Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Markus André Jacobsen

Privacy-Preserving Decentralized Calculation in Digital Contact Tracing

Master's thesis in Applied Computer Science
Supervisor: Mariusz Nowostawski
June 2021

NTNU
Norwegian University of
Science and Technology

Markus André Jacobsen

Privacy-Preserving Decentralized Calculation in Digital Contact Tracing

Master's thesis in Applied Computer Science
Supervisor: Mariusz Nowostawski
June 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Abstract

Contact tracing is a manual process used by health agencies to map individuals' social interactions and is performed when individuals become infected to locate others at-risk. The Covid-19 pandemic increased the need to digitalize this process and shift data processing onto hand-held devices such as personal mobile phones. Digital contact tracing applications are divided into centralized and decentralized models – the latter attempts to strengthen privacy. A drawback of the decentralized model is that it limits the ability of health agencies to gather statistics and granular data. In this thesis, the goal was to provide a possible solution to this problem and study the solution's performance. We present a method of using Bluetooth Low Energy with controlled flooding and Homomorphic Encryption to gather data in a distributed, connectionless, privacy-preserving manner. Through simulation, we have modeled and tested an optimized solution. Our main findings conclude that a clustered network with an aggressive data drop strategy delivers the best performance. We determined that control measures such as Time-To-Live and dropping duplicate packets were required to achieve an optimized solution. Further, a clustered topology with controlled flooding increases the overall performance by 99.6% compared to a fully connected topology. However, a small increase in node participation requirements creates a disproportionate increase in traffic overhead.

Key words: decentralization, Bluetooth Low Energy, digital contact tracing, connectionless networks, controlled flooding

Sammendrag

Kontaktsporing er en manuell prosess som brukes av helsebyråer for å kartlegge individers sosiale interaksjoner, og utføres når enkeltpersoner blir smittet for å identifisere andre i faresonen. Covid-19 pandemien økte behovet for å digitalisere denne prosessen og flytte databehandling til håndholdte enheter som personlige mobiltelefoner. Applikasjoner for digital kontaktsporing er delt inn i sentraliserte og desentraliserte modeller - sistnevnte prøver å styrke personvernet. En ulempe med den desentraliserte modellen er at den begrenser helsevesenets evne til å samle statistikk og detaljerte data. I denne oppgaven var målet å gi en mulig løsning på dette problemet og studere løsningens ytelse. Vi presenterer en metode for å bruke Bluetooth Low Energy med kontrollert *flooding* og homomorfisk kryptering for å samle inn data på en distribuert, tilkoblingsfri, personvernbevarende måte. Gjennom simulering har vi modellert og testet en optimalisert løsning. Våre hovedfunn konkluderer med at et klynget nettverk med en aggressiv datadroppstrategi gir best ytelse. Vi har kommet frem til at det er nødvendig med kontrolltiltak som Time-To-Live og å forkaste dupliserte pakker for å oppnå en optimalisert løsning. Videre øker en klynget topologi med kontrolltiltak den totale ytelsen med 99,6% sammenlignet med en fullstendig tilkoblet topologi. En liten økning i kravene til nodedeltakelse skaper imidlertid en uforholdsmessig økning i trafikk.

Nøkkelord: desentralisering, Bluetooth Low Energy, digital kontaktsporing, tilkoblingsfritt nettverk, kontrollert *flodding*

Acknowledgements

This thesis concludes my master's degree in Applied Computer Science at NTNU Gjøvik. It also marks the end of my six-year attendance at the school. Having researched a subject directly tied to the Covid-19 pandemic has been an insightful endeavor in a time many find difficult. I want to thank my supervisor Mariusz Nowostawski for a good collaboration. I also want to thank my employer Marcello AS and our client, the Agency for Improvement and Development at the City of Oslo, for enabling and encouraging me to pursue the completion of my master's degree part-time. Finally, I want to thank my fiancee for supporting me throughout the work.

Markus André Jacobsen
Oslo, Norway. 29.05.2021

Contents

Figures

Tables

Code Listings

Chapter 1

Introduction

Decentralization in computer sciences is an architecture shift aimed at removing central points of trust and distributing storage, processing, and communication in a peer-to-peer fashion, i.e., the users participate in a trustless system. Taking a centralized system and decentralizing it is often used to democratize systems using the Internet protocol stack as the communication medium. However, the same principles can be applied in short-range communication, such as Bluetooth. Today, Bluetooth Low Energy (BLE) is used for transporting small data packets over short distances, using very little energy, making it an attractive protocol for hand-held- and sensor devices.

The Coronavirus (Covid-19) outbreak started in Asia and rapidly transformed into a global pandemic. Covid-19 is a highly infectious disease that transmits through saliva. A traditional measure against contagious diseases is contact tracing; this is the systematic mapping of an infectious individual's social interactions. The purpose of contact tracing is to investigate if these social interactions could have led to more infected individuals. Contact tracing is handled by trained and certified personnel through interviews. The scale of Covid-19 led to personnel shortages all over the world, and training is a slow process. As a response, many countries started developing *digital contact tracing* applications and frameworks.

Various solutions to digital contact tracing were proposed and developed, but one of the most popular is using BLE to record proximity and exposure time between individuals. The resulting *contact events* would then be shared with the central health agency if and when an individual was diagnosed with Covid-19. Academics and privacy collectives all over the world argued that this solution was hastily modeled and left much to be desired in regards to digital privacy. As a result of this discontent, digital contact tracing utilizing contact events split into two architectural camps: Centralized and decentralized. The decentralized architecture limits the type of information that a central agency receives by shifting much of the data processing over to personal devices.

Decentralized architecture strengthened privacy, but a drawback is that it offers less accurate risk assessment and optimization [1]. By limiting the data that central agencies have at hand, it also constrains the available data for further ana-

lysis. The shortcomings of varied data collection in decentralized architecture are the problem we will look at in this thesis.

1.1 Justification and motivation

Modern decentralization in computer science is an important concept, especially within the realm of privacy and security. A centralized approach implies that some application critical computation or data storage is handled at a central point, e.g., a central server. All users of the system connect to the same server or cluster of servers to use the service. A critical computation may be the handling of authentication, routing, accessing data, with more. By contrast, a decentralized system enables some or all of these functions to be handled by the individual nodes or collectively within a decentralized network. This enables the individual nodes to store and serve information, basically retaking ownership of data, such as a personal profile, and can at any point revoke this access. Decentralization in digital contact tracing is built upon this idea that the identification of participating nodes is never revealed to a central point of trust unless explicitly granted access to do so, i.e., a diagnosed individual reports to the central server.

Expanding on decentralized digital contact tracing, this paper wants to explore the possibilities of providing central agencies with more quantitative and qualitative data through peer-to-peer calculation of core metrics without compromising peer privacy. The ability to provide more detailed and varied data would aid those in control to strategize better. One primary advantage of having lots of data is the ability to predict trends more precisely. This would give decision-makers the ability to enact more accurate, proactive, and localized measures than reactive and generalized measures. Given that we can provide comparable or better data than centralized-based architecture could also incentivize a wider adoption of decentralized digital contact tracing. The first step in this direction is to map out the enabling technologies and practices to enable such a feat. **Goal**: Develop a protocol that would allow for privacy-preserving distributed calculation in decentralized digital contact tracing.

1.2 Research Questions

Objective: Extend existing decentralized digital contact tracing solutions to support better data generation in a privacy-preserving manner without a central point of trust. The solution should be optimal in terms of complexity, efficiency, and privacy.

Tasks: (T1) Provide an overview of existing digital contact tracing solutions, frameworks, and applications. (T2) Investigate possible solutions to share data without a central point of trust while maintaining privacy. (T3) Test the trade-offs and complexity of the solutions found in (T2). (T4) Suggest an overall architecture of the extensions.

The objective and tasks this thesis address can be divided into two Research Questions:

1. How can a privacy-preserving arithmetic calculation be carried out in the context of decentralized digital contact tracing?
2. What trade-offs, in terms of privacy, efficiency, complexity, the answer(s) for RQ1 have?

1.3 Thesis outline

The rest of the thesis is structured in the following way. Chapter 2 presents related background theory and concepts. Chapter 3 will expand more on the research design, methods of data analysis, and validity. Followed by chapter 4 that will present our proposed solution and describe the implementation of the data collection tool (simulator). Followed by chapter 5, presenting the outline of our experiments and the significant experimental findings. Chapter 6 discusses the mentioned findings, limitations, future research directions, and other reflections regarding the solution. Lastly, chapter 7 concludes the thesis.

Chapter 2

Background

This chapter presents theory and concepts related to the problem description and forms the theoretical foundation for the rest of the thesis. The thesis deals with a complex issue in that multiple fields of science are relevant to the overall picture. Digital contact tracing is also a relatively recent but very fast-growing field. A consequence of this is that much research with varying quality is published. We have made an effort to extract what is essential and where multiple sources agree. For more information on sources and search queries, see Appendix C.

2.1 Contact tracing

Contact tracing is a method deployed by health agencies to review with whom and where a disease-infected individual has been. The purpose of the tracing is to determine if others should be contacted and issue different recommendations based on the circumstances or the use of local strategies such as locking down public areas, e.g., a restaurant or mall. Traditional contact tracing requires a considerable amount of staff and time as each infected individual goes through a separate procedure of interviews [2]. Significant findings will need additional follow-up. Training new staff also takes time as each contact tracer needs to be professionally trained. During a large outbreak, this can accumulate in health services running out of capacity, further delaying contact tracing work.

During the Covid-19 outbreak, many countries struggled with the ever-increasing load, and digital contact tracing tools were suggested and swiftly developed and deployed. The idea of digital contract tracing is not new and existing studies on the subject exists [3]. Use case diseases such as Ebola and Influenza have been studied in an effort to automate or partially automate the contact tracing process. Previously studied solutions are either based on proximity, location, or recall, and the most common technologies are Bluetooth, GPS, WiFi, or other radio frequencies. Further, the different suggestions feature different degrees of automation; in some studies, everything is automated; in others, the application merely allows the users to add manual entries [1]. One of the more common

manual entry methods deployed in public areas and businesses is special QR-codes. In this method, the users will scan the QR-code and register a visit at the location of the QR-code. As with traditional contact tracing, the goal of digital contact tracing applications is to autonomously gather data about contact events and the duration of contact and calculate the population's *at-risk* factor. This factor estimates how likely it is that someone has been exposed to an infected person and is based on the concept of social graphs. The at-risk factor will determine if, e.g., an individual should go into quarantine or not. Digital contact tracing gives the public more freedom of movement in that more accurate measurements can be deployed locally. Areas with a high percentage of infection have more restrictions than areas with a low disease footprint.

2.1.1 Digital contact tracing

There are multiple factors to consider when modeling a digital contact tracing application. The application has to log locations, timestamps, contacts, and contact duration. Most existing tracing applications use Bluetooth Low Energy (BLE), see Section 2.2, to determine the distance between another person by recording a contact event. The exposure duration can be determined by performing this exchange in intervals and log the calculated proximity along side a timestamp. Gathering accurate proximity can be achieved through distance measured from the received strength (RSSI) of Bluetooth [4]. All Contact tracing applications of this kind are deployed as an application on a mobile device. For the rest of the thesis, this is referred to as an app or application. It is common for an app to require the device holder to register using some government-issued identification, and as such, each device and application can only have one unique person registered.

Model-based simulation suggests that digital contact tracing is an effective tool to mitigate the impact of an epidemic [5]. The efficiency of digital contact tracing depends heavily on the number of people using it and the density of people using it [6]. High uptake is needed because if large parts of the population are not partaking, the contact tracing will not be able to reach all close contacts, i.e., everyone that makes up a unique social graph. The need for a high uptake has led to an ethical debate if digital contact tracing applications need to be compulsory or voluntary.

In essence, the discussion concerning if participation in digital contact tracing should be compulsory or voluntary, is based around the dilemma of diagnosed individuals being forced to report their test results, i.e., should it be mandatory for a recently infected individual to report it through the contact tracing application. As mentioned previously, the more people participating in the contact tracing, the more effective it gets. Because of this, the idea of making it mandatory prevails. However, by making it compulsory, the public health indirectly becomes an individual responsibility [6]. It is remarked that when businesses require visitors to register using, e.g., the QR-method or refused service, the involvement in digital contact tracing becomes *de facto* mandatory [7].

Another lesser-used technique is that of location sharing. Applications based on this approach will regularly upload a GPS coordinate to a central server. In the case of a reported infected individual, all of the locations stored in the central server will be used to calculate to whom might be at risk of exposure and notify these individuals. This method has been critiqued because it is less privacy concerning and has worse precision than other alternatives [8].

For the sake of clarity, the rest of this chapter and thesis will only discuss digital contact tracing using BLE to record contact events.

2.1.2 Architecture

The architecture of digital contact tracing does not refer to the specifics of how a certain application is implemented, i.e., technical architecture or system architecture. Instead, digital contact tracing architecture refers to: what kind of information are communicated between the users and the system, what responsibility the system has, and what responsibility the users have. In other words, where in the system is the trust placed. In most European countries, the system owner is a governmental health agency and/or a collective of national institutes [9], but these are not necessarily the system developers, maintainers, or operators. The health agencies usually have the role of data extractors and sets requirements. The inclusion of third-party actors has an impact on how trustworthy academics and privacy advocates perceive the applications.

Digital contact tracing is split into two camps divided in their view of how it should be implemented architecturally. The first available solutions were so-called centralized architectures. The other prevalent group wants a decentralized architecture. The advocates for a decentralized system are worried about the lack of privacy protection in centralized systems. They argue that an inadequate amount of time has been invested in securing that the app-based tracing applications do not contain any privacy leakages. Both architectures require a central server to notify peers in the system, but the information sharing, information flow, and information generation differ.

One of the most significant differences in the architectures is how reporting data to the health agencies function. In a centralized architecture, when an infected individual reports infection status to the app, it will submit the Ids of every other encountered app. This means that the server will receive information on all of the contacts that the infected individual had recorded with the tracing application. This differs in a decentralized architecture. An infected individual only reports its own Ids, meaning the central server has only the Ids of infected individuals. A different crucial difference is how the Ids are generated in the first place. In centralized systems, the backend server generates a limited set of unique Ids that are only intended to be used in a limited timespan and distribute this set to a single app. Now the central server will know something about the different Ids firsthand and could be misused if a malicious actor extracts them. Privacy concerns are further discussed in Section 2.6. Whereas in decentralized systems, the app-

holder generates a limited set of temporary Ids, and thus the central server does not know them. The Decentralized Privacy-Preserving Proximity Tracing (DP-3T) collective highlights this as their reasoning for calling their protocol "decentralized" - the backend servers are just there to provide availability and do not act as a central point of trust for security and privacy[1].

The choice of architecture has consequences for the at-risk calculation. The at-risk calculation is an automated estimate of how likely, given the registered contacts, that an individual has been exposed to an infected individual [10]. In a central architecture, the server can calculate an individual's at-risk score based on the information the server already holds. The server can do this because it holds what is effectively a user's complete social graph. Whereas decentralized app holders have to download all the Ids of reported infected and cross-reference with the Ids the app has registered locally. The combined differences in information flow are presented in Figure 2.1.

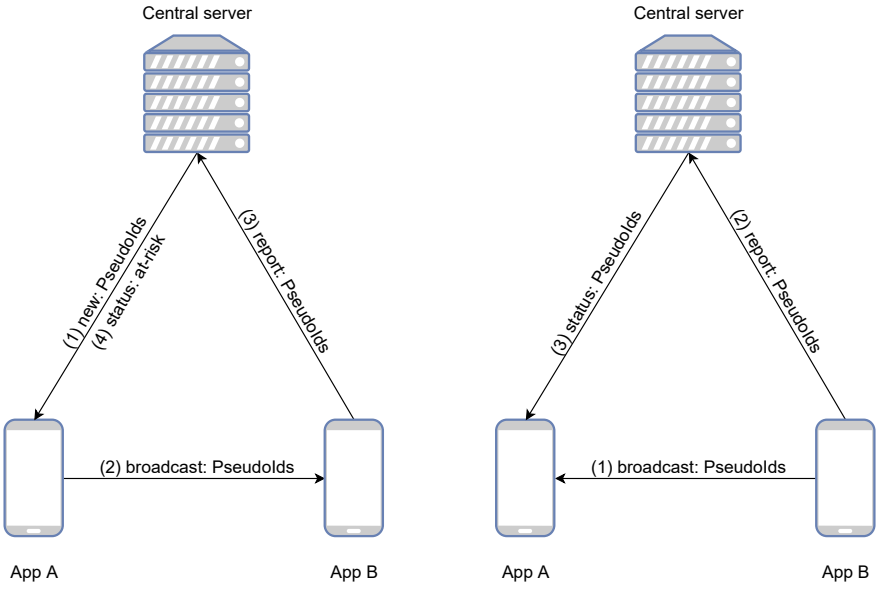
Many digital contact tracing frameworks have emerged through 2020, Martin *et al.* [9] studied 12 different frameworks. Of these, four were based on a centralized architecture, seven were based on a decentralized architecture, and one was based on a hybrid architecture. One of the earliest organizations on contact tracing frameworks was the Pan-European Privacy-Preserving Proximity Tracing (PEPP-PT) group that advocated for standardizing centralized architecture. Previous PEPP-PT members created the group DP-3T because of disagreements over digital contact tracing architecture. Later actors in developing frameworks are the joint operation by Google and Apple. Their framework is heavily inspired by the DP-3T framework but features better interoperability between mobile Operative Systems (OS) and more secure handling of privacy data. Martin *et al.* [9]'s study of the European countries' effort with digital contact tracing identified that between the 22 countries currently having an operational application, 15 of them utilize a decentralized framework, and seven a centralized framework. All of the 15 decentralized-based applications use the Google/Apple Exposure Notification framework[2]. A study by Wen *et al.* [11] includes findings outside of Europe; of the 18 studied apps, 12 were based on a centralized architecture, whereas six were based on a decentralized architecture.

2.2 Bluetooth

Bluetooth (defined in the IEEE 802.15.1 standard) is a Personal Area Network (PAN) communication protocol that sends data using radio frequencies. It is a low-power, short-distance, low-rate protocol [12], but does have the ability to send data over more significant lengths using powerful dedicated emitters. The most common radio transmitter in use is Class 2, which has an effective range of 10 meters with a transfer speed of around 1Mbps [13]. Initially designed as a

[1] `https://github.com/DP-3T/documents/blob/f9c5ba50726652f914869dab8ebf07877aa4a81d/FAQ.md#p7-why-do-you-call-your-design-decentralized-while-having-a-backend`

[2] The numbers reflected the situation in Europe in Sept. 2020



(a) Centralized **(b)** Decentralized

Figure 2.1: Difference in contact tracing architecture. In centralized architecture (2.1a): 1) Server generates a unique set of temporary pseudoIds (also referred to as Ephemeral Identity) and send these to a single app (App A). 2) App A uses the Ids to participate in proximity tracing with App B. 3) App B is diagnosed and submits all the recorded pseudoIds to the server. 4) The server recognizes App A's pseudoIds and calculates an "at-risk" score based on the information at hand and sends this back to App A. In decentralized architecture (2.1b): 1) App B uses the locally generated pseudoIds to participate in proximity tracing with App A. 2) App B is diagnosed and submits its own pseudoIds to the server. 3) App A periodically pulls the available pseudoIds and cross-references these with the Ids encountered during proximity tracing. Figure adapted from Vaudenay [10]

"cable replacement" during the 90s, today's usage of the technology is plentiful. It is infrastructure-less, meaning that no access point or base station is involved in the network. The mobile and the Internet of Things (IoT) communities have fully embraced the technology, pushing innovation further.

Bluetooth as a technology is split into two branches - Bluetooth Classic and Bluetooth Low Energy (BLE). BLE came after Bluetooth Classic and functions similarly. However, its usage is intended for low power, low throughput devices such as sensors, beacons, and other IoT devices [14]. BLE has been extensively adapted by the fitness and health industry, but Zeadally *et al.* [15] predicts that by 2023 90% of devices will be shipped with BLE capabilities. In other words, earlier BLE adoptions utilized devices such as sensors and smartwatches, whereas nowadays, most computers and mobile devices are capable of both Bluetooth Classic and BLE.

The Bluetooth standard is overseen by The Bluetooth Special Interest Group (Bluetooth SIG). The Bluetooth SIG has not created any reference implementation

of Bluetooth, neither Bluetooth Classic nor BLE. Although, they have specified how it should function and to which requirements any implementation must adhere. The goal of this is to create high interoperability between all manufacturers. As such, they have the responsibility of developing the standard and deal with licensing to manufacturers.

2.3 Bluetooth Low Energy

As mentioned earlier, BLE provides the ability to do short-range, small-size data communication. One of the main advantage is that it drains considerably less power compared to Bluetooth Classic. Most smartphones support both Bluetooth Classic and Bluetooth Low Energy via dual-mode, but the Bluetooth protocol stack requires different implementations of the layers. Note that in dual mode, the Bluetooth Stack must implement all the version-specific layers. For BLE, these are the Attribute Protocol (ATT layer), the Generic Attribute Profile (GATT), and the Generic Access Profile (GAP). A distinction: a protocol, in BLE, is a building block in the stack that defines "the different packet formats, routing, multiplexing, encoding and decoding that allow data to be sent effectively between peers." [16], meaning a protocol defines how a packet should be transported and how it should be treated by a recipient. A profile defines how a protocol should be used to achieve some objective.

A discovering device uses the ATT layer to determine the services provided by its neighbors, i.e., devices within range. ATT specifies two roles: client and server; these dictate the communication flow between devices. The GATT profile builds upon ATT and provides a framework that enables communication of the specifics of profiles. A profile is the highest concept in the BLE stack. A profile specifies one or more services used in an application. A service is build using characteristics and conveys information about functionality. A Characteristic is the lowest unit and represents a value and information about the value. The GAP is a profile that every Bluetooth devices need to implement and adhere to, and specifies the roles, defines discoverability, connection, and security modes [17].

BLE is able to function in three distinct network topologies: Point-to-point (one-to-one), Broadcast (one-to-many), and Mesh (many-to-many) [14]. Point-to-point topology is most often used for applications requiring prolonged and stable streaming of data such as mobile to audio device settings. Broadcast topology is used to provide localized data sharing and is the topology used in many digital contact tracing solutions. Mesh topology is useful in where up to a thousand devices have to share information bi-directionally, such as in factory automation architectures.

At its core, BLE devices can communicate using two modes, broadcasting and connections. The broadcast mode is connectionless, i.e., the broadcaster is not directly connected (paired) to any devices. In this mode, the broadcasting device sends out advertising packets that can be picked up by any listening device, often referred to as observers. This mode is the fastest way for a device to send out

data to multiple peers. However, the broadcast mode is not fitted for the communication of any sensitive data as it does not contain any security or privacy mechanisms. Broadcasting has two intended purposes: sending data to applications where a full active connection is not needed and when initiating a pairing in the connection mode [18].

2.4 Bluetooth topology

How peers connect to each other is often determined by the transport technology being used. In the case of Bluetooth, two modes exist connections and connectionless.

2.4.1 Connections

Connection-based routing means that a connection is made between two or more nodes. This can be used to build more complex topologies such as tree or star formations. In Bluetooth Classic and BLE, this is achieved through a pairing process where two nodes agree to the bi-directional connection before data can be exchanged, as seen in Figure 2.2a. Connections are split into two types, pairing and bonding [16]. When a device pairs with another, a short-term connection is established using temporary keys. If a device moves out of range, the two devices can no longer connect to each other.

On the other hand, bonding creates long-term keys such that the devices can reconnect with each other when in range. After an initial connection is established, the devices will remember each other, and the next time they meet, no pairing is required. As long as the devices stay within range, the connection is active. Such connections are what have been used when a mobile device pairs with a peripheral device such as a headset or smartwatch. In more ad-hoc settings, this connection method becomes slow and often requiring active user input, as more security features are present in this mode. In both Bluetooth Classic and BLE, connections can be used to create more complex topologies. The most used Bluetooth Classic topology is that of a piconet. Piconets are created from one acting master device having a maximum of seven slave connections [19]. Slaves can communicate with each other through the master device.

Further, piconets can be chained with bridge nodes to create more extensive networks, as seen in the cluster network of Figure 2.3. Both the master node and the slave nodes can act as bridge nodes. This method can also be used in BLE. One step further is BLE Mesh topologies, where many-to-many connections can be established. In mesh, all the devices can communicate bi-directionally without a central point/device.

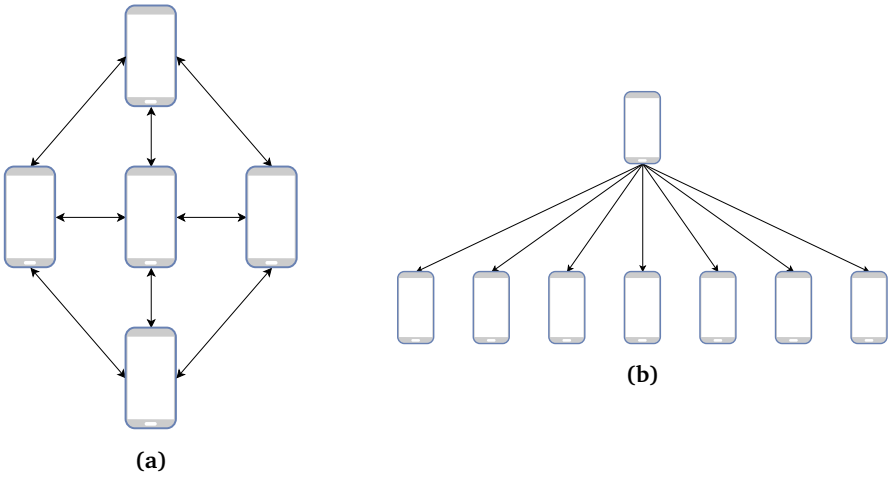


Figure 2.2: Connections and connectionless communication. Connections enable bi-directional communication whereas connectionsless primarily involves a broadcasting node sending information in one direction

2.4.2 Connectionless

In connectionless communication, a device can communicate data to any device, scan for data or devices that are within the listening range [16]. This can be used to share information with any device in range, as seen in Figure 2.2b. This kind of data transport is called a broadcast. The other devices can listen for such broadcasts and process the data on their own accord. The device sending out data has the role of the broadcaster, while the receiving node has the role of the observer. Broadcasting is the only way for a device to transmit data to more than one device simultaneously. BLE broadcast uses two different types of transport packets: advertising- and data packets. Advertising packets are used in, e.g., discovery mode, but can also be used to send smaller amounts of data [16]. Therefore, in a topology using broadcasting to transmit data, it is beneficial to place the most operation-critical data in the advertising packet. The advertising packet has a dedicated slot of 31 bytes reserved as custom data. Depending on how the advertising packets are configured, a scanning or listening node can request additional follow-up data from the original broadcaster node. By design, the broadcasting node has no way of knowing if the data reaches any nodes, and scanning or listening nodes have no guarantee of receiving any information.

2.4.3 Broadcasting and flooding

Expanding on the concept of Bluetooth Broadcasting, this section describes how broadcasting can be used to flood a network with packets and the application of this technique. Broadcasting is when one peer has to send packets to multiple destinations. If the destination peers are known, a simple solution would be to make N packets, each destined for a single node. Where N represents the number

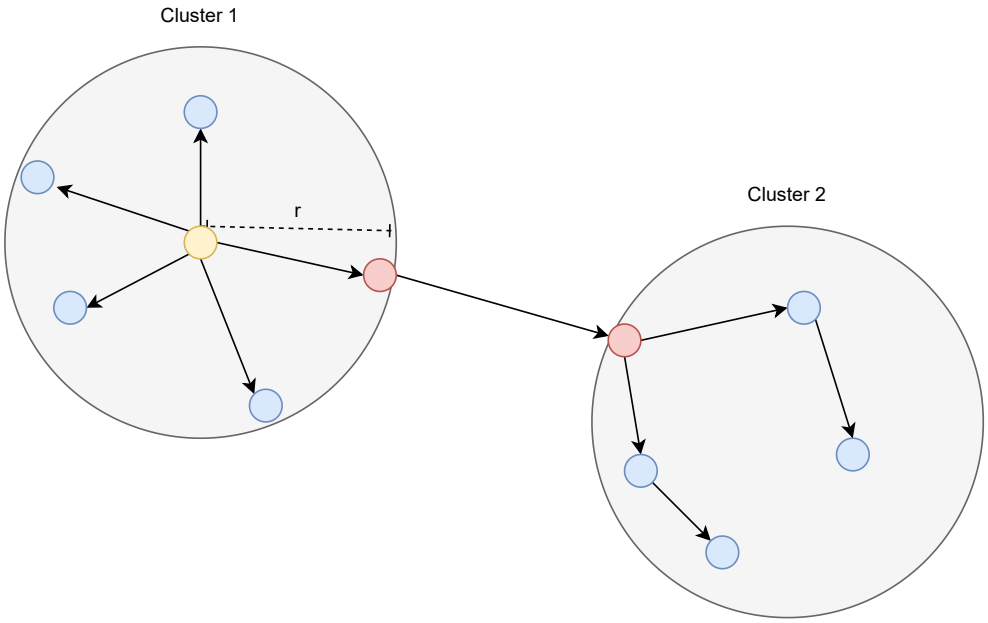


Figure 2.3: Bluetooth clustered network. The yellow node is the original packet sender. The red nodes are the bridge nodes between the clusters. The blue nodes represents reachable entities within the cluster. The arrows indicate communication direction. Each cluster sphere represents the range from origin. Not shown: Each node in the network forms it's own range cluster sphere.

of single-hop neighbours, effectively duplicating a single packet and sending each copy down a single branch. The intermediary nodes will then re-broadcast the received messages if the destination addresses do not match its own. A drawback of this method is that the destination addresses have to be known beforehand. Peers also run the risk of becoming a choke-point if it is the only peer connecting one of the destination peers. In an ad hoc network, the destination addresses is not necessarily a known property without some prior exchange. If the communication is intended for all the nodes in the network, then routing by address becomes cumbersome. A simple solution to the address problem is to flood the network with packets instead. In its simplest form, the technique is called uncontrolled flooding.

Uncontrolled flooding is when a node receives a message and then re-broadcasts the message to all its neighbors. The idea behind flooding is that any node in the network functions both as a sender and receiver. With BLE, the first sending node will broadcast the packet to all nodes in the listening range, and the next receivers will broadcast it further. In other words, given that each node has new neighbors, the reach can grow exponentially. A packet is guaranteed to reach the destination. This only works when the destination node is in range of the broadcast chain. Flooding generally has little per node overhead [20]. If the network contains cycles, some packets can be broadcasted back and fourths in-

definitely. With BLE, such cycles are bound to happen due to overlapping range spheres. One possible solution to deal with this is to include either a Time-To-Live (TTL) or age parameter in the packet. When a node process a packet, it will update the state of these parameters and forward the updated packet. This way, a peer will eventually drop the packet when the TTL or age value expires. Another way to tackle the deadlock problem is to enable a peer to determine if it has previously handled a distinct packet. This is achieved through controlled flooding. One method involves embedding a sequence number in each packet, a peer keeps track of the sequence number after processing, and if that sequence number is reencountered, the packet is dropped. Other methods include Reverse-Path Broadcast (RPB) and spanning-tree broadcast [12], but these are out of this thesis's scope.

Primitive broadcasting routing protocols do create substantial traffic within a network, especially if two nodes are communicating through intermediaries such as in flooding mesh [21]. A network may end up multiplying the number of packets resulting in a network becoming so ineffective to the point of being useless [12]. Therefore some controlling measures are required in creating a more efficient network. The ability to reach a destination node and large overhead in the form of traffic can be seen as a trade-off.

2.5 Anonymity and privacy

A decisive part in why digital contact tracing split into two factions is the disagreement of how well the centralized architecture manages privacy. As digital contact tracing operates on sensitive personal data such as medical records and data exchange with strangers, privacy should not be taken for granted as data leaks can have a considerable impact. One of the earliest traditional definitions of privacy comes from Warren and Brandeis: "[privacy] is the right to be let alone.". The initial concern of privacy was based on the separation of the public and private spheres — Protections against unwanted disclosure of private facts, thoughts, and emotions [22]. Moore [23] emphasizes the aspect of control "A right to privacy is a right to control access to, and uses of, places, bodies, and personal information.". This also expands to the specific context in which some information may be shared [23, 24]. This means that some information shared, e.g., information disclosed to the family, may not be shared with the rest of the world or give the family any right to share that information with someone else.

The current technological age has made the definition and handling of privacy more complex. Especially considering governments and public institutions need to balance national security with personal privacy [23]. Trust plays a crucial role in digital privacy management, trust being a combination of social trust and technical trust. Users of a system display a level of trust when handing over personal information to either an organization or governmental body [25] — Trusting both the fair use of transferred data and the systems processing the data. Digital privacy protection also envelopes other kinds of aspects, such as access to digital

metadata. This could be information gathered by a machine and sent to a machine, such as traffic data [25]. This metadata can be sensitive since it reveals information about the user or the system used to send data. Metadata about the user can, e.g., be assigned static Ids that can be traced back to a physical identity.

Various parts of the world threats the concept of digital privacy differently, e.g., the right to personal digital privacy has different standings in the US and Europa. With the latter having enacted strict rulings and guidelines on how personal data must be handled through GDPR[3]. The EU further emphasized their commitment to protecting personal data when the Schrems II verdict decided that the commercial transfer of personal data to the US through cloud providers became prohibited[4]. By contrast, the trade-off between personal privacy and national security is skewed more towards national security in the US [25].

Anonymity comes from the concept of un-named, where the idea is that other individuals or the society do not know who or what "you" are. This is not the same as being unknown, but the composition of traits that makes up an identity is unknown or partly unknown [26]. These traits can be split into two concepts in regards to the digital age. Firstly, the social aspect of an individual: Name, age, zip code, sex, social security number, with more. Secondly, the digital traits: Internet Protocol (IP) address, online persona, hardware, with more. Anonymity is a method for preserving privacy and enables users of a system to do so without revealing their identity. Some services require us to reveal some traits, but they also have a responsibility to protect those traits from other users [27]. When a user signs up to a forum or registers for a newsletter; they give up a trait, e.g., their email, for the ability to partake.

2.6 Privacy concerns in digital contact tracing

Centralized- and decentralized architecture in digital contact tracing have different types of privacy concerns. As mentioned earlier, the decentralized architecture was mainly created out of privacy concerns with the centralized architecture. However, the former is not void of privacy concerns either.

One of the main concerns is the revelation of a person's social graph. A social graph is simply a map of social relations. This is in essence when the contact events registered with a contact tracing application leak, and some unauthorized entity could inspect which other nodes a node has been in close contact with. Further, if the information used in creating the pseudo identifiers are leaked, then the node's real identity can be discovered [10]. In centralized models, the ID generation is performed at the server level. If the servers are compromised, the identities of all nodes would potentially be leaked.

[3]GDPR described by the Norwegian government: `https://www.regjeringen.no/no/tema/statlig-forvaltning/personvern/ny-personopplysningslov/id2340094/`

[4]Schrems II verdict summarized by the European Parliamentary Research Service: `https://www.europarl.europa.eu/RegData/etudes/ATAG/2020/652073/EPRS_ATA(2020)652073_EN.pdf`. PS: The impact of this verdict is very much an ongoing discussion

The identifier used in registering a unique individual should be a pseudo identifier. They should not be tied to a physical identity. Moreover, they should be temporary, lasting only minutes or hours. A contact tracing application needs multiple sets of identifiers to: Communicate the application type, in the contact events, and as sender identifier. Wen *et al.* [11] found that many existing implementations did not have rotating or random identifiers for either the user or application identification. The applications are then open for fingerprint attack where an agent adopts the Ids and imitates the legitimate actors. A consideration in Bluetooth communication is also the issue of static MAC addresses, where it could be possible to backtrack previous communication based on a physical device.

Privacy concerns in contact tracing are split into macro-level and micro-level concerns. At the micro-level, there are attacks against the communication protocol, i.e., Bluetooth. Attacks such as man-in-the-middle, fingerprint, data mining, and the spreading of corrupt data will currently work against digital contact tracing. These attacks are difficult to scale and, consequently, only affect a limited set of users. On the other hand, macro-level concerns surrounding the security of the central servers used. The servers collecting and generating data can be subjected to hacking due to weak security or coercion from within. Centralized-based models are more at risk from this kind of attack since the servers hold much more information than decentralized servers. Centralized servers generated all the pseudo Ids and all close contacts. Leaking this data would disclose the social graphs. Whereas the decentralized holds the pseudo Ids of infected individuals, but this is more difficult to connect to a physical identity [10].

Some academics have voiced their concern that digital contact tracing could lead to governments taking advantage of the technology and, in turn, use it to perform surveillance on the public. Further, the data could either be misused or be available for an unauthorized party, without the users knowing [1]. There is also a question regarding the role *big tech* companies should have in designing and implementing such applications. As mentioned in Section 2.1.2 companies such as Google, Apple, Microsoft, and Facebook in the west and Alibaba and Baidu in China have played a massive role in shaping the Covid pandemic response [7].

2.7 Homomorphic Encryption

Homomorphic Encryption (HE) is one of the fundamental building blocks within the field of Secure Multi-party Computation. HE enables a multi-party system to share data and perform calculations on said data without disclosing any plaintext values to any of the nodes in the network. This opens up tremendous opportunities within networks that are untrusted or with dishonest nodes.

HE allows the manipulation, i.e., arithmetic operations, on encrypted data. An encryption scheme has a homomorphic property if it mathematically ensures that the operations applied to an encrypted text will result in the exact decrypted text as if the operations were applied on plaintexts. A trending use case for this is in cloud architecture. A cloud user would want to store and manipulate data at

rest in the cloud without the cloud provider viewing the plaintext data. HE would assure the cloud user's privacy is preserved since the data is never in its decrypted state outside the user's environment [28]. Some HE implementations could also allow the cloud user to send the operation to be executed in an encrypted state, such that the cloud provider does not learn what kind of manipulation the user runs on the encrypted data — leaving the cloud provider entirely separated from the data and actions being performed by its users. Effectively giving the cloud users a complete black box in the cloud.

Encryption schemes, in general, can be split into two categories: Symmetric and asymmetric. Symmetric Encryption Schemes, often called secret-key encryption, uses some security parameters to generate a single key that is used both in encryption and decryption. Asymmetric Encryption Schemes, often called public-key encryption, uses some security parameters to generate a set of keys. A public key (*pk*) and a secret key (*sk*), the pk can be distributed publicly and is used to encrypt data, whereas the *sk* must be kept secret and is used for decryption [29]. HE schemes can be either symmetric or asymmetric. With the asymmetric version, the *pk* can be distributed and allow other nodes to use that to encrypt data and merge it into the ciphertext without interaction from the original key generator node or knowledge of the original plaintext.

HE can be divided into three levels of implementation, Partially Homomorphic Encryption (PHE), Somewhat Homomorphic Encryption (SWHE), and Fully Homomorphic Encryption (FHE) [30]. The differences lie in what kind of mathematical operations they allow on the ciphertext and how many times it can be applied without corrupting the ciphertext. See Table 2.1 for a summary of the differences. Traditional encryption schemes typically support three algorithms: Key Generation, Encrypt, and Decrypt. HE must support an additional algorithm: Evaluate; this is a set of operations (multiplication, addition, subtraction, with more) that can be applied to the ciphertext. The three levels of implementation will have differences in the Evaluate algorithm by definition, and various implementations within each level can also have differences. Since a homomorphic property is not an algorithm, but a mathematical proof, leaving many valid implementations.

Table 2.1: The different levels of Homomorphic Encryption [30]

Type	Description
Partially Homomorphic Encryption	Only allows the usage of one operation, but that can be applied to the cipher an unlimited number of times
Somewhat Homomorphic Encryption	Allows the usage of some operations, but these can only be applied to the cipher a limited number of times
Fully Homomorphic Encryption	Allows all operations and these can be applied to the cipher an unlimited number of times

Operations in SWHE schemes can only be applied a limited number of times. This is due to the fact that Encrypt and Evaluate add some noise parameter to the ciphertext and increases the size of the ciphertext. The more operations applied, the higher the noise will become, and if the noise level is too high, the decrypt will no longer return a reliable plaintext. In other words, the resulting plaintext will not correspond to the answer given by the operations over the plaintexts. A possible negation strategy for this is to allow the scheme itself to encrypt the ciphertext such that the cipher is encrypted twice and then decrypt the inner layer. This way, the original noise is removed, and as long as the re-encrypt adds less noise than originally introduced, more operations can be added without compromising the output of decryption [29].

Encryption schemes have some attributes that must be optimized for widespread use, these being: Secure, efficient, and low complexity. Security is needed such that an attacker cannot break the encryption. Encryption schemes are not necessarily unbreakable, but breaking them requires an extreme amount of resources such that cracking the scheme becomes unfeasible. An efficient scheme is needed for usability. If encryption or decryption takes an exceedingly long time, the user will typically prefer usability over security. Low complexity enables the developers and users to implement and use it correctly. If, e.g., the mathematical foundation of the scheme is too complex, the implementations will become untrue to the design and thus flawed [30]. This can lead to situations where an implementation can be subjected to vulnerabilities not foreseen by the designers, and introduced unknowingly by the developers. Alternatively, the system can be challenging to use or allow the users to take shortcuts, weakening the security.

PHE and SWHE schemes are being used in various commercial applications presently. FHE, on the other hand, only has some commercially used implementations, but they remain mainly unused in real-world applications due to the lack of efficiency and very high complexity. In FHE key generation, encrypt, decrypt, and re-encrypt often execute in the magnitude of minutes and hours and the size of the public key in the magnitude of MBs and GBs [30]. This level of implementation is therefore only used in highly specialized fields that need the achieved functionality and are able to tolerate the mentioned performance metrics. PHE, on the other hand, is easy to implement and has low overhead, and is already in use in various application [31].

To summarize, of the three distinct levels of HE, only PHE and SWHE are currently the available schemes to use in real-world scenarios. These offer the most worthwhile trade-off between security, efficiency, and complexity. It is an algorithm's ability to preserve privacy while still allowing computation within the limitation of mobile devices and BLE that we will explore further in this thesis. In the next section, we will present the Paillier scheme.

2.8 Paillier cryptosystem

The Paillier cryptosystem was proposed in 1999 by Pascal Paillier and is based on the *Decisional Composite Residuosity Class Problem (DCR)* which is a type of number-theoretical computational hardness assumption. This problem is based on the belief that computing N-th residue classes are computationally difficult [32]. For context, if we define a set $\mathbb{Z}_n$ of non-negative integers less than n.

$$\mathbb{Z}_n = \{0, 1, ..., n-1\}$$

Each integer in $\mathbb{Z}_n$ represents a residue class. Integers that have the same reminder r when divided by m are in the same residue class [33]. The DCR problem is based on the difficulty of identifying an N-th residue $mod\, n^2$ from a random element of $\mathbb{Z}^*_{n^2}$ [34]. Paillier is an asymmetric encryption scheme of type PHE and has an additive homomorphic property, supporting addition and multiplication on a ciphertext. The only real operation applied is multiplication, with addition being a special case of multiplication with 1. The system can be broken into key generation (Listing 2.1), encryption (Listing 2.2), decryption (Listing 2.3), and eval (Listing 2.4).

A comparative study by Farah *et al.* [35] on the performance of Paillier found that for plaintext of size 68KB, the time it takes to encrypt stays below 500ms. However, the performance of both encrypt- and decrypt time on plaintexts larger than 124kb decreases rapidly, making the overall throughput on larger plaintexts worse than that of, e.g., RSA. However, for smaller plaintexts, Paillier stays efficient and stable. Paillier also has the property of plaintext size matching the size of an encrypted cipher, meaning the objects generated by Paillier should not grow, given the plaintext occupy the same or more space than the generated keys.

Code listing 2.1: Paillier Key Generation Algorithm copied from [28]

```
Key generation
    1. Select two large prime numbers p and q where gcd(pq,(p−1)(q−1)) = 1
    2. Calculate n = pq
    3. Calculate  λ = lcm(p−1,q−1)
    4. Select g as a random integer where g ∈ Z*_{n^2}
    5. Define L(x) = (x−1)/n
    6. Ensure n divides the order of g by checking the existence of the
    following modular multiplicative inverse
    7. u = (L(g^λ mod n^2))^{−1} mod n

    Public Key = (n, g)
    Private Key = (λ, u)
```

Key generation is what provides the scheme security, and as such, the security is directly linked to how large the selected initial key generation primes are.

Code listing 2.2: Paillier Encrypt Algorithm copied from [28]

```
Encryption of message M where M ∈ Z_n
    1. Select r as a random integer where r ∈ Z*_{n^2}
    2. Calculate cipher = g^m x r^n mod n^2
```

Code listing 2.3: Paillier Decrypt Algorithm copied from [28]

```
Decryption of message cipher where cipher ∈ Z_n
    1. Calculate M = L(cipher^λ mod n^2) x u mod n
```

Code listing 2.4: Paillier Eval Algorithm

```
    1. Calculate C_3 = C_1 x C_2 mod n^2
```

Using the described algorithms we can apply them to perform an practical demonstration of how the Paillier cryptosystem works[5]. Note that to increase the readability, relatively small numbers have been chosen for p and q. In an real-world application much higher numbers must be used to achieve any kind of proper security.

The first step is to apply the Key generation algorithm, Listing 2.1, to generate our Public- and Private key.

1. Select $p = 23$ and $q = 31$, $gcd(713, 660) = 1$
2. $n = 713$
3. $\lambda = lcm(22, 30) = 330$
4. $g = 3226$
5. $u = (L(3226^{330}\ mod\ 713^2)^{-1}\ mod\ 713 = 38$

$$PublicKey = (713, 3226)\ and\ SecretKey = (330, 38)$$

In the next step we will chose a plaintext message to encrypt with the algorithm from Listing 2.2.

1. Select message $M_1 = 228$
2. $r_1 = 312$
3. $C_1 = 3226^{228}\ x\ 312^{713}\ mod\ 713^2 = 137456$

In the next step we will demonstrate the additive property of the Paillier scheme by choosing another plaintext message, encrypt it using the Public key, and add it to the original ciphertext by using the eval function from Listing 2.4.

1. To be able to add a cipher to c_1 we first have to use the PK to encrypt a new message $M_2 = 22,\ r_2 = 534,\ C_2 = 333166$
2. $C_3 = C_2\ x\ C_1\ mod\ n^2 = 137456\ x\ 333166\ mod\ 713^2 = 261069$

The last step is to apply the decrypt algorithm, Listing 2.3. The resulting ciphertext C_3, consist of the $C_1 and C_2$, as seen when decrypting, the resulting plaintext matches up to the original two plaintexts.

1. $m = (L(261069^{38}\ mod\ 713^2)\ x\ 38\ mod\ 713 = 250$

[5] Some excellent sources on practical Paillier is `https://asecuritysite.com/encryption/pal_ex` for rundown of a Python implementation, and `https://cryptocalc.giondesign.com.au/` for interactive calculators for all the steps and standard modulus used in the calculations.

2.9 Summary

In this chapter, we have presented the theoretical groundwork that we will use to design and prototype a proof-of-concept system that can perform privacy-preserving distributed calculations in digital contact tracing.

Digital contact tracing serves as our primary use case and is essential in the problem description. As mentioned, a substantial drawback with decentralized-based architecture is that the data collecting agencies lose some of the data support. This architecture is nevertheless an improvement towards better privacy protection compared to centralized-based architecture. The definition and importance of privacy and anonymity to digital contact tracing have been presented. We have also covered the necessary background to understand how BLE Broadcasting can be used as a communication medium to deploy controlled flooding to stateless share data with surrounding peers. Homomorphic Encryption can be utilized to achieve distributed calculations without compromising on individual node's privacy. This can be an entry point to provide a better data foundation without breaching the aforementioned privacy concern.

In the next chapter, the research methodology is presented.

Chapter 3

Methodology

The following chapter presents the methodology and design we will use to investigate our research questions. This research will use a mixed research design, utilizing both aspects of qualitative and quantitative research. The solution proposed in this thesis is formulated through the exploratory findings in the background theory. We will gather quantitative data using a developed computer simulator to determine the performance of the proposed solution in terms of complexity, efficiency, and privacy.

3.1 Research design

A set of tasks and two research questions were presented in Chapter 1 Section 1.2. T1 and T2 are covered in Chapter 2, whereas T3 and T4 are going to be covered in the subsequent chapters. These remaining tasks will need different methodologies. T4 can be seen as a case study using the findings from Chapter 2 as the basis for formulating a possible solution to RQ1. This means taking existing theories in different fields and combine them into a novel solution to the problem. Table 3.1 displays the tasks in relation to the research questions and their significance.

The exploratory question asked in RQ1 can be answered using qualitative methods. The purpose is to establish a novel solution to a new use case, where there is very little information to quantify. It is more a question of feasibility than expanding existing theory. We need to find existing theories and information, and consolidate these so that the expected functionality of RQ1 can be accomplished. If this was an already established field, we could compare the performance of the different solutions, but since this is not the case, we have to propose a completely new solution.

In order to answer RQ2, we have to perform T3, where we intend to gather data using a computer simulator and, with the different simulation runs, try to identify cause-and-effect relationships between the runtime- and configuration variables. A computer simulator is a piece of software that is used to emulate a real-world scenario. This enables us to control all the aspects of the scenario and

supply the simulator with a near endless combination of starting data and configurations. The main benefit of this is that we can emulate different scenarios and extract distinct characteristics of the modeled system. A simulator allows us to rapidly try out new theories by re-running a case with new parameters. Further, as it is a controlled environment, we can more precisely and autonomously collect data. During the modeling of the proposed protocol, see Chapter 4, we will consider the plausible factors that might impact the performance of the system, and then by using an experimental research design, try to control for these factors.

According to Leedy and Ormrod [36], a correlational study tries to find out how much change in one variable is associated with the difference on one or more other variables. E.g., a correlation exists between two or more variables if a change in one lead to the others decreasing or increasing in an statistically significant manner. For RQ2, this means identifying particular characteristics about the proposed solution and see how these impact each other. Further, working hypothesis developed during experimenting will be tested and expand the proposal iteratively. The tool used for analyzing the data will be *IMB SPSS*[1]. The whole process can be viewed as the sequence illustrated in Figure 3.1.

Table 3.1: Research task summary

Task	Contributes towards	Significance
T1	R1/R2	Categorizes the enabling and limiting factors that make up digital contact tracing. The findings function as the theoretical foundation for building the design and architecture.
T2	R1/R2	Investigates the theories that will enable us to achieve privacy-preserving distributed calculation.
T3	R2	An iterable task that enables us to determine the best optimizing factors for the performance of the proposed solution.
T4	R1	The product of the research will be the optimized proposal with feasibility factors.

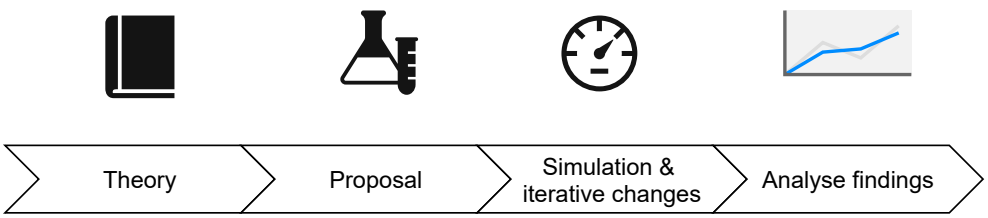


Figure 3.1: Overall research design

[1] https://www.ibm.com/analytics/spss-statistics-software

3.2 Evaluating performance

Performance is an essential concept in regards to the research questions, but the term can be defined in many ways. The description of how this research defines performance is given in the following bullet points:

- The ability to minimize overhead. If resources are spent in processing or communication that none will be able to consume/utilize, it is considered wasted resources and a overhead. By minimizing the overhead, more resources will produce value for the system, thus less overhead.
- The performance also encapsulates the speed it takes to solve a problem. If the solution is not able to finish quickly enough, the solution displays poor performance. What is sufficient resolution time depends on the environment and application.
- Performance can mean to both minimize or maximize certain traits depending on the context or measurement. E.g., high performance in messaging is the ability to process as many messages as possible in the shortest period. On the other hand, high performance in path navigation is the ability to minimize the distance an actor has to travel before reaching its destination.

The following descriptors from the research questions is also operationalized: Complexity and efficiency. Complexity is an intrinsic attribute of a system. Systems with high complexity can be more difficult to understand or describe, whereas systems with low complexity are easy to understand. Complexity can also be seen as an additive property; the more layers or facets a system has, the higher the overall complexity becomes. Efficiency is the ability to optimize operations, this can either be to maximize or minimize them. Systems with high efficiency can process data quickly, without unnecessary steps, and without overhead.

3.3 Internal validity

The simulations will be completely computerized and can be viewed as a controlled setting where we can account for every change in the simulation behavior. Further, we will only change a single variable or elaborate on which variables change and why, per simulation run. This gives us the ability to state the reason(s) why an eventual cause-and-effect relationship exists accurately. A possible threat to our internal validity is the hardware on which the simulations run. All computers have an element of randomness to them, either being memory accesses or context switches that can impact the observed results. We intend to avoid this situation by running each simulation configuration multiple times, such that eventual outliers or abnormalities can be handled accordingly. Additionally, every simulation is done using a single computer, such that we do not introduce errors by using multiple sets of hardware.

3.4 External validity

Since our research is applied in a very controlled environment, the external validity can be questioned. In software development, a problem rarely has a single possible implementation, and our research is the same. We merely propose a solution that fits our criteria, and that is feasible within our limitations. The real-world aspect comes from the assumptions we state about the existing environment. These assumptions are rarely enough to capture the complex nature of reality, i.e., a digital model of reality is often simplified to either: Make the model easier to understand/less complex or more feasible as complete modeling requires sizeable computational power. Further, assumptions can be made about the environment that is only partially true or based on some misconception. This can impact how the findings can be generalized to a broader use case or a real-world scenario.

Chapter 4

Protocol design

4.1 Distributed Calculation Protocol

In Chapter 2 we explained a set of theories and concepts that will help in answering our problem description. The following proposal is a product of combining all the provided background theory into a single protocol. The proposed protocol, Distributed Calculation Protocol (DCP), is based on expanding existing distributed contact tracing designs to do privacy-preserving decentralized calculations using Homomorphic Encryption cryptosystems via BLE network flooding. Each contact tracing application-holder is viewed as a single entity, henceforth called a node. Via the contact tracing, each node accumulates various data that is interesting for a central health agency to collect. Existing contact tracing data collection is concerned with contact events but can be expanded to include other metrics such as the number of times a node leaves the residence, time spent outside the residence, visits to businesses or other residential areas, number of reoccurring contact nodes, with more. The goal of DCP is to anonymize this data such that it can be viewed collectively without breaching individual privacy. Anonymizing this data can be achieved by calculating the mean value over a set of nodes. In DCP, each node holds a *data object* reflecting the results of some completed or uninitiated *calculation process*. A node can initiate a calculation process where the data object is encrypted using Homomorphic Encryption, giving us a *calculation object* and then broadcasted in conjunction with the node's public key. Depending on the type of calculation, listening nodes will contribute with their data and re-broadcast the calculation object to other nodes. When the data is considered adequately anonymized, the original broadcaster will be able to decrypt the calculation object and update its own data object with the new calculated data. This data object will then reflect what the node knows about its environment.

The described scheme will allow us to perform distributed calculations in a connectionless network without the participating nodes knowing what data the other nodes contribute.

In the following chapter, we establish the requirements and explain the logical components making up the proposed protocol. Following that, we present the

simulator that has been developed to test the proposal.

4.2 Protocol limitations

This protocol is intended to work in decentralized systems and will not introduce the usage of additional centralized servers or third-party systems. As stated in section 2.1.2, most digital contact tracing applications today use BLE as the transport protocol. This protocol does not limit itself to BLE but any transport protocol that supports broadcasting and flooding. However, some assumptions are drawn on the anticipation of a BLE implementation, such as range, transport- speed and size.

4.3 Requirements

As we stated in Section 2.5, privacy revolves around the idea that an individual has a right to keep specific attributes hidden from the public. One reason why the digital contact tracing community divided into two architecture factions was privacy concerns in centralized architecture. The goal of the proposed system in terms of privacy must be to not degrade the existing improvements in privacy preservation and not introduce new attack vectors within decentralized architecture.

Based on the problem description and description of DCP, a set of requirements have been established, both related to functional and privacy. Privacy and functional requirements (Table 4.1 and 4.2) describes the things the proposed protocol must satisfy. Additionally, some of the known attack vectors on the Bluetooth stack and existing digital contact tracing applications were discussed in Section 2.6.

Key concepts that are left implementation-specific, i.e., not specified in the protocol, are the choice of communication technology, the choice of the Homomorphic Encryption scheme, and the calculation process trigger criterion.

4.4 Calculation Object

Within our service, the term *Calculation Object* denotes a specific node-held object. This object holds all the necessary data for achieving distributed calculation. A node will encode this object and broadcast it to nearby nodes. The required building blocks of this object are discussed later in this chapter. Some properties are required, while others are dependent on the type of calculation performed by the nodes, which means that an implementation can expand the calculation object to support multiple types of calculations involving different properties.

The Calculation Object will reflect the information that a single node knows about its environment. Concerning digital contact tracing, information about the environment can be the size of the contact list, geographical data, time spent outside of the residence, to name a few. Each node in the system will have one or more Calculation Objects depending on how many types of calculations the nodes

perform. Further, each node in the system can initiate a calculation process, i.e., distribute its object to others.

Table 4.1: Privacy requirements

#	Requirement
1	The plaintext input must not be visible to the other nodes, or deducible based on some property of the input.
2	The calculation object cannot contain any identifiable properties, e.g., historical records of input into the calculation or any identification from the node, e.g., pseudonyms or other identification. All input data has to be non-identifiable numerical data.
3	The original sender cannot process the calculation object if the number of involved nodes is too small to anonymize the data sufficiently.

Table 4.2: Functional requirements

#	Requirement
1	A node should periodically broadcast the calculation object to nearby peers.
2	A node should be able to listen to incoming broadcasted objects.
3	A node should be able to process a calculation object by contributing with its own data and then re-broadcast the modified calculation object.
4	The original sender should be able to decrypt the accumulated calculation object.
5	The nodes must be able to inspect the state of their own calculation object and data object.

4.5 Communication

In this section, we explain how the proposed solution will handle the transmission of messages in the network. It defines how data will be transported between the nodes. Section 2.1.2 revealed that almost all of the existing digital contact tracing applications today use BLE and broadcasting to communicate. These applications exchange the contact tracing identifiers and various other parameters needed for proximity measurement with the other nodes. It is natural to use BLE and broadcasting in DCP as a starting point since we can utilize existing logic of common contact tracing applications.

Broadcast is a fast and easy way to communicate with multiple nodes at the same time as it is connectionless. It saves time by not having to perform handshakes and avoid the security aspects of automated pairing and bonding. Broadcasting also limits the amount of information each node knows about the other, increasing privacy. BLE and broadcasting enable us to share the calculation object

with other nodes, with a couple of assumptions. The nodes must process the calculation object with such an efficiency within a local cluster of nodes so that the original sender has time to receive the calculation object back with the acceptation criterion fulfilled, more details are found in Section 4.10.

Further, the size of data packets has to be feasible within the boundaries of BLE. Data size is not a problem when using a connected topology as the transmission can be segmented into multiple packets. However, since we are only using connectionless broadcasting, transmission segmentation cannot be used as the sender does not know which nodes need which segment, i.e., the sender cannot direct traffic towards a specific peer. The recipients can also not request missing segments, which makes performing calculations based on cryptography impossible.

As broadcasting is one-directional, the flow can be broken into two steps - handling a Calculation Object and broadcasting of the Calculation Object. As seen in Figure 4.1, all receiving peers repeats this process. The original sender can also see the re-broadcast but should reject the object if there have been too few participating nodes.

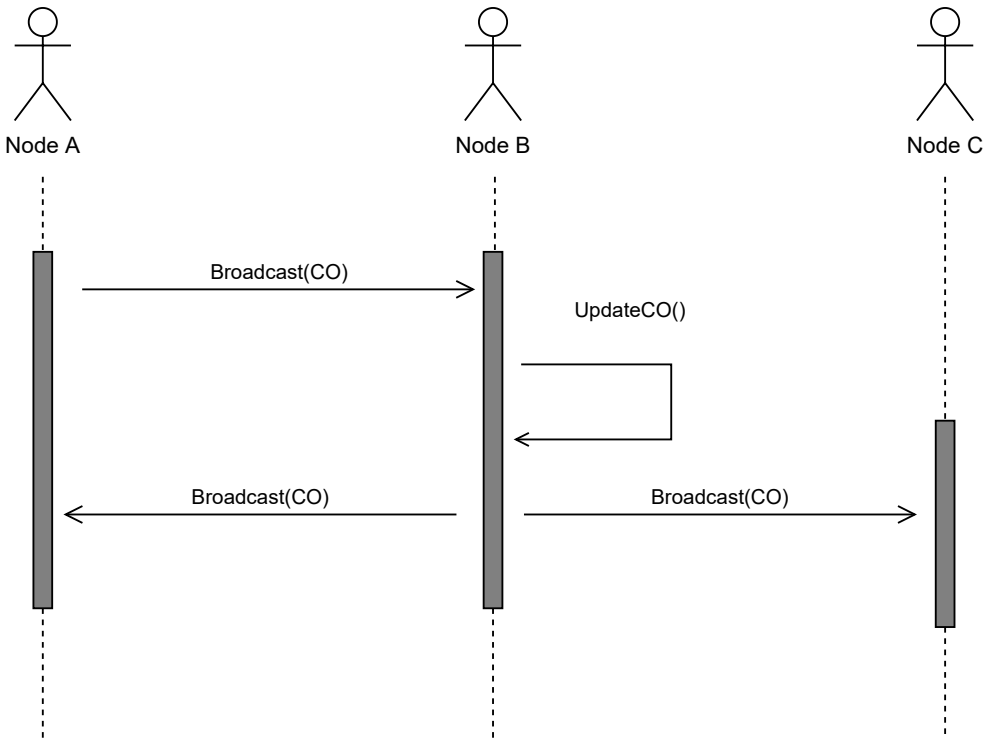


Figure 4.1: The two main node processes - broadcasting and updating the calculation object

4.6 Triggering the process

A trigger criterion dictates on what condition a node should initiate a calculation process. Trigger criteria can vary depending on what type of calculation process being performed. If the process uses some internal state as plaintext in the calculation, the process can be triggered when the internal state receives an update. This can be a good solution if the particular used property does not receive frequent updates. Another option can be to run the calculation process on a timer, meaning that the process will trigger with a specific interval. A third option is to have it trigger depending on how often the listener logic triggers within a particular timeframe. A reactive trigger mechanism will ensure that the process is not triggered when the node has no or few nearby nodes and maximize the number of potential nodes taking part in the calculation process. Only triggering the process when many nodes are nearby can propagate to the other nodes and lead to multiple nodes starting a calculation processes. Potentially growing the network noise exponentially. Table 4.3 summarizes the mentioned trigger criterion with the benefits and drawbacks of each method.

The choice of trigger criterion can be viewed as the trade-off between performance and consistency. Whenever a node initiates a calculation process, the node adds noise to the network. If many nodes are often processing, the network could become very busy, leading to wasteful computation. A node rarely processing will mean that the data object is not up-to-date with the node's environment. The choice of triggering criterion is left to the implementation, but the choice can potentially significantly impact the solution's overall performance.

The logic a node performs when initiating a calculation process can be seen in Figure 4.2.

4.7 Handling the Calculation Object

After the initial broadcaster has sent out its calculation object, it is the nearby nodes' job to decode the data and to handle the received calculation object. The handling logic involves performing the assigned calculation by adding some data to the existing object and re-broadcast it. A single node should not contribute to a calculation object more than once, or else the validity of the object becomes weak.

Since the Calculation Object is broadcasted, all nodes within range can inspect the object's content. This would lead to the exposure of private data, e.g., the data necessary to perform a calculation. An important problem to address is how to shield the contents from this kind of insight while still enable the necessary manipulations. Our solution utilizes Homomorphic Encryption (HE). As described in Section 2.7, HE enables us to perform arithmetic operations directly on a ciphertext. A prerequisite for making HE work is that the original sender node has to distribute a per calculation process generated public key alongside the rest of the data.

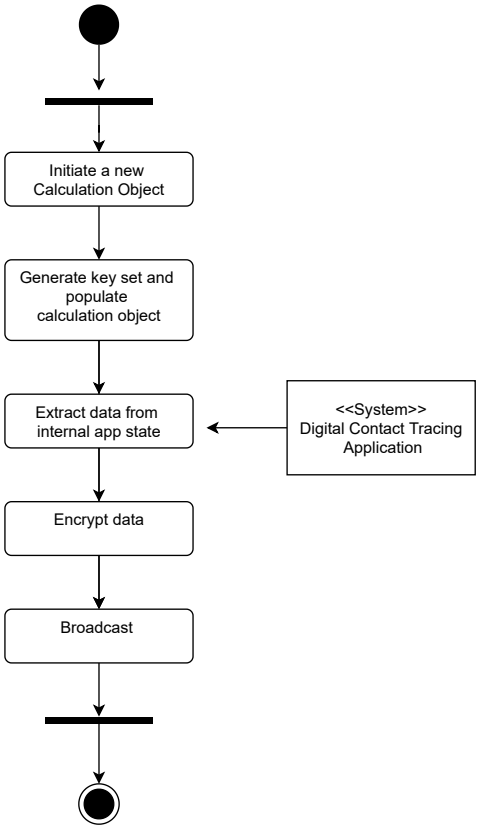


Figure 4.2: Original Calculation Object sender process

4.8 Encryption

As we presented in Section 2.7, three different levels of Homomorphic Encryption schemes exist. These allow for different types of operations and limits on the number of times the operations can be applied. FHE schemes are currently not feasible in our proposed solution since they are not efficient enough and with a high degree of complexity. Making the schemes not practical in a hand-held device application. SWHE can only run the operations a limited number of times before the ciphertext size and the noise parameter become too great and corrupts the possibility of extracting a reliable plaintext.

SWHE schemes might not support our use case; however, the additional support of operations other than multiplication and addition might mean that more complex or diverse distributed calculations can be possible. PHE supports either multiplication or addition, and these operations can be applied an unlimited number of times. Having the encryption stay reliable no matter how many times it has been manipulated means that a single calculation process can, theoretically, involve an unlimited number of nodes. Therefore, the system does not have to handle the use case of unreliable cyphertexts. Further, by enabling many nodes to

Table 4.3: Possible Calculation Process triggers

Criteria	Benefit	Drawback
Update frequency	Can ensure that the data reflected in the environment are always the most up-to-date information.	If based on a frequently updated property the network can become unnecessarily noisy.
Interval	Predictable and consistent, independent from the node's environment. If triggered with few nearby nodes, the process should end quickly with relatively little overhead.	The process will be triggered even when a node is in a typically low density area such as the residence leading to some wasted resources.
Node density	Increases the chance of a node successfully completing a calculation process and maximizes the data collected	If a node infrequently visits areas with a high node density the process may rarely trigger. I.e., different areas may require different node density "sensitivity".

participate we increase the overall quality of collected data. High data quality can be viewed as properly anonymized and produces a more precise mean value. The proposed solution does not limit or dictate exactly what kind of calculation an implementation can support. To recap, both SWHE and PHE can be good candidates to use, depending on the implementation-specific use cases.

Further, various implementations of the PHE and SWHE schemes might be used to achieve different calculations. It is conceivable that an implementation can support an array of different HE solutions to not limit the system to certain types of calculations. However, only one encryption scheme can be used in a single calculation process, these cannot be mixed. The value of Homomorphic Encryption in our proposal lies in its ability to anonymize collected data, e.g., the data cannot be tracked to a single origin after a calculation process has finished, not by the original sender nor any actor inspecting the data.

4.9 Reception handling

When the initial sender node first broadcasts the calculation object, multiple receivers might start processing the object simultaneously. Since we do not want any nodes to update the calculation object multiple times, each node must track

which objects it has processed. A problem arises when multiple nodes are processing the same calculation object. When the nodes re-broadcast, multiple nodes will reject the incoming calculation object as they have logically already processed it. This scenario can quickly result in a deadlock situation or infinite loop where the nodes send the object between them with no update, as illustrated in Figure 4.3. A solution to this is to view the initial recipients as separate branches. This way, multiple branches will exist and, in turn, receive updates and a higher chance of returning to the original sender. The initial sender does not know in advance how many nodes will listen to the initial broadcast, so the branch Id assignments must be delegated to the initial receivers. This can be accomplished by the initial sender leaving the branch Id property on the calculation object empty.

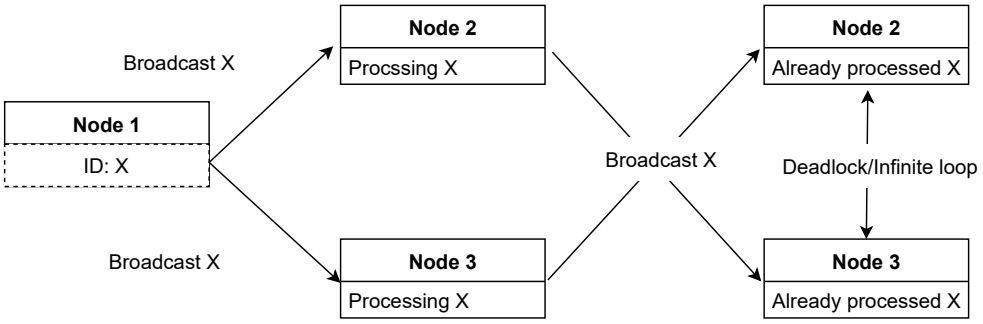


Figure 4.3: Calculation Object single Id branching problem

If the initial broadcaster moves out of range from all participating nodes, a mechanism must be implemented to drop the calculation object gratuitously. Our solution includes a Time-To-Live (TTL) counter in the calculation object, and each receiving node decrements this counter. When the calculation object's TTL value reaches zero, the node currently processing it will drop the object by not re-broadcasting it. The optimal value of the TTL parameter is examined in Section 5.2.2.

A receiving node will re-broadcast a calculation object if the node has either: Updated the state of the calculation object or if the node has already processed the calculation object.

4.10 Return-to-sender

Before the original broadcaster can accept the calculation object, the system must guarantee that the data is adequately anonymized. The original sender cannot decrypt the Calculation Object before enough nodes have partaken in the process. If the Calculation Object returns to the initial broadcaster after a single node has partaken, the receiving node can quickly infer the single node's input values, thus breaching privacy. If two nodes have contributed, the original sender can assume the input values of the two other nodes. Following this logic, the more nodes involved in the process, the less accurate the original broadcaster's estimation of

the input values becomes. The number of involved nodes before the sender can accept the calculation object depends on the type of calculation being performed and how easy it is to guess the input values. E.g., if the input values contain elements of either randomness or are limited to a fixed collection, the knowledge gained from inferring the input value diminishes. The optimal decrypt threshold, e.g., most privacy-preserving or most efficient, is explored later in Section 5.2.3.

When the Calculation Object satisfies the requirement of involved nodes, the original sender can decrypt the calculation object and update its own internal state. If the calculation object does not satisfy the decrypt threshold, the broadcaster node can either drop the packet early, similar to when the TTL value reaches zero, or re-broadcast the calculation object again. The effect of this choice is explored later in Section 5.2.2. Figure 4.4 displays the complete process of handling a Calculation Object, both from the original broadcaster's and the participating nodes' perspective.

4.11 Updating the Calculation Object

When the original sender node receives the Calculation Object back, and the participation counter is high enough. The node can decrypt the object, the output values contained in the calculation object will be used to update the node's internal values, such that the next time a calculation process is initiated, a new calculation object will be populated with the previous result, plus any internal updates the node have done since the last process.

In broadcasting, multiple processed objects might return to the initial sender at different rates. This uncertainty introduces a challenge as the different calculation objects do not contain any record of which nodes have partaken in the process. If we were to merge all the received objects' results, duplicates would happen and then, in return, may void the overall calculation. This is because if multiple duplicates exists, the duplicate entries will have a greater impact on the mean calculation, than the single entry nodes, and it is impossible to counter this effect. As the initial node can never see the inputs used, removing duplicates is not an alternative. A possible solution is to say that first come, first serve and only accept the first received calculation object that satisfies the decrypt threshold. This will mean that most of the branches have performed wasteful work, and depending on the decrypt threshold, the contribution of many nodes will not be included in the resulting data. A second option is to keep accepting objects and choose the object with the highest involved node counter and update the initial sender's internal state. This ensures that the least amount of work is wasted.

4.12 Extracting data

As there is no central point of trust in the system, data extraction from the system must be performed by a node in the network. The proposed solution does not

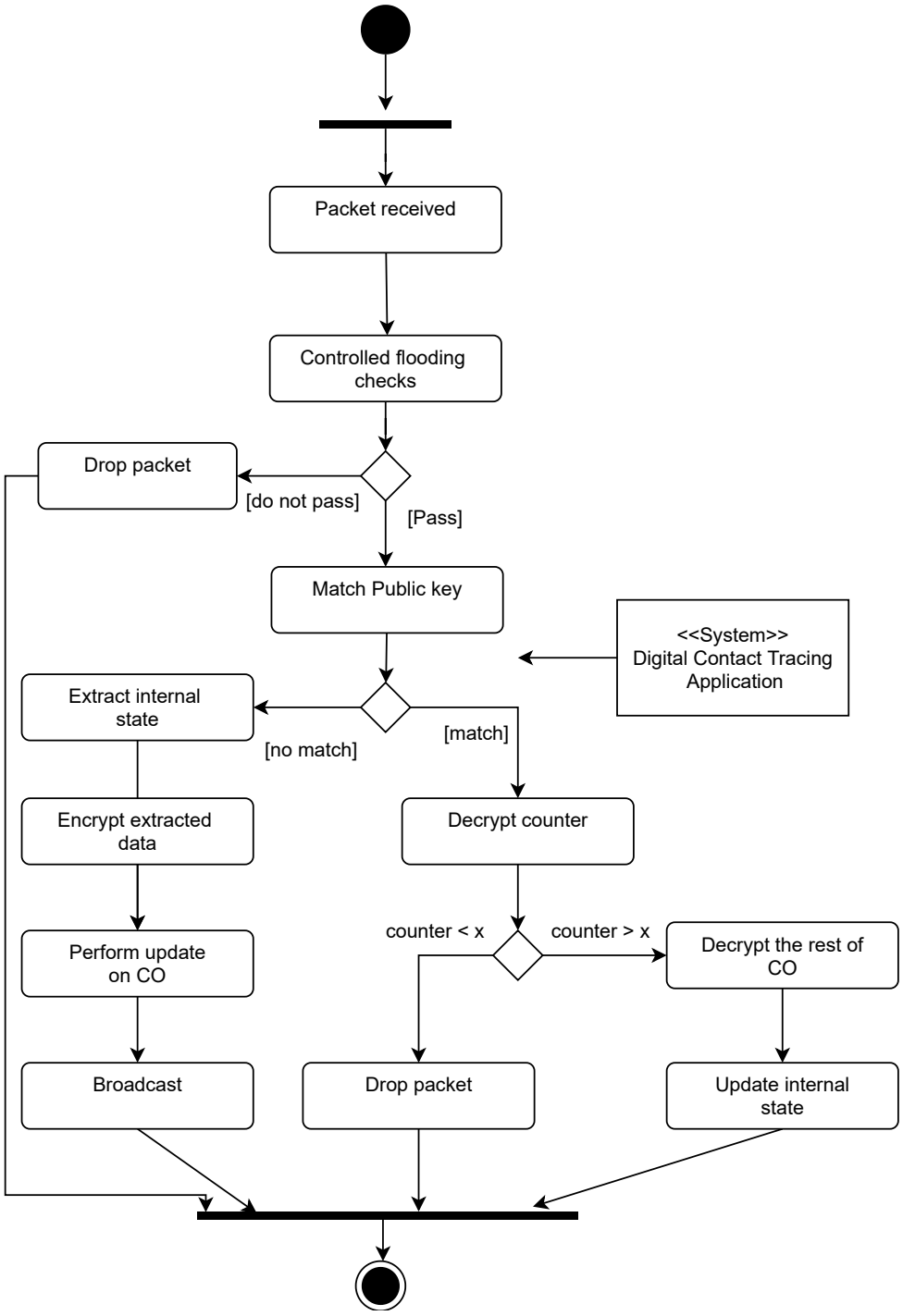


Figure 4.4: Recipient Calculation Object process

envision a way to extract data from the mass of the system centrally. Each node in the system will own a Calculation Object that reflects the information which

a single node knows about its environment. As such, any node in the system can extract the localized data at any time. Further, the nodes have to perform a calculation process to obtain data in the first place. The strength lies in combining the data from multiple calculation processes by performing their own. Thus, the calculation object may contain data from multiple previous calculation processes and be of much more considerable significance than the nodes involved in a specific calculation process.

An alternative to the "in-network" extraction, especially considering contact tracing, is implementing conditions for the node to upload its calculation object to an existing central server. This should be reasonable to do, due to the calculation object becoming anonymized through the calculation process. If a node satisfies some condition, it could upload the calculation object without breaching the privacy of the contained data. This can be done by having the node upload the calculation object at the same time as uploading the pseudonyms to the central agency. On the condition that the app-holder have recently been diagnosed with some disease. In other words, submitting the calculation object becomes an added side-effect when performing the normal information flow in decentralized contact tracing. This could also ensure that the central agency receives data relevant to a specific outbreak. If we want to keep the data transparent and public, a solution could be to have nodes uploading to a public repository instead. Now all nodes can inspect all data, and all nodes knows what happens to the data. A downside of this is that it creates risks at the macro-level instead of the otherwise micro-level system. Our solution assumes the implementation of the first described extraction method, but we do not dismiss the feasibility of the two others.

4.13 Simulator implementation

The previously described protocol needs to be tested according to our research questions. The simulation tool is written using the Golang programming language[1]. Golang is a fast, statically-typed, compiled general-purpose language usually used in server programming. The reason why Golang is used so much in server programming is that it performs well with concurrency. This feature allows a program to run multiple processes at the same time through Goroutines. A single Go application can run millions of Goroutines concurrently. This is used in our simulation to separate, as much as possible, the processing performed by each node. A node should not have to wait for another node to complete some processing to be able to advance its own processing. The simulation tool portfolio consists of a library containing the DCP logic, a simulator to emulate a network and environment, and a data processor to convert the resulting simulation reports to more manageable data and various helper programs[2].

[1] `https://golang.org/`

[2] The developed programs and code can be reviewed at `https://github.com/DCP-DCT`

4.13.1 Nodes

Each node in the simulation is an instance of the structure CtNode, Listing 4.1. The CtNode structure implements the interface ICtNode, Listing 4.2. All the nodes will process concurrently by having the listeners and broadcast functions running as Goroutines. Each instance of CtNode also contains a reference to the WorkerPool. This variable holds a limited number of available workers, where the workers are used in conjunction with Goroutines. This is to safeguard against the complete resource depletion in the simulation hardware. If all the workers are busy, each worker have the ability to form a queue, so no processing can be dropped.

The Config attribute is our way of adjusting the various run and experimental parameters, which will dictate how the solution should behave, more on this in Section 4.13.4.

The calculation process itself involves using the length of the Ids array as input in the calculation. In contact tracing, this is used to represent all the node's registered contact events. The modelled use case is described more in-depth in Section 5.1.1.

Diverting from the proposal, each node in the simulation keeps track of some runtime metrics. These consists of the number of times all the individual actions and conditional checks being performed by the node when performing the HandleCalculationObject routine; more on this in Section 4.14.

Code listing 4.1: Node structure

```
type CtNode struct {
        Id               uuid.UUID                  `json:"id"`
        Do               DataObject                  `json:"data_object"`
        Co               *CalculationObjectPaillier `json:"calculation_object"`
        Ids              []string                    `json:"ids"`
        ProcessRunning   bool                        `json:"process_running"`
        HandledBranchIds map[uuid.UUID]struct{}     `json:"-"`
        TransportLayer   *ChannelTransport          `json:"-"`
        Config           CtNodeConfig                `json:"config"`
        Diagnosis        *Diagnosis                  `json:"diagnosis"`
        WorkerPool       *grpool.Pool                `json:"-"`
}
```

Code listing 4.2: Node interface

```
type ICtNode interface {
        InitRoutine(Prepare) error
        Broadcast()
        Listen()
        HandleCalculationObject(interface{}) bool
        Print()
}
```

4.13.2 Calculation Object and processing

Each node creates a single instance of 'Calculation Object' per calculation process. As this is the object transmitted between the nodes when processing, each run is

started with a new set of public- and secret keys. The main properties of this object are a ciphertext, a counter, the public- and private key, an Id, and a branch Id. A node will convert this object to bytes and transmit it by broadcasting. Each receiving node will use the public key to encrypt its own Ids length and add it to the ciphertext and increase the counter and decrease the TTL property. Listing 4.3 shows that the private key is treated as a private field and will never be included in the byte conversion. This is important as the public key enables a node to encrypt data, but only the private key can be used to decrypt data, and as such, it must only be known to the original key generator node.

Code listing 4.3: CalculationObject structure

```
type CalculationObjectPaillier struct {
        Id          uuid.UUID            `json:"id"`
        BranchId    *uuid.UUID           `json:"branch_id"`
        Counter     int                  `json:"counter"`
        privateKey  *paillier.PrivateKey
        PublicKey   *paillier.PublicKey  `json:"public_key"`
        Cipher      *paillier.Cypher     `json:"cipher"`
        Ttl         int                  `json:"ttl"`
}
```

Code listing 4.4: CalculationObject interface

```
type ICalculationObject interface {
        Add(cipher interface{})
        Mul(cipher interface{})
        Encrypt(int) error
        Decrypt(cipher interface{}) *big.Int
        KeyGen() error
        Serialize()
}
```

The ciphertext is encrypted using the Paillier cryptosystem[3]. We choose to use an existing implementation written in Go[4]. The Paillier cryptosystem is part of the PHE type and supports the addition and multiplication operations. The keys are generated based on two (p, q) independently chosen 128-bit length random primes. In a real-world, routed application, the bit length of the primes must ideally be higher. However, we have not explored the effect of having less secure primes in this type of local ad hoc network. Listing 4.4 shows the interface *ICalculationObject* that has to be implemented by any calculation object. In a real-world implementation, the counter should ideally also be encrypted. This would further obfuscate the contents of the object for any node that is not the original sender. In the proposed system and simulation, a node will only add data to the counter variable and never read as plaintext. Only the original broadcaster will read this during processing. Nevertheless, to keep the counter values inspectable during simulation runtime and to ensure a readable counter in the data report, this value is not encrypted.

[3] https://en.wikipedia.org/wiki/Paillier_cryptosystem

[4] github.com/didiercrunch/paillier

4.13.3 Transport layer

The transport layer is implemented in an effort to replicate Bluetooth broadcasting. Listing 4.5 displays the functions which a Transport Layer type has to implement. The layer can trigger both broadcasting and listening. Listing 4.6 presents the implementation used in the simulator. Data is communicated through go channels that accept byte arrays. This way, the exchanged calculation objects can be serialized and emulates the nodes being separate entities. Compared to passing the complete object instance as a function parameter. The Channel Transport Layer also keeps track of reachable nodes by having a reference to a reachable node's data channel. This serves as the only inter-node dependency.

The listen function is triggered as a Go routine, meaning a non-blocking, asynchronous execution of code. This results in all of the listeners being able to execute without waiting for any other code to finish. This is to emulate further that the nodes are independent entities without the interference of the other nodes in the network. A Goroutine statement is denoted by the keyword *go*, followed by a function to execute.

Communication over channels will naturally be much faster than hardware BLE communication using hardware. Therefore, the simulation allows a throttling parameter that will debounce the broadcast execution for the given time. This way we can emulate a more realistic latency.

Code listing 4.5: Transport interface

```
type Transport interface {
        Listen(nodeId uuid.UUID, handler Handler)
        Broadcast(nodeId uuid.UUID, obj []byte, onTrigger OnTrigger)
}
```

Code listing 4.6: ChannelTransport structure

```
type ChannelTransport struct {
        DataCh          chan []byte
        ReachableNodes  map[chan []byte]struct{}
        SuppressLogging bool
        Throttle        *time.Duration
}
```

4.13.4 Initialization and runtime

The simulation is initialized with the parameters defined in Table 4.4. The topology parameter defines how the nodes can reach other nodes. The "All" topology is defined as a node being able to directly reach all other nodes in the system, effectively creating bi-directional connections between all nodes. The "Cluster" topology sets a maximum number of adjacent nodes before a hop is needed to reach another local cluster. The maximum number of reachable nodes is hardcoded as eight. This is the maximum number of nodes possible in a Bluetooth Piconet, and as such represents a realistic cluster population. The two linking nodes between

clusters have a bi-directional link. When a simulation run is started, every node in the network starts its calculation process simultaneously.

During runtime, a worker pool is used to manage available Goroutines. Each worker has an additional queue not to drop any requests. This is a memory optimization as high load runs can very quickly exhaust the available resources. This should make the runs safer to complete and provide consistent data but could extend the time a run takes to complete. Particular heavy runs acquire a type of sequential processing.

A run report concludes each simulation run. This report takes the nodes and their data and outputs the JSON equivalent. Only the properties that contain a JSON format tag in the structures are includes in the final output. The complete simulator runtime logic can be seen in Figure 4.5.

input : Run Configuration, startTime, and assigned runtime

Result: Writes Simulation run report to file

initialization;

if *Topology is 'Cluster'* **then**
    EstablishNodeRelationshipsLocalCluster(*nodes*, *clusterSize*);
else
    EstablishNodeRelationshipsAllInRange(*nodes*);
end

for *node in nodes* **do**
    go *node*.Listen();
    go CalculationProcessInitiator(*node*);
end

make channel *closeMonitor*;
for do
    if <- *closeMonitor* **then**
        generateReport(*nodes*, *runConfig*);
    else
        if *runtime is expired* **then**
            close(*closeMonitor*);
        end
    end
end

Figure 4.5: Run simulation algorithm

Table 4.4: Simulation configuration parameters

Parameter	Default value
Nr. of nodes	100
Transport layer throttle/latency	30ms
Decrypt threshold	3
TTL	10
Topology All, Cluster	All

4.13.5 Simulation limitations

To ensure consistency through the runs, dynamically adding or removing reachable nodes during runtime is not implemented. This would more realistically simulate that the nodes are moving during the run. The node's reachable nodes must be established at start-up. This does not mean that all nodes must be able to reach all nodes. We consider the impact of moving nodes to be covered by the TTL parameter and the range of implemented transport layer and thus not of particular interest for the current evaluation of the proposal.

Further, during a simulation run, each node will only carry out a single calculation process, which means that we have not experimented with the impact of triggering criteria or the effect of a long term running system in regards to data quality.

4.14 Evaluation

When the simulation receives the shutdown signal, a report is generated for that run. The report contains all the collected diagnostic data from each node and information about the run configuration. See Table 4.5 for a breakdown of the collected data. These numbers form the foundation for evaluating the performance and revealing distinct properties of the proposed system. Additionally, various timers are used to benchmark the performance of different parts of the application. This data will be used in the evaluation of the application. Evaluation will consist of proving feasibility, overall performance, and the impact of adjusting the simulation parameters, see Table 4.4.

4.15 Summary

In this chapter, we have explained the envisioned architecture and how the protocol should function at the different levels and states. An in-depth explanation of how the protocol handles the different processes has also been given and the technical choices we have made in terms of encryption schemes, transport protocol, and routing strategy.

Additionally, we have described a thorough overview of the implementation

choices and technical aspects of the developed simulator. We have also explained the technical foundation for data collection and quantitative analysis.

In the next chapter, we will present the framework of our experiments, the reasoning of input parameters, and the results from the simulation runs.

Table 4.5: Node state tracking

Collection	Description	Abbreviation
NumberOfBroadcasts	The number of times a node has broadcasted an object	B
NumberOfExternalUpdates	The number of times the node has added it's own value to an incoming calculation object	EU
NumberOfRejects	The number of time a node has rejected a calculation object due to the calculation object not satisfying the decrypt threshold condition	R
NumberOfDuplicates	The number of times a node encounters a calculation object that the node has already processed	D
NumberOfPkMatches	The number of times the calculation object is returned to the original sender node. I.e., calculation object *pk* equals node *pk*.	PK
NumberOfInternalUpdates	The number of times a node accepts a calculation object and updates internal state	IU
NumberOfPacketsDropped	The number of times a calculation object is dropped by a node due to the TTL parameter reaching zero or other drop conditions	PD
Control	A node register each external update as a historical entry. This enables us to verify the correctness of calculation objects ciphertext vs. plaintext	-
Timers	Contains all the timed segments a node tracks	-

Chapter 5

Experiments and results

In this chapter, we present how we used the developed simulation stack from Section 4.13 to test the *DCP* solution. The first subject presented is the experiment plan. The plan ensures that each experiment is executed equally and that others can replicate the experiments. We then present the findings discovered through the experimentation, both related to system performance and what we have found about *DCP* BLE feasibility.

5.1 Experiment plan

We have used simulation to test the performance of the *DCP* proposal. Each simulation run consists of a different configuration, where the goal is to isolate the effect of changing one or more input parameters. All of the performed runs and parameters are found in Table A.2. Internal validity is maintained by ensuring that only the variable being tested changes between the runs.

As we have mentioned in Section 4.14, each simulation run gathers data about how many times a node performs distinct actions throughout a run. From these counters, we can determine the characteristics of a particular simulation configuration. Further, these also indicate how the performance of the composition was. Since we do not have a run baseline metric of *DCP* or similar solutions, we can only compare a run with the other runs and experiments.

The proposed system has, on purpose, left some implementation decisions open, those being the structure of the calculation object and what kind of calculation routine to perform. For the experiments, we have used the hypothetical use case described in the next section.

The simulation tool cannot match the speed of actual implementation, i.e., in a real-world application, each node consists of a single hand-held device, compared to a single computer simulating all nodes. Further, since the system is designed to run indefinitely, we define an end to a run when the traffic between nodes ceases. The runs will differ in how long it takes to complete depending on the number of simulated nodes, topology, and other behavioral properties. Finding

out the time each run takes is trial and error. The runs are iterated upon until three runs yield a consistent result. The required run times is listed in Table A.4. Each simulation is then executed three times to ensure consistency and that the data can be reproduced.

All the runs have been executed on a Linux Mint v20.1 machine[1]. The machine runs on an Intel i7-855u (8 core) @4.0GHz CPU with 16GiB of RAM.

5.1.1 Simulated environment and use case

In the following section, we explain the use case emulated by the simulator and the case used as bases in our analysis. A candidate use case is the distributed calculation of means. A mean can be calculated using any numerical information contained within the digital contact tracing application. A health agency can then use the results to generalize a local population. We demonstrate this by using the size of the contact events table as inputs in the distributed calculation. This table contains information about the encountered Ids, exposure time, and other variables used in the "at-risk"-calculation. Different digital contact tracing applications include different variables to increase the accuracy of the proximity measurement. The other variables can, e.g., be the model and make of the handheld device to account for differences in hardware specifications when performing proximity measuring.

The contents of the contact events table are attractive to gather as it tells us about how many unique individuals an app-holder comes into contact with, usually in a 14 days time span. This data is considered private information as the leakage of it would reveal a user's social graph. The content of the contact events is reported to a central agency in existing centralized digital contract tracing model when an individual gets diagnosed positive. However, this information is never shared by the applications in a decentralized architecture. The described case enables this type of data to be collected in a privacy-preserving way in decentralized applications.

5.1.2 Choice of input ranges

It is not feasible to test for every combination of simulation input values. From an analytical standpoint, each combination of input values has to be associated with a hypothesis. In other words, the input values should not be chosen at random. An essential step in the experiments is to confine each tested variable to a reasonable and specific set of values. These values have to be justified and, as closely as possible, be reasonable real-world values. Proper input states ensure that the findings have higher external validity than if random values were selected. Further, the choice of input variables impacts the simulator's behavior considerably, so reasonable values give a more valid behavior.

[1]Kernel: 5.4.0-67-generic

The node size dictates how many nodes are created for a run and the purpose of this variable is to see how the proposed system performs under various loads. The simulations contain node size in the set of $\{25, 100, 200\}$ nodes. Smaller than 25 nodes would likely make it easier to do more in-depth analyze the actions of the individual nodes, but in return, make it more difficult to generalize the results. The node sizes of 100 and 200 were chosen to simulate city-like population density[2] and are of such magnitude that it is easier to generalize the findings and benchmark the solution properly.

Closely related to the node size is the choice of topology. Since our system uses broadcast/listener to communicate and not a direct connection, the topology refers to the range in which the various nodes map, i.e., how far can a single node reach in a single hop in the network. We have defined two different topologies: *all* and *cluster*. In the all topology, all nodes can reach all other nodes; this is achieved with the nodes forming a bi-directional connection to every node. This simulates a single large cluster formation. In the cluster topology, each node can have a maximum number of adjacent neighbors before an extra hop is needed to reach the next cluster. The idea of the cluster topology is that as the node size increases, it is more likely to form smaller clusters of nodes naturally. In our opinion, the cluster topology model real-world scenarios more accurately. Even more so in a pandemic situation as with the Covid-19 social distancing. Health agencies recommend people to keep a distance of two meters from each other to minimize the possibility of infection[3]. This significantly limits the possible number of people being in range simultaneously using, e.g., Bluetooth as communication technology. The simulator uses a maximum cluster size of eight for all runs.

An essential variable in our simulations concerning privacy is the decrypt threshold variable. This variable dictates how many nodes that have to contribute to a specific calculation object before the original sender can decrypt the data. With a very low decrypt variable, the original sender is more likely to be able to decrypt, whereas a higher decrypt variable makes the calculation process more dependent on the density of the nearby population. Moreover, a low decrypt threshold means that if a certain percentage of the population conspire with each other and shares their data, the other inputs can be easier deduced, and the level of privacy decreases. If two or more nodes want to find out another node's input, they all have to conspire against that node, and we can say that the network has to be 20% honest to preserve the privacy of the honest nodes. For example, if the network consists of ten nodes, then if two of the nodes are honest, the other eight cannot find out the exact input of the two honest nodes, only the average between them. If 30% of the network is honest, then the remaining nodes can find out the average of the three nodes. In other words, the more honest the network is, the less accurate an attack gets. The decrypt variable was tested with the range $[3, 6] \Rightarrow \{x \in \mathbb{Z} : 3 \leq x \leq 6\}$.

[2]Since each node has a range sphere, the theoretical number of nodes in a single network is unlimited. Nevertheless, we determined that the chosen sizes to be sufficient

[3]`https://www.fhi.no/nettpub/coronavirus/fakta/avstand-kontakter/`

The TTL variable determines how many hops a calculation object can take before being dropped by a node, where we defined a *hop* as the calculation object being transported from one node to another node. Justifying a sensible start and cutoff point to the TTL value is not straightforward because we have no prerequisite to tell how it will impact the proposed solution. Despite it being difficult to predict, we chose to test the TTL variable in the range of $[3, 18] \Rightarrow \{x \in \mathbb{Z} : 3 \leq x \leq 18\}$, any higher than this, and it is likely to produce noise due to the range of the decrypt variable not making it beneficial. Similarly, a decrypt threshold smaller than three, and the calculation process would not be able to finish due to the lower value of the decrypt threshold.

The last simulation variable that we have defined is network latency. The latency variable is to simulate different network speeds, and the goal of this variable is to see how the proposed system deals with delays. Good and bad latency is specific to the usage, e.g., badly perceived latency when gaming does not necessarily mean bad latency when browsing the Internet or streaming video. We have performed the tests with the latency in the set of $\{30ms, 100ms, 200ms\}$.

5.2 System performance

Recalling R2 - "What trade-offs, in terms of privacy, efficiency, complexity, the answer(s) for RQ1 have?" the system performance can be broken into three key categories: Privacy, efficiency, and complexity. The complexity performance metric is based on how well the solution copes with larger node sizes and the impact of topology choices. Next, the efficiency performance metrics test how the different controlled flooding mechanics impact the solution's performance. Finally, the privacy performance metric tells how far we can increase the decrypt threshold before the solutions suffer from poor performance. Table 4.4 displays the runtime variables the simulation uses and Table A.2 displays all the performed runs and input values. The following sections present the experiments and key findings.

5.2.1 Complexity

Complexity is difficult to describe using an absolute definition, even more so when the term is used differently depending on the field of science. We interpret complexity to mean how the larger parts of a system interact and function. In testing complexity, we want to see how the proposed system copes with high node counts and how the choice of topology impacts the performance of the solution. As such, the topology type and node size become our independent variables, the rest of the simulation inputs stay controlled. Finally, the node diagnosis data are the dependent variables.

The first experiment we performed was to see if there was any difference in performance between the fully connected and clustered topology. The independent samples t-test showed that the full connectivity and cluster topologies

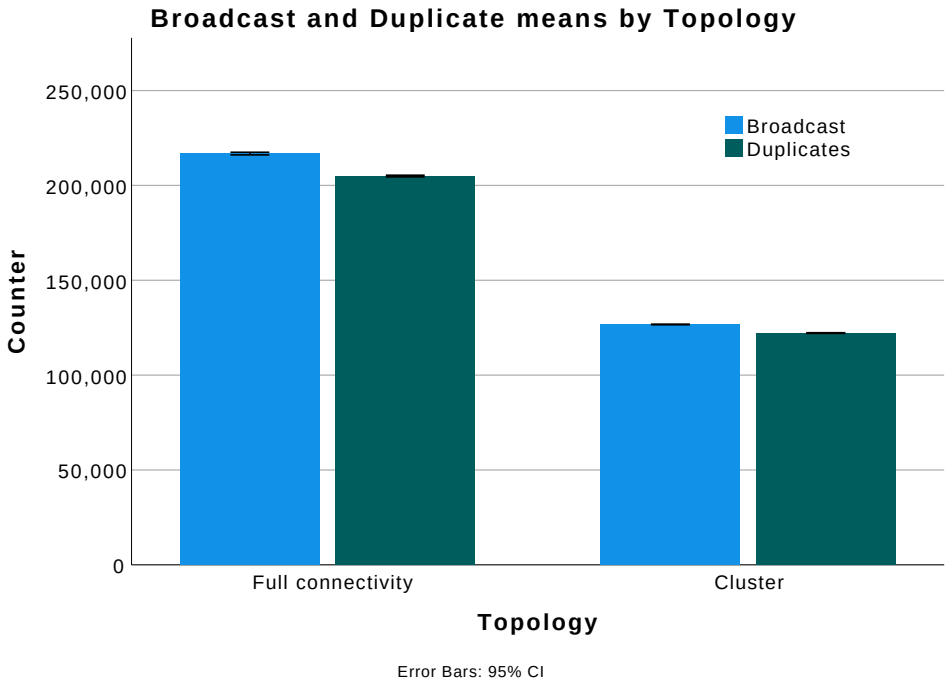


Figure 5.1: Difference in the number of broadcasts and duplicates over the two topologies. The error bars are narrow due to the large ticks.

yielded two different populations. The following was found: There was a significant difference in mean broadcast count between the full connectivity and cluster topology ($t_{1383.189} = 232.963, p < .001$). With a cluster node performing on average 91148 fewer broadcasts than a fully connected node. There was also a significant difference in mean duplicate Calculation Object handled between the full connectivity and cluster topology ($t_{1890.337} = 282.333, p < .001$). Here a cluster node encountered on average 84018 fewer duplicate calculation objects. Figure 5.1 illustrates the difference between full connectivity and cluster topology in regards to broadcasts and duplicates. There was also a significant difference in mean public key matches between the full connectivity and cluster topology ($t_{1831.409} = -46.995, p < .001$). With a cluster node performing on average 16022 more public key matches than a fully connected node. There was a significant difference in mean internal updates between the full connectivity and cluster topology ($t_{1943.567} = -39.857, p < .001$). With a cluster node performing on average 11666 more internal updates than a fully connected node. Figure 5.2 shows the difference between internal updates and public key matches in the two topologies. The full group statistics of this analysis can be seen in Figure 5.3, we will not provide any more fully detailed group statistics tables, but these can be located in Appendix B.

After finding that nodes in the cluster topology far outperforms the fully-

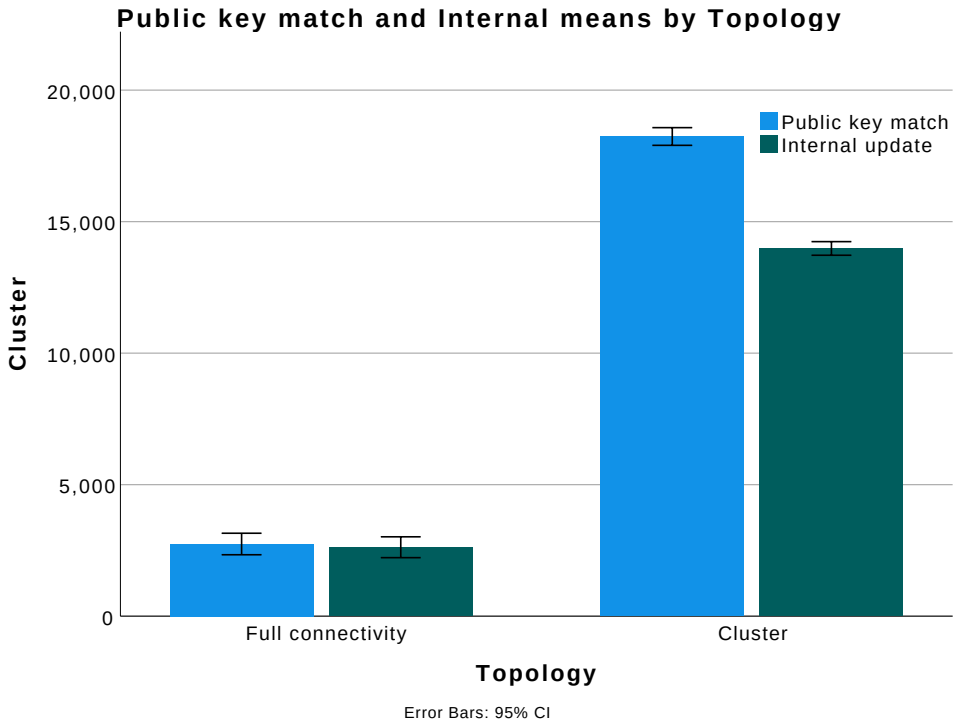


Figure 5.2: Difference in the number of public key matches and internal updates over the two topologies

connected topology, we wanted to explore one of the unique characteristics of a clustered network. This experiment tested if the nodes connecting the clusters (hence referred to as a bridge node) act as bottlenecks in the solution. An independent samples t-test was performed to compare the load on bridge nodes and non-bridge nodes. There was a significant difference in the average number of public key matches for bridge nodes (M=58.58, SD=11.02) and non-bridge node (M=47.42, SD=7.80); $t_{126.508} = -9.228, p < .001$. Further, there was a significant difference in the average number of internal updates for the bridge node (M=50.75, SD=10.34) and the non-bridge node (M=40.58, SD=7.02); $t_{123.820} = -9, p < .001$. The rest of the dependent variables did not yield a significant difference in the two populations. The increase in public key matches and internal updates suggest that a bridge node receives its calculation object in return more often than a non-bridge node, but this does not increase a bridge node's ability to complete a calculation process compared to the control.

The next experiment was designed to establish if a delay in the broadcasting would impact the proposed solution. Our working hypothesis was that due to how the simulator was implemented, the latency should not impact the solution's performance. An independent samples t-test proved that there was, in fact, no difference between the control runs and long delay runs. One explanation for this is that since the nodes are stationary in the simulation - there is no risk that the

Group Statistics

	T	N	Mean	Std. Deviation	Std. Error Mean
B	Full connectivity	975	216779.86	11059.661	354.193
	Cluster	975	125631.39	5190.332	166.224
EU	Full connectivity	975	11734.01	4932.586	157.969
	Cluster	975	248.71	46.599	1.492
R	Full connectivity	975	122.81	257.010	8.231
	Cluster	975	4478.63	2449.241	78.438
D	Full connectivity	975	204922.03	7121.287	228.064
	Cluster	975	120903.05	5969.272	191.170
PK	Full connectivity	975	2743.65	6509.186	208.461
	Cluster	975	18766.27	8424.098	269.787
IU	Full connectivity	975	2620.83	6306.750	201.978
	Cluster	975	14287.64	6615.510	211.866
PD	Full connectivity	975	76767.43	28698.794	919.097
	Cluster	975	54582.06	10447.670	334.593

Figure 5.3: Topology group statistics

nodes will move out of range of each other and, as a consequence, be impacted by bad latency. Moreover, it illustrates that the nodes are not dependent on each other or otherwise throttle a node's performance.

The last complexity experiment we performed was to test the effect of node size in the clustered topology network. By performing Levene's test, we could conclude that the sample sizes are unequal, this is to be expected since all the runs were performed three times, and we tested the actual impact of node sizes. Instead we used an ANOVA test with Welch statistic which showed that across the node sizes the following dependent variables differ significantly: Mean broadcast, $F_{Welch}(2, 164.80) = 58.67$, $p < .001$, mean duplicates, $F_{Welch}(2, 169.88) = 56.64$, $p < .001$, mean external updates, $F_{Welch}(2, 458.11) = 1975.89$, $p < .001$, and mean packets dropped, $F_{Welch}(2, 175.2) = 115.05$, $p < .001$. The complete multiple comparisons table can be found in Appendix B Figure B.6, this shows that the number of broadcasts, duplicates, and external updates grows as the node size increases, whereas the number of packets dropped decreases. What is equally interesting is that we accept the null hypothesis for the number of rejects, public key matches, and internal updates. We interpret this as when the node size increases, the traffic and external contributions increases, but that does not mean that a node in a large network has a higher possibility of completing a calculation process, i.e., the size of the network should not impact a nodes ability to collect data.

5.2.2 Efficiency

Efficiency can be understood as how optimized a system is, where optimized means the fastest, lowest footprint, and the least amount of overhead. To achieve an efficient solution, the system has to be fine-tuned, i.e., the most efficient state has to be found and this requires iterative testing. From our theory, the way to achieve efficiency is by utilizing techniques to remove unnecessary computation. In testing efficiency, we want to see how the proposed system performs with controlled flooding. The TTL variable acts as the independent variable, the rest of the inputs stays controlled, and the nodes' diagnosis data are the dependent variables.

The first efficiency experiment we performed was to see the impact of various TTL values on the performance of the proposed solution. As Figure 5.4 displays, we observed an increase in the number of broadcasts and duplicates as the TTL value increases. Whereas the number of packets dropped stabilizes when TTL is in the range $[6..15]$, with only a small increase. When the TTL variable becomes larger than this, the trend shows that the number of packets dropped decreases. A possible explanation for this drop is that as the TTL value becomes significantly large, the calculation objects overcome the threshold of expiring. As the error bar shows, some nodes statistically never drop a single packet. Figure 5.5 shows that the number of public key matches, calculation object rejects, and internal updates steadily increase as the TTL value increases. Moreover, all the variables have a significant standard deviation; we suspected this to be caused by the bridge nodes performing more work than the non-bridge nodes. The complexity experiments showed that a bridge node performed on average 21% more public key matches and 22.3% more internal updates than the non-bridge nodes. However, the complexity experiments do not explain the high standard deviation in the average number of rejects. The complexity runs were all completed with a TTL-variable of six, and as Figure 5.5 illustrates, the high standard deviation in calculation object rejects is not as prominent as when the TTL value increases beyond the a value of nine. Since the TTL variable controls how far a calculation object can travel before being dropped, the longer distance can result in more packets being able to reach the original sender and thus be subjected to a threshold reject.

To see if we could further improve the efficiency, we tried experimenting with the concept of early drop-off. Here, a node can be configured to drop a calculation object if the same branch Id has been encountered by a node once before. In our early packet drop experiment, we found that all of the early drop nodes' diagnostic counters decreased compared to the non-early drop control nodes[4]. The most interesting findings were that there was a statistically significant difference in mean broadcast between the early drop and control $t_{399.004} = 20.301, p < .001$. Here the early drop node performed on average 73042 fewer broadcasts than the control nodes. We also found a statistically significant difference in the mean number of duplicates handled $t_{399.172} = 19.836, p < .001$, with an early drop node performing on average 68869 fewer duplicate checks than a control node. Figure 5.6

[4]Full group statistics can be seen in Figure B.2

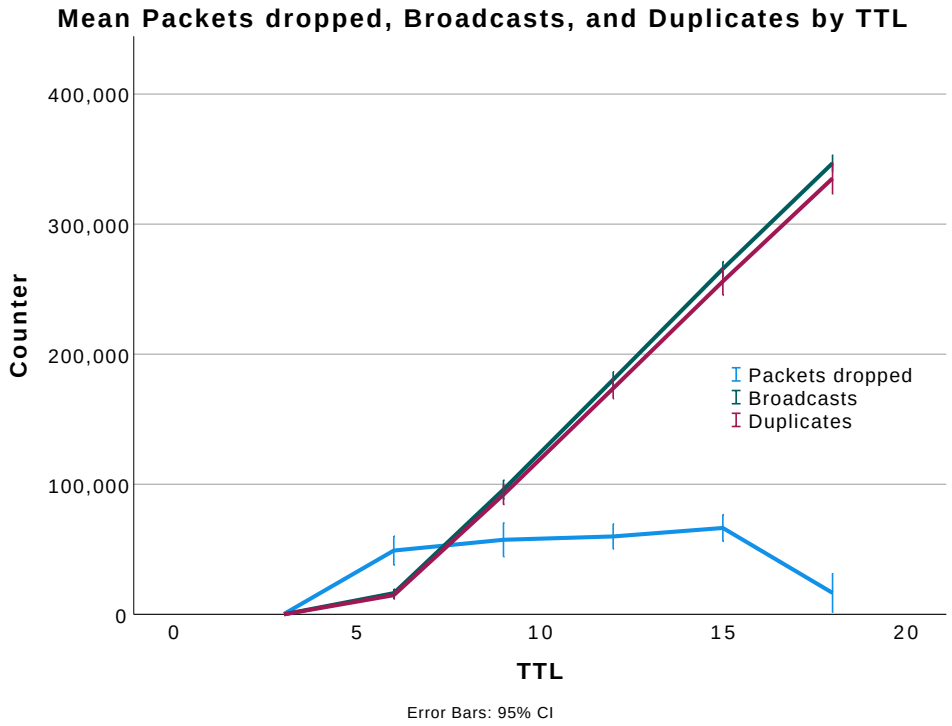


Figure 5.4: Mean number of packets dropped, broadcasts, and duplicates by the Time-To-Live (TTL) value. This graph makes it very evident that the number of duplicates closely follows the number of broadcasts for all tested values of TTL.

illustrates this difference. This indicates that by utilizing the early drop technique, we significantly decrease the traffic in the network. However, it does not decrease a node's ability to complete a calculation process. The early drop nodes still performed on average (M = 43.02, SD = 9) internal updates and (M = 50.1, SD = 9.9) public key matches. This is significantly less than that of a control node (M = 8711, SD = 8643) internal updates and (M = 11682, SD = 12091) public key matches, but still sufficient enough to complete a calculation process. Additionally, internal updates and public key matches are performed by a single node on its calculation object. The early drop internal updates and public key matches findings further suggest an increased efficiency over the control. An interesting find is that the average number of times an early drop node contributes to an external calculation object increases from (M=179.52 SD=88.7) to (M=249.9 SD=159), which represents an increase of 32.7%. This increase can be explained by examining the difference in packets dropped. A control node did drop on average 40545 more packets than an early drop node. This, seen in conjunction with the earlier reported numbers in broadcasts and duplicates — the early drop network seems to allow more value-producing packets to propagate and already handled, noisy packets to be dropped. In other words, the early drop clause flushes the network, while the TTL parameter allows non-handled packets to propagate deeper into

the network compared to the control.

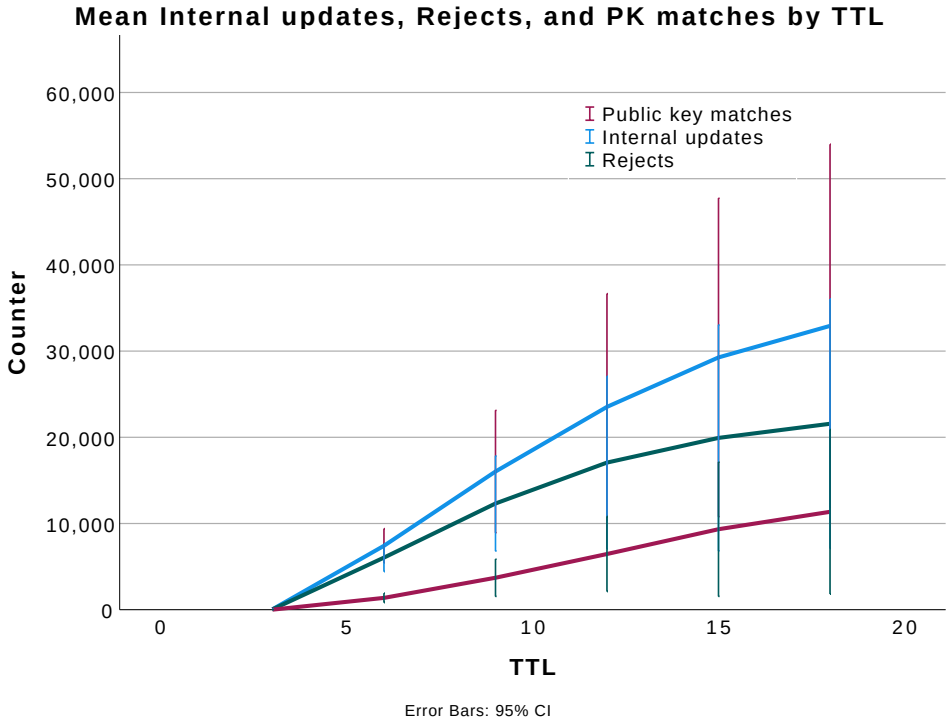


Figure 5.5: Mean number of internal updates, rejects, and public key matches by Time-To-Live (TTL). In this graph we can see that the errors bars is really wide, suggesting that we either have some few extreme outliers, or that the we have some nodes that on average perform these action more than the other nodes.

5.2.3 Privacy

As mentioned earlier, how privacy-preserving our calculations are is determined by how many nodes contribute to a calculation object. This is because the more nodes participating, the harder it becomes to single out the input values of an individual node. The simulator enforces this relationship by only allowing the calculation object owner to decrypt the object if a set number of nodes have participated. This is what we call the decrypt threshold. A higher decrypt threshold should, in theory, mean higher privacy, but we need to test how the solution deals with higher values. In the results, the decrypt threshold acts as the independent variable, the rest of the inputs stays controlled, and the nodes' diagnosis data are the dependent variables.

We observed that a minimal increase in the decrypt threshold from the control value of three significantly decreased a node's ability to conclude a calculation process during earlier testing. A finished calculation process requires that a node has done at least one internal update. When we increased the threshold to four,

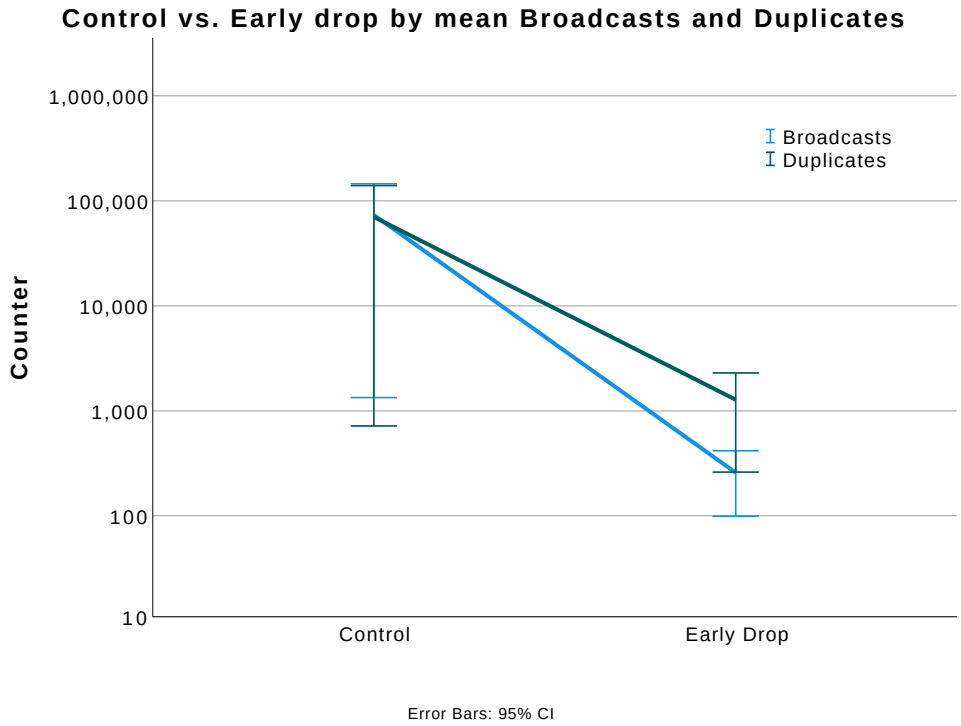


Figure 5.6: Mean broadcasts and duplicates difference between the early drop nodes and control nodes. Note that the y-axis is logarithmic with base 10. This also makes it very clear that the error bar for the control group is significant, ranging from over 100000 to as little as 1000.

only 6% (6/100) nodes reached the required threshold. The hypothesis that due to the increase in decrypt threshold, the default TTL value of 10 was insufficient and that all the packets were being dropped was tested. But, increasing the TTL value did not yield any statistically significant difference against the control. We then tried to include the early drop property from the efficiency experiments and converted it to a counter, as compared to the previous "if handled, drop packet"-logic. Now, a node can be configured not to drop a duplicate calculation object before it has been re-broadcasted X times. This means we also had to change the simulation run completion criteria from the cease in traffic to when all the nodes have completed at least a single internal update. In other words, a node has to decrypt an object that has reached the required decrypt threshold before it is considered finished in the subsequent experiment.

In the following experiment, we wanted to see if an increase in decrypt threshold leads to a change in our dependent variables. Figure 5.7 shows that the relationship between the number of broadcasts and the decrypt threshold can be illustrated as an exponential function. The same is observed when looking at the duplicates, as showcased by Figure 5.8. A strong increase can be seen in both cases when going from a decrypt threshold of five to six. If we view the counters as

an logarithmic scale the relationship is nearly linear, see Figure B.4 and Figure B.5 in the Appendix. The average number of external updates and internal updates do not share this exponential property, as seen in Figure 5.9. When increasing the decrypt threshold from three to six the observed external updates increased with 151.7% (DT = 3, M = 178.42, SD = 3.62) to (DT = 6, M = 449.1, SD = 8.79). Whereas in internal updates we saw the opposite relationship with a decrease of 90.3% (DT = 3, M = 43.84, SD = 0.951, DT = 6, M = 4.25, SD = 0.621). The increase in external updates may be explained by the fact that nodes participate in calculation processes initiated further away in the cluster.

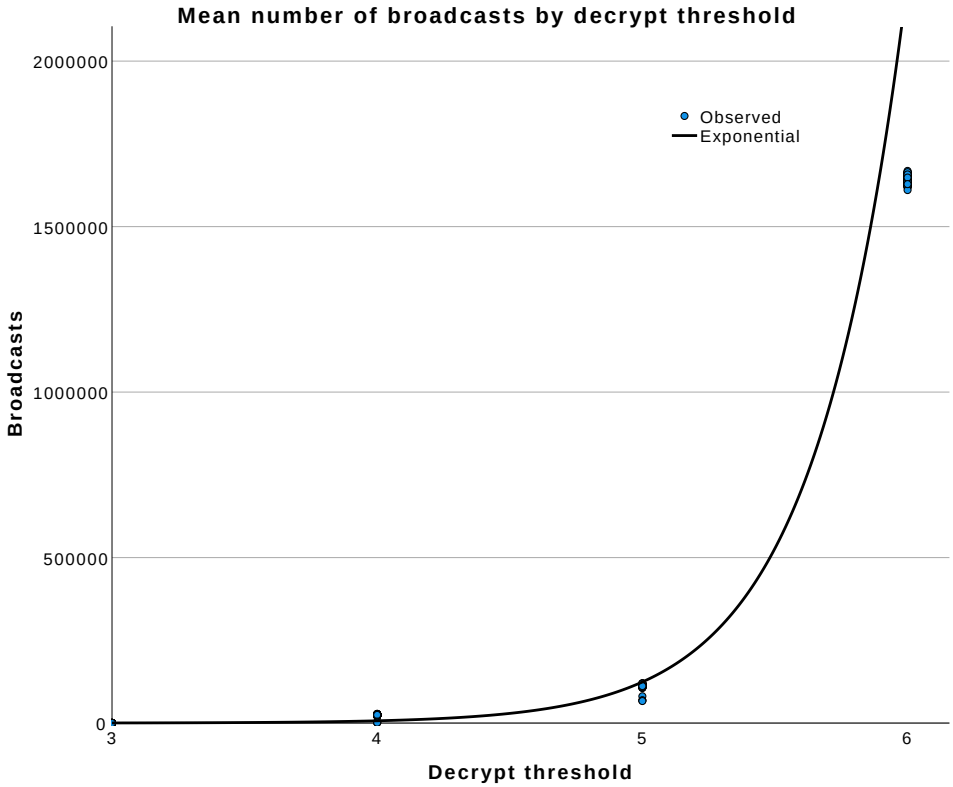


Figure 5.7: Mean node broadcasts by decrypt threshold. With observed values and curve fit estimation. The standard deviation for the observer broadcast values were ±36.8, ±4624.9, ±8801.1, and ±12358.1 for decrypt threshold 3, 4, 5, and 6 respectively. Model summary: $F(1, 398) = 7080.45$; $p < .001$

Reiterating on the bridge node versus non-bridge relationship, we found that there was no statistically significant difference in the mean number of broadcasts performed between bridge nodes and non-bridge nodes, $t_{398} = -0.036, p = 0.972$ when we include the whole decrypt threshold range.

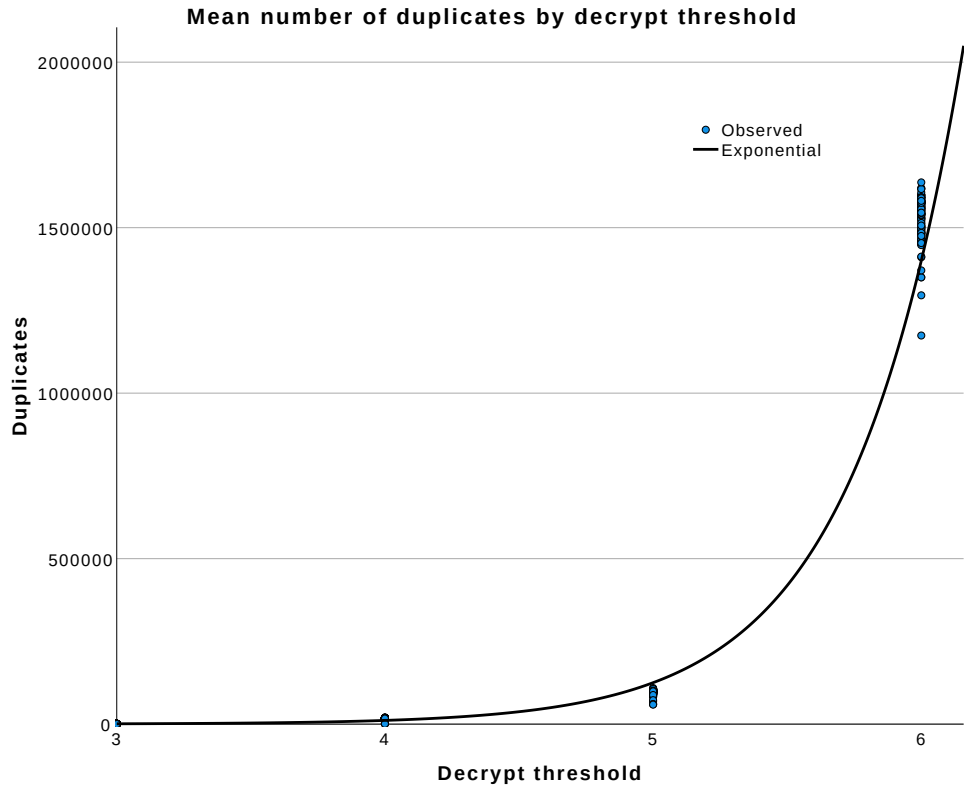


Figure 5.8: Mean node duplicates by decrypt threshold. With observed values and curve fit estimation. The standard deviation for the observer broadcast values were ±207.2, ±3290.2, ±9087.9, and ±72325.5 for decrypt threshold 3, 4, 5, and 6 respectively. Model summary: $F(1, 398) = 18256.46$; $p < .001$

5.3 BLE advertisement feasibility

The last set of experiments we performed was an inquiry into the feasibility of using the proposed system with only BLE Advertisement packets. As we described in Section 2.4.2, if possible, utilizing only the BLE advertising packets would save time by the nodes not having to perform a handshake. The BLE advertising packet has a variable data payload slot of 31 bytes. Before running any experiments, we looked into the properties of our calculation object. In the solution, we generate a key set, where each key has the same size, but only the public key is used in communication. In our simulator, we had chosen 128-bit length primes to create our composite keys or a total of 32 bytes of keys. In other words, the public key alone will not fit into a BLE advertisement packet. Further, both the Id and BranchId take up 16 bytes each. Nevertheless, we tested how the calculation object grows when homomorphic ciphers are added, this simulates that other nodes contribute with an external update to the calculation object. Figure 5.10 displays the growth in Calculation Object as more additions are added to the ciphertext.

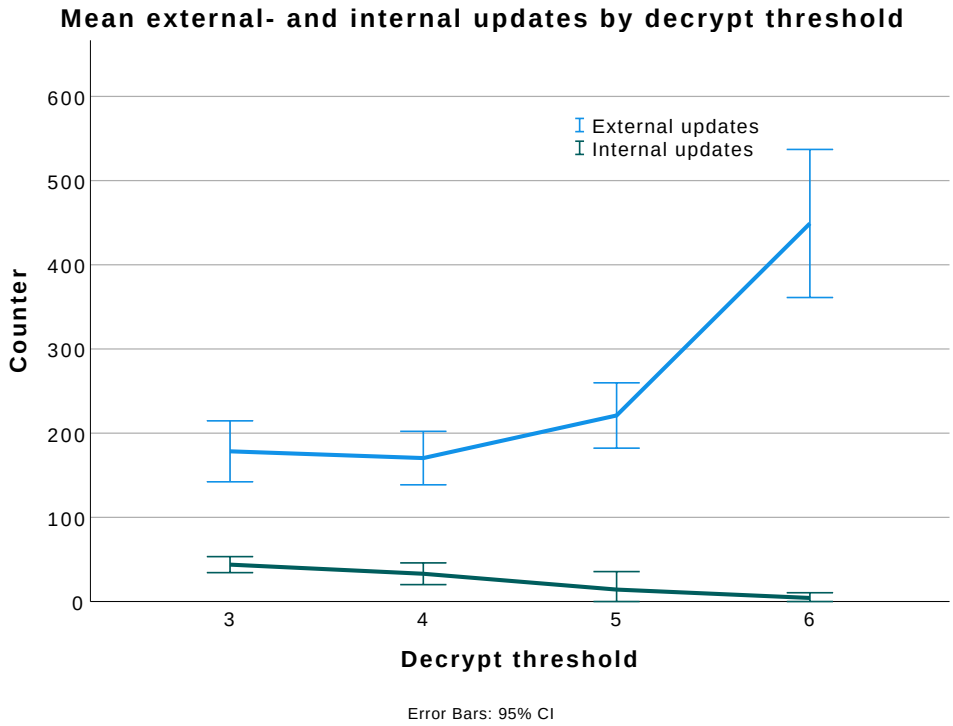


Figure 5.9: External- and internal updates by the decrypt threshold. As the decrypt threshold increases the mean number of external updates a node performs increases, whereas the mean number of internal updates a node performs decreases. Interestingly, the same relationship can be seen in the error bars.

At C_0, the Calculation Object takes up 227 bytes, and this is the encoded size of the object with all properties initialized, including the public key. When the first encrypted message is added, the size increased to 384 bytes, increasing by 154 bytes or 69.1%. However, the size of the calculation object stays almost consistent through all the remaining additions, with C_{100000} having 388 bytes. Some possible solutions for the size issue are discussed in Section 6.1.

5.4 Summary

In this chapter, we have described the experiment framework and the justification of the chosen simulation configuration values. Further, we have reported on the main findings from the experiments. Measuring the performance was presented by three main categories, complexity, efficiency, and privacy. The main conclusions from the experiments are that the cluster topology is superior to the full connectivity topology. Further, in the clustered topology, the bridge nodes perform some extra work compared to the non-bridge nodes. We also discovered that allowing the nodes to drop already handled calculation objects instead of propagating them

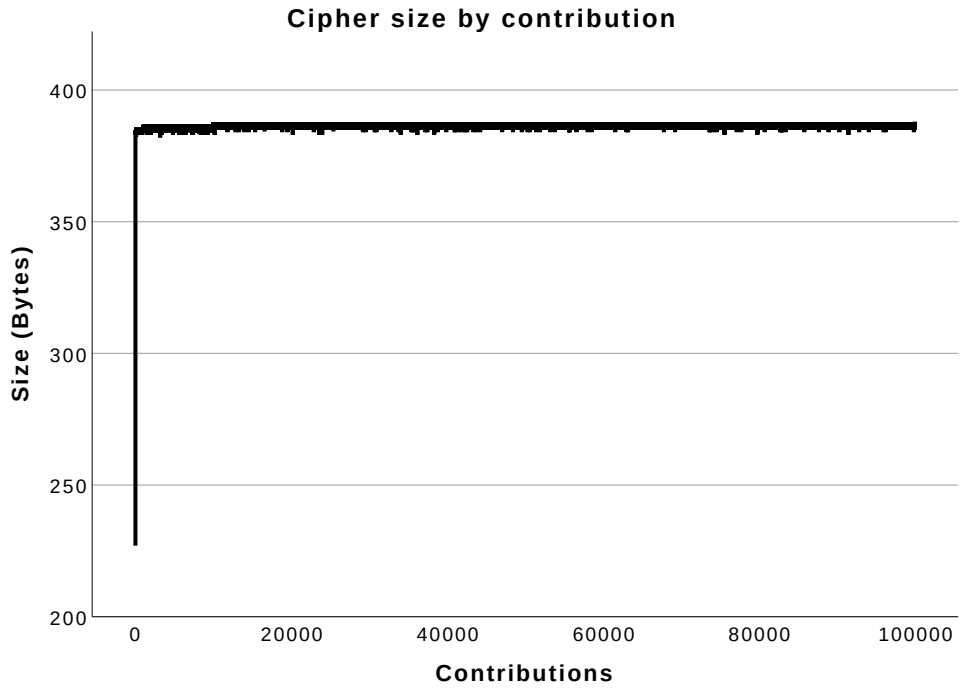


Figure 5.10: Calculation object contribution growth. Each contribution adds a new encrypted message to the Calculation object's ciphertext. There was only eight unique size values with the following frequency: (Size: 387, Freq: 80.7%), (Size: 386, Freq: 16.5%), (Size: 385, Freq: 2.4%), (Size: 384, Freq: 0.3%), (Size: 383, 382, 227, 388, Freq: f $<$ 0.1%).

further increases the efficiency and allows the nodes to perform more external updates. Our last findings in the performance experiments were that small increases in decrypt threshold grow the overhead exponentially, and a doubling from the default decrypt threshold increases the number of broadcasts by a factor of 8878. In terms of advertising feasibility, we found that we cannot fit everything in a single BLE advertisement packet due to the size of the public key and the ciphertext.

In the next chapter, we take a closer look at how the findings impact the proposed solution's overall performance and feasibility. We also discuss the limitations of the findings and experimental work, directions for future work, and reflections on the thesis.

Chapter 6

Discussion

In the following chapter, we discuss the significance of our findings and what these reveal about the proposed solution. Both concerning the feasibility and the performance. Further, we comment on some potential future research directions and immediate improvements to the proposed solution, together with its recognized limitations. Then, we discuss some other areas and use cases where this sort of system can be utilized. Lastly, we give some reflections on the work that has accumulated over the project period.

6.1 Feasibility

Recalling RQ1: "How can a privacy-preserving arithmetic calculation be carried out in the context of decentralized digital contact tracing?", this question is answered through our proposed system. Combining Homomorphic Encryption and BLE broadcasting allows a system to extend current digital contact tracing practices to gather more extensive data than currently being collected in decentralized-based architecture. Homomorphic Encryption protects the privacy of the participating nodes, and BLE enables us to communicate in a non-persistent and low overhead approach. By low overhead, we mean that BLE functions using very low energy over shorter distances, so the load of utilizing it on a hand-held device is small.

We did not achieve the necessary packet size with the current implementation to include everything needed for operation in a single BLE advertisement packet. Moreover, the keys used for Homomorphic Encryption have to be of a certain length to provide adequate security, and these on their own do not meet the size limitation. This finding does not mean that BLE is not feasible but that the advertisement packets need to be connectable and that the communicating nodes have to perform a handshake to receive the rest of the information from the broadcaster. This thesis has not explored the performance impact of the nodes having to perform a handshake. However, in our opinion, the need to perform a handshake could decrease the available communication time window since it would increase the time it takes to establish a connection. This could mean, e.g., two passing by

nodes moving at high speed would risk not being able to finish the exchange, and the receiving node has to register this and disregard the partly received data.

Apart from the size of the Ids, public key, and cipher, few optimizations are available. A possible optimization could be to change the counter and TTL type from Int to Int8, saving 6 bytes, but this is a minuscule optimization compared to the size of the Ids and cipher elements. With the required length of cipher elements, there seems to be no way to fit all of the information into the advertisement packet. However, a possible solution to this is to select only a subset of the calculation object properties and broadcast these in the advertisement packet., If necessary, we can apply compression techniques to make these values fit in the 31 bytes data slot. The sub-set should enable the nodes to make some decisions before it has all the information available, such as the calculation object Id and branch Id. This information will enable a node to determine if it is necessary to request the rest of the information by moving the early drop logic. In the proposed solution, these two values are coded as UUIDs and take 16 bytes each, collectively being only 1 byte short of fitting into the advertising packet, making it likely that some compression or changing how the UUIDs are represented, e.g., change from decimal representation to hexadecimal representation, would save the required space.

6.2 Performance

The performance experiments looked into how the DCP solution executes under various starting conditions and logical components. The experiments tested the areas of complexity, efficiency, and privacy. In the following section, the findings are discussed collectively.

The results from the fully connected versus cluster topology experiment uncovered a couple of things. Firstly, a fully connected node will perform significantly more broadcasts than a cluster node. This is coupled with a fully connected node encountering more duplicate calculation objects. The actions of broadcast and duplicate seem to follow each other very closely in a fully connected topology. We interpret this as a fully connected node performing far more resource wasted work in that the extra computations do not result in any value for the original sender node. Secondly, a node in the cluster topology performed more internal updates and more frequently encountered a successful public key match than the control. This, on the other hand, increases the likelihood of a cluster node being able to finish a calculation process, i.e., increasing value-producing traffic and minimizing wasted resources. Also, in terms of complexity, we found that the size of the network does not impact the performance negatively; both smaller networks and more extensive networks perform similarly. Although we have not tested very small or very large networks. As long as the network is larger than the decrypt threshold value, we do not see any challenges with such networks. We have not found any alarming issues with extensive networks either, but we cannot say for certain what effects that could have.

We have tested out some mechanics to deal with the occurring broadcast storm. One such mechanic is the impact of a TTL field in the packets. Our results showed that having a larger TTL increased the mean number of public key matches, nodes rejecting calculation objects, and nodes performing internal updates. However, it also showed that the mean number of broadcasts and duplicates increased at a much higher rate as the TTL value increased. In other words, we saw diminishing returns of value-producing work when increasing the TTL value. We observed that the middle ground between the TTL and value-producing traffic is when the TTL value is between three and six. In other words, a high TTL is not a sought-after property, and limiting the distance a packet can travel before being dropped does not limit a node's ability to produce valuable results. An implementation should aim for the bare necessary values for control factors and instead design for a network tolerating more loss.

Another mechanic we experimented with was making the nodes drop already handled calculation objects before the TTL variable has expired to decrease the number of broadcasts and duplicates flooding the system. The control node group re-broadcasts duplicates to reach nodes positioned further away in the topology and new nodes joining the network. The results showed that the nodes that dropped already handled calculation objects immediately had a 99.6% decrease in the number of broadcasts — Indicating that most of the traffic in the control group is unnecessary. In the control group, the calculation objects are circulated in a limited set of nodes or nodes that have previously handled the object. Both cases create a deadlock situation where the nodes circulate the packets until the TTL property expires. This is bound to happen as the flooding has no direction; it does not differentiate between the nodes or paths.

Finally, the privacy experiments showed that most dependent variables grow exponentially as the decrypt threshold increases. Consequently, as the proposed solution stands now, there is a relatively low crossover between usability and privacy. Since high privacy is a desirable trait, it also introduces overhead and makes the solution slower, i.e., decreasing usability. However, a very high decrypt threshold is not needed to achieve privacy, assuming a high degree of honest nodes in the network. Considering that the mean number of value-producing actions performed, i.e., external- and internal updates, did not significantly increase at the same rate as the rest of the metrics, most of the growing traffic is wasted traffic. The added overhead most likely comes from the nodes having to propagate the calculation object to enough nodes to meet the higher decrypt threshold, i.e., the calculation object has to travel further to reach enough nodes that can contribute to the calculation and then travel the same distance back to the original sender.

Combining all our findings together, we can answer RQ2. In terms of complexity, a clustered topology is to prefer over a fully connected one. This can be concluded based on the observed decrease in wasteful communication in the clustered topology. Including controlled flooding techniques, such as TTL and branching Ids, further decreases the number of noisy actions performed and increases the overall efficiency. Moreover, the controlling factors should be as aggressive as possible.

The resulting loss of data is not data of value as it only contributes to noise and is nonetheless dropped at a later stage due to the controlled flooding mechanisms.

Further, the node size of a network does not seem to impact the overall performance of the individual node. The choice of decrypt threshold will significantly impact how well the proposed system performs. With a higher decrypt threshold, most evaluation characteristics increase exponentially without increasing practical actions to justify the increase in traffic.

6.3 Future research

In this section, we present some possible future research directions that we envision for the proposed system. As addressed earlier, increasing the decrypt threshold results in a disproportionate increase between value-producing traffic and wasteful traffic. New solutions to control the flooding need to be investigated. One proposal might be to more accurately direct the flow of communication. More specifically, utilize the TTL variable to limit backward broadcasting, backward meaning communication that propagates away from the original sender. The nodes can limit backward communication when the TTL value reaches a certain threshold. This can be achieved by working out an appropriate half-life for the TTL and having nodes perform a cutoff when this threshold is reached. This can work since at a certain point the TTL would be less than likely required to reach back to the original sender.

Another technique can be to include the inverse of a TTL variable, namely a hop variable. Each calculation object has a hop counter; when passing the object to a new node, the hop variable increases. However, when a node encounters a branch Id it has previously handled, the node decreases the hop variable. The nodes that have handled that specific calculation object becomes dead ends, whereas nodes that have not processed it takes it further. This method could function as the shortest path to the original sender and direct where the calculation object goes.

We have showcased that uncontrolled flooding without active routing in a multi-hop ad hoc network generates an exponential overhead when the decrypt threshold increases and depending on various natural forming networks such as the clustered and full connectivity topologies. A future improvement to the large observed overhead could be to investigate to what degree the RSSI measured distance can control the direction of packets and how the nodes should accept incoming packets. Future research could investigate if the cumulative measured distance can be used, similar to the previously described hop variable, to direct the direction of the packets. Nodes closer to the broadcast initiator would reduce the overall measured RSSI distance, while nodes further away would increase it. The nodes could then use this to decide whether to drop or accept an incoming packet. Then the nodes would be able to actively route broadcasts backward when some threshold or requirement is satisfied. This could potentially increase the chance of a successful calculation process and decrease wasteful operations. Further, if

the cumulative distance can be used to make smarter assumptions about routing, then machine learning algorithms can be used to power the decision-making based on previously observed results. Effectively, making the connectionless routing stronger than simple assumptions about the state based on a metric with a high range.

The current proposed system has no notion of data persistence. The nodes do not retain any information about the network or packets for any period. A logical extension of the proposed system is to make the nodes perform caching of expiring calculation objects, i.e. when the TTL variable is reaching zero. This way, a node can re-trigger the broadcasting of the cached object at a later moment. The caching method could negate some of the effects of nodes moving outside of the network before the calculation process is finished. The exact re-broadcast trigger criteria and optimal cache time must be investigated by future research. A possible starting point can be to look into the Google Geolocation API[1] for device location without the use of GPS. The cache time and trigger criteria should consider when the caching node is most likely to encounter the original sender node. Certain times of the day, or days, i.e., weekdays versus weekends, can increase the likelihood of such an encounter. An intuitive situation is to limit the re-broadcasting only to when the node is suspected to be in commute. This is information that common navigational tools, such as the Google Route API[2], can supply.

Additionally, the current proposal assumes no connections, it is connectionless, and in that sense, the network contains no destination nodes, as in traditional connected communication protocols. However, an argument can be made that the original sender at some point in a calculation process's lifetime becomes a destination node. This calls for better optimization in the return-to-sender conditions. Therefore, it is a reasonable step for a future project to investigate establishing permanent connections and moving the solution towards an ad hoc connected network. This would enable researchers to compare the performance of the proposed solution with established routing protocols developed for connected ad hoc networks, of which there are plenty.

To summarize, we have provided just a handful of the possible future research directions. Most of our immediate concern lies in reducing the traffic overhead, and a logical starting point is to look into ways of providing smarter connectionless routing and whether or not a node should handle a specific calculation object.

6.4 Limitations

We have now covered our findings and future research directions; next, we present some of the limitations identified about the proposed solution and simulation.

In a connectionless system, ensuring that the nodes will form clusters is not trivial. There is no way for a node to determine if it is part of a cluster other

[1] `https://developers.google.com/maps/documentation/geolocation/overview`

[2] `https://cloud.google.com/maps-platform/routes`

than analyzing the number of unique incoming broadcasts. A cluster would not be formed by a logical rule in a real-world scenario but rather the node density in a specific area, and this will fluctuate depending on the node's environment. In Section 5.1.2 we stated that some natural clustering could likely happen in a real-world scenario due to the limited broadcast range. Simulating a real-world case with moving nodes and realistic ranges could give insight into how clusters can form. It could also give insight into how the network would handle nodes joining and leaving the network's range sphere, especially when a broadcast initiator leaves, as briefly discussed in the previous section. Our experiments wanted to test the core internal program logic and not the communication protocol itself. However, network-specific simulators exist, e.g., the NS-2 network simulator[3]. Such programs can emulate moving nodes and can be configured to accurately emulate Bluetooth Low Energy behavior, transfer speeds, and packet sizes.

We have previously stated that our proposed system is not constrained to using BLE but is heavily modeled after the intent of using it. Expanding on this, no experiments have been conducted to prove that BLE is compatible with the required communication speed and throughput, other than the inquiry into the feasibility of using advertisement packets. This is also true for the choice of the Homomorphic Encryption scheme. We used Paillier in our proposed solution as it is relatively simple to understand and supported the requirements in our case study. Paillier has many advantages over other solutions, like the length of the primes being relatively small compared to, e.g., RSA[4]. However, if an implementation requires other operations than those supported by Paillier, another Homomorphic Encryption scheme must be used. This is not necessarily a limitation of the overall system, but using Partially Homomorphic Encryption limits the type of calculation operators available to the system.

A significant concern in digital contact tracing is the appearance of a man-in-the-middle attack, where a third-party actor can intercept a message and change its content before arriving at its intended recipient. Alternatively, a legitimate actor injecting a piece of false or corrupt information and passing it on to other nodes. The proposed solution does not address these problems and suffers the same concern as digital contact tracing. Especially the ability of a third-party actor to inject corrupt data into a calculation object could jeopardize the overall validity of a calculation. Since all nodes in the network perform independent calculations, the overall risk of this happening is constrained to the micro-level, in that some of the nodes in the network can end up with corrupted calculations, but if a central agency is to collect data over a large area, the corrupted data should be detected as outliers. Future investigation should be made into adding deniability such that the system itself can reject invalid entries or detect tampering attempts.

[3] `https://www.isi.edu/nsnam/ns/`

[4] For RSA, NIST recommends a key size of 1024 bits, while the same security can be achieved in Paillier with 160 bits [35]

6.5 Other applications

Apart from digital contact tracing, the privacy-preserving decentralized calculation in the proposed solution can be utilized in other settings. The data generated through distributed calculation have to gain the users of the system to some extent. The ideas can be used in decentralizing other currently centralized services, where the users receive the benefit without allowing third-party actors to access the users' data. Apart from the privacy-preserving aspect, such calculations do not need to send the data to any central collection point. Possible areas include services such as localized information about, e.g., busy hours, that can be achieved without including third parties such as Google or Apple and without the network revealing the location of any node. This solution does require additional expansion to extract the data. Another possible application can be in bartering prices with vendors or private marketplaces where the buyer collects best-case offers without revealing the sellers. The resulting bid will be the best-case trade-off between delivery time, price, ability to reimbursement, with more. This type of system would allow the buyer to gather offers without the sellers impacting each other's bids, pushing businesses to offer the buyers the best deals they can offer. Resulting in a supply chain that is more difficult to establish monopolies in, and in return, give the best service to the buyers.

Additionally, the technology can be used for what we will term "good faith" data mining while still preserving privacy and as publicly available data. Data mining is not inherently wrong, but the lack of transparency and obfuscated user agreement is. If the miners were to find a way to incentivize the users to participate in data mining, our system would allow data mining where the contributors both agree to the collection and have insight into what is being collected.

Another direction is to extend digital contact tracing to more persistent diseases, such as seasonal epidemics. The proposed solution could be implemented as a notification hub of nearby infected individuals spanning the plectra of diseases. An application would enable a user to register a status if that user becomes infected with, e.g., the common flu. By utilizing distributed calculation, the nodes would calculate the nearby density of infected individuals. This results in an immediate "at-risk"-score calculation, but the score will not reflect how likely it is that the node has contracted a disease, but rather the local risk of meeting an infected node. This differs from current digital contact tracing, where the infectious status is reported to a central agency, which then, depending on the architecture, distributes this to the other nodes. Such a solution might require more strict privacy demands. However, none of the nodes would learn anything about who is infected. Instead, the solution notifies the node to be extra cautious if the density of infected is high.

As discussed in Section 2.6, digital contact tracing has come under academic scrutiny over the potential of misuse. The proposed system could enable even more detailed and encroaching surveillance of citizens. Since the solution could be modeled to gather any data as long as it is numerical. Suppose this type of

functionality becomes standardized by hand-held manufacturers, i.e., embedded as part of the standard library in mobile Operative Systems. In that case, the technology should be upheld to the same security requirements as, e.g., how Android handles its accessibility tools. These are placed in a special type of permission system where the technology users have to give explicit approval of its use. Further, the proposed solution requires complete transparency in what data are collected, which should be upheld in other implementations.

The field of digital contact tracing itself establishes some limitations on how the current system should behave. Namely, avoiding the inclusion of node identifiers in the calculation object. More information could be included in the payload in a less privacy concerned setting if it would benefit the routing or enable other use cases. However, the purpose of including Homomorphic Encryption was to ensure privacy, so if a theoretical payload were to include identifying properties, it would violate our privacy requirements, listed in Table 4.1.

6.6 Reflections

Even though the field of Bluetooth and Homomorphic Encryption is not novel in itself, the field of Covid-19 digital contact tracing is. The release of new research in this area is very rapid, especially in the year 2020. The quality and scope of released work are, therefore, notably fluctuating. As the field appeals to a large set of research fields and domains, many different disciplines try to explain and map out the fairly technical questions surrounding digital contact tracing. The consequence is a complicated landscape as there does not seem to be an established truth about much, and different researchers state conflicting facts about the same problems. Since the time elapsed from the creation of hand-held digital contact tracing applications and the pandemic is, as of 2021, an ongoing situation, few surveys and meta-studies have been published. This does not mean that we question the reliability of the sources we have cited but instead make it a complicated field to research. This will most likely be easier as the field matures.

Further, as data is transient, the available data only reflects a snapshot of time, and with time the field will rapidly grow and establish itself. Although consortiums such as DDT3 and the Google/Apple collaboration have tried to establish some standardized and properly backed frameworks, there is still a wast pool of diverse implementations solving problems differently. This may contribute to the varying quality of research since there are many possible solutions, and which solution that will take precedence in the future is not clear. However, we predict that due to Google and Apple's market influence, their exposure notification framework is a candidate for market dominance. This would mean that decentralized-based architecture becomes the norm and that our proposed solution and functionality will be a relevant research topic in the future.

In hindsight, the upper limits of nodes tested in our simulations, 200 nodes

at max[5], are too few to represent the performance of a massive network accurately, such as in a metropolitan area. There is a genuine possibility of a clustered topology naturally forming with thousands of active nodes in this such an area. Selecting the upper limit came down to the logical increase in nodes, but also simulation hardware limitations. The 200 nodes, fully connected topology runs consumed approximately 14 GiB of RAM, exhausting the delegated memory capacity imposed by the worker pool. Without the worker pool, the memory footprint would have been higher[6]. A possible way to negate this could have been sourcing out a more task-specific platform such as a GPU farm or utilizing a cloud service such as the *Microsoft Azure Batch* service[7]. This would have enabled us to test with more extreme amounts of nodes and faster. An eventual meta-study of our findings should utilize simulation/big data-processing specific platforms.

As mentioned in Section 4.14, the simulator includes a set of timers that measure the time it takes to execute different parts of the main algorithm. Each node also tracks an internal history of all the processed calculation objects to validate the resulting data. However, during testing, these timers prove unreliable due to the limitations of the simulator hardware. Since some of the simulations runs executed on the resource limit, we cannot guarantee that the timers stay valid if the CPU has paused execution or if the program had to store memory in the much slower swap memory space. As such, we chose to exclude these from the analysis. If the simulation runs were to be replicated using *Microsoft Azure Batch*, the timers could potentially become reliable and included in the analysis. Lastly, we also chose to exclude the analysis of the historical records since each node processes an immense amount of calculation objects, the sheer memory size of the history entries exhausted all available resources, slowing down execution to a standstill. This is, in our opinion, not a significant loss, as we did not have to prove Homomorphic Encryption, but as mentioned in Section 6.3, future work should try to develop a method for the nodes to validate the data.

[5]The full connectivity experiments were never able to finish with the originally planned 600 nodes, due to hardware constraints.

[6]The high memory consumption was the reason why we implemented a worker pool in the first place.

[7]`https://docs.microsoft.com/en-us/azure/batch/`

Chapter 7

Conclusion

We conclude the thesis by summarizing the main findings and contributions, followed by a summary of our recommendations for future work. The research questions we have addressed are divided into two parts. 1. How can a privacy-preserving arithmetic calculation be carried out in the context of decentralized digital contact tracing, and 2. What trade-offs, in terms of privacy, efficiency, complexity, the answer(s) for RQ1 have. We chose to divide the research into two approaches. Firstly, to be able to answer RQ1, we have sourced information in the fields of contact tracing, privacy, Bluetooth Low Energy, and Homomorphic Encryption. We then used this background information to model an approach to investigate and resolve RQ1. Next, we developed a simulation tool to explore the trade-offs regarding privacy, efficiency, and complexity of the modeled solution. The next section presents the contributions towards each of the research questions.

7.1 Contribution

In this thesis, we have designed and prototyped a proof-of-concept system that demonstrates that Homomorphic Encryption can be used in decentralized digital contact tracing to increase the capability to gather more complex data without breaching existing contact tracing privacy concerns. This method could be used in centralized- and decentralized-based models but is first and foremost an argument for decentralization. The system design and model used for simulations are novel in combining three existing fields of study, Homomorphic Encryption, contact tracing, and Bluetooth Low Energy, in a way not examined before. Through simulation experiments, we have highlighted some of the properties of our system proposal and further described its strengths and weaknesses. Our model utilizes connectionless broadcast/listening communication, and initial simulations with a fully connected network became our baseline benchmark. Connectionless routing produces much overhead; to reduce this, we have tested various methods for minimizing unnecessary traffic. The simulations prove that the concept is achievable and that proper optimization of the controlled flooding mechanisms can lead to

a low overhead system. As discussed in Section 6.2, we found that the most optimized system is when the network consists of smaller locale clusters. Where the packets have a short Time-To-Live value, and the nodes disregard duplicate data immediately. However, as it is presented in this paper, the model underperforms with relatively small privacy requirements. That is, when the node contribution to a calculation increases, the traffic generated increases exponentially. We can summarize our contributions in the following way:

- Presented an extensive overview of the enabling theory and background material.
- Provided a viable solution to perform privacy-preserving distributed calculation using short-range communication in a decentralized architecture in the context of digital contact tracing.
- Given recommendations derived from the observed traits on tuning the solution based on complexity, efficiency, and privacy.
- Recommended how future research can proceed based on observed potential and shortcomings in the solution.

7.2 Further work

We have previously given an extensive discussion on future research directions in Section 6.3. In our opinion, future work should be divided into two branches, the first focusing on enhancing performance in high privacy requirements, while the second should be looking more extended into the feasibility and eventually a proof-of-concept application targeting hand-held devices. Higher privacy should be of significant priority as it gives the solution its appeal and public trust. This is especially important in the context of contract tracing, as higher public trust could mean that more people are willing to participate in the collective responsibility. Equally important is the development of simulations that focuses on the impact of emulating the behavior or the Bluetooth protocol. This would eventually lead to a proof-of-concept application that would confirm the feasibility and potentially uncover new research areas. Our recommendations for future work can be summarized as the following:

1. Explore new methods of handling the exponentially growing traffic as the calculation process node contribution requirement increases.
2. Research the proposal using more realistic communication, e.g., make the nodes communicate through a Bluetooth simulator.
3. Research the impact of non-stationary nodes in the topology. It can be seen in conjunction with Point 2.
4. Investigate (1) the effect of implementing expiring packet caching in the nodes and (2) the feasibility of re-broadcasting the cached packets based on the livelihood of the node meeting the original sender node.

Bibliography

[1] I. Braithwaite, T. Callender, M. Bullock and R. W. Aldridge, 'Automated and partly automated contact tracing: A systematic review to inform the control of COVID-19,' English, *The Lancet Digital Health*, vol. 2, no. 11, e607–e621, Nov. 2020, Publisher: Elsevier, ISSN: 2589-7500. DOI: 10.1016/S2589-7500(20)30184-9. [Online]. Available: https://www.thelancet.com/journals/landig/article/PIIS2589-7500(20)30184-9/abstract (visited on 02/02/2021).

[2] CDC, *Case Investigation and Contact Tracing _ Part of a Multipronged Approach to Fight the COVID-19 Pandemic _ CDC*, en-us, Feb. 2020. [Online]. Available: https://www.cdc.gov/coronavirus/2019-ncov/php/principles-contact-tracing.html (visited on 30/01/2021).

[3] S. H. S. Lai, C. Q. Y. Tang, A. Kurup and G. Thevendran, 'The experience of contact tracing in Singapore in the control of COVID-19: Highlighting the use of digital technology,' en, *International Orthopaedics*, vol. 45, no. 1, pp. 65–69, Jan. 2021, ISSN: 1432-5195. DOI: 10.1007/s00264-020-04646-2. [Online]. Available: https://doi.org/10.1007/s00264-020-04646-2 (visited on 29/01/2021).

[4] Y. Zhang, J. Weng, R. Dey and X. Fu, 'Bluetooth Low Energy (BLE) Security and Privacy,' en, in *Encyclopedia of Wireless Networks*, X. (Shen, X. Lin and K. Zhang, Eds., Cham: Springer International Publishing, 2020, pp. 123–134, ISBN: 978-3-319-78262-1. DOI: 10.1007/978-3-319-78262-1_298. [Online]. Available: https://doi.org/10.1007/978-3-319-78262-1_298 (visited on 11/01/2021).

[5] J. Almagor and S. Picascia, 'Exploring the effectiveness of a COVID-19 contact tracing app using an agent-based model,' en, *Scientific Reports*, vol. 10, no. 1, p. 22 235, Dec. 2020, Number: 1 Publisher: Nature Publishing Group, ISSN: 2045-2322. DOI: 10.1038/s41598-020-79000-y. [Online]. Available: https://www.nature.com/articles/s41598-020-79000-y (visited on 02/02/2021).

[6] M. Klenk and H. Duijf, 'Ethics of digital contact tracing and COVID-19: Who is (not) free to go?' en, *Ethics and Information Technology*, Aug. 2020, ISSN: 1572-8439. DOI: 10.1007/s10676-020-09544-0. [Online]. Avail-

able: https://doi.org/10.1007/s10676-020-09544-0 (visited on 24/01/2021).

[7] F. Lucivero, N. Hallowell, S. Johnson, B. Prainsack, G. Samuel and T. Sharon, 'COVID-19 and Contact Tracing Apps: Ethical Challenges for a Social Experiment on a Global Scale,' en, *Journal of Bioethical Inquiry*, vol. 17, no. 4, pp. 835–839, Dec. 2020, ISSN: 1872-4353. DOI: 10.1007/s11673-020-10016-9. [Online]. Available: https://doi.org/10.1007/s11673-020-10016-9 (visited on 14/01/2021).

[8] T. Sharon, 'Blind-sided by privacy? Digital contact tracing, the Apple/-Google API and big tech's newfound role as global health policy makers,' en, *Ethics and Information Technology*, Jul. 2020, ISSN: 1572-8439. DOI: 10.1007/s10676-020-09547-x. [Online]. Available: https://doi.org/10.1007/s10676-020-09547-x (visited on 12/01/2021).

[9] T. Martin, G. Karopoulos, J. L. Hernández-Ramos, G. Kambourakis and I. Nai Fovino, *Demystifying COVID-19 Digital Contact Tracing: A Survey on Frameworks and Mobile Apps*, en, Review Article, ISSN: 1530-8669 Pages: e8851429 Publisher: Hindawi Volume: 2020, Oct. 2020. DOI: https://doi.org/10.1155/2020/8851429. [Online]. Available: https://www.hindawi.com/journals/wcmc/2020/8851429/ (visited on 02/02/2021).

[10] S. Vaudenay, 'Centralized or Decentralized? The Contact Tracing Dilemma,' Tech. Rep. 531, 2020. [Online]. Available: http://eprint.iacr.org/2020/531 (visited on 12/01/2021).

[11] H. Wen, Q. Zhao, Z. Lin, D. Xuan and N. Shroff, 'A Study of the Privacy of COVID-19 Contact Tracing Apps,' en, in *Security and Privacy in Communication Networks*, N. Park, K. Sun, S. Foresti, K. Butler and N. Saxena, Eds., ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Cham: Springer International Publishing, 2020, pp. 297–317, ISBN: 978-3-030-63086-7. DOI: 10.1007/978-3-030-63086-7_17.

[12] J. F. Kurose and K. W. Ross, *Computer Networking A Top-Down Approach*, 6th ed. Pearson Education Limited, 2013, ISBN: 978-0-273-76896.

[13] S. Cheruvu, A. Kumar, N. Smith and D. M. Wheeler, 'Connectivity Technologies for IoT,' en, in *Demystifying Internet of Things Security: Successful IoT Device/Edge and Platform Security Deployment*, S. Cheruvu, A. Kumar, N. Smith and D. M. Wheeler, Eds., Berkeley, CA: Apress, 2020, pp. 347–411, ISBN: 978-1-4842-2896-8. DOI: 10.1007/978-1-4842-2896-8_5. [Online]. Available: https://doi.org/10.1007/978-1-4842-2896-8_5 (visited on 13/10/2019).

[14] B. SIG, *Topology Options*, en-US, 2019. [Online]. Available: https://www.bluetooth.com/bluetooth-technology/topology-options/ (visited on 02/10/2019).

[15] S. Zeadally, F. Siddiqui and Z. Baig, '25 Years of Bluetooth Technology,' English, *Future Internet*, vol. 11, no. 9, p. 194, 2019, Num Pages: 194 Place: Basel, Switzerland Publisher: MDPI AG. DOI: http://dx.doi.org/10.3390/fi11090194. [Online]. Available: https://search.proquest.com/docview/2429918141/abstract/E5C780FBC90E4039PQ/1 (visited on 21/12/2020).

[16] K. Townsend, C. Cufí, R. Davidson and Akiba, 'Getting Started with Bluetooth Low Energy,' en, in *Getting Started with Bluetooth Low Energy*, O'Reilly Media, Inc., May 2014, ISBN: 978-1-4919-4951-1. [Online]. Available: https://www.oreilly.com/library/view/getting-started-with/9781491900550/ch01.html (visited on 15/02/2021).

[17] D. A. Gratton, *The Handbook of Personal Area Networking Technologies and Protocols*. Cambridge: Cambridge University Press, 2013, ISBN: 978-0-521-19726-7. DOI: 10.1017/CBO9780511979132. [Online]. Available: https://www.cambridge.org/core/books/handbook-of-personal-area-networking-technologies-and-protocols/6519CB7035222B38C149411BB7FD4392 (visited on 15/02/2021).

[18] J. Tosi, F. Taffoni, M. Santacatterina, R. Sannino and D. Formica, 'Performance Evaluation of Bluetooth Low Energy: A Systematic Review,' eng, *Sensors (Basel, Switzerland)*, vol. 17, no. 12, pp. 2898–, 2017, Place: Switzerland Publisher: MDPI AG, ISSN: 1424-8220. DOI: 10.3390/s17122898.

[19] M. P. Singh, M. D. Sharma and M. S. Agrawal, 'A Modern Study of Bluetooth Wireless Technology,' 2011.

[20] J. Muñoz-Benítez, R. González, A. Otero, L. A. Herrera, M. Huerta and G. Sagbay, *A flooding routing algorithm for a wireless sensor network for seismic events*. Oct. 2015. DOI: 10.1109/ICCSAT.2015.7362953.

[21] M. Labib, A. Ghalwash, S. Abdulkader and M. Elgazzar, 'Networking solutions for connecting bluetooth low energy devices - a comparison,' *MATEC Web of Conferences*, vol. 292, p. 02 003, Jan. 2019. DOI: 10.1051/matecconf/201929202003.

[22] J. DeCew, 'Privacy,' in *The Stanford Encyclopedia of Philosophy*, E. N. Zalta, Ed., Spring 2018, Metaphysics Research Lab, Stanford University, 2018. [Online]. Available: https://plato.stanford.edu/archives/spr2018/entries/privacy/ (visited on 12/02/2021).

[23] A. D. Moore, *Privacy, Security and Accountability: Ethics, Law and Policy*, en. Rowman & Littlefield International, 2016, ISBN: 978-1-78348-475-1.

[24] J. DeCew, 'Privacy,' in *Stanford Encyclopedia of Philosophy*, 2008.

[25] G. Buttarelli, 'Privacy matters: Updating human rights for the digital society,' en, *Health and Technology*, vol. 7, no. 4, pp. 325–328, Dec. 2017, ISSN: 2190-7196. DOI: 10.1007/s12553-017-0198-y. [Online]. Available: https://doi.org/10.1007/s12553-017-0198-y (visited on 16/04/2021).

[26] K. A. Wallace, 'Anonymity,' en, *Ethics and Information Technology*, vol. 1, no. 1, pp. 21–31, Mar. 1999, ISSN: 1572-8439. DOI: 10.1023/A:1010066509278. [Online]. Available: https://doi.org/10.1023/A:1010066509278 (visited on 09/02/2021).

[27] R.-Y. Xiao, 'Survey on Anonymity in Unstructured Peer-to-Peer Systems,' en, *Journal of Computer Science and Technology*, vol. 23, no. 4, pp. 660–671, Jul. 2008, ISSN: 1860-4749. DOI: 10.1007/s11390-008-9162-7. [Online]. Available: https://doi.org/10.1007/s11390-008-9162-7 (visited on 09/02/2021).

[28] M. A. Will and R. K. L. Ko, 'Chapter 5 - A guide to homomorphic encryption,' en, in *The Cloud Security Ecosystem*, R. Ko and K.-K. R. Choo, Eds., Boston: Syngress, Jan. 2015, pp. 101–127, ISBN: 978-0-12-801595-7. DOI: 10.1016/B978-0-12-801595-7.00005-7. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128015957000057 (visited on 03/04/2021).

[29] C. Gentry, 'Computing arbitrary functions of encrypted data,' en, *Communications of the ACM*, vol. 53, no. 3, pp. 97–105, Mar. 2010, ISSN: 0001-0782, 1557-7317. DOI: 10.1145/1666420.1666444. [Online]. Available: https://dl.acm.org/doi/10.1145/1666420.1666444 (visited on 05/03/2021).

[30] A. Acar, H. Aksu, A. S. Uluagac and M. Conti, 'A Survey on Homomorphic Encryption Schemes: Theory and Implementation,' *ACM Computing Surveys*, vol. 51, no. 4, 79:1–79:35, Jul. 2018, ISSN: 0360-0300. DOI: 10.1145/3214303. [Online]. Available: https://doi.org/10.1145/3214303 (visited on 05/03/2021).

[31] C. Zhao, S. Zhao, M. Zhao, Z. Chen, C.-Z. Gao, H. Li and Y.-a. Tan, 'Secure Multi-Party Computation: Theory, practice and applications,' en, *Information Sciences*, vol. 476, pp. 357–372, Feb. 2019, ISSN: 0020-0255. DOI: 10.1016/j.ins.2018.10.024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025518308338 (visited on 19/02/2021).

[32] P. Paillier, 'Public-Key Cryptosystems Based on Composite Degree Residuosity Classes,' en, in *Advances in Cryptology — EUROCRYPT '99*, J. Stern, Ed., ser. Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 1999, pp. 223–238, ISBN: 978-3-540-48910-8. DOI: 10.1007/3-540-48910-X_16.

[33] R. Johnsonbaugh, *Discrete Mathematics*, 7th. Pearson Education Limited, 2014, ISBN: 978-1-292-02261-1.

[34] T. Jager, 'The Generic Composite Residuosity Problem,' en, in *Black-Box Models of Computation in Cryptology*, Wiesbaden: Vieweg+Teubner Verlag, 2012, pp. 49–56, ISBN: 978-3-8348-1989-5 978-3-8348-1990-1. DOI: 10.1007/978-3-8348-1990-1_5. [Online]. Available: http://link.springer.com/10.1007/978-3-8348-1990-1_5 (visited on 03/04/2021).

[35] S. Farah, Azhara, M. Javed, Shamim and T. Nawaz, *An experimental study on Performance Evaluation of Asymmetric Encryption Algorithms*. Dec. 2012.

[36] P. Leedy D and J. E. Ormrod, *Practical Research Planning and Design*, 11th ed. Pearson, 2015, ISBN: 1-292-09587-3.

Appendix A

Simulation run data

Table A.1: Planned simulation run configurations

#	Impact area	Nr. of nodes	Latency	Decrypt Threshold	TTL	Topology	Topology size
1	Complexity	25	30	3	10	All	-
2	Complexity	100	30	3	10	All	-
3	Complexity	200	30	3	10	All	-
4	Complexity	600	30	3	10	All	-
5	Complexity	25	30	3	10	Cluster	8
6	Complexity	100	30	3	10	Cluster	8
7	Complexity	200	30	3	10	Cluster	8
8	Complexity	600	30	3	10	Cluster	8
9	Complexity	100	100	3	10	All	-
10	Complexity	100	200	3	10	All	-
11	Complexity	100	100	3	10	Cluster	8
12	Complexity	100	200	3	10	Cluster	8
13	Efficiency	25	30	3	6	All	-
14	Efficiency	25	30	3	12	All	-
15	Efficiency	100	30	3	6	All	-
16	Efficiency	100	30	3	12	All	-
17	Privacy	25	30	3	10	All	-
18	Privacy	25	30	6	10	All	-
19	Privacy	25	30	9	10	All	-
20	Privacy	100	30	3	10	All	-
21	Privacy	100	30	6	10	All	-
22	Privacy	100	30	9	10	All	-

Table A.2: Performed simulation run configurations. Note that configurations are ordered by time of execution and does not necessarily correspond to the same ordering found in the code.

#	Impact area	Nr. of nodes	Latency	Decrypt Threshold	TTL	Topology	Topology size
1	Complexity	25	30	3	10	All	-
2	Complexity	100	30	3	10	All	-
3	Complexity	200	30	3	10	All	-
4	Complexity	600	30	3	10	All	-
5	Complexity	25	30	3	10	Cluster	8
6	Complexity	100	30	3	10	Cluster	8
7	Complexity	200	30	3	10	Cluster	8
8	Complexity	600	30	3	10	Cluster	8
9	Complexity	100	100	3	10	All	-
10	Complexity	100	200	3	10	All	-
11	Complexity	100	100	3	10	Cluster	8
12	Complexity	100	200	3	10	Cluster	8
13	Efficiency	100	30	3	3	Cluster	8
14	Efficiency	100	30	3	6	Cluster	8
15	Efficiency	100	30	3	9	Cluster	8
16	Efficiency	100	30	3	12	Cluster	8
17	Efficiency	100	30	3	15	Cluster	8
18	Efficiency	100	30	3	18	Cluster	8
19	Privacy	100	30	3	6	Cluster	8
20	Privacy	100	30	4	6	Cluster	8
21	Privacy	100	30	5	9	Cluster	8
22	Privacy	100	30	6	60	Cluster	8

N	L	DT	TTL	T	TS	B	EU	R	D	PK	IU	PD
25	30	3	10	All		8653	494	29	8130	543	478	7125
100	30	3	10	All		11253	1765	5	9471	104	98	15931
25	30	6	10	All		9034	402	475	8008	483	1	7406
100	30	6	10	All		12516	1715	34	10465	70	21	17470
25	30	9	10	All		8602	360	319	7848	319	0	8488
100	30	9	10	All		11711	1748	97	9731	105	5	15918
25	30	3	6	All		3791	449	23	3315	277	259	4517
100	30	3	6	All		5819	1540	4	4217	102	99	13084
25	30	3	12	All		10935	410	20	10506	426	416	6791
100	30	3	12	All		16312	1937	2	14329	70	69	17468
25	30	3	10	All		8910	440	27	8450	487	457	7318
100	30	3	10	All		10559	1717	2	8826	71	68	15034
200	30	3	10	All		12626	3161	1	9442	34	33	23655
600	30	3	10	All		97979	25154	1	72786	3	2	0
25	30	3	10	Cluster	8	6367	117	166	6094	1158	949	4058
100	30	3	10	Cluster	8	7148	158	258	6696	1094	799	4980
200	30	3	10	Cluster	8	7049	169	254	6619	1080	817	4600
600	30	3	10	Cluster	8	6662	172	212	6239	953	739	3697
100	100	3	10	All		13275	1774	5	11499	118	113	17759
100	200	3	10	All		12090	1806	6	10255	143	137	17198
100	100	3	10	Cluster	8	7148	147	236	6701	1035	788	4722
100	200	3	10	Cluster	8	7140	170	242	6710	943	726	4632

Table A.3: Raw data exported from IBM SPSS. N = Nr of nodes, L = Latency (ms), DT = Decrypt Threshold, TTL = Time-To-Live, T = Topology, TS = Topology Size, B = Broadcasts, EU = External Updates, R = Rejects, D = Duplicates, PK = Public Key matches, IU = Internal Updates, PD = Packets Dropped

Table A.4: Required run complete time

# of nodes	Topology	Seconds
25	Full connectivity	100
100	Full connectivity	1000
200	Full connectivity	4400
600	Full connectivity	Unable to finish
25	Cluster	60
100	Cluster	200
200	Cluster	600
600[a]	Cluster	1000

[a]Not used in the analysis as the 600 nodes, fully connected run was not able to finish

Appendix B

Analysis

Report

DT		B	EU	R	D	PK	IU	PD
3	Mean	186.50	178.42	7.08	949.40	50.92	43.84	165.28
	N	100	100	100	100	100	100	100
	Std. Deviation	36.849	36.208	.895	207.243	10.394	9.512	49.407
4	Mean	23583.53	170.41	7274.29	16137.83	7307.28	32.99	86254.57
	N	100	100	100	100	100	100	100
	Std. Deviation	4624.930	31.759	1978.212	3290.213	1984.590	12.898	20567.199
5	Mean	111491.87	221.00	15712.50	95557.37	15726.70	14.20	165329.24
	N	100	100	100	100	100	100	100
	Std. Deviation	8801.138	38.802	6256.468	9087.920	6257.814	21.370	14778.338
6	Mean	1642446.13	449.07	114657.49	1527338.57	114661.74	4.25	1594251.35
	N	100	100	100	100	100	100	100
	Std. Deviation	12358.403	87.945	69460.442	72325.561	69460.255	6.206	76857.176
Total	Mean	444427.01	254.72	34412.84	409995.79	34436.66	23.82	461500.11
	N	400	400	400	400	400	400	400
	Std. Deviation	693833.120	125.938	58228.431	647927.223	58218.447	20.671	658654.463

Figure B.1: Decrypt threshold - Report

Group Statistics

	ED	N	Mean	Std. Deviation	Std. Error Mean
B	Control	400	73300.67	71959.696	3597.985
	Early Drop	400	258.02	159.181	7.959
EU	Control	400	179.52	88.690	4.435
	Early Drop	400	249.94	159.036	7.952
R	Control	400	2970.30	3884.556	194.228
	Early Drop	400	7.08	.892	.045
D	Control	400	70149.86	69431.051	3471.553
	Early Drop	400	1280.17	1018.907	50.945
PK	Control	400	11682.20	12091.199	604.560
	Early Drop	400	50.10	9.893	.495
IU	Control	400	8711.90	8643.690	432.185
	Early Drop	400	43.02	9.047	.452
PD	Control	400	42085.94	26424.964	1321.248
	Early Drop	400	1550.23	1011.714	50.586

Figure B.2: Group statistics Early drop

Report

TTL		B	EU	R	D	PK	IU	PD
3	Mean	59.00	50.92	7.08	.00	47.64	40.56	327.46
	N	300	300	300	300	300	300	300
	Std. Deviation	9.428	8.546	.892	.000	7.861	7.066	62.252
6	Mean	16277.98	174.77	1371.46	14730.74	7417.20	6045.74	49030.61
	N	300	300	300	300	300	300	300
	Std. Deviation	3186.144	33.504	550.045	2947.764	1982.458	1630.778	11153.364
9	Mean	95945.93	230.29	3708.18	92006.46	16028.52	12320.33	57317.96
	N	300	300	300	300	300	300	300
	Std. Deviation	7304.508	41.467	2169.592	7550.693	7100.161	5524.382	13055.302
12	Mean	180435.95	262.56	6460.81	173711.58	23528.92	17068.12	59878.57
	N	300	300	300	300	300	300	300
	Std. Deviation	6058.080	50.019	4356.630	7932.794	13132.367	10014.789	9692.705
15	Mean	265650.50	286.45	9336.49	256026.56	29265.95	19929.47	66404.74
	N	300	300	300	300	300	300	300
	Std. Deviation	5523.979	55.978	7793.636	10559.141	18476.628	13114.755	10255.734
18	Mean	346850.63	305.81	11355.90	335187.93	32923.55	21567.65	16494.18
	N	300	300	300	300	300	300	300
	Std. Deviation	6470.489	60.741	9544.088	12217.420	21072.862	14479.428	15246.332
Total	Mean	150870.00	218.47	5373.32	145277.21	18201.96	12828.64	41575.59
	N	1800	1800	1800	1800	1800	1800	1800
	Std. Deviation	126858.449	97.080	6778.053	122857.669	17454.809	12030.908	26807.232

Figure B.3: TTL - Report

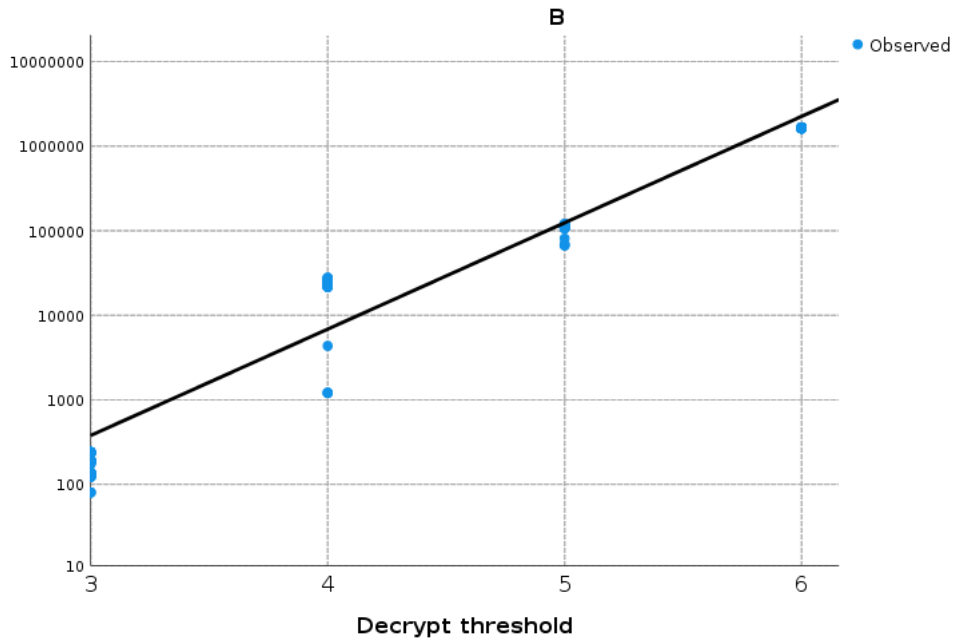


Figure B.4: Mean broadcast by Decrypt threshold, logarithmic

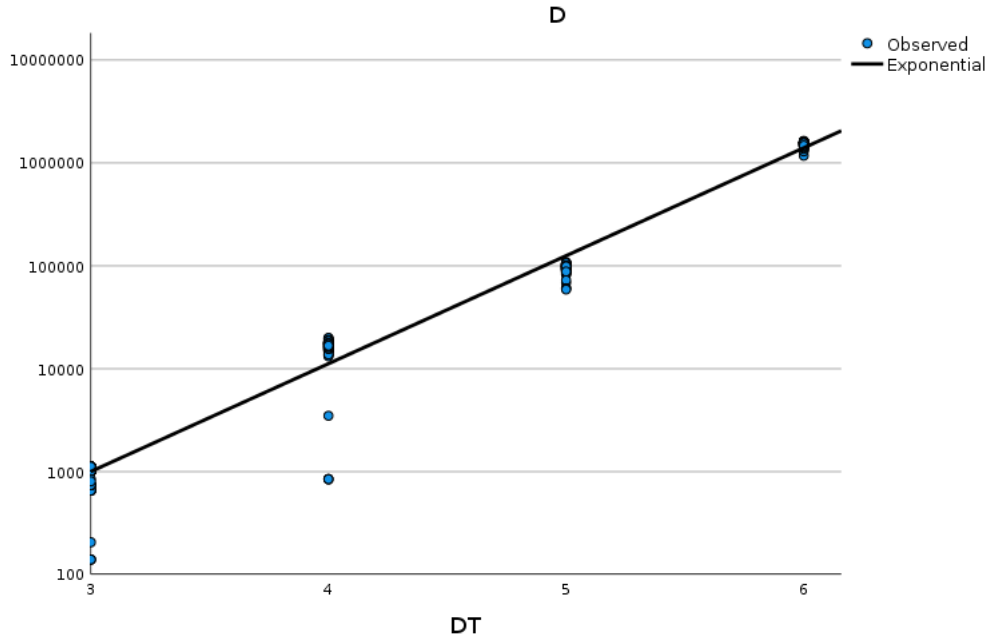


Figure B.5: Mean duplicates by Decrypt threshold, logarithmic

Post Hoc Tests

Multiple Comparisons

Games-Howell

Dependent Variable	(I) NS	(J) NS	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval Lower Bound	95% Confidence Interval Upper Bound
B	25	100	-5819.360*	1001.773	.000	-8202.71	-3436.01
		200	-8325.947*	932.885	.000	-10556.18	-6095.72
	100	25	5819.360*	1001.773	.000	3436.01	8202.71
		200	-2506.587*	390.801	.000	-3426.54	-1586.64
	200	25	8325.947*	932.885	.000	6095.72	10556.18
		100	2506.587*	390.801	.000	1586.64	3426.54
EU	25	100	-86.037*	2.773	.000	-92.56	-79.51
		200	-107.098*	1.776	.000	-111.28	-102.92
	100	25	86.037*	2.773	.000	79.51	92.56
		200	-21.062*	2.938	.000	-27.97	-14.16
	200	25	107.098*	1.776	.000	102.92	111.28
		100	21.062*	2.938	.000	14.16	27.97
R	25	100	453.283	449.015	.573	-616.12	1522.69
		200	627.472	432.392	.320	-404.99	1659.93
	100	25	-453.283	449.015	.573	-1522.69	616.12
		200	174.188	173.966	.576	-234.73	583.10
	200	25	-627.472	432.392	.320	-1659.93	404.99
		100	-174.188	173.966	.576	-583.10	234.73
D	25	100	-6186.607*	1046.258	.000	-8673.88	-3699.34
		200	-8846.320*	966.491	.000	-11155.84	-6536.80
	100	25	6186.607*	1046.258	.000	3699.34	8673.88
		200	-2659.713*	450.079	.000	-3718.84	-1600.59
	200	25	8846.320*	966.491	.000	6536.80	11155.84
		100	2659.713*	450.079	.000	1600.59	3718.84
PK	25	100	1342.250	1431.016	.618	-2063.84	4748.34
		200	1107.677	1371.641	.699	-2166.17	4381.53
	100	25	-1342.250	1431.016	.618	-4748.34	2063.84
		200	-234.573	602.660	.920	-1651.04	1181.90
	200	25	-1107.677	1371.641	.699	-4381.53	2166.17
		100	234.573	602.660	.920	-1181.90	1651.04
IU	25	100	888.967	1096.699	.697	-1720.42	3498.36
		200	480.205	1046.211	.891	-2016.63	2977.04
	100	25	-888.967	1096.699	.697	-3498.36	1720.42
		200	-408.762	478.518	.669	-1533.50	715.98
	200	25	-480.205	1046.211	.891	-2977.04	2016.63
		100	408.762	478.518	.669	-715.98	1533.50
PD	25	100	13639.757*	1537.370	.000	9988.85	17290.67
		200	19206.088*	1401.858	.000	15858.19	22553.98
	100	25	-13639.757*	1537.370	.000	-17290.67	-9988.85
		200	5566.332*	738.573	.000	3828.77	7303.89
	200	25	-19206.088*	1401.858	.000	-22553.98	-15858.19
		100	-5566.332*	738.573	.000	-7303.89	-3828.77

*. The mean difference is significant at the 0.05 level.

Figure B.6: Multiple comparison table with Games-Howell of the impact of Node Size

Appendix C

Search protocol

When we searched for material related to the problem description and research question, multiple consortiums were used. The following Digital sources were used when collecting theory: Link Springer, The ACM Digital Library, and Google Scholar. Additional sources include Bluetooth SiG published documents, information documents published by various health agencies and government bodies, physical books, and some new articles to help with the narrative.

Table C.1: Identified search words

Search word
Contact Tracing
Digital Contact Tracing
Bluetooth
Bluetooth Low Energy
Ad hoc routing
Network Flooding
Privacy
Anonymity
Multi-party Computation
Homomorphic Encryption
Paillier
proof of work
shared objects
non interactive proof of work
zero knowledge

Not all the search words produced results that ended up in the paper. Further, the keywords target different parts. Some are related to technical topics, architecture, scenario processes, networking, security, with more.

Markus André Jacobsen

Privacy-Preserving Decentralized Calculation in Digital Contact Tracing

NTNU
Norwegian University of
Science and Technology