and pick up messages—smart nudging server also acts like a client and deposits message and pickup messages from a message-box. Data processors and data providers (i.e., SN system) leave messages in virtual locations. Data processors and data providers will be able to participate in data communication rounds since finishing data processing within a small round time will be difficult for process-heavy jobs. Data process will be happened in sessions and destroyed after the result is encrypted and enveloped with the client's public key in SN servers.

In PPSN, there are four critical zones depicted in figure 4.1. They are client zone, SN backend zone, context zone, and obscure zone. All the user lives in the client zone and SN backend zone in a sense, a big static client who is responsible for data processing and data providing—SN system. Context zone handles the public APIs and heterogenous IoT sensing and merging them into a general format and prepare the data for data processors and data providers in SN backend. Context zone is directly connected to the outside world of IoT sensing and Public API, and one of the main tasks of context zone is to prepare the data to be dispatched on demand—in a pub-sub fashion.

4.3.1 Protocol schematics

PPSN's two main protocol to complete sense, analyse, inform and nudge between the users and Smart Nudging (SN) backend is the "communication protocol" and "commencing protocol". Both of the protocols communicate through message-box. Message-boxes are virtual locations on PPSN server array—located on the last server in the chain. SN backend and client agree on a message-box to communicate in a round and SN backend deposits messages in message-boxes and clients (i.e., users) picks the messages up. It can also happen in the other way around where client deposits messages and SN backend picks up messages from message-box. However, the total accesses counts of the message-box for both the client and SN backend will be identical each time to reduce one more variable. For example, in one round of data exchange between the clients and SN backend, first the client and SN backend agree on a randomly chosen message-box through commencing protocol. After that, in a communication protocol, two endpoints—SN backend and users—can deposit and retrieve data from message-box. These message-boxes are 128-bit IDs, and honest client should never collide on the same message-box. In our PPSN system, each user is identified by their public-key. As part of the commencing protocol, each user is assigned to new message-box in every round to hold communication with SN backend. All the message-boxes get destroyed as the round comes to an end. Each users and SN backends periodically polls data and leaves data in the message box.

Cover traffics are added to obscure the message-box access pattern. In addition to that, all message-box are ephemeral meaning they are persisting only within a particular round. When a new round begins, new message-boxes are also created. So for each round of communication PPSN get new set of message-boxes to communicate through. The first server on the chain on PPSN server array is responsible for commencing a round. It waits for a certain time (i.e., 10s round time) to get requests and response for a certain message-box. Within that time frame, the server collects all the request from the clients and deposit messages in message-boxes and retrieve messages from message-boxes through the chain of servers. It is not possible to access that same message-box after the round is over. As a result, if a client goes offline, then eventually it losses the ability to send receive messages for that specific round. These issues are handled by retransmission of the messages in the next rounds. This round-based communication in PPSN makes it difficult for adversaries to relate message-box access over time as the communication protocol chooses different message-boxes in different rounds. Following sections will outline the two major protocols of PPSN system.

Commencing protocol

Commencing protocol is responsible for initiating the communication. In other words, before any communication between users and SN backend, they need to go through this protocol. Commencing protocol basically exchange the message box's pseudo-random string id between users and SN backend. Every round of communication first goes through this commencing in order to have the message-box id.

As discussed in §2.3.7—user uses PPSN system's mix-net public key and put user's public key inside, indicating that this user is commencing for communication. This message travels through the PPSN system and ends up in a message box with the user's public key and encrypted response address. This encrypted message is a layer of encryption where every server in the PPSN array removes each layer of encryption.

While each user ends up in a message-box, SN backend's random connection—which is acting as a client—picks up messages from message-box. Two important things happen here. One, the virtual endpoint of SN backend gets the message-box id and user's public key. Two, response for the user is constructed with the message box id. After that, each user in each message box gets their message-box id. This message box id is ephemeral—meaning it is round specific. Message boxes are destroyed after each round and commencing protocol repeats the process. Therefore, commencing protocol is used in the PPSN system to let the user and SN backend to know their communication point for each round.

Communication protocol

The communication protocol is responsible for all the communications that are happening between users and SN backend. After going through the commencing protocol, user and SN backend endpoint agree on a message-box. For a limited time(e.g., 10 seconds) all the communication between users and SN backend happens through that specific message-box given that the integrity of the other measures—shuffling requests and responses and encryptions of messages—holds. All communication is identified using the user's public keys.

4.3.2 SN backend endpoints

Counter-intuitively, in PPSN system, SN backend[2] acts as a client for PPSN server array—performing requests and responses for each round. To handle all the request and responses from users SN backend needs have larger bandwidth. For a single connection between SN backend and PPSN reveals meta-data about SN backend—because of it is increased size of request and response to handle all the users. Adversaries will be clearly able to see the difference between message sizes and the frequency of request-response between users and SN

[2] SN backend should not be confused with legacy SN system. SN backend is a superset and modified version of the legacy SN system, which makes the legacy SN system compatible with PPSN system and preserves user privacy thereby.

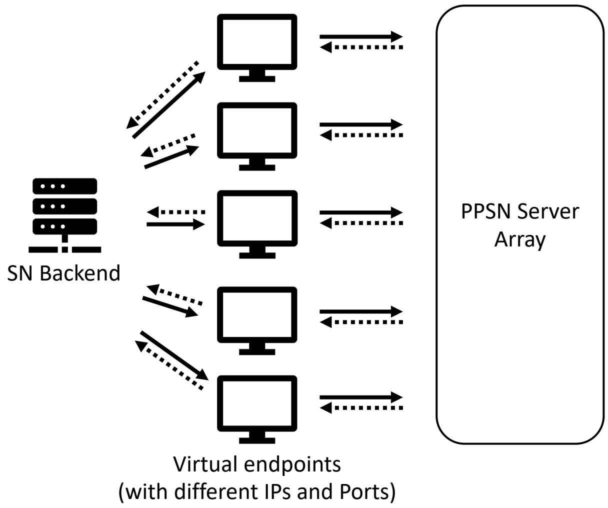


Figure 4.2: A High-level overview of SN backend's virtual endpoints to obfuscate and maintain constant message size of requests and response from SN Backend.

backend. To hide this variable SN backend implements virtual endpoints and identical message sizes and identical request-response frequencies. Virtual endpoints are achieved User Datagram Protocol (UDP) sockets which forwards the request and response to SN internal endpoint. There is a limit of the number of UDP sockets in a system (i.e., 65535). Since each socket is serving each user in a single round of communication, the maximum number of sockets can be easily reached when the number of users in the PPSN system surpasses 65535. As per our assumption, the number of users for the PPSN system and nudging can reach millions. SN backend addresses this problem using virtualization to produce more endpoints of SN backend, and they will stream back the requests and responses to SN internal primary endpoint. Figure 4.2 depicts how SN backend manages endpoints and behave like any other user entity in PPSN system.

4.3.3 Key-exchange

Key-exchange in PPSN system is a critical phase of the whole proposed system. In this phase, user's clients and SN backend comprehend the PPSN system's mix-net and exchange necessary keys to take part in commencing protocol and conversation protocol—discussed in §4.3.1.

There are several ways to manage the public and private keys in PPSN system. One way is to generate and propagate varified keys manually or through application update. Another option could be a local copy of a public database of keys (e.g., PGP key server) [80]. However, updating keys with application update could risk the PPSN system and make the PPSN system's security dependent on external servers (e.g., google play store server[81], app store server[82])

4.3.4 One round of communication

Each round provides users and SN backend to communicate with each other. They communicate through message-box—user leaves messages for SN backend, and SN backend picks up messages for those users from a particular message-box. Then, the user picks up messages from the specific message-box.

In order to complete a successful round of conversation, first, key-exchange happens. Key exchange happens only once for many rounds. Key-exchange is basically for letting the users and SN backend to know about the mix-net encryption and enveloping system. Once, key exchange is done, users and SN backend do not need to repeat it for every round.

Users initiate commencing protocol every round to bind the message box with their client and SN backend endpoints. Each user binds with each endpoint of SN backend—§4.3.2 outlines the endpoints of SN backend. In other words, users leave their public key and encrypted return address in the newly created empty message-boxes. SN backend endpoints randomly pick up these messages from message-box and leave encrypted return address and its public key without any order. Users and SN backend endpoints they both receive necessary parameters for starting communication protocol.

The actual communication and data exchange between users and SN backend happens here in the communication protocol. Communication protocol gets a limited time to perform several message exchange. In communication protocol, SN backend can leave request or response in the form of message in a message box. Users then pick them up periodically within the round. Users can also perform request and response just like SN backend. Request response happens asynchronously—a message can contain both the request and response. What is visible through the network is those encrypted messages that travel between the SN backend and users through PPSN server array and message-box.

Both the commencing protocol and communication protocol has a time out and specific round time (e.g., 10 seconds in our case). The first server in PPSN array—which connects directly to the users and SN backend—responsible for collecting messages for 10 seconds then forwards it to the next server in the

chain. Next server shuffles the messages, remove its encryption layer and adds necessary noises and forwards the messages to the next server in the chain. This process repeats until it reaches to the last server where message-boxes lives. The last server routes the message by evaluating the message-box id. So the complete round of communication is as follows for a user, U and the last server in the PPSN server array, L:
Commencing protocol,
The message from $U \longrightarrow L$:

$$\begin{aligned} K_m(R1, K_u(R0, K_sn(R3, message), K_m(S1, U), K_x), L) \longrightarrow \\ K_u(R0, K_sn(R3, message), K_m(S1, U), K_x) \end{aligned} \tag{4.1}$$

A random SN backend end-point picks up the message (i.e., message-box id and users public-key) and leaves another message in the same message-box (i.e., encrypted SN backend end-point source information and SN backend's public-key)

Reply message from $L \longrightarrow U$:

$$K_m(S1, U), K_x(S0, response) \longrightarrow U, S1(K_x(S0, response)) \tag{4.2}$$

Where: K_u = U's public key, K_m = the PPSN server arrays's public key.

Communication protocol,
Communication protocol is almost the same as commencing protocol that has been discussed earlier. However, it contains a enveloped data part, E' which PPSN server array does not touch and does not know anything about. Here, PPSN server array only acts like a message forwarder for E'.

4.3.5 Content hiding

One of the goals of PPSN system is to encrypt as much data as possible and whatever cannot be encrypted—metadata—will be hidden by adding noise discussed in §4.3.7. Data enveloping—discussed in §2.3.5—has been used throughout the PPSN system as a means of content/data hiding. Data enveloping is used to maximize the performance for encryption and exchanging data through an insecure channel by combining symmetric and asymmetric encryption scheme—discussed in §2.3.2, §2.3.2. Two major algorithms RSA (discussed in §2.3.3) as asymmetric encryption and AES (discussed in §2.3.4) as symmetric encryption has been used in PPSN system's data enveloping. When data envelop travels through the PPSN system's mixnet server array, only the symmetric key has layered encryption as encrypting large data with RSA is expensive—discussed in §2.3.3. Actual data is decrypted inside SN backend in-memory, and before finishing up the process, the data is encrypted back with a one-time random symmetric key. Then that symmetric key is encrypted with the corresponding user's public key. Finally, SN backend forgets the symmetric key and store the data in the database, which makes the data completely unreadable even for the SN backend system. Each time SN backend wants to read that data, it needs to go through the commencing and communication protocol to get back the decrypted symmetric key from the owner as the symmetric key was decrypted using owner's public key.

4.3.6 Untraceable end-to-end communication

Any observable communication between clients and servers leaks metadata. Tracing end to end communication is easy and reveals a story about the situation of the client and the server over time—discussed in §2.2. PPSN system addresses this issue by using mixnet—discussed in §2.3.7, which is an array of a handful of servers creating encryption layers and shuffling request and responses. As a result, PPSN system breaks the link between senders and receivers—making the communications between SN backend and Users untraceable. In addition to that, each server in PPSN system's array only knows about the message it receives from and the destination—discussed in §2.3.7.

4.3.7 Hiding meta-data

§4.3.5, §4.3.6 outlines how PPSN system hides the content by encryption and make end-to-end communication untraceable by using mixnet—makes PPSN system already secure against weaker adversaries. However, at any given time, when an adversary looks into the communication traffic, he knows that these are the real users. Since strong adversaries can take users offline, and online long term attacks could be possible by observing the behaviour of malicious users or servers. Even that adversary cannot read the data; it will reveal a lot about the behaviour of the servers and clients. To take the privacy of PPSN system even further for strong adversaries, the idea of noise comes to play. The noise is fake requests and response messages for commencing protocol and communication protocol to hide the real users among fake ones. Each server in PPSN system's server array gathers requests for a specified round time—discussed in §4.3.1—then adds a predetermined amount of noise(i.e., random fake messages). Then the server shuffles all the requests, including the noises. Therefore, adversaries probability of being certain about any messages—belongs to a real user or SN backend—reduces. This technique of adding noises to the system makes PPSN system differentially private—discussed in §2.3.6. In addition to that, adding noise also protects users from being identified when there are too many malicious users in the PPSN system. Moreover, noises are also beneficial when there are fewer users using the PPSN system. Depending on the number of noise, the more user who is using the PPSN system, the more it increases the probability of being a real user. For keeping things less complicated, PPSN system used a static number of noise (i.e., 300,000 of fake messages for both the commencing and communication protocol).

4.3.8 PPSN system as middleware

PPSN system was designed with loose coupling in mind. Therefore, PPSN system is decoupled from SN backend, which makes PPSN system flexible enough to be used as middleware for providing untraceable communication and privacy for any service like SN system. To make PPSN system compatible with third party systems and clients, they need to follow the API specification of PPSN system—protocol schematics—to interact with PPSN system through key-exchange, commencing protocol and communication protocol—discussed in §4.3.1. This ability of decoupling gives PPSN system a great advantage to use with a legacy system like SN system with little changes.

4.3.9 Data storage and processing techniques

Data storage and processing techniques are vital for preventing content theft and making an almost trust-less system design from the core of the PPSN system. As outlined in §4.3.8, PPSN system behaves like middleware between the users and SN backend. Therefore, PPSN system does not store or cache any data or messages. In addition to that, PPSN system forgets almost everything about what happened previously, but the necessary keys for cryptographic operations. In other words, each round of communication resets the server state in terms of messages.

Data storage takes place inside the user's client and SN backend. However, no plain data is allowed to be stored directly in the database, rather the encrypted form of that data. Users client maintain its own database and store the data in an encrypted form. Each user has only the data related to that specific user. Therefore, user's clients can use conventional data storage technique inside their smartphone or other devices that the user uses.

However, in SN backend, a different approach of storing data into a database is needed for long term purposes. Two of the aspects are addressed; one, users should be able to opt-out from smart nudging without talking to SN backend; two, in case of content theft, and data breach SN backend should be able to provide privacy for users data. Hence, SN backend encrypts the user data in-memory and encrypt the symmetric key with users specific public-key and finally forgets the symmetric key. Finally, the encrypted cypher text is sored into the database. Therefore, this technique leaves SN backend without any clue about the data. In order to use the data, SN backend must communicate with the user to get the actual key to decrypt the data. Also in SN backend, encryption and decryption happen in-memory. After each process inside SN backend, the encrypted result is stored in a database. This method of encryption follows data enveloping technique discussed in §2.3.5. However, here the envelop never leaves SN backend, but the encrypted symmetric key with a particular user's public key which is used to create the envelope. If data, D' is received, and SN backend performs M algorithm on that data, then the data storage takes place as follows:

SN backend receives the encrypted data, D' from a user, U and decrypts it.

$$\begin{aligned} D' + SN_{priv} &\longrightarrow D \\ M(D) &\longrightarrow D_1 \\ D_1 + S0 + SN_s &\longrightarrow D_1' \\ SN_s + U_{pub} &\longrightarrow SN_s' \end{aligned} \tag{4.3}$$

Where SN_s = Random onetime symmetric key produced by SN backend, M = Random data processing algorithm, N_{priv} = Private key of SN backend, U_{pub} = Public key of the user, U, and $S0$ = Random salt produced by SN backend.

SN_s is destroyed as soon as SN_s' is produced. In the meantime, D_1' along with SN_s' is ready to store in SN backend database. Its worth pointing out that SN_s' will not be able to decrypt the cypher text, D_1' as SN_s is destroyed.

Decryption of D_1' involves reconstructing of the destroyed key, SN_s. This decryption process also happens in-memory for running sets of algorithms on that specific data and follows the same principles that have been discussed above. In addition to that, it requires further communication with that specific User, U through PPSN Sytem's protocol to have SN_s' decrypted (i.e., SN_s)[3]. After performing the operations, when the process is finished, it repeats the same process of storing data into SN backend database—encrypts the new derived data and stores it in the database.

[3] All the messages and data that have been received by SN backend is encrypted with SN_{pub} by the original sender to avoid content theft.

5

Implementation

This chapter outlines the barebone prototype implementation of the PPSN system[1]. This proof of concept implementation of the PPSN system design (discussed in §4.3) was developed in Nodejs (§2.4.1)—consists of around 1003 lines of code. Several shell scripts have also been written to run the simulation instantly for a handful of users and servers in server array of the PPSN system.

Asymmetric encryption algorithm, RSA (§2.3.3) and symmetric encryption algorithm, AES (§2.3.4) are one of the fundamental tools that have been used to implement the data-enveloping (§2.3.5), which is the basis of content-hiding in the PPSN system (§4.3.5). Likewise, both the encryption algorithm RSA and AES have also been used for layering the messages to implement untraceable end-to-end communication (§4.3.6) in the PPSN system. The crypto module of Nodejs v14 has been used to implement all the cryptographic functionality—OpenSSL's hash, Hash-based Message Authentication Code (HMAC), cipher, decipher, sign, and verify functions—throughout the implementation of the PPSN system. Specifically, AES-256-CTR as symmetric encryption and RSA_PKCS1 as asymmetric encryption have been used to implement data enveloping and layering of the messages in the PPSN system. RSA key pairs are generated using OpenSSL/ssh-keygen, and key-exchange is done manually before starting the

[1] The source code of the barebone implementation of the PPSN system is publicly available at "`https://github.com/mhusme/ppsn`" under GNU LESSER GENERAL PUBLIC LICENSE (LGPL)

PPSN system.

TCP/IP Stream Sockets (§2.4.1) have been used to build up the connection between the applications (i.e., clients and servers). ZeroMQ (§2.4.1) has been used to implement and simulate the publish-subscriber model between SN backend and the context (i.e., public APIs).

Protocol buffers (§2.4.2) have been used as the standard data format between the servers among PPSN server array and also between the users and the PPSN system—providing an extra layer of security by nature as the schema has to be known beforehand to read the binary data stream. Constant noise generation has been implemented programmatically by randomizing the message and then convert it back to protocol buffers and finally adding all the generated noise into the real requests. Then shuffling of the requests takes place and finally, the server forwards all the requests randomly.

This prototype is a proof of concept of the PPSN system. It implements the minimum viable applications to demonstrate one round of end-to-end communication between users and SN backend through PPSN system. This implementation followed the design principles of the PPSN system discussed in chapter 4 (e.g., the first server of the PPSN server array takes all the requests, then shuffles it and forward to the right adjacent server. Then the message lives in a message-box in the last server to be picked up by SN backend. Subsequently, SN backend leaves a message in the message box, and the user picks up the message.).

Even though this implementation followed the PPSN design—discussed in §4.3—the prototype implementation and the design has some differences. The two protocols, commencing protocol and communication protocol has implemented as one protocol as they are very similar, but only their message size is different. The complete data storage solution with a database has not been implemented. However, sufficient implementation has been done to prove the data storage technique (i.e., encrypting data with a random symmetric key, then encrypt the symmetric key with users public key and finally forget that symmetric key). Conclusively, this implementation does only one round of communication. Therefore, to run the application in multiple rounds, it needs to be executed manually for each round of communication.

Virtualization of SN backend was discussed in §4.3.2 to prevent the running out of TCP connection which is not implemented in this barebone implementation of the PPSN system. Hence, a handful of SN Backend instances has been implemented to handle all the user requests.

Needless to say, this implementation only serves the purpose of proof of concept and benchmarking of the PPSN system, users, and SN backend—this implementation is by no means intended to use in a real-world production environment. In addition to that, this implementation focuses on the proof of concept of untraceable communication and content hiding for SN— it does not implement the whole nudging system by any means (e.g., processing user data, data processing in SN backend, key-exchange mechanism). More precisely, users and SN-backend application exchange dummy nudging messages among them through the PPSN system by following its design schematics. More work is needed to deploy this barebone implementation of the PPSN system for real-world uses.

6

Evaluation

This chapter qualitatively evaluates the PPSN system's privacy and quantitively evaluates the proof of concept implementation of the PPSN system. It also outlines the experimental setup and the method of evaluation. In addition to that, this evaluation answers the following question quantitively:

Q1 – *Does the PPSN system provide privacy and improve the SN (§2.1.2) system? (§6.3)*

Q2 – *Is the PPSN system feasible and acceptable in terms of performance? (§6.4)*

6.1 Method of evaluation

This section outlines and discusses the methods of evaluation that has been used to evaluate the privacy and the performance of the barebone PPSN system.

6.1.1 System latency

End-to-end latency is one of the crucial performance metrics that have been used to evaluate the performance of the PPSN system. The latency of a system is given by the time between dispatching a request and receiving a response.

Therefore, the latency of a system is the inverse of the throughput—the number of processed request/response per unit time—of a system. Furthermore, latency is referred to the waiting time of a client in a client-server architecture to do a unit work.

Latency is usually measured in seconds. Hence, the common way of measuring latency is to record the starting time of a process and record the time after the process is finished. The time difference between the finishing time and the starting time gives us the latency in a unit of time.

Since the PPSN system involves a lot of independent processes and servers, a different approach has been made to measure the latency of the PPSN system. In the PPSN system, end-to-end latency has been measured by subtracting receive time and request time. The sender creates a message with its current system timestamp inside before starting any operation. The receiver receives the message through the PPSN system. When the receiver receives the request, it has the request timestamp and also the current system timestamp as receiving timestamp. The difference between receiving timestamp and requesting timestamp gives the end-to-end latency of the PPSN system.

Let Transmitting message, T_x, receiving message, R_x and time function t. Then the latency L is as follows:

$$L = t(R_x) - t(T_x) \quad where,\ t(R_x) \geq t(T_x)\ and\ L \geq 0 \tag{6.1}$$

More precisely, in the PPSN system a user builds a message with the current system timestamp before starting the actual process (i.e., encryption of the message and layering of the message) and sends it to the SN backend through PPSN system server array.

Then, arithmetic mean is used to increase the measurement precision of the overall end-to-end latency of the PPSN system. The total amount of latency has been calculated for a number of requests in that round, then the arithmetic mean is derived from dividing the total number latency by the total number of processed requests.

For n number of requests, the average mean latency, $\overline{L}$ of the PPSN system is as follows:

$$\overline{L} = \frac{1}{n}\sum_{i=1}^{n} L_i \tag{6.2}$$

As measuring latency involves measuring system time, end-to-end latency subjects to clock synchronization among the servers and clients, which is one of the challenges to overcome to measure the end-to-end latency precisely.

However, the barebone implementation of the PPSN system has been tested locally for performance measuring, and the time synchronization issue did not arise.

6.1.2 System load average

System load is a measure of how much computational work needs to be done in a given time [83]. Thus, the load average is the average system load over a timeframe. In UNIX computing, system load comes in the form of three numbers. If we read the numbers from left to right, then it tells us the system's overall average load for all the cores and threads during the last one-, five-, and fifteen-minute periods. All UNIX like system generates dimensionless metric of three "load average" numbers in the kernel. It can be easily read from shell commands (i.e., uptime, top) or the file /proc/loadavg [83]. An idle system has load number 0, excluding the idle process, and each process, waiting or using the CPU increments the load number by one. On the other hand, a terminating process decrements the load number by one. Only the processes in the running or runnable states are counted in most of the UNIX or UNIX like systems. However, blocked processes due to a busy or stalled I/O system are also counted, which can lead to a different result if there are too many processes in uninterruptible sleep states [83].

Listing 6.1: Current system load average from a Unix shell by running the uptime command

```
$ uptime
22:14:08 up 10:07,1 user , load average : 1.73 , 0.60 , 7.98
```

If we want to interpret the UNIX system load average in a single CPU system then, the numbers "1.73 0.60 7.98" of listing: 6.1 can be interpreted as:

- The system was overloaded by 73% on average during the last minutes. In other words, on average 0.73 processes had to wait for being processed by a single CPU system.

- For the last five minutes, the number 0.60 means the system was idling 40% of the time on average.

- Similarly, during the last 15 minutes, the system was overloaded by 698%—meaning 6.98 processes had to wait for their turn to be processed out of 7.98 processes in a single CPU system.

These days most of the system has two or more CPUs. To calculate the system load avg per CPU, we need to divide the load avg by the number of processors that are available on the system.

CPU load vs CPU utilization

The study of different load indices by Ferrari et al. [84] showed that CPU load information based on process queue length is much better to have an idea about the system's overall load. On the other hand, CPU utilization only tells us the CPU consumption. We have no idea how the system is overloaded once the CPU reaches its maximum utilization [84]. System load information is more useful for load balancing and measures how the overall system will behave under a particular setup (e.g., in our case system running PPSN).

Listing 6.2: SystemInfo.h for calculating CPU Load

```
#ifndef _SYSTEMINFO_H
#define _SYSTEMINFO_H

#include <string>
#include <fstream>
#include <iostream>
#include <vector>

using namespace std;

class SystemInfo
{
public:
  SystemInfo();
  string get_cpuLoad();

private:
  float cpuLoad;
  void fetch_cpuLoad_from_system();
  size_t get_logical_cores();

  float calculate_cpu_usage_percentage
              (float raw_cpu_usage);

  const vector<string> explode
         (const string& s, const char& c);
};

#endif
```

PPSN system load average

To calculate our PPSN system load average, we have made a C++ application to measure system load average, including the OS and streamlined required processes. For easy integration, we have made a custom header "SystemInfo.h" (Listing 6.2) which provides us with necessary methods to get the CPU Load in percentage. "SystemInfo.cpp" implements all the methods that we have in "SystemInfo.h". To get the CPU load, first, we read the kernel scheduling entity file "/proc/loadavg" to read UNIX load average numbers. After that, we read the total number of available system cores, which will be our divisor for getting the per core system load average. To get the number of total logical cores, we are reading the file "/proc/cpuinfo" and extract the number of siblings of the system. Then, we will calculate the percentage and return it from the public function get_cpuLoad().

Listing 6.3: Calculate CPU usage method logic

```
float SystemInfo::
calculate_cpu_usage_percentage(float raw_cpu_usage){

  float calculated_cpu_usage =  100 * (raw_cpu_usage
        / static_cast<float>(get_logical_cores()));

  return calculated_cpu_usage;
}
```

Listing 6.3 shows the logic and calculation process of the "calculate_cpu_usage_percentage" function body which sets the cpuLoad property. Finally, we return the cpuLoad property from "get_cpuLoad" public method.

6.1.3 Noise performance

Noise performance in the PPSN system is defined by the uncertainty of observation, whether it is correct or not. The more uncertain an observation gets, the system achieves the more noise performance. Noise performance is given by a probability function of the number of total active users and the amount of noise in the PPSN system. In PPSN system's noise performance context, if an adversary wants to observe a message and wants to know if that message is from either a real user or fake noise. Then the probability of that message from a real user is the factor of the real active users and the inverse of total active users, including noise (given that all the active users and generated noises participate in that traffic with a single message). Therefore, the metrics of the noise performance of the PPSN system—probability of an observation o being a real user's message is:

$$Pr[o \mid \mathbb{R}u] = \frac{\sum_{i=1}^{n} \mathbb{R}u_i}{(\sum_{i=1}^{n} \mathbb{R}u_i) + \eta} \quad if\ Pr[o \mid \mathbb{R}u] \in \mathbb{R} : 0 < Pr[o \mid \mathbb{R}u] < 1 \tag{6.3}$$

Where, real user's message is $\mathbb{R}u$, total number of active real users in the PPSN system is n, and total number of noise in the PPSN system is η

6.2 Experimental setup

Series of experiments have been done to evaluate the performance of the PPSN system. A single machine is used to perform all the experiments and run the barebone implementation of the PPSN system (discussed in chapter 5). The specification of that experimental machine is as follows: Processor: 8th Generation Intel® Core™ i7 Processor @ 1.80GHz (4.60GHz Turbo, four cores and eight threads) [85], Memory: 16.0GB of DDR4-2400 RAM (Max memory bandwidth 37.5 GB/s), and Graphics: Mesa Intel® UHD Graphics 620 (WHL GT2) @ 300-1200 MHz. The machine runs 64bit Linux operating system (Ubuntu v20.04.1 LTS, kernel version 5.4.0-48-generic), gcc v9.3.0, nodejs v10.19.0, ZeroMQ v3.

A single application runs to emulate user traffic, using multithreading and process forking. SN backend consists of a single application which emulates SN backend endpoints with multithreading and process forking. A handful of TCP/IP socket connections have been used to emulate user traffics and SN backend traffic to avoid running out of source TCP/IP sockets—meaning one single application emulates all the user traffic and multiple application emulates the SN backend endpoints. Three independent applications have been used to emulate The PPSN server array throughout the experiments where the number of users varies. The PPSN system performance with respect to the number of servers in the PPSN server array uses fifty thousand constant users. RSA key length 2048bit and AES key length 256bit have been used throughout all the experiments. The final message size around 5KB each. On average, for any given time, 10% of users communicated through the PPSN system. All the servers in the PPSN server array add noise, $\eta = 10,000$ each (higher amount of noise—Noise, $\eta = 500,000$—stalls the machine).

Experiments have been done for one round of end-to-end communication—

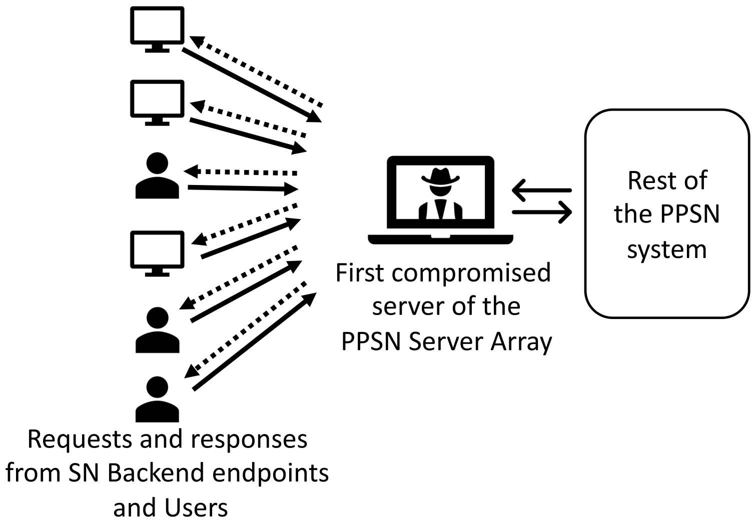

Rest of the PPSN system
First compromised server of the PPSN Server Array
Requests and responses from SN Backend endpoints and Users

Figure 6.1: An adversary can only see the incoming and outgoing encrypted messages if the first server (1^{st} server in the PPSN server array figure 4.1, from top to bottom) in the PPSN server array gets compromised. If at least one server in the PPSN server array is honest, an adversary should not be able to tell about the content of the messages and original destination of the messages.

message from a user to SN backend. A user puts the time in the message before starting the encryption process and sends the message through the PPSN server array. SN backend receives the message and derives the time difference—latency—after decrypting the message. Since every application is running on the same machine, time synchronization does not affect the precision of the latency. Each experiment runs for 1, 5 and 15 minutes. For each run, the total amount of latency and the total number of processed request has been calculated and then finally, average latency has been calculated. System load average for 1, 5, and 15 minutes have also been calculated using a C++ application. Lastly, each test has been performed five times and averaged to reduce the margin of error.

6.3 Privacy evaluation of the PPSN system

Evaluation of the privacy of a system is as difficult as formalizing the privacy of a system as the power of adversary is always unknown. Besides, over time technology improves and situation changes. Today's best privacy-practise might not be ideal tomorrow. With that said, this section qualitatively and quantitively evaluates the privacy achieved by the PPSN system with respect to the PPSN

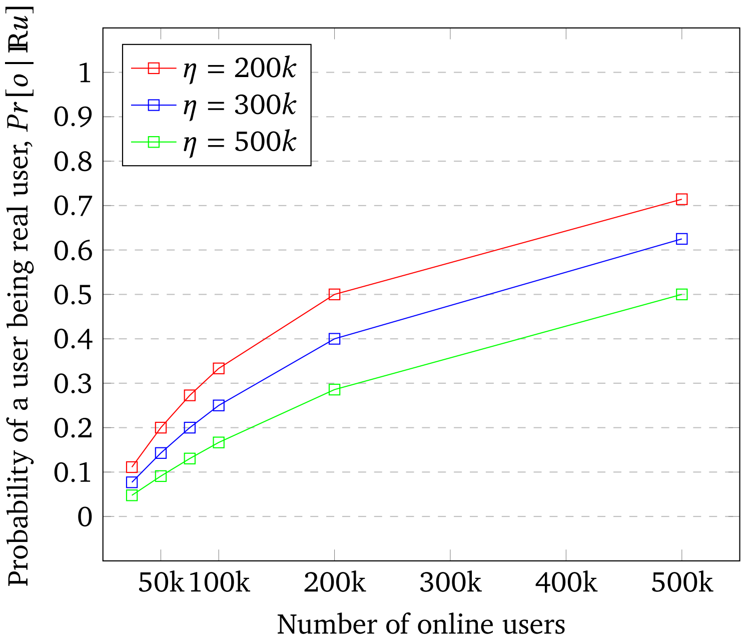


Figure 6.2: Identifiability of a real user in the PPSN system for a different amount of constant noise when varying the number of active users in the system.

system's design goals (§4.1) for the presumed adversary model (§4.2). Furthermore, this section also analyses whether the proposed system architecture (§4.3) can satisfy all the design goals (§4.1) or not.

The PPSN system architecture, depicted in figure 4.1 has two servers with blackhat on them to denote the compromised server. According to the threat model (§4.2) of the PPSN system, at least one server in the server array needs to be honest in order to have the integrity of the PPSN system.

Since the entry server is presumed to be compromised, an adversary can see all the requests and responses—illustrated in figure 6.1. However, an adversary neither know the final destination of the messages nor the content of the messages; it only knows the next hop for the message (i.e., the next server in the PPSN server array). Since there are several users, and SN backend endpoints are connected, it gets more challenging for the adversary to know who is communicating with whom. In addition to that, the presence of noise makes it even more challenging for an adversary even though most of the users are malicious or controlled by that adversary. An adversary on the entry server can identify the SN backend endpoints if SN backend uses the same host over and over again but it does not help as the message size for both the SN backend and users are the same. Therefore, users metadata is preserved and fulfilling the 4th design goal (§4.1) of the PPSN system. Moreover, the message content

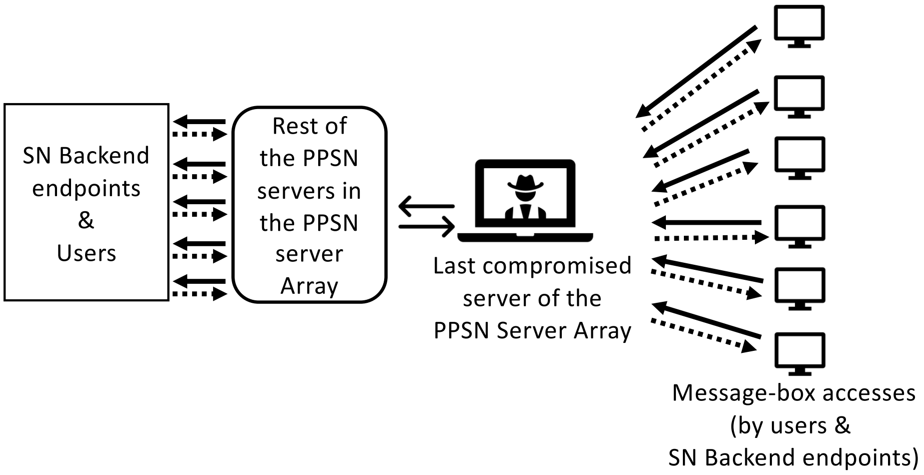

SN Backend endpoints & Users
Rest of the PPSN servers in the PPSN server Array
Last compromised server of the PPSN Server Array
Message-box accesses (by users & SN Backend endpoints)

Figure 6.3: An adversary can only see the message-box access counts if the last (3rd server from figure 4.1, from top to bottom) server in the PPSN server array gets compromised. If at least one server in the PPSN server array is honest, an adversary should not be able to tell about the order of the messages, a correlation between accessing the message-box, the origin of the messages and the content of the messages.

is encrypted, and user data is also hidden, satisfying the 1st design goal (§4.1) of the PPSN system.

Sensitive user data is also hidden inside of the SN backend by encrypting the data with a random symmetric key and then encrypt that symmetric key with user's public-key and finally forgetting the symmetric key. If SN backend gets compromised and if the adversary has no access to the code, then an adversary will not be able to see the user or the data which satisfies the 1st, 5th and 6th of PPSN system's design goals. Since users are identified by their public keys, and every operation in SN backend needs user validation, it becomes RSA and AES encryption breaking problem. In the PPSN system's case, that would take 300 trillion years for breaking the 2048bit RSA key, and it would take 2,117.8 trillion years to break the AES encryption §2.3.4. I addition to that, if an adversary has full control of the SN backend, it will still not be able to trace the user because of the PPSN system's server array. Because each server only knows the adjacent servers, adds noise, shuffles all the message, including noise before passing it on to the next server, which provides untraceable communication satisfying the 2nd and 3rd design goals.

Finally, in figure 4.1, it shows that the 3rd server is also compromised (a blackhat is on the third server in the PPSN system's server array). Figure 6.3, illustrates what an adversary can tell after compromising the last server on the PPSN server array. It gets the messages with all the layer removed and the destination message-box id. However, by the design principles of the PPSN system, the message is still encrypted and can only be decrypted by a valid addressee. In addition to that, the requests are shuffled, and number noises are added from the previous server. As long as the server in the middle is honest and does not reveal it is a shuffle, then the adversary in the 3rd server will not be able to read the content or correlate the message.

Noise is essential for anonymizing users metadata. It serves the purpose of making an adversary confused about a message whether if its from a real user or not. Noise performance is evaluated by the evaluation method discussed in 6.1. Figure 6.2 shows the confidence level of an adversary about an observation being a message from a real user. To illustrate it, if an adversary makes an observation o, then the probability of o being a real user is higher for a lesser amount of noise—meaningless privacy. A higher amount of noise gives a lower probability of o being real user which means higher privacy. Honest users can also be used as noise for a set of messages that an adversary tries to observe. However, for the simplicity of the calculation, it was not included in the noise performance function.

PPSN system also thought about dynamic random noise addition by each server for tighter privacy and metadata hiding. However, the asymptotic performance of the dynamic noise addition is impractical as it increases the time complexity for a substantial user base. For n users, and m servers in the PPSN server array, the asymptotic performance of the system in terms of noise, $T(n,m)$ is:

$$T(n,m) \in O(n \times 2^{m-1}) \quad \text{as } (n,m) \rightarrow \infty \tag{6.4}$$

Where each user requires its own SN backend endpoint and the entry server in the PPSN server array does not add any noise.

6.4 Performance of the PPSN system

In order to evaluate the PPSN system, several experimentations on the barebone implementation of the PPSN system (chapter 5) have been done to observe how the system behaves under different parameters (i.e., number of online users, number of servers in the PPSN server array). All the experimentation has been done using the experimental setup discussed in §6.2. The PPSN system is heavily dependent on asymmetric encryption which is CPU bound. Although adding noise increases the I/O operations between the servers, it is insignificant compared to the CPU cost. However, to keep the $Pr(\mathbb{R}u)$ function below 0.5, the PPSN system adds constant noise with respect to users. Therefore, bandwidth increases significantly for noise addition. Nevertheless, the barebone

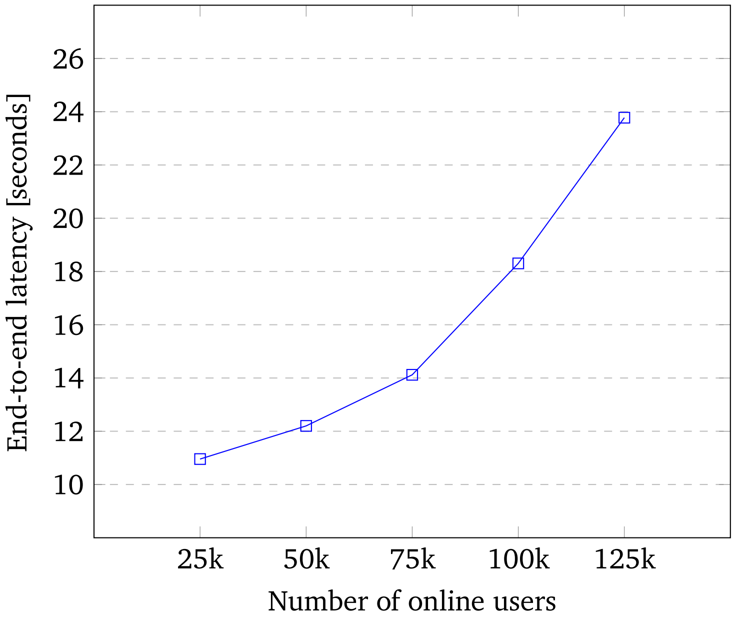


Figure 6.4: Performance of the PPSN system's end-to-end communication while varying the number of users with three servers in the PPSN server array and 10k noise.

implementation of the PPSN system uses a lower constant amount of noise for simplicity and some privacy edge.

End-to-end latency and system load average are the key performance metrics that have been used to evaluate the performance of the barebone implementation of the PPSN system. The latency of the PPSN system has been measured both for a variable number of users and a variable number of servers in the PPSN server array. Likewise, the system load average of the PPSN system has been measured both for a variable number of users and a variable number of servers in the PPSN server array. Method of evaluation discussed in §6.1.1, §6.1.2 has been used to measure the latency and system load average.

6.4.1 End-to-end latency of the PPSN system

The end-to-end latency of the PPSN system gives an idea about the practicality and feasibility of the system in the context of users and the number of servers in the PPSN server array. PPSN barebone implementation (Chapter 5) has been run on the experimental setup discussed in §6.2 to measure the latency of the system by using the method of evaluation discussed in §6.1.1. PPSN system's end-to-end latency is measured for one round of communication. That includes the symmetric encryption of the message, layered asymmetric encryption for

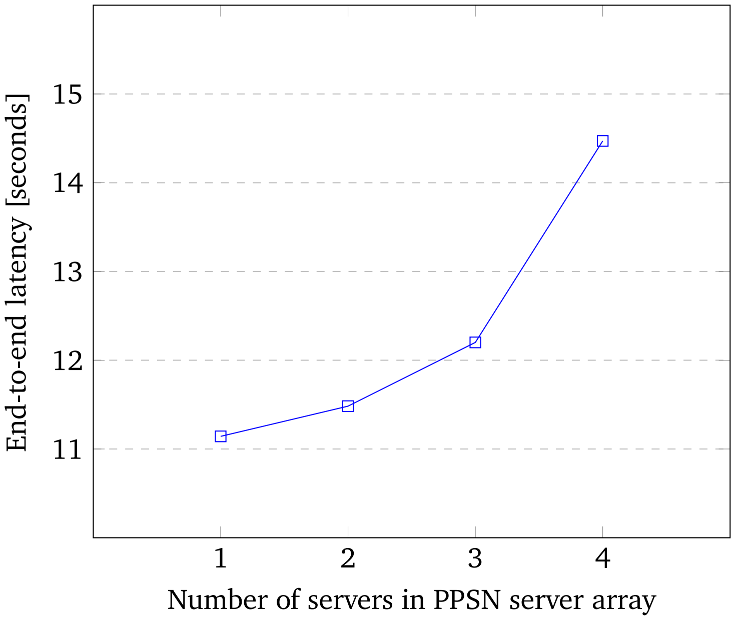


Figure 6.5: Performance of the PPSN system's end-to-end communication when varying the number of servers in the PPSN server array with 50k users and 10k noise.

preparing the message for the PPSN server array in client-side, and layered decryption of the message, adding noise and shuffling in the PPSN server array. In other words, latency is measured for a complete journey of a message from a user to the SN backend in the PPSN system.

In the PPSN system, the latency largely depends on the number of active users and the number of servers in the PPSN server array. Latency also largely depends on the amount of noise each server produce. However, the barebone implementation adds very little noise (i.e., $\eta = 10k$), and therefore the impact on server latency is negligible. Figure 6.4 plots the latency of the PPSN system in seconds with respect to the number of active users in the PPSN system. Likewise, PPSN system latency[1] has also been shown in figure 6.5 with respect to the number of servers in the PPSN server array. The following two paragraphs outline the results analysis of the test for the PPSN system latency in terms of users and the number of servers in the server array.

[1] The method of measuring the PPSN server latency is discussed in §6.1.1

Result and analysis

Figure 6.4 shows that the end-to-end latency of the PPSN system scaled linearly with the increase of the number of active users in the PPSN system where 10% of the users actively participated in the communication and three servers in the PPSN server arrray. Likewise, the end-to-end latency of the PPSN system also rises linearly with the increase of the number of servers used in the PPSN server array depicted in figure 6.5 where 10% of the users actively participated out of 50k active users.

For every 25k of user increase, the end-to-end latency of the PPSN system increased about 2 seconds, then it picks up more quickly (>4s) for 100K users and beyond (figure 6.4). On the other, the series addition of each server in the PPSN system's server array increases the end-to-end latency by a second then it picks up more quickly (>2s) for four servers or more in the PPSN system's server array. Lastly, both of them has an overhead of communication round—the waiting time of a server to collect requests for a round specific time (10s in this case). Figure 6.4 and figure 6.5 also shows that the 1-minute system latency is higher than 5 minutes and 15 minutes system latency. Longer executions (e.g., for 15 minutes) have low end-to-end latency, and shorter runs have a higher end-to-end latency—this phenomenon caused by the peak bursts of process forking and threads in the system. However, with the passage of time, the peak evens out and provides lower end-to-end latency for longer runs like 5 minutes and 15 minutes.

6.4.2 System load of the PPSN system

The system load of the PPSN system gives an idea about the computational power required to run the PPSN system in the context of the number of users and the number of servers in the PPSN server array. The method of evaluation discussed in §6.1.1, and the experimental setup discussed in §6.2 are used to test the system load for the prototype implementation (Chapter 5) by varying the number of users while keeping the number of servers in the PPSN system constant at three.It measures the load of end-to-end message transmission between the users and the SN backend endpoints for a given time. Therefore system load measurement includes the symmetric encryption of the message, layered asymmetric encryption for preparing the message for the PPSN server array in client-side, and layered decryption of the message, adding noise and shuffling in the PPSN server array.

In the PPSN system, end-to-end server latency is proportional to system load as both behave the nearly identical with the increase of the users or the addition of the new servers in the PPSN server array. System load also largely depends

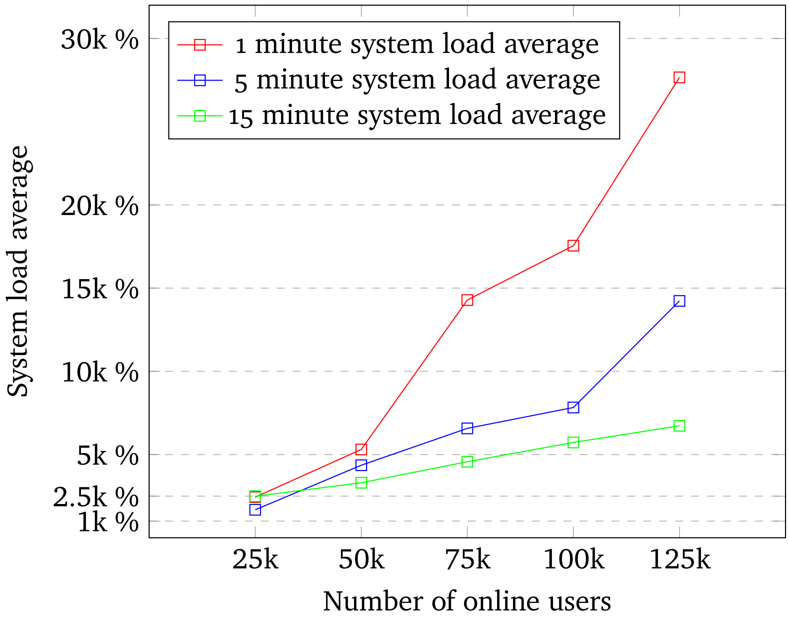


Figure 6.6: PPSN system's overall system load average when varying the number of users with three servers in the PPSN server array and 10k noise.

on the number of noise. For ease of the testing Noise, $\eta = 10k$ is used for both of them. System load with respect to the number of active users in the PPSN system is shown in figure 6.6. Likewise, figure 6.7 shows the PPSN system's load average[2] with respect to the number of servers in the PPSN server array. Results and analysis of the system load average for the PPSN system is as follows:

Result and analysis

System load average in percentage for end-to-end message communication has been shown in figure 6.6 for a variable number of active users and three servers in PPSN server array for 1 minute, 5 minutes and 15 minutes. System load increases as the number of active users in the system increase—where 10% users participated in the actual message transmission (figure 6.6). System load average for 1 minute fluctuates a bit more than the system load average for 5 minutes and 15 minutes (figure 6.6). In contrast to the system load average for 1 minute, the system load average for 15 minutes increases steadily with the number of users (figure 6.6). In addition to that, the system load average for 5 minutes behaved almost the same as 15 minutes system load average with a higher system load average until 100k active users (figure 6.6).

[2] The method of measuring the PPSN server load average is discussed in §6.1.2

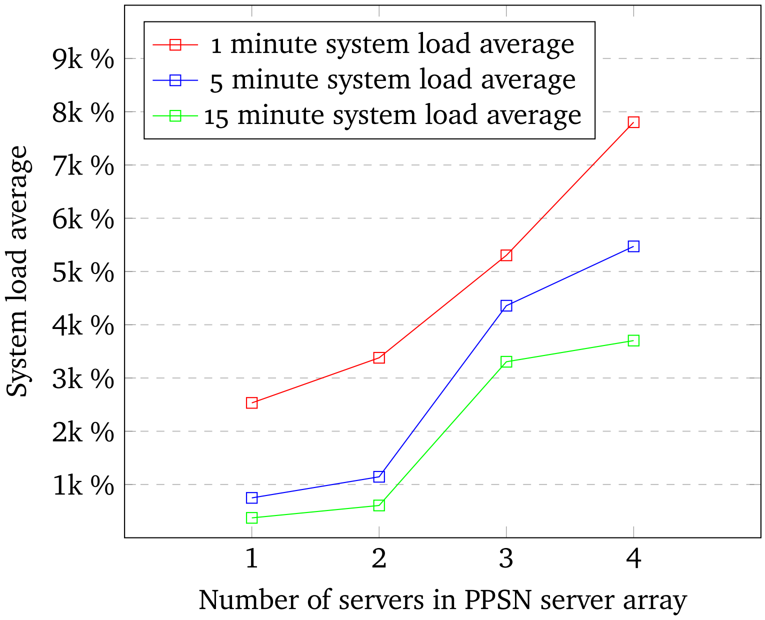


Figure 6.7: PPSN system's overall system load average when varying the number of servers in the PPSN server array with 50k users and 10k noise.

Likewise, figure 6.7 shows the system load average in percentage for end-to-end message communication for a variable number of servers in the PPSN server array for 50k active users (around 10% of the users participated in the message transmission) for 1 minute, 5 minutes and 15 minutes. Furthermore, figure 6.7 depicts how system load increases with an increased number of servers in the PPSN server array. The slope of the system load average remains roughly the same until the PPSN server has two servers in its server array (figure 6.7). However, for three or more servers in the PPSN server array results in higher slopes—meaning higher system load average for 5 minutes and 15 minutes system load average (figure 6.7). Surprisingly, figure 6.7 shows that the peak load average—1-minute system load average—does not fluctuate significantly compared to 5-minute and 15-minute system load for variable servers in the PPSN server array.

Lastly, from figure 6.6 and figure 6.7, the 1-minute system load is higher than 5 minutes and 15 minutes system load. In other words, longer runs have lesser system load, and shorter runs have a higher system load. This is happening because of the peak load of the testing (i.e., the bursts of process forking and threads in the system), and over time the peak spreads out and provides lower system load for longer runs.

7

Discussion

This section outlines the discussion of the privacy arguments of the PPSN system, its design choices, implementation and evaluation. Conclusively, this section also frames the discussion about the objective of this thesis.

Privacy argument. The PPSN system does not provide a factual list of which data are private, but rather gives an idea of what kind of data is possibly private and public, and a concept of data segregation. However, the PPSN system does encrypt all the user data. Needless to say, PPSN does not need to know which data from a user should be encrypted or not, since any data that is not protected, reveals metadata. The PPSN system followed a naive data segregation approach—any data from users are considered as private, and any data from public APIs are considered as public data. Consequently, all the private data is protected, and the public data stream directly connects to the SN backend since it does not require data hiding. In some exceptional cases, public APIs may need protection, but the PPSN system does not support this feature.

System performance. End-to-end system performance of the PPSN system has been measured and evaluated in terms of end-to-end system latency and end-to-end system load in §6.4. The PPSN server latency and system load behave as expected—higher latency for a higher number of users in the system and higher system load for higher servers in the PPSN server array. However, the end-to-end system latency and system load for more servers in the PPSN server array has a smaller change in latency and system load compared to increasing users. This phenomenon occurs because users are more computation heavy than the servers in the PPSN server array, as the users need to perform the message enveloping and layered encryption. However, the RSA algorithm encryption process is slightly faster than the decryption process. On the other hand, servers in the PPSN server array only takes off one layer of encryption and adds noise. This scenario would have been changed if the users emulated from a different machine and PPSN server array in another machine as each server in the PPSN server array needs to process $(n + \eta)$ requests.

Design choices. Data enveloping is one of the core design choices in the PPSN system, which reduces the encryption time significantly by combining the asymmetric encryption and the symmetric encryption. RSA is used as asymmetric encryption, and AES is chosen for symmetric encryption. However, there are other alternatives for asymmetric and symmetric encryption like DH [86], algorithm RC5 [87], DES [88] and more. Nevertheless, the performance of those algorithms do not vary that much. In this thesis, the current standards of encryption methods are chosen. Protocol buffer has been used instead of JSON to shrink the message size and to provide data security to some extent, as it needs a schema to decode the binaries.

Prototype trade-offs. The barebone implementation of the PPSN system does not implement the whole proposed system, but instead implements the minimum viable product to demonstrate and experiment with it. Therefore the implementation serves as a proof of concept. It implements a round of end-to-end communication between users and SN backend. Moreover, it does not implement the commencing protocol, but it implements the communication protocol. SN backend is a dummy server application which implements only the interface of the PPSN system. However, any legacy SN system can be used as an SN backend in the PPSN system with the PPSN interface implemented in it.

Scalability. The PPSN system design does not allow adding servers for scaling purpose. Adding more servers in the PPSN server array, would increase privacy as it adds another layer of encryption and more noise addition. Not to mention one more extra server for an adversary to compromise. However, this cost of extra encryption and noise adds up—meaning scaling it down by adding more servers instead of getting higher throughput and lower latency. There could be a workaround where one can think each server in the PPSN server array as a block of servers. These blocks are connected in series and servers inside the block are connected in parallel. Therefore, scaling is possible in this setup by adding more servers in a block. However, this will create a bottleneck because other blocks have not been expanded. Therefore, every block in the

PPSN server array needs to have the same amount of server expansion to achieve scalability. It is not a cost-effective solution, and scaling up the PPSN system is expensive.

Fault-tolerance. Unfortunately, the PPSN system has no fault tolerance in the PPSN server array, as all the servers have to participate in making the communication untraceable. If one server in the PPSN server array fails, then the entire system goes offline. Fault-tolerance could be achieved by the same block of server concept discussed in the earlier paragraph Scalability. However, SN backend has some degree of fault tolerance by nature of its virtual endpoint design, where each SN backend endpoint is assigned to serve a user. In case of one SN backend endpoint failure, users can request for another round with different SN backend endpoint.

An Ideal experimentation. In this thesis, all the tests have been performed inside a single machine. The experimental setup should have run on a different server in the different data centre to simulate the real-world scenario. However, one of the subjects in this thesis is the proof of concept of the PPSN system design, and it does that. Nevertheless, user bottlenecking has been tried to reduce by throttling down the user requests as the experiments run on a single machine. Network bandwidth performance has not been explored since the PPSN system is CPU bound and ran on a single machine.

Settling the research problem. This thesis began with a research problem (§1.3) of ensuring user privacy in SN system (§2.1.2) and sought to answer the research questionaries throughout the thesis. It answers the question about privacy issues in the SN system (§2.1.2) in chapter 3. Subsequently, it answers the question of technical requirement to make the SN system private yet practical and feasible in chapter 4. Finally, it answers how to implement (Chapter 5) and evaluate (Chapter 6) the privacy-preserving smart nudging (PPSN) system that satisfies the identified privacy requirements for SN system. Indeed, this thesis answers all the three research questions that it aimed to answer.

8

Future work

This thesis only implements a barebone prototype of the PPSN system, which lacks features. Hence, a complete implementation of the PPSN system is essential to evaluate it in a real-world setup (e.g., distributed server in a distributed data centre and with real users) is left for future work.

Clients (i.e., users) are found to be computational heavy from the results of experimentation. The enveloping and layered encryption could be done in a better way, which requires further study.

The experimentation in this thesis uses only a single machine to experiment with the PPSN system. Ideally, for more precise performance evaluation, the experimentation should take place on a different physical server, and it is left for the future.

Finally, this thesis only theoretically explains how the PPSN system achieves privacy. No attacks have been performed practically to test how secure the PPSN system is. Performing various kind of practical attacks on the PPSN system could reveal some security holes in real-life, which are left for future work.

/9 Conclusion

The contribution of this thesis, privacy-preserving smart nudging system (PPSN) is the first of its kind in smart nudging for green transportation system that achieves GDPR compliance and provides strong privacy by hiding both the content and the metadata. Furthermore, PPSN system protects users and their data from traffic analysis and data breach from strong adversaries.

This thesis achieves it by thoroughly studying the SN system, GDPR, various cryptographic schemes, secure communications, and various private messaging systems. Then delivers a comprehensive design and a proof of concept—the PPSN system that satisfies the design goals and solves the privacy issues of legacy SN system.

It accomplishes this level of privacy by identifying unprotected variables it needs to hide then encrypting as much data as possible and then adding noise in them whatever is not encryptable. Finally, the PPSN system tries to minimize the difference between the states of the system over time by employing round conversations and data padding. Altogether these techniques help the PPSN system to accomplish adequate privacy and security for both the users and the SN system in terms of GDPR, yet making the system practical and feasible enough for real-world usage.

Bibliography

[1] Henk CA Van Tilborg. *Fundamentals of cryptology: a professional reference and interactive tutorial*, volume 528. Springer Science & Business Media, 2006.

[2] Ari Luotonen and Kevin Altis. World-wide web proxies. *Computer Networks and ISDN systems*, 27(2):147–154, 1994.

[3] Chris Greamo and Anup Ghosh. Sandboxing and virtualization: Modern tools for combating malware. *IEEE Security & Privacy*, 9(2):79–82, 2011.

[4] Suvda Myagmar, Adam J Lee, and William Yurcik. Threat modeling as a basis for security requirements. In *Symposium on requirements engineering for information security (SREIS)*, volume 2005, pages 1–8. Citeseer, 2005.

[5] Adam Shostack. *Threat modeling: Designing for security*. John Wiley & Sons, 2014.

[6] Paul Voigt and Axel von dem Bussche. *The EU General Data Protection Regulation (GDPR)*. Springer International Publishing, 2017.

[7] Anders Andersen and Randi Karlsen. *Privacy Preserving Personalization in Complex Ecosystems*, pages 247–261. Springer Berlin Heidelberg, Berlin, Heidelberg, 2018.

[8] Randi Karlsen and Anders Andersen. Recommendations with a nudge. *Technologies*, 7(2), 2019.

[9] Anders Andersen, Randi Karlsen, and Weihai Yu. *Green Transportation Choices with IoT and Smart Nudging*, pages 331–354. Springer International Publishing, Cham, 2018.

[10] A. Rusbridger. *The Snowden leaks and the public*. The New York Review of Books, 2013.

[11] Anders Andersen and Randi Karlsen. User profiling through nfc interactions: Mining nfc-based user information from mobile devices and back-end systems. In *Proceedings of the 14th ACM International Symposium on Mobility Management and Wireless Access*, MobiWac '16, page 173–176, New York, NY, USA, 2016. Association for Computing Machinery.

[12] Yehuda Lindell and Benny Pinkas. Secure multiparty computation for privacy-preserving data mining. *Journal of Privacy and Confidentiality*, 1(1), Apr. 2009.

[13] EU Commission et al. Green paper, towards a new culture for urban mobility. *European Union, Brussels*, 2007.

[14] US EPA. Emission facts: Greenhouse gas emissions from a typical passenger vehicle, 2005.

[15] Oliver Storz, Adrian Friday, and Nigel Davies. Supporting content scheduling on situated public displays. *Computers & Graphics*, 30(5):681–691, 2006.

[16] Markus Weinmann, Christoph Schneider, and Jan vom Brocke. Digital nudging. business & information systems engineering 58, 6 (dec. 2016), 433–436, 2016.

[17] Matthias Lehner, Oksana Mont, and Eva Heiskanen. Nudging–a promising tool for sustainable consumption behaviour? *Journal of Cleaner Production*, 134:166–177, 2016.

[18] Min Gao, Kecheng Liu, and Zhongfu Wu. Personalisation in web computing and informatics: Theories, techniques, applications, and future research. *Information Systems Frontiers*, 12(5):607–629, 2010.

[19] MA Razzaque, M Milojevic-Jevric, A Palade, and S Clarke. Middleware for internet of things: a survey. ieee internet things j. 3, 70–95 (2016), 2015.

[20] Dieter Uckelmann, Mark Harrison, and Florian Michahelles. An architectural approach towards the future internet of things. In *Architecting the internet of things*, pages 1–24. Springer, 2011.

[21] Charith Perera, Yongrui Qin, Julio C Estrella, Stephan Reiff-Marganiec, and Athanasios V Vasilakos. Fog computing for sustainable smart cities: A survey. *ACM Computing Surveys (CSUR)*, 50(3):1–43, 2017.

[22] Daniel J Solove. Conceptualizing privacy. *Calif. L. Rev.*, 90:1087, 2002.

[23] Daniel J Solove. A taxonomy of privacy. *U. Pa. L. Rev.*, 154:477, 2005.

[24] Daniel J Solove. I've got nothing to hide and other misunderstandings of privacy. *San Diego L. Rev.*, 44:745, 2007.

[25] Chris Clifton, Murat Kantarcioglu, and Jaideep Vaidya. Defining privacy for data mining. In *National science foundation workshop on next generation data mining*, volume 1, page 1. Citeseer, 2002.

[26] Parliament and council regulation (eu) 2016/679 of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation). *O.J.*, L 119:1–88, 2016-04-27.

[27] Ronald L Rivest. Cryptography. In *Algorithms and complexity*, pages 717–755. Elsevier, 1990.

[28] Fu-Guo Deng and Gui Lu Long. Secure direct communication with a quantum one-time pad. *Physical Review A*, 69(5):052319, 2004.

[29] Ronald Rivest and S Dusse. The md5 message-digest algorithm, 1992.

[30] Robert R Schaller. Moore's law: past, present and future. *IEEE spectrum*, 34(6):52–59, 1997.

[31] Monika Agrawal and Pradeep Mishra. A comparative survey on symmetric key encryption techniques. *International Journal on Computer Science and Engineering*, 4(5):877, 2012.

[32] Gary C Kessler. An overview of cryptography. *the Handbook on Local Area Networks, Auerbach*, 1998.

[33] Zaid Kartit, Ali Azougaghe, H Kamal Idrissi, Mohamed El Marraki, Mustapha Hedabou, Mostafa Belkasmi, and Ali Kartit. Applying encryption algorithm for data security in cloud storage. In *International Symposium on Ubiquitous Networking*, pages 141–154. Springer, 2015.

[34] H Delfs and H Knebl. Introduction to cryptography principles and applications (2007).

[35] Steve Burnett and Stephen Paine. *The RSA security's official guide to*

cryptography. McGraw-Hill, Inc., 2001.

[36] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[37] Jakob Jonsson and Burt Kaliski. Public-key cryptography standards (pkcs)# 1: Rsa cryptography specifications version 2.1. Technical report, RFC 3447, February, 2003.

[38] Anton Iliev, Nikolay Kyurkchiev, and Asen Rahnev. A note on adaptation of the knuth's extended euclidean algorithm for computing multiplicative inverse. *International Journal of Pure and Applied Mathematics*, 118(2):281–290, 2018.

[39] Breaking rsa encryption – an update on the state-of-the-art. `https://www.quintessencelabs.com/blog/breaking-rsa-encryption-update-state-art/`. Accessed: 2020-07-16.

[40] Joan Daemen and Vincent Rijmen. Aes proposal: Rijndael. 1999.

[41] Cynthia Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19. Springer, 2008.

[42] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[43] Michael Hilton. Differential privacy: a historical survey. *Cal Poly State University*, 2002.

[44] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3–4):211–407, August 2014.

[45] Seidu Inusah and Tomasz J Kozubowski. A discrete analogue of the laplace distribution. *Journal of statistical planning and inference*, 136(3):1090–1102, 2006.

[46] Wikipedia:biographies David Chaum. `https://en.wikipedia.org/wiki/David_Chaum`. Accessed: 2020-04-28.

[47] David L Chaum. Untraceable electronic mail, return addresses, and digital

pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[48] Claudio A Ardagna, Sushil Jajodia, Pierangela Samarati, and Angelos Stavrou. Privacy preservation over untrusted mobile networks. In *Privacy in Location-Based Applications*, pages 84–105. Springer, 2009.

[49] George Danezis. Mix-networks with restricted routes. In *International Workshop on Privacy Enhancing Technologies*, pages 1–17. Springer, 2003.

[50] Michael G Reed, Paul F Syverson, and David M Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Selected areas in Communications*, 16(4):482–494, 1998.

[51] David Goldschlag, Michael Reed, and Paul Syverson. Onion routing. *Communications of the ACM*, 42(2):39–41, 1999.

[52] Bernd Conrad and Fatemeh Shirazi. A survey on tor and i2p. In *Ninth International Conference on Internet Monitoring and Protection (ICIMP2014)*, pages 22–28, 2014.

[53] Ye Zhu, Xinwen Fu, Bryan Graham, Riccardo Bettati, and Wei Zhao. On flow correlation attacks and countermeasures in mix networks. In *International Workshop on Privacy Enhancing Technologies*, pages 207–225. Springer, 2004.

[54] Paul Syverson. Sleeping dogs lie in a bed of onions but wake when mixed. *4th Hot Topics in Privacy Enhancing Technologies (HotPETs 2011)*, 2011.

[55] Vitaly Shmatikov and Ming-Hsiu Wang. Timing analysis in low-latency mix networks: Attacks and defenses. In *European Symposium on Research in Computer Security*, pages 18–33. Springer, 2006.

[56] About node.js. `https://nodejs.org/en/about/`. Accessed: 2020-06-30.

[57] Andres Ojamaa and Karl Düüna. Assessing the security of node. js platform. In *2012 International Conference for Internet Technology and Secured Transactions*, pages 348–355. IEEE, 2012.

[58] Bjarne Stroustrup. *The C++ programming language*. Pearson Education, 2013.

[59] Brain Beej Hall. Beej's guide to network programming: using internet sockets, 2001.

[60] Pieter Hintjens. *ZeroMQ: messaging for many applications*. " O'Reilly Media, Inc.", 2013.

[61] Pieter Hintjens. Ømq-the guide. *Distributed Messaging-zeromq*, 2014.

[62] Protocol Buffers a language-neutral, platform-neutral extensible mechanism for serializing structured data. `https://developers.google.com/protocol-buffers`. Accessed: 2020-07-03.

[63] Extensible markup language (xml). `https://www.w3.org/standards/xml/core`. Accessed: 2020-06-29.

[64] The json data interchange standard. `https://www.json.org/json-en.html`. Accessed: 2020-06-28.

[65] K. Maeda. Performance evaluation of object serialization libraries in xml, json and binary formats. In *2012 Second International Conference on Digital Information and Communication Technology and it's Applications (DICTAP)*, pages 177–182, May 2012.

[66] Protocol buffers encoding. `https://developers.google.com/protocol-buffers/docs/encoding#structure`. Accessed: 2020-07-14.

[67] David Isaac Wolinsky, Henry Corrigan-Gibbs, Bryan Ford, and Aaron Johnson. Dissent in numbers: Making strong anonymity scale. In *Presented as part of the 10th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 12)*, pages 179–182, 2012.

[68] Henry Corrigan-Gibbs, Dan Boneh, and David Mazières. Riposte: An anonymous messaging system handling millions of users. In *2015 IEEE Symposium on Security and Privacy*, pages 321–338. IEEE, 2015.

[69] Jelle Van Den Hooff, David Lazar, Matei Zaharia, and Nickolai Zeldovich. Vuvuzela: Scalable private messaging resistant to traffic analysis. In *Proceedings of the 25th Symposium on Operating Systems Principles*, pages 137–152, 2015.

[70] Glenn R Bernard. Stadium game for fans, September 25 2001. US Patent 6,293,868.

[71] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. Technical report, Naval Research Lab Washington DC, 2004.

[72] Michael O Rabin and Christopher Thorpe. Time-lapse cryptography. 2006.

[73] Andreas Kapsner and Barbara Sandfuchs. Nudging as a threat to privacy. *Review of Philosophy and Psychology*, 6(3):455–468, Sep 2015.

[74] Valerie Issarny, Vivien Mallet, Kinh Nguyen, Pierre-Guillaume Raverdy, Fadwa Rebhi, and Raphael Ventura. Dos and don'ts in mobile phone sensing middleware: Learning from a large-scale experiment. In *Proceedings of the 17th international middleware conference*, pages 1–13, 2016.

[75] Oliver Storz, Adrian Friday, and Nigel Davies. Supporting content scheduling on situated public displays. *Computers & Graphics*, 30(5):681–691, 2006.

[76] Matthias Lehner, Oksana Mont, and Eva Heiskanen. Nudging–a promising tool for sustainable consumption behaviour? *Journal of Cleaner Production*, 134:166–177, 2016.

[77] Richard H Thaler and Cass R Sunstein. *Nudge: Improving decisions about health, wealth, and happiness*. Penguin, 2009.

[78] P Guldborg Hansen and A Maaløe Jespersen. Nudge and the manipulation of choice. *European Journal of Risk Regulation*, 3(1):3–28, 2013.

[79] Andrew S Tanenbaum and Maarten Van Steen. *Distributed systems: principles and paradigms*. Prentice-Hall, 2007.

[80] PGP BAL's. Public key server. *The Computer Law Resource*, pages 1–2, 1996.

[81] Android developer guides. `https://developer.android.com/guide`. Accessed: 2020-06-29.

[82] Apple app store guide. `https://developer.apple.com/app-store/review/guidelines/`. Accessed: 2020-06-29.

[83] Ray Walker. Examining load average. *Linux Journal*, 2006(152):5, 2006.

[84] Domenico Ferrari and Songnian Zhou. An empirical investigation of load indices for load balancing applications. Technical report, CALIFORNIA UNIV BERKELEY DEPT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE, 1987.

[85] Intel® core™ i7-8565u processor product specifications. `https:`

//ark.intel.com/content/www/us/en/ark/products/149091/intel-core-i7-8565u-processor-8m-cache-up-to-4-60-ghz.html. Accessed: 2020-08-29.

[86] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

[87] Ronald L Rivest. The rc5 encryption algorithm. In *International Workshop on Fast Software Encryption*, pages 86–96. Springer, 1994.

[88] Don Coppersmith. The data encryption standard (des) and its strength against attacks. *IBM journal of research and development*, 38(3):243–250, 1994.