Privacy-Preserving Data Aggregation with Probabilistic Range Validation

F. W. Dekker

TU Delft

Privacy-Preserving Data Aggregation with Probabilistic Range Validation

by

F. W. Dekker

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday, September 25, 2020, at 13:00 PM.

Student number:	4461002	
Project duration:	November 14, 2019 – September 25, 2020	
Thesis committee:	Dr. ir. Z. Erkin,	TU Delft, supervisor
	Dr. ir. S. Picek,	TU Delft
	Dr. ir. M. F. Aniche,	TU Delft

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Abstract

Privacy-preserving data aggregation protocols have been researched widely, but usually cannot guarantee correctness of the aggregate if users are malicious. These protocols can be extended with zero-knowledge proofs and commitments to work in the malicious model, but this incurs a significant computational cost on the end users, making adoption of such protocols less likely.

We propose a privacy-preserving data aggregation protocol for calculating the sum of user inputs. Our protocol gives the aggregator confidence that all inputs are within a desired range. Instead of zero-knowledge proofs, our protocol relies on an asynchronous probabilistic hypergraph-based detection algorithm with which the aggregator can quickly pinpoint malicious users. Our protocol is robust to user dropouts and is non-interactive apart from the registration phase. We describe several optional extensions to our protocol for temporal aggregation, dynamic user joins and leaves, and differential privacy. We analyse our protocol in terms of security, privacy, and detection rate. Finally, we compare the runtime complexity of our protocol with a selection of related protocols.

Preface

This thesis describes my alternative solution to detecting malicious users in a privacy-preserving data aggregation protocol. My decision to dive into this field of research came after following the Privacy-Enhancing Technologies course by dr. ir. Erkin—who would later supervise this very thesis—and being inspired by the almost-magical properties of homomorphic encryption and oblivious transfer, amongst others. During my thesis, I first read up on privacy-preserving data aggregation for three months. Then, I decided to research the issue of detecting malicious users because the solutions I found were more like workarounds than proper solutions. So, I came up with my own solution, and then spent four months investigating my own solution to find out if it was any good. Luckily for me, it was, so I spent the final three months writing my thesis. Meanwhile, I wrote a paper about my solution, the second attempt of which is yet to be reviewed. All in all, this thesis is the culmination of 310 days of work, an estimated 1.063% of my entire life until my projected death. It is the crowning jewel on my 1846 days (6.189%) as a computer science student at TU Delft, and while it is neither perfect nor exhaustive, it is something I am proud of.

I thank Zekeriya Erkin especially for keeping me engaged and motivated, particularly during my time in isolation because of some sort of worldwide pandemic; working from home does not suit me in the least. I thank Oğuzhan Ersoy for his mathematical and cryptographic help during both the design and the analysis of my solution. I thank Dion Gijswijt for his mathematical insights, even though I was too late to process them into this work. Finally, I thank my family, girlfriend, friends, and colleagues for their patience on the occasions when I prioritised my thesis over them.

F. W. Dekker
Delft, September 2020

Contents

1 Introduction

> As we move toward greater surveillance, we need to figure out [...] how to design systems that make use of our data collectively to benefit society as a whole, while at the same time protecting people individually.
>
> *Bruce Schneier* [58]

In the age of (mis)information, data is power, and nothing shows this more than when Cambridge Analytica arguably helped disrupt democracy during the 2016 United States presidential election, the United Kingdom's Brexit vote, and many more elections [40]. Meanwhile, well over 300 companies provide similar data-driven political campaigns throughout the world [44]. These companies make use of vast amounts of privacy-sensitive data, sold to them by specialised companies that combine personal data from trackers, government censuses, and more [52]. Governments have started to recognise that this is a threat to democracy and regulations have started to appear, but governments are equally guilty of privacy invasion, now for surveillance purposes. One solution to both problems may lie in the adoption of privacy-preserving data aggregation protocols, which allow companies to continue to analyse privacy-sensitive data while ensuring that these data remain confidential. The adoption of these protocols is not without its problems, and it is important these protocols are designed with the requirements of the aggregating parties in mind to make them as appealing as possible.

1.1. Data aggregation

We cannot deny that data analytics are immensely useful for communities and individuals alike. At the same time, the data underpinning these analyses must come from somewhere, and the required sacrifice of privacy ranges from trivial to invasive.

As an example, Google Maps informs users of nearby traffic jams, interesting locations, and predicts how busy shops will be at a given moment. More generally speaking, participatory sensing, a form of crowdsensing, allows an aggregator to create a whole body of knowledge by combining measurements from many users [18]. For example, crowdsensing can be used to learn about the current temperature, humidity, and air pollution, the locations of unprotected Wi-Fi access points or speed cameras, or information about health-related issues such as depression and stress conditions [18, 45]. Oftentimes, the aggregator makes the results publicly accessible through a dedicated application. The business models of these aggregators may be to sell the individual users' data, to include advertisements in a dedicated application, or to license a subset of the data to third parties. The spatio-temporal measurements provided by users explicitly contain private information which can be used to de-anonymise users [18]. It is not feasible for the aggregator to work with anonymised measurements from users, not only because this would destroy the business model but also because the aggregator wants users to be accountable for their measurements.

As another example, applications such as Amazon Echo, Siri, and Google Assistant give accurate answers to questions posed by the user by contacting an underlying knowledge base, contextualised by the identity of the speaker. The collected data can be sold to advertisement technology companies to build a profile of

the user to create targeted advertisements; in 2018, this market was worth over US$100 billion in the USA alone [33]. Similarly, YouTube and Netflix track what their users are watching and what they think of the content [3, 49]. In each of these cases, the data processor builds a profile based on usage data which can be compared to other profiles to create recommendations or to predict user behaviour. Because collecting private information is at the core of these companies' business models, these companies exploit the power imbalances between themselves and their users to nudge users into sharing even more data using so-called "dark patterns" [2]. Applications embed dark patterns into their user interface to trick users into doing things they might not otherwise have done, for example by choosing specific default settings for the application, or using ambiguous or biased language when describing the consequences of a user's actions. At the same time, profiles are not always accurate, and underlying biases in the model may exacerbate existing discrimination [15]. The requirement that these profiles are compared with each other makes it hard to safeguard privacy in recommender systems.

Another example of the usefulness of data aggregation is telemetry, a system wherein application vendors automatically receive information on user behaviour. The developers can use the collected data to better understand how users interact with the software to improve the user experience. Google Chrome makes extensive use of telemetry and collects data such as the user's current settings, crash reports, and search predictions [1]. While Google Chrome also stores information such as bookmarks to help users with synchronisation between devices, this data can be encrypted using a user-specific key to prevent spying from Google's side. Telemetry, however, cannot be stored under plain encryption because the telemetry data needs to be processed to be useful to Google's developers. Even if Google anonymises the telemetry data before it is sent, it might still be possible to identify individual users based on the exact contents of the data [48].

A more traditional example of data aggregation is the smart grid, in which users report their current power consumption at a regular time interval [31]. The aggregator is interested in simple metrics such as the total amount of power consumed by its customers. With these data, the aggregator can precisely adjust and predict the power usage of the network in real time, allowing it to reduce overheads and decrease prices for its customers. However, by giving the aggregator access to live power consumption, the aggregator can see exactly which devices turn on at which time. These data may reveal people's daily schedules, cultural background, and even religion [31, 46].

While some companies do not shy away from collecting and processing intimate private information, such as when mental health websites sell behavioural data of their visitors [53], when Amazon records every sound it can obtain [21], or when Grindr reveals people's HIV status to third parties [9], even companies with the users' best interests in mind eventually fail. Developers make poor design decisions all the time, and data breaches occur regularly. Unsurprisingly, user trust in the cybersecurity abilities of companies and governments has been decreasing steadily [54].

1.2. Privacy...

While the benefits exchanged for these data are measurably positive, analytics may be seriously harmful to one's privacy, and the consequences are far-reaching.

On one hand, privacy is necessary to safeguard the dignity and integrity of individuals. A certain level of privacy is required for people to feel safe enough to explore themselves, their opinions, and their surroundings without being subjected to the judgment of others. Without privacy, shame would leave no room for learning and self-development, and the world would become a blander place. The feeling of being watched inhibits actions the watchee deems sinful or wrong, even if it would make them a better person. Even people who claim that they have "nothing to hide" would probably feel uncomfortable sharing their diary, the data from a diet app, or their browser history with their colleagues. At the same time, people are prepared to share this same information with app vendors, apparently under the impression that the short-term benefits outweigh the long-term privacy risks. However, even if the vendor does not intend to do more than just storing the minimal amount of necessary data, the application may have severe flaws. For example, some women's health apps fail to use authenticated connections, allow for remote code execution, and suffer from database leaks [55]. Meanwhile, dignity and integrity are also important in the personal sphere. For example, privacy protects the marriage equality of men and women in an abusive relationship by giving them a private place to store their thoughts and communicate with others without being spied upon and controlled by their partners [4].

On the other hand, privacy resolves the imbalance between individuals and powerful entities such as governments and large corporations by protecting these individuals from surveillance and suppression. The data

leaks provided by Edward Snowden are a prominent example of how widespread government surveillance is. The NSA has been working with telecommunications companies for many years to collect the private information of virtually all American citizens [35], and additionally collude with intelligence agencies throughout the world to increase their coverage [43]. At the same time, the Chinese government has initiated controversial privacy-intrusive schemes, such as a social credit system and Uyghur re-education camps [22] to enforce cultural standards onto the population. In many countries in the world, governments suppress or even persecute minorities [38] or those with dissenting opinions [57]. In these cases, privacy is important because it protects one's right to be who they are and to express themselves as they are regardless of what the government deems correct or legal today. It protects political freedom and exploration and protects independent journalism. It protects freedom of thought and freedom to protest. It protects oppressed minorities from persecution. Privacy is essential for any functional democracy and is an essential right for any human being.

Indeed, privacy has been enshrined in the Universal Declaration of Human Rights back in 1948. It is only recently that governments have started to adopt clear legislation regarding the processing of privacy-sensitive data. As of 2019, hundreds of governments throughout the world have adopted legislative work describing the data rights of its people [34]. Most notably, the GDPR went into effect in May 2018 in the European Union, providing its citizens with the right to know how and why their privacy-sensitive data are collected, amongst others. Furthermore, the European Union's planned ePrivacy Regulation will replace the old ePrivacy Directive to further streamline European citizens' data rights. With these and similar regulations, data control is moved from powerful data processors back to the data subjects, thereby fixing the power imbalance between them. Data processors will have to be more transparent and careful with privacy-sensitive data, increasing their accountability. However, for businesses relying on the collection of massive amounts of this kind of data, compliance is no easy feat and fundamental changes must be made before we see any real change.

1.3. . . . by design

We cannot expect privacy to be solved just by creating regulations. Companies must be convinced to embrace privacy in the applications they create, but bolting on control mechanisms and naively encrypting data does nothing to stop the invasion of privacy. Instead, privacy must be embedded deeply into applications and their design to prevent privacy violations in the first place. This approach is known as privacy by design.

The first description of privacy by design was published as far back as in 1995, in a report about the use of so-called privacy-enhancing technologies (PETs) [66]. PETs provide rigorous guarantees about certain privacy properties and can be embedded transparently into existing applications. For example, the anonymous credentials PET allows users to authenticate themselves using only the data that is strictly necessary for the authentication. If someone wants to prove that they are of legal age when purchasing alcohol, this PET allows them to prove just their age; details such as name, nationality, and even date of birth remain confidential. Another example is multi-party computation, a PET that allows different parties to jointly compute a specified function on privacy-sensitive inputs without revealing the inputs to each other. This can be used to track the spread of malware in networks operated by different providers without these providers having to reveal details about their networks to each other.

While writing the 1995 joint report, the authors realised that the mere availability of PETs does not guarantee privacy. Without clear guidelines on when and how to use PETs, it is easy to abuse or misuse PETs in such a way that nothing changes. Simply gluing PETs onto existing applications does very little to change the deeper privacy issues that an application has, such as the collection of unnecessary privacy-sensitive data or the use of the previously mentioned dark patterns. It is for these reasons that the report is about more than just PETs; it is about a thorough systems engineering approach where privacy is central to the design of the application. The corresponding framework, published in 2009 [16], summarises the goals succinctly in its seven foundational principles. Amongst these principles are the requirements that privacy should be proactive rather than reactive, privacy should be the default setting in applications, privacy should be considered from the very first steps of the application's design, and privacy should be considered in all aspects of the system, ranging from the underlying infrastructure to the responsibilities carried by the business.

The privacy by design framework has been criticised [23, 36] for failing to stress the importance of data minimisation. These critics argue that without a precise description of the principles underlying data minimisation, the privacy by design framework risks legitimising the collection of copious amounts of privacy-sensitive data as long as the data collector expresses the purpose of the collection and provides appropriate controls. For example, without clear data minimisation guidelines, a data collector may argue that it is necessary to collect pictures of user passports to verify that a user is of legal age, ignoring the opportunity to use

the anonymous credentials PET. Even when the application gives the user control over these data, there is no guarantee that the data will not leak or be abused. Data minimisation principles such as "minimise data collection" imply the need to avoid collecting these data through the deployment of PETs. Even with regulations in place that recognise the need for privacy by design, it is important to continue research into the development of PETs to create new opportunities to reduce data collection.

A PET of particular interest to us is privacy-preserving data aggregation (PDA). PDA protocols allow an aggregating party, called the aggregator, to calculate aggregate statistics on privacy-sensitive data from its users without having to access that data. Users encrypt their data according to the protocol's specifications and send this to the aggregator at a regular interval or upon a specific trigger. The aggregator collects the received inputs and calculates the desired aggregates, such as the sum, mean, or maximum of the user data. At the same time, the protocol's design guarantees that the aggregator cannot derive any single user's private value. Essentially, by using a PDA protocol, the aggregator can continue aggregating and analysing user data as it would have before, but now it can guarantee that users' private data remain confidential.

PDA protocols may achieve these privacy guarantees in various ways. Some protocols, such as [29, 31], use a combination of homomorphic encryption and secret sharing. With homomorphic encryption, users encrypt their data such that the aggregator can perform specific calculations on the ciphertexts without the need to perform intermediate decryptions. It is only once all data have been combined that the aggregator needs to decrypt the ciphertext to obtain the aggregate. By secret sharing the inputs before homomorphically encrypting it, the protocol guarantees that intermediate decryptions by the aggregator reveal no sensitive information. Other PDA protocols, such as [7], insert a proxy in between the users and the aggregator. Users encrypt their data with the aggregator's key and send it to the proxy, which shuffles data of several users and forwards these to the aggregator. This way, the proxy is unable to see individual values, and the aggregator does not know which datum belongs to which user. There are many more ways to implement PDA protocols, but they all allow the aggregator to calculate useful metrics without sacrificing user privacy. These privacy properties usually come at the cost of increased computation complexity or bandwidth usage.

1.4. Adopting new techniques

PDA protocols seem to solve the issue of combining data aggregation with privacy by design, so all that is left is for companies to implement them into their applications. PDA protocols are attractive because they remove the need for data processors to even come into contact with privacy-sensitive data, diminishing worries about data breaches. In terms of performance, existing protocols provide relatively good performance for both users and the aggregator, and while the introduced overhead is not trivial, these protocols have only been optimised more and more recently. In fact, with Google having deployed a custom PDA protocol for telemetry purposes in its Google Chrome browser [11, 30], the practicality of PDA protocols is evident.

However, without widespread adoption of PDA protocols, it is clear something is lacking. One drawback of existing PDA protocols is that if all individual data points are confidential, the aggregator cannot perform input validation, making PDA protocols useless for applications that cannot trust their users. Consider crowdsensing, where the aggregator wants to filter out users that, say, report moving faster than light or users that claim to be inside the core of the earth. Similarly, recommender systems want to detect and remove fake accounts before continuing their analyses, and in smart grids, the network operator wants to detect when smart meters have been modified to underreport power consumption. Regardless of whether these invalid measurements come from faulty hardware, from malicious users with political motives, or from something else, aggregators want to identify the inputs that should be thrown out. For all these reasons and more, we need a PDA protocol that also allows the aggregator to perform some sort of input validation. Unfortunately, many existing PDA protocols, such as [29, 42, 67], assume that users always honestly follow the protocol. Without a protocol that can deal with malicious users, aggregators have little reason to transition to PDA protocols, leaving end users vulnerable to privacy violations.

Several protocols allow an aggregator to perform input validation. Notably, zero-knowledge proofs (ZKPs) allow users to create mathematically rigorous proofs that their values satisfy the aggregator's specifications, without disclosing any information about the value itself. ZKPs can easily be integrated into existing PDA protocols to allow them to work with malicious users, as suggested in [29, 42, 60]. This integration requires that a "glue" is applied between the ZKP and the original PDA protocol to ensure that the two are the same; this glue can be applied in the form of a commitment scheme. However, ZKPs often require either a trusted setup or significant computation resources from users [47], which makes this solution unappealing or even infeasible for aggregators working with resource-constrained users.

1.5. Research goal

In our research, we focus specifically on sum-based aggregation where the aggregator wants to obtain the sum of all user inputs. Sum-based aggregation is sufficient as a primitive to build complex algorithms such as principal component analysis, singular-value decomposition, and decision tree classifications if the aggregator can express the aggregate as an aggregate-sum query, as explained in [12]. Furthermore, we focus on range validation, where the aggregator wants to ensure that all inputs are within a pre-defined numerical range. Our research question is thus as follows.

> How can range validation be incorporated into a privacy-preserving data summation protocol without using zero-knowledge proofs?

1.6. Contributions

We propose a privacy-preserving data aggregation protocol with probabilistic range validation that sums user inputs. Users start by registering with the aggregator, who divides the users into a multitude of overlapping groups such that each user is in a unique set of groups, and negotiates the exchange of information between users such that users do not have to interact with each other directly. Users report a privacy-sensitive measurement at a regular interval called a round in such a way that the aggregator obtains the sum of user inputs for each group. The aggregator finds the total sum by combining the sums of the right groups. The aggregator verifies that each group's aggregate is within a pre-defined valid range, and looks at the intersection of all groups that fails this verification to pinpoint exactly which users contributed invalid measurements. Our grouping mechanism guarantees that the aggregator never misidentifies users, but the probabilistic nature of the validation means that a group with malicious users may not be detected right away. Our protocol guarantees privacy in the standard model using a combination of secret sharing and commitments.

Our protocol boasts several important properties. Firstly, the scheme can be customised to strike a balance between privacy, complexity, and detection rate. For example, one can increase the protocol's resistance to user collusions, but this necessitates more work per round from the aggregator. Secondly, our protocol is realistic. It does not require a trusted setup and is non-interactive apart from the registration phase: Users simply send their encrypted values to the aggregator, who then aggregates and validates asynchronously. Thirdly, our protocol is an efficient solution for aggregators relying on resource-constrained users; users are subject to logarithmic computational complexity in the number of users. Fourthly, the grouping structure of our protocol gives the protocol robustness as the aggregator can continue to operate even when users fail to submit their measurements. Finally, our protocol can be used as a primitive to build complex algorithms such as principal component analysis, singular-value decomposition, and decision tree classifications by writing the inputs as aggregate-sum queries, like in [12].

1.7. Outline

This thesis is structured as follows. In Chapter 2 we treat the underlying mathematical concepts of our protocol and related works. In Chapter 3 we provide an overview of related work. In Chapter 4 we present the design of our privacy-preserving data aggregation protocol with probabilistic range validation. In Chapter 5 we extend our design to support dynamically adding and removing users, and make our protocol differentially private. In Chapter 6 we prove the correctness and security of our design, and evaluate its detection rate and complexity on a real-world dataset. Finally, in Chapter 7 we discuss our results and provide some closing remarks.

2

Preliminaries

In this chapter we discuss the preliminary knowledge required to understand subsequent chapters. We briefly treat the subjects of secret sharing, commitment schemes, homomorphic encryption, differential privacy, and hypermeshes.

2.1. Secret sharing

Sometimes plainly encrypting a secret is not enough. Instead, the secret should remain inaccessible until multiple parties agree to its access. Naively splitting up the secret is not sufficient because partial data may still reveal some information. For example, giving half a key to each of two parties means that for either party the problem of finding the full key has been reduced for to finding the remaining half of the key, which is significantly easier to brute force.

Secret sharing schemes can be used to securely divide arbitrary secrets. A (t, n)-secret sharing scheme divides a secret S into n shares such that any combination of at least t shares is sufficient to reconstruct S. Furthermore, any combination of fewer than t shares reveals no more information about S than zero shares would. Here, t is referred to as the threshold, and $0 < t \leq n$.

A well-known secret sharing scheme is by Shamir [59]. Shamir's Secret Sharing is based on polynomial interpolation, and works by embedding both the secret S and the secret shares into a polynomial. The scheme relies on the fact that t points are sufficient to reconstruct the unique polynomial of degree $t-1$ that goes through these points, while there are infinitely many polynomials of degree $t-1$ that go through fewer than t points. To instantiate a (t, n)-secret sharing scheme, a trusted dealer with access to the secret S constructs a suitable polynomial of degree $t-1$ by randomly choosing a number a_i for each $i \in [1, t)$, and setting $a_0 = S$. The polynomial is then

$$f(x) = \sum_{i=0}^{t-1} \left(a_i x^i \right). \tag{2.1}$$

The trusted dealer then creates a secret share by choosing a point $(x_i, f(x_i))$ on the polynomial for each $i \in [0, n)$, where each $x_i$can be chosen arbitrarily but must be distinct. To reconstruct the secret S, a coalition of at least t parties pool their shares together, reconstruct the unique polynomial from their shares, and solve for $S = x_0 = f(0)$. Any set of at least t shares suffices, while any set of fewer than t shares reveals no information about S because there is no unique polynomial going through that set of points.

Another commonly used form of secret sharing is additive secret sharing, which requires that $t = n$. The goal of additive secret sharing schemes is to create a share s_i for each $i \in [0, n)$ such that

$$\sum_{i=0}^{n-1} s_i = S. \tag{2.2}$$

A trusted dealer can create appropriate shares by setting s_i to a random number for each $i \in [0, n-1)$ and then setting $s_{n-1} = S - \sum_{i=0}^{n-2} s_i$. Clearly, these shares satisfy Equation 2.2. The scheme ensures confidentiality because any combination of the shares retains at least one random component, except for the sum of all shares which negates them. We discuss a number of strategies to generate additive secret shares of the secret $S = 0$ in Section 3.1.2.

2.2. Commitment schemes

Consider the following conundrum. Alice and Bob want to flip a coin using text messages. They decide that Alice should make a guess and Bob should flip the coin. However, if Alice guesses first, Bob can lie and claim that the coin flipped in his favour; and if Bob flips first, then Alice can always claim to have guessed correctly even if she did not. How can Alice and Bob flip a coin fairly if they do not trust each other?

Alice and Bob can solve this problem with the use of a commitment scheme. With such a scheme, Alice sends a commitment of her guess to Bob without revealing her actual guess, and later reveals her guess in such a way that Bob can verify that her commitment really corresponds to that guess. In general, commitment schemes roughly work as follows:

1. Alice guesses x.
2. Alice sends $y = c(x)$, a commitment to x, to Bob.
3. Bob flips a coin and reveals the result to Alice.
4. Alice reveals x to Bob.
5. Bob verifies that $y = c(x)$.

For this scheme to work as intended, the commitment function c must have two properties:

- **Hiding**: Bob cannot determine x from $c(x)$.
- **Binding**: Alice cannot reveal $x' \neq x$ such that $c(x) = c(x')$.

Depending on the cryptographic primitives used to construct the commitment scheme, the properties can hold computationally or information-theoretically. If a property holds computationally only, then the property may be violated, but it is not feasible to do so for a computationally bounded adversary. On the other hand, if a property holds information-theoretically, no adversary can violate the property regardless of its computational power. Importantly, commitment schemes cannot be both information-theoretically binding and information-theoretically hiding: A scheme that is information-theoretically hiding must have at least two messages with the same commitment, both of which can thus be found by a computationally unbounded adversary, which proves that the scheme not information-theoretically binding.

A commonly used commitment scheme is by Pedersen [51]. This scheme relies on a group G of large prime order q in which the discrete log problem is intractable. Given two public generators g, h of G, a commitment on a message m is constructed as

$$g^m h^r \tag{2.3}$$

for random $r \in \mathbb{Z}_q$. Clearly, the scheme is information-theoretically hiding because the component h^r is random, and therefore perfectly hides the component g^m. During the reveal, Alice sends the tuple (m, r) to Bob, who re-calculates Equation 2.3 to verify the commitment. As such, the scheme is only computationally binding: Alice would have to solve the discrete log problem to find another tuple $(m', r') \neq (m, r)$ such that $g^m h^r = g^{m'} h^{r'}$, which is feasible if Alice has unlimited computational power.

2.3. Homomorphic encryption

Whereas normal encryption requires access to plaintexts in order to perform meaningful operations on them, homomorphic encryption allows one to compute with ciphertexts. This property can be used to outsource expensive operations on privacy-sensitive data to a third party. Early homomorphic encryption schemes allowed for only a single operation such as addition or multiplication, but in 2009, Gentry [32] proposed the first homomorphic encryption scheme that supports evaluation of arbitrary circuits on encrypted data.

To understand homomorphic encryption, consider a function f takes an element from the group $(P, +)$ and returns an element from the group $(C, *)$; these groups may or may not be the same. The $+$ and $*$ can represent any particular operators, not just addition and multiplication. The function f is a homomorphism if it satisfies for any two inputs $x_1, x_2 \in P$ the equation

$$f(x_1) * f(x_2) = f(x_1 + x_2). \tag{2.4}$$

An encryption scheme that uses a function like f is a homomorphic encryption scheme, and a third party can safely be given $f(x_1)$ and $f(x_2)$ to calculate $f(x_1 + x_2)$ without being able to find x_1 or x_2. If the third party

is malicious, additional measures should be taken to verify the correctness of the result when it is returned, but we do not consider that issue here.

An example of a simple (but insecure) homomorphic encryption scheme is unpadded RSA. Given RSA public key (n, e), we have for messages $m_1, m_2 \in \mathbb{Z}_n$ that

$$enc(m_1) * enc(m_2) = m_1^e * m_2^e = (m_1 * m_2)^e = enc(m_1 * m_2) \mod n. \tag{2.5}$$

A homomorphic encryption scheme of particular interest is the Paillier cryptosystem [50], which is additively homomorphic. The scheme is used in several protocols described in Chapter 3. The asymmetric scheme relies on a public key $pk = (n, g)$ such that $n = pq$ for primes p, q with $\gcd\big(pq, (p-1)(q-1)\big) = 1$; and $g \in \mathbb{Z}^*_{n^2}$ randomly. The private key is simply $sk = \lambda = lcm(p-1, q-1)$. Then, encryption of a message $m \in \mathbb{Z}_n$ is calculated as

$$enc_{pk}(m, r) = g^m r^n \mod n^2 \tag{2.6}$$

for some random $r \in \mathbb{Z}^*_n$. This encryption scheme is indeed additively homomorphic since

$$enc_{pk}(m_1, r_1) * enc_{pk}(m_2, r_2) = \left(g^{m_1} r_1^n\right) * \left(g^{m_2} r_2^n\right) = g^{m_1+m_2}(r_1 r_2)^n = enc_{pk}(m_1 + m_2, r_1 r_2) \mod n^2. \tag{2.7}$$

Consequently, it is also possible to find the product of a plaintext m with a constant e under encryption as

$$enc_{pk}(m)^e = enc_{pk}(m * e). \tag{2.8}$$

Decryption is a much more involved process and is explained in [50].

The security of the Paillier cryptosystem relies on the decisional composite residuosity assumption, which states that it is not feasible to determine whether $z = y^n \mod n^2$ for unknown y, given n and z. Under this assumption, the Paillier cryptosystem provides IND-CPA security and IND-CCA1 security [8], but not IND-CCA2 security because of its malleability. Note that it is generally not assumed that the discrete log problem holds in the Paillier cryptosystem.

Damgård and Jurik [20] propose a generalisation of the Paillier cryptosystem that can function as a threshold cryptosystem. Similar to secret sharing, (t, n)-threshold cryptography allows for any combination of at least t out of n parties to decrypt a plaintext in a cooperative protocol. In the cryptosystem by Damgård and Jurik, the private key is divided into n parts. To decrypt a ciphertext, each user performs a partial decryption of the ciphertext using the partial key. Any combination of at least t partial decryptions can be combined to trivially reconstruct the original plaintext.

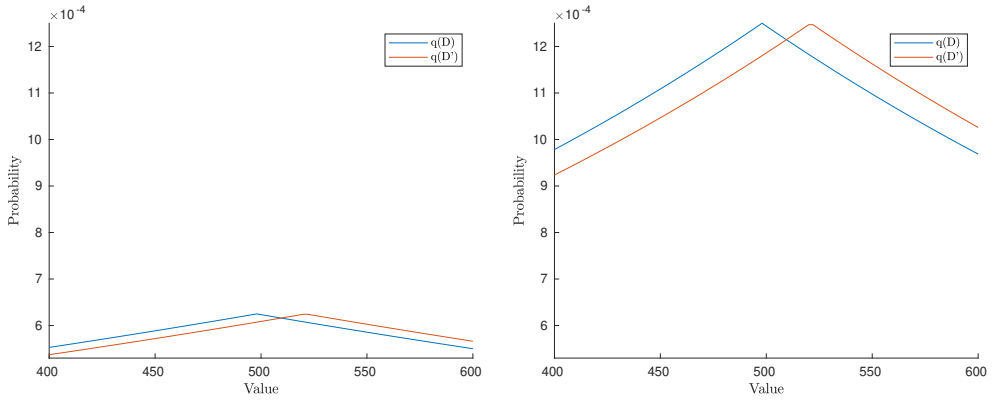
2.4. Differential privacy

Consider a database with user records that is publicly queriable. Eve queries the database and sees that the sum of all ages is 498. The next day, Eve knows that a single user has been added to the database, re-queries the database, and sees that the sum of all ages is now 521. Eve now knows that the newly added user is 23 years old. Attacks like these allow adversaries to infer sensitive information about users based on the difference of aggregates and must be taken into account when modelling privacy.

Differential privacy was first formalised by Dwork et al. [25] in 2006 as a formal privacy model to ensure that participating in a protocol in itself does not harm a user's privacy; a differentially private database disallows observers from determining whether a particular user in the database. Differential privacy can be achieved by perturbing statistical results before publishing them so that the exact value is not visible. Formally speaking, given two databases D, D' that differ in exactly one element, a query function q, and the image Q of q, the function q is ε-differentially private if

$$\forall t \in Q : \Pr[q(D) = t] \leq \exp(\varepsilon) \cdot \Pr[q(D') = t]. \tag{2.9}$$

Intuitively, this means that the probability distribution of the query function should be not change by more than a factor of ε after changing one element in the database. To achieve ε-differential privacy, the perturbation should be proportional to the influence a single database element has on the query's output. This influence is expressed as the sensitivity S of the query function q, which is defined as the largest possible difference between $q(D)$ and $q(D')$ for any databases D, D' that differ in exactly one element. We illustrate the effect of different values of ε in Figure 2.1. Decreasing the privacy parameter ε ensures that the values $q(D)$ and $q(D')$ are statistically harder to distinguish, but this also makes the query results less useful because they are less accurate. Naturally, an adversary can approximate the real value if they can reverse-engineer the



(a) 0.05-differentially private query function q

(b) 0.1-differentially private query function q

Figure 2.1: Effect of Laplace-based differential privacy on an aggregation function q with sensitivity $S = 40$ before ($q(D)$, unperturbed sum is 498) and after ($q(D')$, unperturbed sum is 521) a new user has been added, showing that a lower ε provides better privacy but lower utility

probability density function of the perturbation function by repeating the same query many times. Therefore, additional protective mechanisms are required to protect public databases, but we do not discuss those here.

As a relaxation of regular ε-differential privacy, Dwork et al. [24] propose (ε, δ)-differential privacy. Given variables D, D', q, Q as before, q is (ε, δ)-differentially private for some $0 \leq \delta < 1$ if

$$\forall t \in Q : \Pr[q(D) = t] \leq \exp(\varepsilon) \cdot \Pr[q(D') = t] + \delta. \tag{2.10}$$

This relaxation provides leniency towards large changes in the probability of a particular value as a result of a change in the database. This generalised model is useful when regular ε-differential privacy cannot be proven for a system only because there are some unlikely transformations of the database that result in larger changes to the probability density function.

Instead of using a database, a different architecture can be used in which an aggregator interacts with many users to publish statistics on the privacy-sensitive data of these users. In the case of a malicious aggregator, users cannot trust the aggregator to add differential noise before looking at the privacy-sensitive data. A naive solution would be to apply ε-differentially private noise to each datum before submitting it to the aggregator, but the cumulative noise would make the published statistics virtually useless. An alternative solution is ε-local differential privacy, in which individual users perturb their data such that the individual perturbation is not enough to protect an individual response, but the cumulative perturbation is sufficient for the aggregate. This scheme can then be combined with a secret sharing scheme to ensure that the aggregator cannot see intermediate aggregates of the data. We discuss some privacy-preserving data aggregation protocols that use differential privacy in Section 3.1.1.

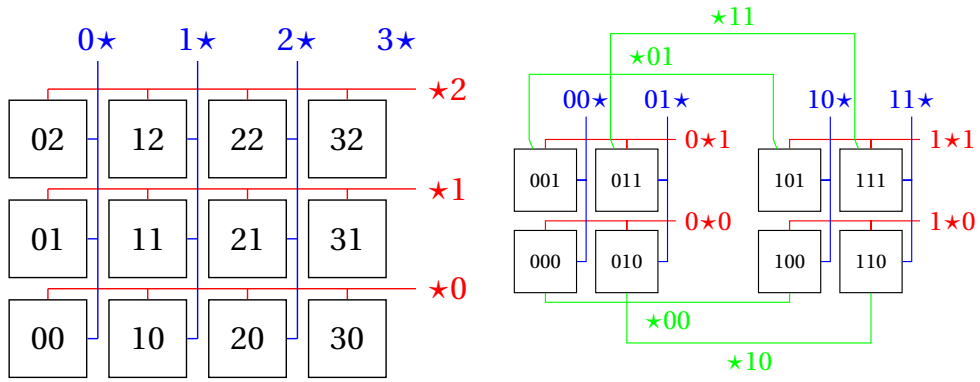
2.5. Hypermeshes

Wittie [65] proposes "spanning-bus hypercubes" as a network layout for parallel communication. Later, this structure would become known as the hypermesh, as in [63, 64]. A hypermesh can be modelled as an orthogonal hypergraph with ℓ dimensions. Each dimension has a corresponding radix (or base) b_i, and all bases together form the vector $b = (b_{\ell-1}, b_{\ell-2}, \ldots, b_0)$. If all bases are equal, the hypermesh is regular, in which case the hypermesh can be described as a b^ℓ-hypermesh for some integer $b \geq 2$. Otherwise, the hypermesh is irregular, and is described as a b-hypermesh for the vector of bases b. Unless noted otherwise, we assume that hypermeshes are irregular. Several examples of hypermeshes are shown in Figure 2.2.

A b-hypermesh contains $n = \prod_{i=0}^{\ell-1} b_i$ nodes, where each node can be described by an ℓ-digit identifier string $d_{\ell-1} d_{\ell-2} \ldots d_0$ such that $d_i \in [0, b_i)$ for all $0 \leq i < \ell$. Equivalently, each identifier can be considered the node's coordinates in an ℓ-dimensional Euclidean space. Alternatively, an identifier string can be interpreted as a mixed-radix representation, and can be converted to a decimal number in the range $[0, n)$ as

$$f(d_{\ell-1} d_{\ell-2} \ldots d_0) = f(d_{\ell-2} d_{\ell-3} \ldots d_0) b_{\ell-1} + d_{\ell-1}, \tag{2.11}$$

where $f() = 0$.



(a) A (4,3)-hypermesh, with 12 nodes and 7 hyperedges

(b) A (2,2,2)-hypermesh (which is the graph of a regular cube), with 8 nodes and 12 hyperedges

Figure 2.2: Examples of hypermeshes

The edges in a b-hypermesh are hyperedges, which are edges with any number of endpoints, including none at all or just one. In particular, there is a hyperedge for each set of nodes that differ in identifier by exactly one digit. As such, a hyperedge can be given an identifier that matches the digits of the nodes in it, with the variable digit replaced by the wildcard symbol $\star$. If the wildcard symbol is in the ith position in the hyperedge's identifier string, the hyperedge is said to be aligned along the ith dimension of the hypermesh. A hyperedge along the ith dimension has b_i endpoints. Hyperedge identifiers can be written compactly as a tuple (p, v), where $p \in [0, \ell)$ is the index of the wildcard symbol and $v \in [0, N)$ is the numerical value of the hyperedge identifier after replacing the wildcard symbol with a 0 and putting it through Equation 2.11.

G describes the set of all hyperedges, and U describes the set of all nodes. Along the ith dimension, there are $\frac{N}{b_i}$ hyperedges, which together contain all nodes. There are exactly ℓ sets of disjoint hyperedges that contain all nodes; the ith set contains all hyperedges aligned along the ith dimension. There is a total of $\sum_{i=0}^{\ell-1} \frac{N}{b_i}$ hyperedges. The set U_j describes the nodes connected by hyperedge j, the string identifiers of which can be found by replacing the wildcard symbol in the hyperedge identifier with the respective values $[0, b_i)$. Conversely, the set G_i describes the hyperedges to which node i is connected, the string identifiers of which are can be found by taking the string identifier of node i and replacing the respective symbols with the wildcard symbol.

Two nodes that are connected to the same hyperedge are neighbours, i.e. two nodes i, k are neighbours if $i, k \in U_j$ for some $j \in G$. The set N_i describes the $\ell(b-1)$ neighbours of node i, and can be found by taking $\bigcap_{j \in G_i} U_j$. A pair of nodes $i \neq k$ can be related in a number of ways:

1. If the identifiers differ by exactly one digit, $G_i \cap G_k$ contains exactly one hyperedge, i.e. the nodes are neighbours in exactly one hyperedge.

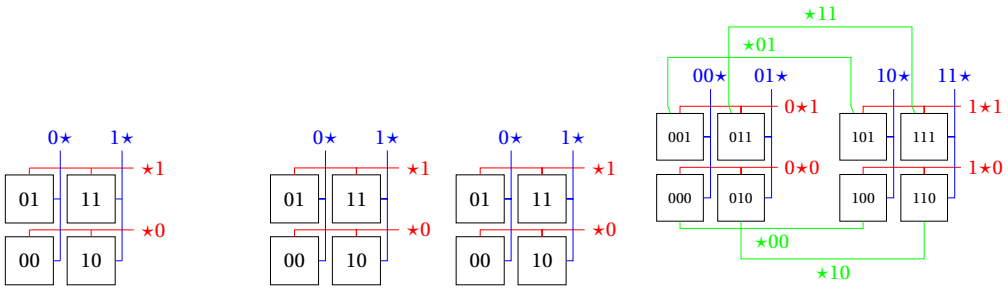
2. If the identifiers differ by exactly two digits, $G_i \cap G_k$ is empty and $N_i \cap N_k$ contains exactly two nodes, i.e. the nodes have two neighbours in common. These neighbours also differ in two digits from each other.

3. Otherwise, $G_i \cap G_k$ and $N_i \cap N_k$ are empty, so i, k are independent.

Two nodes cannot be connected by more than one hyperedge, and two hyperedges cannot overlap in more than one node.

An ℓ-dimensional hypermesh contains b_i disjoint $(\ell-1)$-dimensional hypermeshes along its ith dimension, for a total of $\sum_{i=0}^{\ell-1} b_i$ sub-hypermeshes in a b-hypermesh. The (i, j)th sub-hypermesh is equivalent to a b'-hypermesh where b' is the same as b but with the ith element removed. This hypermesh contains the nodes that match the identifier $\star^i j \star^{\ell-i-1}$, where $\star^x$ denotes a sequence of x wildcard symbols. Intuitively, the (i, j)th sub-hypermesh is a "slice" of the b-hypermesh along the ith dimension at the jth point, highlighting that the b-hypermesh is just a sequence of b_i connected b'-hypermeshes aligned along some dimension. In Figure 2.2b, the $(2,1)$th sub-hypermesh is $\star\star1$ and contains nodes $001, 011, 101, 111$.

Note that hypermeshes are identical to Hamming graphs, except that hypermeshes use hyperedges instead of regular edges. Similar to Hamming graphs, b-hypermeshes can be constructed recursively from the sub-hypermeshes contained within it, as we show in Figure 2.3. To recursively construct a b-hypermesh,



(a) The initial (2,2)-hypermesh from which larger hypermeshes can be constructed

(b) Two copies of the original (2,2)-hypermesh

(c) A (2,2,2)-hypermesh constructed by connecting the two copies of the (2,2)-hypermesh and relabeling their nodes

Figure 2.3: Recursive construction of a (2,2,2)-hypermesh from a (2,2)-hypermesh

first consider a (b_0)-hypermesh, which contains b_0 nodes connected by a single hyperedge. Then, given a $(b_{i-1}, b_{i-2}, \ldots, b_0)$-hypermesh H, the $(b_i, b_{i-1}, \ldots b_0)$-hypermesh H' can be constructed by taking b_i copies of H, identifying each copy with a unique number in $[0, b_i)$, connecting all nodes that have the same identifier with a new hyperedge, and prefixing each node identifier with the identifier of the copy of H they are in. Consider, for example, the (2,2,2)-hypermesh in Figure 2.2b, which can be constructed by taking two (2,2)-hypermeshes, and then adding a hyperedge between every pair of nodes that has the same last two digits.

3

Related work

In the past decade, a large volume of privacy-preserving data aggregation (PDA) protocols have been proposed. This chapter details a selection of these works in approximate chronological order. First, in Section 3.1, we investigate the fundamental PDA protocols, separated into three branches. Then, in Section 3.2, we look at protocols that combine these fundamental ideas into hybrid approaches. Finally, in Section 3.3, we consider existing solutions for range validation in PDA protocols.

3.1. Privacy-preserving data aggregation protocols

The need for privacy-preserving data aggregation arose somewhere at the start of the previous decade. Researchers considered a setting in which an aggregator wants to process the private data of several users without violating user privacy. The aggregator regularly collects data from the users in rounds, either based on a regular interval or after some trigger. Several authors independently proposed solutions to this issue, which can be categorised into three types: differential privacy, secret sharing, and anonymisation. These solutions arose from slightly different requirements and therefore have different advantages and disadvantages.

3.1.1. Protocols using differential privacy

Differentially private techniques as described in Section 2.4 rely on the aggregator to perturb the aggregate's output to protect individual privacy. However, this makes the technique unsuitable in cases where private data must be protected from the aggregator itself. Dwork et al. [24] presented early efforts at remedying this shortcoming, but Rastogi and Nath [56] argue that these techniques do not scale well with many users, nor do they account for the high correlation that is present in time-series data; both of which are significant shortcomings in the data aggregation setting.

As a solution, Rastogi and Nath propose the PASTE framework to address these issues using a custom distributed noise generation algorithm. The framework uses the observation that, given i.i.d. normal random variables Y_i with mean 0 and variance λ for $i \in [0,3)$, the variable $Y_1^2 + Y_2^2 - Y_3^2 - Y_4^2$ is a Laplacian random variable with mean 0 and variance $2\lambda^2$. This equation allows a construction in which each user adds a bit of noise so that the total noise is ε-differentially private. Because the individual perturbations are not enough to provide privacy to a single user, the PASTE framework uses threshold Paillier encryption to prevent the aggregator from decrypting partial aggregates. On a high level, the protocol works as follows. At the start, each of the N users is given the encryption key and one share of the (N,N)-secret shared decryption key. To submit the private value m_i to the aggregator, each user i generates a random value r_i and sends the encryption of $m_i + r_i$ to the aggregator. User i then generates a normal random variable $y_{i,j}$ with mean 0 and variance λ for each $j \in [0,3)$, and consequently engages in an interactive protocol so the aggregator finds the encryption of $Y_i^2 = \left(\sum_{i=0}^{N-1} y_{i,j}\right)^2$ for each $j \in [0,3)$. The aggregator takes all these components and homomorphically sums them together to get the encryption of

$$\sum_{i=0}^{N-1} (m_i + r_i) + Y_1^2 + Y_2^2 - Y_3^2 - Y_4^2. \tag{3.1}$$

To decrypt this value and remove the blindings r_i, the aggregator sends the ciphertext to each user, each of whom partially decrypts the ciphertext using their share of the private key and their secret blinding, and re-

turns the result to the aggregator. The aggregator combines these partial decryptions to obtain the plaintext, which corresponds to the differentially private sum of private data. The protocol is secure because the secret blindings ensure that decrypting intermediate values is not useful, and the protocol has additional safeguards that invalidate the ciphertext if the aggregator uses a different linear combination of random variables Y_i. A drawback of this approach is that it requires interactivity between the aggregator and the users to generate the noise and to decrypt the aggregate, which results in additional bandwidth and computation overheads.

Shi et al. [60] propose another protocol that does not need interactivity to decrypt the aggregate. Instead, the aggregator directly decrypts the obtained aggregate ciphertext. Additionally, the authors formally prove their security assumptions. The protocol provides (ε,δ)-differential privacy given a lower bound γ on the ratio of honest users. At the start of the protocol, a trusted dealer generates 0-sum additive secret shares for each user. Then, each user i samples noise r_i using a symmetric geometric distribution $\text{Geom}(\alpha)$ as

$$\begin{cases} \text{Geom}\left(\exp\left(\frac{\varepsilon}{S}\right)\right), & \text{with probability } \frac{1}{\gamma N}\log\frac{1}{\delta} \\ 0, & \text{otherwise,} \end{cases} \tag{3.2}$$

given the query function's sensitivity S and the number of users N. Then, users add r_i to their private datum before homomorphically encrypting it using their secret share as the key. Users send the encrypted noisy measurements to the aggregator, who adds them together such that the keys cancel out and the aggregator is left with a value of the form g^x where x is the noisy sum of private values. The aggregator can decrypt this term using a discrete logarithm operation. The secret shares ensure that the aggregator cannot decrypt any other linear combination of ciphertexts. However, computing the discrete logarithm is an expensive operation and is only feasible when the plaintext space is small. Chan et al. [17] describe an extension of the protocol that provides robustness but retains the expensive decryption and the need for a trusted third party to generate the secret shares during the setup.

3.1.2. Protocols using secret sharing

Garcia and Jacobs A different solution to create a PDA protocol is by Garcia and Jacobs [31], who respond to the then-recent concerns voiced by the Dutch Senate regarding the privacy of smart metering data. Garcia and Jacobs argue that it is not strictly necessary for operators and utility providers to know each smart meter's consumption data when they are only interested in aggregates of these data. In an attempt to direct research at moving "power to the meter", Garcia and Jacobs propose a PDA protocol that does not rely on centralised trust. At the start of the protocol, the aggregator disseminates users' public keys amongst the N users. Then, in each round, each user i creates additive secret shares $x_{i\to j}$ of their private datum x_i for each user j such that

$$x_i = \sum_{j=0}^{N-1} x_{i\to j}. \tag{3.3}$$

Each user i homomorphically encrypts the secret share $x_{i\to j}$ with the public key of user j, and sends these data to the aggregator. Upon receiving these data, the aggregator forwards the secret shares to the right users based on some metadata that user i provided with the ciphertext. Each user i then adds the homomorphic ciphertexts together and decrypts them to obtain the sum of the shares sent to that user. This sum is then sent to the aggregator without encryption, who sums all these data together to obtain

$$\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} x_{j\to i} = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} x_{i\to j} = \sum_{i=0}^{N-1} x_i. \tag{3.4}$$

By distributing private data in the form of secret shares, no user learns anyone else's private data. Furthermore, the encryption prevents the aggregator from seeing secret shares while they are being distributed. However, the protocol has several drawbacks, such as its lack of robustness to disappearing users, high communication complexity, and high computation complexity for users. Regardless, the protocol kickstarts a new area of research focused on creating improved PDA protocols, initially focused on smart meters but later generalised to all data aggregation.

Kursawe Kursawe [41] proposes an alternative method of creating 0-sum secret shares, amongst others. The protocol works by randomly selecting p leaders from the set of N users. After the public keys of all users have been disseminated in some way, each non-leader i generates one random value $r_{i\to j}$ for each leader j,

and sends it to that leader encrypted with that leader's public key. Each leader j then generates one random value r_j such that

$$r_j + \sum_{i=0}^{N-1} r_{i\to j} = 0. \tag{3.5}$$

When sending a measurement to the aggregator, each non-leader i simply blinds the measurement with

$$\sum_{j=0}^{p-1} r_{i\to j}, \tag{3.6}$$

while each leader j blinds their value using only r_j. When the aggregator sums the blinded values, the blindings cancel out and only the sum of inputs remains.

While Kursawe's design removes the high communication complexity present in the protocol by Garcia and Jacobs, it no longer provides forwards secrecy. Additionally, the difference in two blinded values sent in subsequent rounds is exactly the difference between the unblinded values, which may leak privacy-sensitive information. Regenerating the blindings regularly resolves this issue, but reintroduces the high communication complexity.

Kursawe et al. Kursawe et al. [42] provide several protocols, the first of which is identical to the protocol in [41]. The other three protocols focus on comparison of private values rather than summation, but also use a new method to obtain 0-sum secret shares, which works as follows. At the start of the protocol, each user has a Diffie-Hellman key pair, and shares the public key with all other users. Each user i then performs key agreement using their private key and the public key of another user j to obtain a key $k_{i,j}$ such that $k_{i,j} = k_{j,i}$. User i then sums the key agreements together as

$$b_i = \sum_{j\neq i} \left((-1)^{j<i} k_{i,j}\right), \tag{3.7}$$

where $j < i$ is 1 if $j < i$, and 0 otherwise. As such, $k_{i,j} + k_{j,i} = 0$ for any pair of users i, j. User i then uses b_i to blind a private datum before sending it to the aggregator. When the aggregator adds all blinded values together, the sum contains for every pair of users (i, j) both $k_{i,j}$ and $-k_{j,i}$, negating the blindings and leaving only the sum of private data. One major drawback of this approach, like other additive secret sharing schemes, is that it is not robust to user dropouts.

Ács and Castelluccia [5] independently propose the same method in a protocol combining secret sharing with differential privacy.

Erkin and Tsudik Erkin and Tsudik [29] propose a protocol that allows aggregating time-series data both spatially (i.e. sum of all users in one round) and temporally (i.e. sum of one user in multiple rounds). Furthermore, the role of aggregator can be performed by any user and there can be any number of aggregators. In the following description of the protocol, we assume that there is a single aggregator, who is not a user.

At the start of the protocol, the aggregator creates a Paillier key pair and sends the public key to all users. Like other protocols of this type, the N users then proceed to generate 0-sum additive secret shares. To this end, each user i sends a random value $r_{i\to k}$ to each other user k, after which the blinding used by user i is

$$b_i = \sum_{k=0}^{N-1} \left(r_{i\to k} - r_{k\to i}\right). \tag{3.8}$$

The sum of all b_i is then 0 because for every pair of users i, k the secret share b_i contains $r_{i\to k}$ while b_k contains $-r_{i\to k}$. If the random values are replaced with seeds for a random number generator, blindings can be re-generated in each round without any communication overhead.

To encrypt the value m_i, user i performs a modified Paillier homomorphic encryption as

$$c_i = g^{m_i} h^{n+b_i} \mod n^2, \tag{3.9}$$

where (g, n) is the public Paillier key and h is a random value in $\mathbb{Z}_n^*$ that depends on the current round number. This encryption retains the security properties of normal Paillier encryption, and because the exponent is not n as it would be with unmodified encryption, the ciphertext cannot be decrypted, even with the private Paillier key. By choosing a value h that depends on the round number, the aggregator cannot reuse old ciphertexts in its calculations.

To aggregate spatially, the aggregator simply multiplies the ciphertexts together and decrypts the result to obtain

$$\prod_{i=0}^{N-1} c_i = \prod_{i=0}^{N-1} \left(g^{m_i} h^{n+b_i} \right) = g^{\sum_{i=0}^{N-1} m_i} h^{\sum_{i=0}^{N-1} (n+b_i)} = g^{\sum_{i=0}^{N-1} m_i} \left(h^N \right)^n, \tag{3.10}$$

which the aggregator can decrypt to obtain the sum of all m_i. Aggregating temporally is more involved because h is different in each round, and because the sum of b_i over multiple rounds is not necessarily 0. Therefore, the aggregator asks the user to send a value that cancels out the blindings from the ciphertexts when multiplied with the product of the user's ciphertexts, making the result a valid ciphertext.

While the protocol has low communication complexity, it requires intervention of a trusted third party when smart meters fail to respond. Additionally, like several other protocols in this category, it relies on expensive 2048-bit homomorphic encryption to achieve security in practice.

3.1.3. Protocols using anonymisation

A third category of PDA protocols is proposed by Efthymiou and Kalogridis [26], also in relation to privacy concerns about smart metering data. Rather than relying on secret sharing, homomorphic encryption, or differential privacy, this branch of protocols is based on the idea that the aggregator does not need to know who sent the data, as long as the data has been authenticated by a reliable entity. Efthymiou and Kalogridis propose the use of a trusted third party which functions as a proxy. To this end, users send two types of data: low-frequency data, used for billing purposes, and high-frequency data, used for real-time data aggregation. This can be considered analogous to the spatial and temporal aggregation described in the later work by Erkin and Tsudik [29]. The user submits high-frequency data to the proxy, who forwards the data to the aggregator only after anonymously authenticating the user with the aggregator. On the other hand, the user sends low-frequency data directly to the aggregator. As long as the aggregator and proxy do not collude, the aggregator does not learn high-frequency data about users, but is still able to perform real-time analysis and billing. However, the use of a trusted third party is not always feasible, and the protocol cannot guarantee that low-frequency data reported by a user corresponds to that user's high-frequency data.

Molina-Markham et al. [46] start with an analysis of the privacy sensitivity of smart metering data. The authors show how to infer the number of people in a household at a given time or the type of breakfast that was eaten in the morning, using only smart metering data. Concerned by these results, Molina-Markham et al. propose a new anonymisation-based protocol that improves upon the work of Efthymiou and Kalogridis by including an interactive zero-knowledge proof system that allows the aggregator to verify that the low-frequency report contains the sum of the high-frequency reports. The interactive proof can be repeated multiple times to increase the aggregator's confidence that the values are correct. However, the interactivity of the proof incurs a higher bandwidth consumption, and as the number of proof rounds increases so does the performance impact on the user.

An independent proposal by Applebaum et al. [7] is similar to the previous two works in that it uses a proxy for anonymisation. However, the problem statement is slightly different, instead focusing on aggregating databases at scale without considering time-series data. In this protocol, users send encrypted data to the proxy in each round, which shuffles these data before forwarding them to the aggregator. The aggregator can decrypt and aggregate, but does not know the origin of any datum. Meanwhile, the proxy does not have the decryption key, so it cannot read any data. The protocol uses various additional techniques that allow the aggregator to simultaneously aggregate values according to key-value pairs, but we do not discuss those here. A disadvantage of this protocol is that it requires that the proxy and aggregator do not collude, which is not always a feasible assumption.

3.2. Hybrid protocols

After the fundamental privacy concepts of privacy-preserving data aggregation had been established, new works appeared with hybrid approaches for modified problem statements. We highlight a few of these works that relate to our protocol and investigate their underlying ideas.

3.2.1. A protocol for dynamic groups of users

Erkin [28] considers the issue of robustness towards disappearing users without relying on additional protocols or third parties. His proposal is based in part on his previous work in [29], but works by dividing users into disjoint groups. His protocol description assumes groups of only two users, without loss of generality.

Instead of calculating the total sum of all private inputs directly, the aggregator first calculates the sum for each group. To this end, all group aggregates are encoded into a single value using the Chinese Remainder Theorem (CRT) so that the aggregator needs to decrypt only once. At the start of the protocol, the aggregator assigns a unique prime number p_j to each group j to be used as that group's factor in the CRT encoding. In each group j, users generate 0-sum additive secret shares $b_{i,j}$ for each user i in that group, for example using a key exchange protocol as proposed by Kursawe et al. [42]. User i in group j then encodes a message m_i as

$$c_i = m_i \frac{P}{p_j} \left(\left(\frac{P}{p_j} \right)^{-1} \bmod p_j \right) \mod P, \tag{3.11}$$

where P is the product of all p_j, and encrypts c_i with the blinding $b_{i,j}$ using the modified Paillier encryption scheme by Erkin and Tsudik [29]. The aggregator multiplies all ciphertexts together and decrypts the result to get T. To obtain the sum of values of users in group j, the aggregator computes $T \bmod p_j$; to get the sum of all users the aggregator simply takes the sum of all group sums.

Privacy is guaranteed in a manner similar to [29], and robustness is implied by the aggregator's ability to exclude groups of which not all users sent a (valid) ciphertext. Additionally, the protocol allows users to dynamically join the protocol because the user can simply be added to an existing group; only users in this group have to renegotiate the 0-sum additive secret shares. Similarly, users can dynamically leave the protocol. The protocol does not provide any protection against malicious users sending measurements using the wrong group's prime number, however.

An extension of this protocol is presented in [39], which uses Shamir secret sharing instead of additive secret sharing to achieve increased robustness. However, this extension requires that the aggregator is not a user and relies on a random user in each group to ensure the privacy of users in that group.

3.2.2. A protocol with high robustness

Some real-world data aggregation systems need massive scalability and have to work even when many users disappear. Additionally, these systems aim to avoid direct communication between users because it is difficult or sometimes impossible to establish. Bonawitz et al. [13] propose a solution addressing these issues. The protocol uses multiple instances of secret sharing to achieve high robustness; with n users, the protocol uses (t, n)-secret sharing for some threshold t.

During the setup, each user i generates two asymmetric key pairs: $\left(c_i^{PK}, c_i^{SK}\right)$, used for secure communication between users, and $\left(s_i^{PK}, s_i^{SK}\right)$, used for key agreement-based 0-sum secret sharing. Each user sends both public keys to the aggregator, who stores them and forwards them to all other users. Once all keys have arrived, each user i chooses a seed b_i, used for generating random numbers. User i then creates (t, n)-secret shares of both the seed b_i and of s_i^{SK}. For each other user k, user i selects the kth secret share of b_i and the kth secret share of s_i^{SK}, and encrypts them using c_k^{PK} before forwarding these to user k through the aggregator. Once the secret shares of other users have arrived, user i generates a 0-sum secret-shared blinding p_i based on the keys s_*^{PK} of all other users using the key agreement-based protocol by Kursawe et al. [42].

With the setup complete, users can start sending private measurements to the aggregator. To send the measurement m_i, user i takes p_i and a random number generated using b_i, and sends $m_i + p_i + b_i$. When the aggregator sums all these blinded values together, they obtain

$$\sum_{i=0}^{N-1} \left(m_i + b_i + p_i \right). \tag{3.12}$$

If all users sent their values, then the p_i cancel out, but the b_i remain. For every user i that did not send a value, b_i is not present, but p_i is not there to cancel out other users' key agreements either; fortunately, these key agreements can be reconstructed by the aggregator if they know s_i^{SK}. Either way, the aggregator needs to subtract one value for each user that shared their public keys at the start. To do this, the aggregator contacts each user i and asks a secret share of each user k: If user k is present in Equation 3.12 the aggregator asks for the secret share of b_k, otherwise the aggregator asks for a secret share of s_k^{SK}. If the aggregator obtains at least t secret shares for each user, the aggregator can reconstruct the necessary values to negate all blindings to obtain the aggregate.

The protocol has a few optional extensions. One extension provides differential privacy for user data, and another ensures integrity and privacy when the aggregator is malicious. However, the latter extension requires users to sign and verify large amounts of data, growing linearly with the amount of users.

Even without extensions, the design of the protocol guarantees that the aggregator cannot uncover the private datum of a single user as long as there are fewer than $\lceil \frac{2N}{3} \rceil$ users colluding with the aggregator. While the protocol provides strong robustness guarantees, the long interactive sessions between the user and the aggregator incur high bandwidth and computation requirements on users, also growing linearly with the number of users.

3.3. Protocols with range validation

The majority of PDA protocols described above assume that users send well-formed inputs. A variety of modifications and protocols have been suggested that help the aggregator ensure private inputs are within a predefined range.

Jiang et al. [37] propose a protocol based on negative survey. The aggregator divides the valid range into several classes and lets users submit one of these classes in each round. As such, it is not possible for users to submit a value that is outside the valid range. Instead of submitting the right class corresponding to their measurement, however, each user randomly sends one of the incorrect classes—hence the name negative survey. The aggregator then derives an estimation of the true distribution of classes using certain statistical methods. However, if the true class does not change over time, the aggregator could derive a user's true class by looking at which class has not been submitted. Therefore, whenever the true class changes, users should choose a new subset of incorrect classes and continue to select a random class from this subset until the true classes changes again. One drawback of the negative survey approach is that accurate results require a large number of classes, but this in turn requires more inputs if the probability distribution is to remain accurate.

Kursawe [41] proposes a scheme that allows an aggregator to gain confidence that all private values are valid by checking that the sum of inputs approximately equals the actual aggregate. However, this protocol cannot identify *which* user sent the invalid value and requires the aggregator to know an approximation of the aggregate beforehand, which is not always feasible.

Sun et al. [62] present APED, a PDA protocol that can detect defective smart meters using a method similar to ours. In APED, a trusted third party does the setup. First, the third party chooses a group of large prime order p and finds a generator g. Then, the third party generates for each user i the random key k_i, divides all users into w random sets of disjoint pairs such that each user is in w pairs at once, and sends for each pair of users i, j the key $k_{i,j} = -(k_i + k_j)$ to the aggregator. Each round, each user i sends an encryption of their measurement m_i as $c_i = g^{m_i} h^{k_i}$, given a round-dependent pseudo-random value h. After receiving the ciphertexts, the aggregator chooses one of the w pairing sets, and decrypts the product of each pair of users i, j by calculating

$$c_i c_j h^{k_{i,j}} = \left(g^{m_i} h^{k_i}\right)\left(g^{m_j} h^{k_j}\right) h^{-k_i - k_j} = g^{m_i + m_j} h^0 \tag{3.13}$$

and then taking the discrete logarithm in g. If decryption fails, at least one of the two users must be defective, but the aggregator does not know which one yet. As such, the aggregator continues to run the protocol but uses a different set of pairings in the next round. Eventually, the aggregator can infer which users are defective from the overlap of invalid pairs. An extension of the protocol, DG-APED [61], uses groups of arbitrary size. Both protocols have several drawbacks. Firstly, they rely on a trusted third party to create groups and generate key material. Secondly, because the protocol is tailored to defective users, its detection algorithm is unsuitable for malicious users that do not send an invalid value in each round. Finally, decryption does not scale with larger inputs because of the discrete logarithm operation.

Ahadipour et al. [6] propose a protocol that reduces the amount of private data the aggregator has access to. Users are divided into disjoint groups, and the aggregator obtains the sum of each group's values in addition to a random subset of users' private values. The aggregator directly looks at these private values to determine which users sent invalid values. While this reduces the privacy impact on the users, giving the aggregator access to even a single private value is not tolerable for sensitive data.

Yang and Li [67] propose a protocol that can identify out-of-range values using re-encryption. The aggregator divides users into disjoint groups, and when it finds that the aggregate of a group is out of range, it re-encrypts and shuffles the values of the violating group and sends them to a random user in that group. The random user decrypts the values and reports back which values are out of range. The main drawback of this scheme is that it is especially vulnerable to collusions, as a single collusion between the aggregator and the random user suffices to reveal all private values of an entire group to the aggregator.

Finally, there is a multitude of proposals that assume that users are honest-but-curious, but note that zero-knowledge proofs could be used to perform input validation [29, 42, 60]. With zero-knowledge proofs, users can mathematically prove that their value is within a particular range without having to reveal their

value. However, generic zero-knowledge proofs such as SNARKs require a trusted setup, which is often not a realistic assumption. Its cousin, the STARK [10], resolves this problem, but this comes at the cost of increased communication complexity. Corrigan-Gibbs and Boneh [19] introduce SNIPs to allow users to prove that input is valid according to an arbitrary circuit, but this solution requires a multitude of cooperating servers, of which all must be honest to guarantee correctness and at least one must be honest to guarantee privacy for the user; furthermore, client-side communication costs grow linearly with the complexity of the validation circuit. Range proofs are a specific form of zero-knowledge proof specific to range checking. Even though range proofs such as Bulletproofs by Bunz et al. [14] are more efficient than generic zero-knowledge proofs, they still incur a relatively high complexity for the users (i.e. the provers) [47], and must be used in addition to the privacy-preserving data aggregation protocol and a cryptographic link between the two such as a commitment scheme.

4

A privacy-preserving data aggregation protocol with probabilistic range validation

We consider a setting with n users and a single aggregator, similar to related work in Chapter 3. Users continuously submit privacy-sensitive measurements to the aggregator at regular intervals called rounds; we assume that users and the aggregator have access to a synchronised clock. We work in the standard model, under the assumption that the discrete log problem is intractable. Some users are malicious and may deviate from the protocol; these are exactly the users the aggregator wants to identify. All other users are honest-but-curious (also known as semi-honest). We assume that the aggregator is honest-but-curious, an assumption made in several related works, including [28, 29]. This assumption makes sense in a business-driven setting, in which a malicious aggregator would be faced with negative publicity and a loss in consumer trust if its malicious behaviour were discovered. Still, we allow users and the aggregator to collude by exchanging information or coordinating behaviour. Finally, we assume that the security, integrity, and authenticity of all messages is guaranteed. The notation used to describe our protocol is shown in Table 4.1.

Our protocol consists of four phases: registration, submission, aggregation, and detection. Registration occurs only once, at the start of the protocol. Submission happens once in each round. The aggregation and detection phases occur asynchronously and can be scheduled freely by the aggregator. We give a brief overview of the phases below, and describe the individual phases in more detail in subsequent sections.

- *Registration*: Each user sends a message to the aggregator indicating that they want to register. Once all users have registered, the aggregator divides the users into overlapping groups. It then sends information such as the public parameters and the group configuration to all registered users.
- *Submission*: Every round, the users in each group secret share the value 0. Then, each user takes copies of their private value and blinds each copy with the secret share of a different group. The user sends the blinded copies in addition to commitments to the secret shares to the aggregator.
- *Aggregation*: The aggregator verifies that the secret shares of each group sum to 0 and verifies that each user used copies of a single private value, remembering which users and groups failed verification. Next, the aggregator computes the sum of private values of each group, and remembers which groups have aggregates that are out of bounds. Finally, the aggregator combines all group aggregates to find the sum of all private values.
- *Detection*: After completing aggregation, the aggregator looks back at which groups exhibited malicious activity over the past several rounds, and derives from their overlap which users caused the malicious behaviour. As the protocol progresses, the aggregator identifies more and more malicious user.

4.1. Registration

The goal of the registration phase is to determine the parameters under which the protocol will run and to exchange any information that is necessary for the protocol to run each round. Broadly speaking, the registration phase works in two steps. Firstly, the aggregator generates the public parameters pp, and at the

Table 4.1: The notation used in the description of our protocol

Symbol	Meaning	Notes
n	Number of users	$n = b^\ell$
b	Grouping base/radix = users per group	$b \geq 2$
ℓ	Grouping dimensionality = groups per user	$\ell \geq 2$
$[min, max]$	Valid range of a single private value	$min < max$
g	Generator for commitments	
pp	Public parameters, contains all of the above	
U	Set of all user identifiers	
G	Set of all group identifiers	
G_i	Set of identifiers of groups of user i	
U_j	Set of identifiers of users of group j	
N_i	Set of identifiers of neighbours of user i	
(sk_i, pk_i)	User i's key pair	
t	Round number	$t \in \mathbb{N}$
$m_{i,t}$	User i's private value in round t	
$c_{i,j,t}$	User i's encryption of $m_{i,t}$ for group j	
$M_{j,t}$	Sum of private values of users in group j in round t	
M_t	Sum of all private values in round t	
$r_{i \to j,t}$	User i's random value for neighbour j in round t	
$s_{i,j,t}$	User i's secret share for group j in round t	
$d_{i,j,t}$	User i's commitment to $s_{i,j,t}$	
V	Set of identifiers of groups that aggregator has marked as malicious	
W	Set of identifiers of users that aggregator has marked as malicious	
$\|x\|$	Size of x	
$x[:i]$	First i elements of x	
$x[-i:]$	Last i elements of x	

same time, users generate some key material and several random numbers. Secondly, the users register with the aggregator, which shares the public parameters with the users and negotiates the exchange of random numbers between users.

The most important parameters are n, g, min, and max. The aggregator chooses application-specific values for n and $[min, max]$ such that $min < max$. The aggregator also chooses g to be a random generator of an algebraic structure in which the discrete log problem is hard, such as a Diffie–Hellman group or an elliptic curve. The order of this group must be sufficiently large to store at least one user submission. The remaining parameters relate to the grouping algorithm and the secret sharing scheme, and we describe their selection in subsequent sections.

If users disagree with the aggregator's choice of parameters, they can refuse to participate in the protocol by not sending any further messages. Therefore, the aggregator should take care to choose the public parameters such that users can be expected to accept them. Malicious users can at most refuse to participate despite a choice of good parameters, at which point the aggregator immediately knows the identity of that malicious user; the aggregator adds the groups of this user to the set V, an action that we describe in more detail in Section 4.3.

4.1.1. Parameters for the grouping algorithm

The grouping algorithm is a crucial component of our protocol because it is the foundation upon which the detection algorithm is built. Principally, the detection algorithm looks at the behaviour of groups to infer the behaviour of their users: If a group exhibits malicious behaviour, then at least one user in that group must be malicious. The grouping algorithm ensures that users are placed in a unique combination of groups such that the aggregator can pinpoint malicious users based on the set of misbehaving groups. Different grouping algorithms provide different guarantees and have varying privacy implications. They may also differ in the number of malicious users they can successfully detect.

We propose a grouping algorithm based on the structure of hypermeshes. The aggregator creates a hypermesh with n nodes such that each user corresponds to a random node. The hyperedges in the hypermesh

are then the groups to which users belong. As such, each user i is in groups G_i, and each group j contains users U_j. The honest-but-curious aggregator chooses the bases b of the hypermesh based on a trade-off between performance, privacy, and detection rate; these parameters can be tailored to the protocol's specific application. The aggregator sends the bases b to each user as part of the public parameters, and communicates the user's hypermesh identifier to each user. This is sufficient information for users to reconstruct all properties of the hypermesh. We prove a number of theorems regarding our grouping algorithm's complexity, correctness, and performance in Chapter 6.

4.1.2. Exchanging data for secret sharing

Our scheme uses secret sharing to prevent the aggregator from decrypting ciphertexts unless all ciphertexts of a group have been aggregated. We apply the procedure for creating 0-sum additive secret shares used by Erkin and Tsudik [29] to each group in G. We avoid direct communication between users by forwarding messages through the (honest-but-curious) aggregator, but use public-key encryption to ensure that the aggregator cannot see the actual random values being transmitted. Our goal is to obtain secret shares $s_{i,j,t}$ for each user $i \in U$ in each group $j \in G_i$ in each round t such that

$$\forall j \in G : \sum_{i \in U_j} s_{i,j,t} = 0. \tag{4.1}$$

While the following description assumes that users exchange random numbers each round, such excessive communication can be avoided by having users exchange their seed for a random number generator once during registration.

First, each user i generates an asymmetric key pair (sk_i, pk_i), and includes pk_i when sending the registration message to the aggregator. Once all n users have registered, the (honest-but-curious) aggregator sends to each user i the public keys $\{pk_k \mid k \in N_i\}$. These key pairs can be reused and do not need to be exchanged again in future rounds. Later, in each round t, user i generates a random number $r_{i \to k,t}$ for each neighbour $k \in N_i$. To ensure that these random numbers can be used to properly hide private data, its bit length should be the maximum bit length of a private datum plus security parameter κ, say $\kappa = 80$. User i then encrypts each $r_{i \to k,t}$ with pk_k, and sends this value together with the identity k to the aggregator, who forwards the message to user k. Once user i has obtained $r_{k \to i,t}$ for each neighbour $k \in N_i$, user i creates a secret share

$$s_{i,j,t} = \sum_{k \in U_j} \left(r_{i \to k,t} - r_{k \to i,t} \right) \tag{4.2}$$

for each $j \in G_i$. That is, for each group the user subtracts the values it received from the values it sent. This construction satisfies Equation 4.1 because

$$\sum_{i \in U_j} s_{i,j,t} = \sum_{i \in U_j} \sum_{k \in U_j} \left(r_{i \to k,t} - r_{k \to i,t} \right) = \sum_{i \in U_j} \sum_{k \in U_j} r_{i \to k,t} - \sum_{k \in U_j} \sum_{i \in U_j} r_{i \to k,t} = 0. \tag{4.3}$$

If at least one user in group j other than i is honest, then $s_{i,j,t}$ cannot be predicted by the aggregator even if the other users in group j collude with the aggregator by sharing their random values. We analyse this property formally in Section 6.2.1.

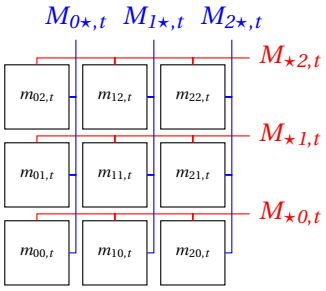
4.2. Submission

In round t, each user i should submit a private value $m_{i,t}$ to the aggregator. Our goal is to let the aggregator obtain the aggregate $M_{j,t}$ of each group without being able to see or calculate the $m_{i,t}$ of any user. We propose a scheme in which each user sends one copy of their measurement $m_{i,t}$ for each group $j \in G_i$ they are in, as illustrated in Figure 4.1. Using the secret sharing scheme described in Section 4.1.2, we can naively achieve these goals using the encryption function

$$c_{i,j,t} = m_{i,t} + s_{i,j,t} \tag{4.4}$$

to have each user i send $\{c_{i,j,t} \mid j \in G_i\}$ to the aggregator. The aggregator then sums these messages per group so that the secret shares cancel out and only the sum of private values remains.

To ensure that malicious users cannot insert inconsistencies into the system, users additionally provide commitments to the secret shares they use. That is, given a malicious user i, we prevent this user from sending $m_{i,t} + s_{i,j,t}$ in group j and $m'_{i,t} + s_{i,j',t}$ in group $j' \neq j$ such that $m_{i,t} \neq m'_{i,t}$. While the secret shares still cancel out correctly if they are used honestly, the malicious user contributes different values to different groups,



$$M_t = M_{0\star,t} + M_{1\star,t} + M_{2\star,t} = M_{\star 0,t} + M_{\star 1,t} + M_{\star 2,t}$$

Figure 4.1: Overview of the desired aggregates in a (3,3)-hypermesh

resulting in different total aggregates along the hypermesh's dimensions. Clever malicious users could use this to their advantage to avoid being detected while still nudging the aggregate in their favour. Therefore, we use a simple homomorphic commitment scheme that is information-theoretically binding and computationally hiding: To commit to a value x, a user sends g^x. Recall that g was chosen by the honest-but-curious aggregator so that finding x from g^x is computationally intractable. Then, each user i computes commitments $d_{i,j,t} = g^{s_{i,j,t}}$ and sends $\{(c_{i,j,t}, d_{i,j,t}) \mid j \in G_i\}$ to the aggregator. Note that $g^x = g^{x \bmod |g|}$, where $|g|$ is the order of g, because the exponentiation operation is by definition modulo $|g|$.

4.3. Aggregation

The aggregator does not need to wait for all inputs to arrive before aggregating. Instead, the aggregator considers the data that have arrived so far to perform intermediate validation and aggregation. This method of operation is necessary considering that malicious users may refuse to submit their data. The goal of the aggregation phase is to calculate the sum of private values in that round, and to mark groups that have shown malicious behaviour by adding their identifiers to the set V. The aggregator looks at V during the detection phase to find out which users caused the malicious behaviour as described in Section 4.4.

At a regular interval, possibly independent of the interval that determines rounds, the aggregator looks at the user submissions that have arrived. The aggregator marks each user i that has failed to submit their data for a previous round within a certain timeout as malicious by adding each group in G_i to V. Optionally, the aggregator can choose to be lenient by adding these groups to V only if user i fails to submit their data with some regularity. Then, for each current or past round t, the aggregator tries to calculate that round's aggregate M_t as described below. If not all submissions are available for that round, the aggregator computes an approximation of M_t. The aggregator needs to recompute M_t for a round t only if more submissions have arrived for that round since the last time the aggregator computed M_t.

Before calculating the aggregate of round t, the aggregator verifies that the inputs are well-formed. Given the user submissions for round t that have arrived so far, the aggregator first verifies for each group $j \in G$ that

$$\prod_{i \in U_j} d_{i,j,t} = \prod_{i \in U_j} g^{s_{i,j,t}} = g^{\sum_{i \in U_j} s_{i,j,t}} = g^0 = 1 \tag{4.5}$$

to ensure that users committed to secret shares of the value 0. If a group j fails this check, at least one user in this group must have been malicious, so the aggregator adds j to V. This check does not occur for groups of which not all users have sent their data yet; the aggregator simply postpones this check until the missing data have arrived, or adds the disrupting user's groups to V as described above otherwise. Next, the aggregator verifies that for each user i the set

$$\left\{ g^{c_{i,j,t}} \left(d_{i,j,t}\right)^{-1} \mid j \in G_i \right\} = \left\{ g^{m_{i,t} + s_{i,j,t}} g^{-s_{i,j,t}} \mid j \in G_i \right\} = \left\{ g^{m_{i,t}} \mid j \in G_i \right\} \tag{4.6}$$

contains only one value, i.e. that all its elements are the same. This check ensures that each $c_{i,j,t}$ for user i uses the same underlying $m_{i,t}$, enforcing the consistency requirement of the protocol. We prove that this requirement is satisfied in Section 6.1.1. If a user i fails this check, all groups in G_i are added to V. Again, users that have not sent all of their data for round t are exempt of this check.

After the aggregator has completed its verifications, it starts aggregating. For each group $j \in G$, the aggregator calculates

$$M_{j,t} = \sum_{i \in U_j} c_{i,j,t} = \sum_{i \in U_j} \left(m_{i,t} + s_{i,j,t} \right) = \sum_{i \in U_j} m_{i,t}. \tag{4.7}$$

If an aggregate $M_{j,t}$ is not in the range $\left[|U_j| \cdot min, |U_j| \cdot max \right]$, at least one user must have sent a value that is not in $[min, max]$, so the aggregator adds j to V. This check can be adjusted to support use cases with a varying range by choosing a range $\left[min_{i,t}, max_{i,t} \right]$ for each user i in each round t and then verifying that

$$M_{j,t} \in \left[\sum_{i \in U_j} min_{i,t}, \sum_{i \in U_j} max_{i,t} \right]. \tag{4.8}$$

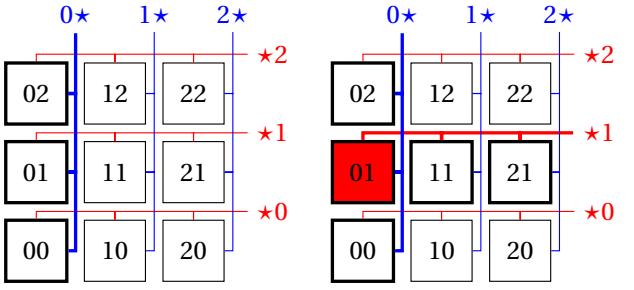
We analyse the detection rate of this method in Section 6.4.

The sum of all private values can be calculated by taking the sum of all group sums. Recall that the set of all groups along the same dimension contain all users once, so the set of all groups along all dimensions contains all users ℓ times. Hence, the sum of all group sums should be divided by ℓ. Therefore, the aggregator calculates

$$M_t = \frac{\sum_{j \in G \setminus V} M_{j,t}}{\ell}, \tag{4.9}$$

which is the average of the total sums along each of the hypermesh's ℓ dimensions, excluding the malicious groups in V. This approximates the sum of only the honest-but-curious users; if all users are honest-but-curious this approximation is perfect. If desired, the aggregator can estimate the sum of all users by including a fake group aggregate for each group in V based on the average of $\{M_{j,t} \mid j \in G \setminus V\}$. We discuss the impact malicious users have on the accuracy of the aggregate in Section 6.1.3.

4.4. Detection



(a) Only the group 0★ has exhibited malicious behaviour, so the malicious user cannot be identified yet

(b) Both group 0★ and group ★1 have exhibited malicious behaviour, so user 01 must be the malicious user

Figure 4.2: Detection in a (2,2,2)-hypermesh, assuming there is a single malicious user, with detected users and groups in bold

The aggregator uses the detection algorithm to identify which users are malicious. Throughout the protocol and across rounds, the aggregator has added groups that displayed malicious behaviour to the set V. Consequently, the set V contains all groups in which at least one user behaved maliciously, for example by using a wrongly constructed secret share, sending out-of-range values, or not sending anything at all. Based on the overlaps of groups in V, the aggregator infers which users caused the malicious behaviour; users that are in exactly ℓ different groups in V are malicious, and the aggregator adds these users to W. After a single round, only a few groups have been added to V and it can happen that the aggregator is unable to detect some malicious users with the current information. As the protocol continues to run its rounds, however, the aggregator adds more groups to V and is able to pinpoint more malicious users. We analyse the detection performance in Section 6.4. The actual actions undertaken by the aggregator after pinpointing a malicious user depend on the application. For example, in a smart metering application, the aggregator may send an inspector to the user's home to collect evidence of tampering.

As an example of how detection works, consider the (3,3)-hypermesh in Figure 4.2. When only a single group has shown malicious behaviour, the aggregator is unable to identify which user is malicious. However, a few rounds later the aggregator may have placed another group in V, at which point the aggregator can pinpoint the malicious user with certainty. In this example, identification can go wrong if there are two malicious

users. For example, if users 02 and 21 are malicious and user 01 is honest-but-curious, it may happen that only groups $0\star$ and $\star 1$ are detected, but user 01 will still be marked as malicious. Until the groups $\star 2$ and $2\star$ are detected, the aggregator cannot see the difference between these two scenarios. We prove in Section 6.1.2 that our detection algorithm does not have false positive detections as long as the number of malicious users is strictly less than ℓ.

5

Extensions for increased flexibility and privacy

We extend the design described in Chapter 4 in four ways to achieve increased flexibility and privacy. Firstly, we show how the aggregator can obtain temporal aggregates in addition to spatial aggregates. Secondly, we increase flexibility in the choice of the number of users n by generalising the hypermesh structure. Thirdly, we build on the generalised hypermesh to allow users to dynamically join and leave. Finally, we improve privacy guarantees by adding distributed differential privacy.

5.1. Performing temporal aggregation

Currently, our protocol supports only spatial aggregation, i.e. the aggregate of a set of users. We extend our protocol to additionally support temporal aggregates, i.e. the aggregate of the submissions of a single user over several rounds. An example use of this extension is in smart metering, where the aggregator is interested in finding the total energy consumption of its customers for billing purposes.

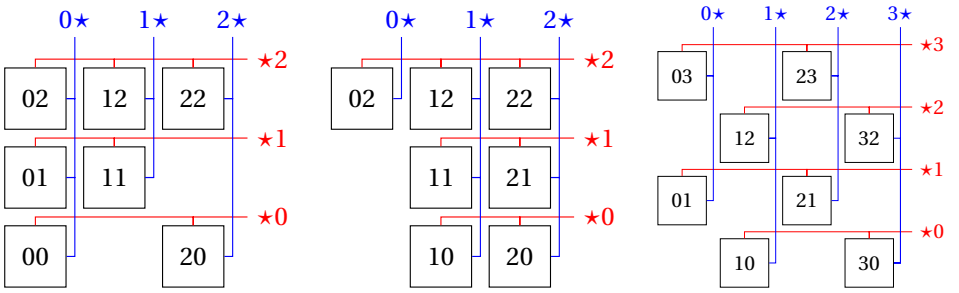
To achieve temporal aggregation, the aggregator creates a new virtual dimension of size b_τ by sending b_τ to each user during registration. The addition of this virtual dimension creates a virtual group of size b_τ for each user. However, unlike regular groups, each virtual group is used by only one user. The aggregate of the virtual group is then the temporal aggregate of that user. Note that the vector of bases b is not adjusted, and as such the equations in Section 4 do not consider this dimension during submission, aggregation, and detection. Instead, the virtual dimension is used as follows.

Submission During submission, each user generates a fresh secret share for their virtual group. Since this user is the only user in this virtual group, there is no need to exchange keys or random numbers as in Section 4.1.2. Instead, the user generates fresh secret shares so that the secret shares of b_τ subsequent rounds sum to 0. To do this, the secret share for rounds t such that $t \not\cong 0 \bmod b_\tau$ is chosen at random. Then, for rounds t such that $t \cong 0 \bmod b_\tau$, the secret share is the negation of the sum of the previous $b_\tau - 1$ secret shares. Each user then uses this secret share to send their submission and the accompanying commitment for the virtual group as with other groups, as described in Section 4.2.

Aggregation and validation The aggregator stores the submissions for virtual groups until it has received a sequence of b_τ submissions, the last of which is in a round t such that $t \cong 0 \bmod b_\tau$. Temporal groups are validated and aggregated in the same way as any other group, except that they are not used in the calculation of the total aggregate. Note that the consistency validation in Equation 4.8 also includes the virtual group. Therefore, if user submissions are transformed for an aggregate-sum query as explained in [12], the temporal aggregate does not represent the sum of the untransformed values.

Detection The detection algorithm is slightly adjusted to make use of the possibility to detect malicious behaviour in virtual groups. If a virtual group is in V, then the user of that group must be malicious and they are added to W. Additionally, all groups of this malicious user are added to V to ensure that the malicious submissions are excluded from future aggregates.

5.2. Increased flexibility with incomplete hypermeshes



(a) A $((3,3),\{10,21\})$-hypermesh, which is valid

(b) A $((3,3),\{00,01\})$-hypermesh, which is invalid because group $0\star$ contains only a single user

(c) A $((4,4),\{00,02,11,13,20,22,31,33\})$-hypermesh, which is invalid because it is disconnected

Figure 5.1: Examples of incomplete hypermeshes

In the design in Chapter 4, the number of users n is the product of the bases b and must hence be a composite number. We generalise the irregular hypermesh as described in Section 2.5 to allow for any number of users $n \geq 4$, and to create more variety in the choice of b given some n. In particular, we allow for incomplete hypermeshes in which a subset of nodes is unused. As a result, n can now be expressed as a composite number minus some other number. This is sufficient to describe any number $n \geq 4$; trivially, any prime number $n \geq 4$ can be described as a $\left(2, \frac{n+1}{2}\right)$-hypermesh with one missing node, though various other configurations are also possible.

Formally, an incomplete hypermesh is described by a set of gaps Γ that contains the identifiers of the unused nodes, in addition to a vector of ℓ bases b like in complete irregular hypermeshes. A (b,Γ)-hypermesh contains $n = \prod_{i=0}^{\ell-1} b_i - |\Gamma|$ nodes. A b-hypermesh is a special case of a (b,Γ)-hypermesh where $\Gamma = \emptyset$. Given a b-hypermesh H and a (b,Γ)-hypermesh H' for the same b, we redefine U, N_i, U_j for arbitrary $i \in U$ and $j \in G$ in the context of H' to return what they would return in the context of H, minus the entries of Γ. Additionally, in H', values G_i and N_i for $i \in \Gamma$ are undefined. With these changes, the formulas from Chapter 4 do not need to be adjusted.

For our extended protocol, we have two additional requirements that the hypermesh must satisfy. We call a hypermesh that matches these requirements a "valid hypermesh". Firstly, a valid hypermesh must not contain any groups with only one node. The aggregator should not aggregate groups with only one user since this would trivially reveal that user's private value. However, excluding this group from aggregation would result in a user that is in fewer than ℓ groups, which precludes our detection algorithm from pinpointing this user. Therefore, we disallow groups with only one node altogether. Secondly, all nodes in a valid hypermesh must be connected. Formally, two nodes i,k are connected if $i \in N_k$, or if there exists a node j such that nodes i,j are connected and nodes j,k are connected. Without this requirement, it becomes much easier for the aggregator to find individual users' private values by solving a system of linear equations. We give some examples of valid and invalid hypermeshes in Figure 5.1. We prove formal privacy guarantees for valid incomplete irregular hypermeshes in Section 6.2.2.

When gaps are introduced, the rank of the hypermesh's incidence matrix decreases. We discuss the relevance of the rank of the incidence matrix with regards to privacy in Section 6.2.2. We require that the incidence matrix has at least $\lambda \geq 1$ unknowns, where λ is a security parameter chosen by the aggregator during registration. As with other parameters, the aggregator chooses the value such that honest-but-curious users are likely to accept the aggregator's choice.

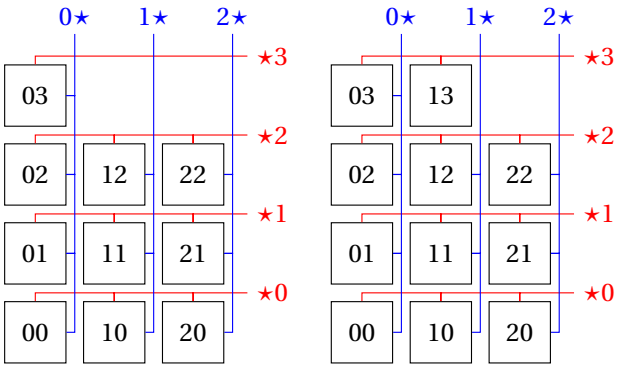
In conclusion, the only change with respect to the standard protocol from Chapter 4 that is required for this extension is to replace the complete irregular hypermesh with a valid incomplete irregular hypermesh. To achieve this, the aggregator has to share Γ and λ as part of the public parameters pp, after which all users and the aggregator should use the adjusted definitions of U, N_i, G_i, U_j for all users i and groups j.

5.3. Support for dynamic user sets

The basic protocol in Chapter 4 marks users that fail to respond as malicious. However, in practice, there may be legitimate reasons for users to leave or join the protocol, or the aggregator may wish to remove users it has detected as malicious to make space for other users. We extend our protocol to support dynamic joins and

leaves of users by using incomplete hypermeshes. The core idea is to insert gaps into the hypermesh when users leave, and to fill gaps when users join. Because we restrict the design to valid incomplete irregular hypermeshes, there are some limitations. For example, a user cannot leave if removing their node would result in a group with only one user, and new users cannot be placed in gaps that would create a disconnected hypermesh. Because the aggregator can see the difference between aggregates before and after users join and leave, we recommend that this extension is used in conjunction with the differential privacy extension described in Section 5.4.

Our extended protocol with support for dynamic user sets works in three steps. Firstly, when a user wants to join or leave, this user sends a message to the aggregator, who adds the request to a queue. The aggregator may additionally insert removal requests for users it has determined to be malicious. Secondly, when the aggregator is ready to process the queue of requests, it determines what changes must be made to the hypermesh's structure to accommodate as many requests as possible. Finally, the aggregator communicates these changes to all users in the hypermesh and re-runs part of the registration phase where necessary. We now describe the latter two steps in more detail.



(a) Adding a single user to a hypermesh without gaps creates a group with only one user, preventing this user from being pinpointed by the aggregator should they be malicious

(b) Adding two users at the same time allows either user to be pinpointed should they be malicious

Figure 5.2: Addition of users to a $((3,3),\emptyset)$-hypermesh when $\Gamma = \emptyset$

After a time-based trigger or when the queue is full, the aggregator processes the queue of pending requests according to the following steps to determine what needs to change.

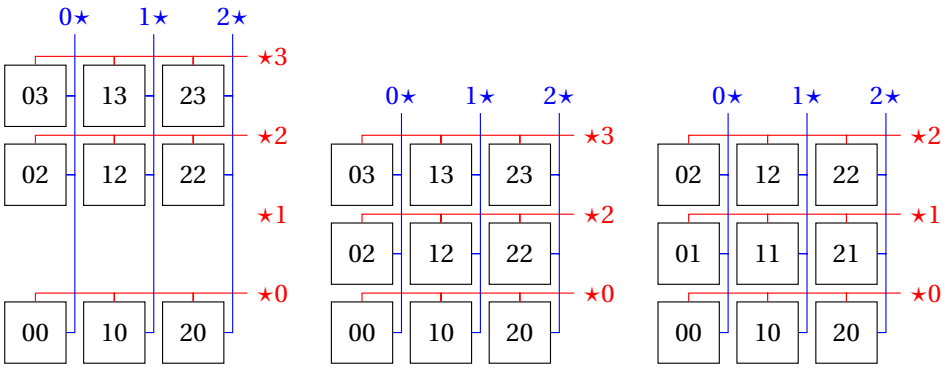
1. The aggregator takes the first join request and the first leave request, and assigns the node of the user to be removed to the user to be added. The aggregator removes these two requests from the queue, and repeats this matching until all requests in the queue are of the same type, or the queue is empty. This way, the structure of the hypermesh does not have to be adjusted; only the identities of the users occupying the hypermesh change.

2. If all remaining requests are join requests, the aggregator processes them as follows. For each request, the aggregator assigns a node from Γ to the user, removing that gap from the hypermesh, but only if this would not result in an invalid hypermesh. Optionally, before handling each request, the aggregator can expand the hypermesh by choosing a dimension $i \in [0,\ell)$, incrementing b_i by one, and adding each node in the newly created $(i, b_i - 1)$th sub-hypermesh to Γ. However, adding only one user in the newly created sub-hypermesh would make that user the only user in all groups in that sub-hypermesh, making the hypermesh invalid. Therefore, when extending the hypermesh to accommodate more users, the aggregator should add at least $2^{\ell-1}$ users at once.

   Effectively, the protocol stipulates only that the aggregator should deliver a best effort at handling join requests, but is not required to provide any guarantees on when it will do so. We illustrate the need to add multiple users at once after growing a hypermesh in Figure 5.2.

3. If all remaining requests are leave requests, the aggregator may process any subset of removal requests by removing the requesters' nodes from the hypermesh, as long as the resulting hypermesh is valid and the rank of the incidence matrix satisfies the security requirements with respect to the parameter λ

as discussed in Section 5.2. The aggregator may also process leave requests such that the (i, j)th sub-hypermesh is empty; the sub-hypermesh will be shrunk in the next step. The aggregator may prioritise removals of malicious users over the removal of other users.



(a) After users 01, 11, 21 have left, the sub-hypermesh ⋆1 is empty and can be removed

(b) The ⋆1 sub-hypermesh has been removed and nodes have been rearranged to form a $((3,3),\emptyset)$-hypermesh

(c) Node identifiers have been updated to fit in the new identifier space

Figure 5.3: Removal of the $(1,1)$th sub-hypermesh contained in a $(3,4)$-hypermesh after all three users of the group ⋆1 have left

After having decided how the requests should be processed, the changes should be implemented and communicated. Since the secret shares used by users in a group no longer sum to 0 after a user has left or joined, some secret shares must be re-generated. The aggregator sends the following information to each user in the hypermesh to inform them of the changes that must be made. These data can be compared to a "diff" of the hypermesh.

- The aggregator sends the updated set of gaps Γ to each user.
- The aggregator communicates changes to the hypermesh size. For each dimension i that is extended, the aggregator sends `EXTEND` i. Subsequently, each receiver increments b_i by one. For each (i, j)th sub-hypermesh that is empty, the aggregator sends `REDUCE` i j. Each receiver decrements b_i by one and renames every user where the ith symbol in the identifier is $\geq j$ by decrementing this symbol by one. We illustrate such a reduction in Figure 5.3.
- Users update their secret shares. Given the secret sharing scheme in Section 4.1.2, users must exchange information that allows them to calculate an updated version of Equation 4.2. If the aggregator has stored the public keys when they were exchanged during the registration, then the aggregator sends to the newly joined user i the public key of each neighbour $k \in N_i$, and user i sends its public key to each neighbour through the aggregator. Otherwise, the secret sharing scheme must be started anew for each group in which a user left or joined. Either way, if the seed-based version of the secret sharing scheme is used, then the missing seeds $r_{i\to k,t}$ must be exchanged through the aggregator as described in Section 4.1.2.
- For each user i that has left or been replaced, the aggregator removes all G_i from V, forcing the aggregator to re-evaluate whether the group contains a malicious user.

Users of whom the neighbours did not change do not strictly have to be informed of changes in the hypermesh since these changes do not affect them. As such, the aggregator can choose to defer informing them of changes until they are affected at a later point. This reduces the amount of communication and computation that is necessary for these users. However, it requires the aggregator to keep track of which changes have not been sent to which user, increasing the space complexity at the aggregator.

5.4. Use of distributed differential privacy

The ability to dynamically add and remove users requires the introduction of differential privacy, as the aggregator would otherwise be able to derive users' private values from differences in the aggregate as users join and leave. Because the goal is to protect the privacy of users from the aggregator, users cannot trust the

aggregator to add the noise, and must instead rely on the distributed generation of differential noise, where each user adds this noise to their private value before submitting it to the aggregator. However, a private value plus some noise may no longer be in the valid range $[min, max]$, possibly causing the aggregator to mark an honest-but-curious user as malicious. We provide a construction that minimises the risk of false positive detection. Our construction functions independently of other extensions.

We use the distributed differential noise algorithm by Shi et al. [60], but other methods may also be applicable if they can guarantee privacy even when some malicious users do not add noise. Using the construction by Shi et al., each user i calculates local differential noise $r_{i,t}$ for round t as in Equation 3.2. Then, each user i adds the noise to their message $m_{i,t}$ before encrypting it as in Equation 4.4. Generating the noise requires knowledge of several variables: the leakage parameter ε, a lower bound on the ratio of honest users γ, and the query function's sensitivity S. Firstly, ε is chosen by the aggregator and is part of pp. Secondly, as we prove in Section 6.2.2, our protocol works correctly with up to ℓ malicious users, so we set $\gamma = \frac{n-\ell+1}{n}$. Finally, the aggregator needs to determine the sensitivity S of the query function. In our case, the most granular data the aggregator has access to is the aggregate of a group, so the sensitivity is the maximum change between the aggregates $M_{j,t}, M_{j,t+1}$ for any group j and round t. In [60], the sensitivity is determined based on the assumption that each user datum is in the range $[0, \Delta]$. We can ensure inputs are in this range with $\Delta = max - min$ by having each user i map their datum $m_{i,t}$ to $m_{i,t} - min$ before transmission, and then having the aggregator map each group aggregate $M_{j,t}$ to $M_{j,t} + |U_j| \cdot min$. If a malicious user i refuses to apply this mapping to their datum, the user effectively sends $m_{i,t} + min$. Then, the sensitivity S is $\delta \cdot (max - min)$, where δ is the largest number of users that can be changed in a group between two consecutive rounds. The value δ is chosen by the aggregator and is sent to each user as part of the public parameters pp. Then, if the dynamic user extension in Section 5.3 is used, the aggregator should not add or remove more than δ users to any single group in between any two consecutive rounds.

The purpose of our un-extended protocol is to detect malicious users that send a private datum that is not in the range $[min, max]$. However, when an honest-but-curious user sends a value that is close to max and then adds noise, it may happen that the resulting value is invalid, after which the aggregator incorrectly marks the user as malicious. Given local noise generation function θ and a transgressing group aggregate $M_{j,t}$, the aggregator can calculate the probability that the distance to the valid range's boundaries is caused by the noise function θ rather than malicious intent. Instead of adding a group to V based on whether the group's aggregate is outside of the valid range, the aggregator looks at the probability that the group's transgressions over multiple rounds are caused by honest-but-curious noise, and adds the group to V when the probability falls below a threshold chosen by the aggregator.

With this extension, our protocol protects the privacy of private data even when users dynamically join and leave. Unlike the un-extended protocol and our other extensions, care should be taken that this extension may cause false positive detections, albeit with low probability.

6

Analyses and proofs

We prove the correctness and privacy of our protocol, and analyse its detection rate and complexity. Because the extensions in Chapter 5 are generalisations of the protocol in Chapter 4, the analyses and proofs in this chapter apply to both the regular protocol and the extended protocol, unless noted otherwise.

6.1. Correctness analyses and proofs

If all users are honest-but-curious, the secret shares neatly cancel out as in Equation 4.7, and the aggregator correctly calculates M_t as in Equation 4.9. However, if some users are malicious, this story changes. In this section we prove that the verifications by the aggregator ensure consistency, we give an upper bound on the number of malicious users that the protocol can handle, and quantify the impact of malicious users to the calculation of M_t.

6.1.1. Proof of aggregate consistency

As per Equation 4.4, each user i blinds their measurement $m_{i,t}$ in round t using the secret share $s_{i,j,t}$ to obtain $c_{i,j,t}$. With the differential privacy extension to our protocol proposed in Section 5.4, the user additionally adds some noise $r_{i,t}$, but for simplicity we consider this to be part of $m_{i,t}$. If users were to send only the ciphertext $c_{i,j,t}$ for each $j \in G_i$, then a malicious user i could send two different values $m_{i,j,t} \neq m_{i,j',t}$ for groups $j \neq j'$ without the aggregator noticing. If the malicious user is intelligent, they can send a valid value in group j and an invalid value in group j'. Then, since the detection algorithm finds only those users of which there are ℓ groups in V, this malicious user will probably not be detected. Meanwhile, the invalid values still influence the calculation of M_t. To prevent malicious users from evading detection, our protocol requires users to send commitments of their secret shares to allow the aggregator to verify along the hypermesh's dimensions.

We show that it is infeasible for malicious users to contribute different values to different groups under the assumption that the discrete log problem is intractable in the group generated by g. This property holds even if all users are malicious. We prove this property in the standard model, and let every user i send $\{(c_{i,j,t}, d_{i,j,t}) \mid j \in G_i\}$ in fixed round t to the aggregator. We denote $s_{i,j,t} = \mathrm{dlog}_g(d_{i,j,t})$ and $m_{i,j,t} = c_{i,j,t} - s_{i,j,t}$ for all $i \in U$ and all $j \in G_i$, regardless of whether users constructed their messages honestly.

Theorem 6.1. *In our protocol and its extensions, users cannot send messages to the aggregator such that there is a user i with $m_{i,j,t} \neq m_{i,j',t}$ for any two groups $j, j' \in G_i$ such that the aggregator's verifications do not fail, assuming the discrete log problem is intractable in the group generated by g.*

Proof. For the sake of contradiction, let there be a user i that violates our theorem. First of all, note that if either user i or any neighbour $k \in N_i$ fails to send their messages, the aggregator's verifications fail, which contradicts our assumption. Otherwise, because we assume that the aggregator's verifications do not fail, it follows from verification of Equation 4.5 that $\sum_{i \in U_j} s_{i,j,t} = 0$, because

$$\prod_{i \in U_j} d_{i,j,t} = 1 \Leftrightarrow \mathrm{dlog}_g \left(\prod_{i \in U_j} d_{i,j,t} \right) = \sum_{i \in U_j} \mathrm{dlog}_g(d_{i,j,t}) = \sum_{i \in U_j} s_{i,j,t} = 0. \tag{6.1}$$

Next, we know from verification of Equation 4.6 that, for fixed $i \in U$ and $t \in \mathbb{N}$, all $m_{i,j,t} = c_{i,j,t} - s_{i,j,t}$ for $j \in G_i$ are equal. However, this contradicts the assumption that user i was able to construct $m_{i,j,t} \neq m_{i,j',t}$ for $j \neq j'$, thereby proving our theorem. □

6.1.2. Proof of correct identification

Over the duration of the protocol, the aggregator adds the identifiers of groups that have exhibited malicious behaviour to the set V. It is important to verify that the aggregator correctly identifies malicious users. We prove that it is not possible for an honest-but-curious user to be misidentified as a malicious user if there are strictly fewer than ℓ malicious users in the protocol. This upper bound holds regardless of the number of colluding users. It follows from our theorem that it is not possible for malicious users to frame an honest-but-curious user. Note that if the differential privacy extension from Section 5.4 is used, there is a small probability that the aggregator incorrectly adds a group to V, in which case an honest-but-curious can be identified as a malicious user.

Theorem 6.2. *In our protocol and all extensions except the differential privacy extension, if there are strictly fewer than ℓ malicious users, the aggregator does not identify an honest-but-curious user as a malicious user.*

Proof. Let V be the groups that have been detected by the aggregator. Note that the aggregator adds a group to V only if it exhibits malicious behaviour, so each group in V contains at least one malicious user.

Firstly, consider the case where there is only one malicious user i. Recall that the detection algorithm selects those users that are simultaneously in ℓ groups. So, if V contains fewer than ℓ groups, the aggregator identifies no users as malicious, so the theorem holds. Otherwise, if V contains exactly ℓ groups, then these must be the groups G_i since these are the only groups in the whole hypermesh that contain at least one malicious user. As such, the detection algorithm marks only user i as malicious, so the theorem holds. Finally, V cannot contain more than ℓ groups since there are only ℓ groups that contain a malicious user.

Otherwise, if there is more than one malicious user, we must have $\ell > 2$ by assumption of the theorem's antecedent. For the sake of contradiction, let there be an honest-but-curious user i whom the aggregator falsely identifies as malicious. Then this user must be in ℓ groups of V, i.e. $|V \cap G_i| \geq \ell$. Since user i is not malicious, each of these ℓ groups must contain at least one malicious user. Because a group contains those users that differ by exactly one digit, two neighbours in one group cannot also be neighbours in another group. Therefore, user i must have at least ℓ distinct malicious neighbours, with at least one in each group in G_i. However, we assumed there are strictly fewer than ℓ malicious users. Therefore, we find that $|V \cap G_i| < \ell$, so user i could not have been identified as a malicious user. □

6.1.3. Analysis of impact of missing aggregates

The aggregator stores the identifiers of groups that have shown malicious behaviour in the set V. By excluding the aggregates of these groups when calculating the aggregate as in Equation 4.9, the aggregator diminishes the effect that malicious users have on the aggregate. However, groups in V may also contain honest-but-curious users. As such, malicious users directly impact the accuracy of the aggregate. In this section we quantify the impact malicious users have on the total aggregate M_t.

Each user i sends their private datum $m_{i,t}$ along a maximum of ℓ dimensions, and the aggregator takes the average of the aggregate along each dimension. Effectively, each user sends a $\frac{1}{\ell}$ share of their datum in each group, and the aggregator sums all shares together to find M_t. In a perfect protocol, the aggregator would remove only the data shares of malicious users i, resulting in the correct aggregate for only the honest-but-curious users. In our protocol, the aggregator removes the data shares of all users in groups in V. In the worst case, V contains all groups of all malicious users, which results in the removal of the data shares both of malicious users and all their honest-but-curious neighbours. Specifically, if V is empty, each user contributes a ratio of $\frac{|G_i|}{\ell}$ data shares, where $|G_i|$ may be less than ℓ for some users in incomplete hypermeshes. When V is not empty, this ratio decreases to $\frac{|G_i \setminus V|}{\ell}$.

Based on our findings in Section 6.1.2, there are at most $\ell - 1$ malicious users. The largest impact on the aggregate occurs when no two malicious users are neighbours because this maximises the total amount of neighbours between the malicious users. At the same time, because there are only $\ell - 1$ malicious users, no user i ever has all of their data shares excluded by the aggregator. Compared to a perfect protocol that excludes only the $\ell(\ell - 1)$ malicious data shares, our protocol excludes a factor of b too many.

With a total of ℓb^ℓ data shares, the maximum impact that malicious users have on the aggregate is relatively insignificant and further diminishes as the number of malicious users decreases or when the aggregator evicts malicious users using the dynamic user protocol from Section 5.3.

6.2. Privacy proofs

By colluding with the aggregator, users may help the aggregator obtain the private value of some users. For example, if all-but-one users in a group send their private value to the aggregator, the non-colluding user's private value is the difference between the group's aggregate and the sum of the colluding users' private values. All privacy-preserving data aggregation protocols are vulnerable to this kind of attack, and there is no feasible method for preventing them. For our protocol not to be vulnerable to these attacks, we must assume that every group contains at least two users that do not collude with the aggregator, so that the aggregator can at worst obtain the sum of the non-colluding users in a group. Under this assumption, we give an upper bound on the number of users that may collude with the aggregator such that the aggregator cannot find the private values $m_{i,t}$ of non-colluding users.

Note that colluding users are not necessarily malicious since communication and collaboration outside of our protocol does not strictly violate our protocol. Also note that users that collude amongst each other but not with the aggregator cannot determine the private value $m_{i,t}$ of another user because they do not have access to any values that are derived from this private value. In the remainder of this section, we say "(non-)colluding user" to mean users that do (not) collude with the aggregator, regardless of whether the user colludes with other users.

6.2.1. Proof of confidentiality of secret shares

The private value $m_{i,t}$ of user i in round t is blinded with the secret share $s_{i,j,t}$ for groups $j \in G_i$. If this secret share is properly random, then it will properly blind the private value. We now prove that the secret share is properly random under some assumptions.

Theorem 6.3. *In our protocol and all its extensions, the aggregator cannot obtain the secret share $s_{i,j,t}$ for any $j \in G_i$ for a non-colluding honest-but-curious user i under the assumption that each group $j \in G_i$ contains at least one non-colluding user other than i.*

Proof. Firstly, note that finding $s_{i,j,t}$ from $d_{i,j,t}$ is not feasible under the assumption that the discrete log problem is hard in the group generated by g.

The secret share $s_{i,j,t}$ is constructed as in Equation 4.2, and its randomness thus depends on the randomness of the components $r_{i\to k,t}$. User i chooses half of the components themselves, and the other users in group j choose the rest. Therefore, as before, if each user $k \in U_j \setminus \{i\}$ colludes with the aggregator by sending $r_{i\to k,t}, r_{k\to i,t}$, then the aggregator can find user i's private value. However, since we assume that each group contains at least two users that do not collude with the aggregator, there always exists a user $k \neq i$ in group j such that the random components they exchanged with each other are unknown to the aggregator. Because user i is honest-but-curious, the component $r_{i\to k,t}$ is properly random, and so are the secret shares generated from these components, regardless of the components the colluding users have generated. Therefore, the secret shares used by user i cannot be known or predicted by the aggregator. □

6.2.2. Proof of confidentiality of private values

In our protocol, each group aggregate is a linear equation of the private values of the users in that group, and since each user is in multiple groups, the aggregator has access to a linear system of equations formed by the group aggregates. If the aggregator is able to solve this system, then it can find the private values of all users. For example, in a $(2,2)$-hypermesh, the aggregator has the system of equations

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} m_{00,t} \\ m_{01,t} \\ m_{10,t} \\ m_{11,t} \end{bmatrix} = \begin{bmatrix} M_{0\star,t} \\ M_{\star 0,t} \\ M_{\star 1,t} \\ M_{1\star,t} \end{bmatrix}. \tag{6.2}$$

Here, the matrix on the left-hand side is the incidence matrix of the hypermesh, with a row for each hyperedge and a column for each user. The aggregator can play with the matrix as much as it wants to express users' private values in terms of each other. However, not every system has a unique solution, depending on the relation between the rank and the number of variables. Since the system of equations is consistent by Theorem 6.1, the rank of the incidence matrix is the same as the rank of the augmented matrix. We can find the rank of the augmented matrix by transforming it to its reduced row echelon form, for example by applying

Gaussian elimination:

$$\left[\begin{array}{cccc|c} 1 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 & M_{\star 0,t} \\ 0 & 0 & 1 & 1 & M_{\star 1,t} \\ 0 & 0 & 0 & 0 & 0 \end{array}\right]. \tag{6.3}$$

While the system has four variables, it has a row rank of only three, meaning there are infinitely many solutions by the Rouché–Capelli theorem. That is, the aggregator cannot find a single user's private value. However, with only one unknown variable, if any single user colludes with the aggregator, all private values can be found. For example, considering the incidence matrices above, if user 00 sends their private value $m_{00,t}$ to the aggregator, then the aggregator can find $m_{01,t} = M_{0\star,t} - m_{00,t}$ and $m_{10,t} = M_{\star 0,t} - m_{00,t}$, and finally $m_{11,t} = M_{1\star,t} - m_{10,t}$. In a $(3,3)$-hypermesh, we find a rank of four, so with nine users at least five users must send their private value to the aggregator to allow it to solve the system of equations. Meanwhile, as we discussed in Section 6.2.1, if two users in the same group in the $(3,3)$-hypermesh send their private value to the aggregator, the aggregator is able to find the private value of the remaining user in that group. However, because the aggregator has determined only three of the five unknowns, this collusion does not affect users in other groups. Still, we maintain that it is necessary for each group to contain at least two users that do not collude with the aggregator.

We prove that there is no unique solution for the linear system of equations emerging from a hypermesh, and give an exact solution to its rank to provide an upper limit on the number of users that may collude with the aggregator. First, we define the rank for complete irregular hypermeshes, and then we discuss the rank of valid incomplete irregular hypermeshes.

Theorem 6.4. *The rank of the incidence matrix of a complete irregular b-hypermesh is $\prod_{i=0}^{\ell-1} b_i - \prod_{i=0}^{\ell-1}(b_i - 1)$.*

Proof. As in the example in Equation 6.2, we denote the incidence matrix such that each row corresponds to a hyperedge, and each column corresponds to a node. We choose this notation because it more intuitively corresponds to the group aggregate equations on which the matrix is based. This choice is arbitrary in our case because the row rank is equal to the column rank.

We recursively construct the incidence matrix, similar to how the hypermesh itself can be constructed. We start with a $b[-1:]$-hypermesh, which is simply a set of b_0 nodes that are connected by a single hyperedge. Its incidence matrix $C_{b[-1:]}$ is a $(1 \times b_0)$-matrix containing only 1s. From this, we can construct incidence matrices for hypermeshes of higher dimensions, starting at $\ell = 2$. In particular, as described in Section 2.5, to construct the ℓ-dimensional $b[-\ell:]$-hypermesh, we take $b_{\ell-1}$ copies of a $b[-\ell-1:]$-hypermesh, connect the nodes that have the same identifier with each other, and then prefix each node with the index of the hypermesh copy they are in. The copies that we combine into a larger hypermesh are identical, and so are their incidence matrices. The $b[-\ell:]$-hypermesh's incidence matrix thus starts with a diagonal of $b_{\ell-1}$ copies of $C_{b[-\ell-1:]}$. Then, to connect the identical nodes in the copies, we create new hyperedges (i.e. new rows) and set the cells in rows of identical nodes to 1. The result is a horizontal sequence of $b_{\ell-1}$ identity matrices $I_{n_{\ell-1}}$, where $n_\ell = \prod_{k=0}^{\ell-1} b_\ell$ is the number of users in a $b[-\ell:]$-hypermesh. We have thus constructed the incidence matrix

$$C_{b[-\ell:]} = \begin{bmatrix} C_{b[-\ell-1:]} & 0 & \dots & 0 \\ 0 & C_{b[-\ell-1:]} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & C_{b[-\ell-1:]} \\ I_{n_{\ell-1}} & I_{n_{\ell-1}} & \dots & I_{n_{\ell-1}} \end{bmatrix}, \tag{6.4}$$

where each 0 represents a matrix of the same size as $C_{b[-\ell-1:]}$ containing only 0s.

We now use complete induction on $\ell = |b|$ to prove that $\operatorname{rank}(C_b) = n_\ell - \prod_{i=0}^{\ell-1}(b_i - 1)$. For the base case, we take $\ell = 1$ and find that $\operatorname{rank}(C_{b[-1:]}) = 1$ because its incidence matrix is a row of 1s, proving our base case:

$$\prod_{i=0}^{0} b_i - \prod_{i=0}^{0}(b_i - 1) = b_0 - (b_0 - 1) = 1. \tag{6.5}$$

Then, for the recursive case, our induction hypothesis is that $\operatorname{rank}(C_{b[-\ell-1:]}) = n_{\ell-1} - \prod_{i=0}^{\ell-2}(b_i - 1)$. Our goal is to write $C_{b[-\ell:]}$ in column echelon form so we can find its rank by counting the number of non-zero columns. Note that this is equivalent to counting the number of non-zero rows in its transpose. First, consider how one can write $C_{b[-\ell-1:]}$ in column echelon form, for example by applying Gaussian elimination. Take the column

operations necessary to achieve this, and apply them to each instance of $C_{b[-\ell-1:]}$ in $C_{b[-\ell:]}$. Note that this also transforms the identity matrices located below the $C_{b[-\ell-1:]}$s. After applying these steps, each instance of $C_{b[-\ell-1:]}$ is in column echelon form, so the rightmost $n_{\ell-1} - \text{rank}(C_{b[-\ell-1:]})$ columns of each instance are all zero columns. Meanwhile, because the identity matrices at the bottom are full rank, its columns are all still non-zero. However, their $n_{\ell-1} - \text{rank}(C_{b[-\ell-1:]})$ rightmost columns are identical between each of the $b_{\ell-1}$ identity matrices. We eliminate these rightmost columns in all identity matrices except the rightmost one using simple column operations. Then, we transform the rightmost columns in the remaining identity matrix into column echelon form, and finally we move all empty columns to the right of the matrix. Thus, we have transformed $C_{b[-\ell:]}$ to column echelon form because we have a diagonal of block matrices in column echelon form, with additional entries below the diagonal and zero columns to the right. The only empty columns are those that were emptied as a result of removing duplicate columns using the identity matrices. Thus, the number of empty columns is

$$(b_{\ell-1}-1)\left(n_{\ell-1}-\text{rank}(C_{b[-\ell-1:]})\right) = (b_{\ell-1}-1)\left(n_{\ell-1}-\left(n_{\ell-1}-\prod_{i=0}^{\ell-2}(b_i-1)\right)\right) = (b_{\ell-1}-1)\prod_{i=0}^{\ell-2}(b_i-1) = \prod_{i=0}^{\ell-1}(b_i-1). \tag{6.6}$$

The rank of $C_{b[-\ell:]}$ is the number of non-empty columns, which is $\prod_{i=0}^{\ell-1} b_i - \prod_{i=0}^{\ell-1}(b_i-1)$. □

The number of unknowns in the system is then $\prod_{i=0}^{\ell-1}(b_i-1)$. As long as fewer than this number of users collude with the aggregator, the aggregator cannot solve the system of linear equations. For example, a regular hypermesh with $b=\ell=2$ could not tolerate a single user colluding with the aggregator, while a system with $b=\ell=10$ could tolerate up to $\frac{9^{10}}{10^{10}} \approx 35\%$ of its users colluding. In Figure 6.1 we show what percentage of users can be malicious as a function of b and ℓ in a regular hypermesh. As the number of groups per user grows, the collusion resistance decreases. This can be compensated for by increasing the number of users per group, but, as we discuss in Section 6.4, this decreases the detection rate. The balance between these variables should be fine-tuned to the requirements of the use case our protocol is applied to.

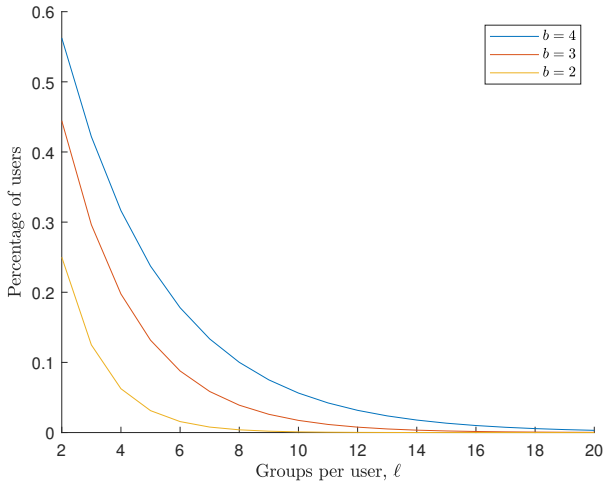


Figure 6.1: Percentage of users that can be malicious as a function of b (users per group) and ℓ (groups per user) in a regular hypermesh

We do not provide a characterisation of the rank of the incidence matrix of a valid incomplete irregular hypermesh. Instead, we note that the rank is such that the number of unknowns is at least $\lambda \geq 1$, by definition of λ. By prohibiting some choices for Γ, our protocol continues to guarantee confidentiality of private values given at most λ malicious users.

6.3. Complexity analysis

We compare our protocol with several related protocols in terms of complexity in Table 6.1. These works relate to our protocol either in purpose or in methodology. We describe these protocols in more detail in Chapter 3. Similar to the complexity analysis by Erkin [28], we express the complexity of the analysed protocols in terms of the number of encryptions, decryptions, multiplications, exponentiations, additions, and subtractions, and the amount of outgoing communication required. We analyse these metrics separately for each user and the aggregator. The results in Table 6.1 are subject to the following assumptions and simplifications:

- For Paillier-based protocols, we include only operations under modulo n^2 since operations under modulo n are relatively trivial.
- To simplify the notation for the protocol in [28], we assume that K is $\mathcal{O}(n)$, and that the aggregator and utility provider are the same entity.
- For the protocol in [19], we assume that the number of multiplication gates is linear to the size of the range.
- In the analysis of our own protocol, we exclude operations related to our grouping algorithm such as finding the users of a group.
- In our protocol, the number of users per group is, on average, $b = n^{\frac{1}{\ell}}$. This amount is maximal when $\ell = 2$, so we say that b is $\mathcal{O}\left(\sqrt{n}\right)$. Similarly, ℓ is $\mathcal{O}(\log n)$.

We begin by discussing the complexity of our protocol and its extensions. After that, we look at Table 6.1 in more detail.

6.3.1. Complexity of our protocol

We now dissect the complexity results for our protocol, starting with the complexity for users. Users send their messages in plaintext, albeit masked with a particular blinding, meaning encryption is trivial. The blindings themselves are calculated from the sum of random numbers, which is trivial as well. On the other hand, the commitments to the secret shares are not trivial to compute. Users create one commitment for each group they are in, resulting in $\mathcal{O}(\log n)$ exponentiations. Finally, the registration phase dominates the communication complexity, assuming we use the optimisation with which users need to exchange information only during the registration. Each user sends one message for each neighbour, resulting in $\mathcal{O}\left(\log n\sqrt{n}\right)$ outgoing messages. However, during each subsequent round, users send only one message for each group they are in, for a per-round communication complexity of only $\mathcal{O}(\log n)$.

The aggregator is subject to a higher overall computation complexity. It needs to verify all incoming messages per group and then aggregate the private values. First, for the verification in Equation 4.5, the aggregator multiplies the contributions of each user to each group; since each user is in ℓ groups this amounts to ℓn multiplications which is $\mathcal{O}(n \log n)$. Subsequently, for the verification in Equation 4.6, the aggregator performs one multiplication and one exponentiation per user, resulting in $\mathcal{O}(n \log n)$ multiplications and an equal amount of exponentiations. Finally, the registration phase dominates the aggregator's communication complexity, with the aggregator forwarding the random components for the secret shares for each user, resulting in $\mathcal{O}(n \log n\sqrt{n})$ outgoing messages. Meanwhile, the aggregator sends no messages during the aggregation rounds, other than some acknowledgements inherent to the underlying transportation protocols.

The extensions in Chapter 5 have little impact on the complexity of our protocol. Incomplete hypermeshes, as described in Section 5.2, reduce complexity by decreasing the number of neighbours a user has. Dynamic user sets, as described in Section 5.3, require the aggregator to update users of changes to the hypermesh's topology, but the associated communications and computations are trivial. Finally, adding differential privacy, as described in Section 5.4, has users add random noise to their datum once per round, which is trivial; the aggregator must additionally perform a number of constant-time operations to determine the likelihood of particular aggregates occurring, which is also trivial.

6.3.2. Comparison with related protocols

We start by comparing our protocol to a selection of similar privacy-preserving data aggregation protocols, as listed in Table 6.1. Several of these protocols impose only constant complexity requirements on their users whereas our protocol has logarithmic complexity. An important difference here is our protocol's ability to detect malicious users, for which we require users to submit their private datum to multiple groups. The other protocols would need to rely on zero-knowledge proofs in order to have this feature. At the same time, our protocol does not make use of heavy cryptography such as Paillier encryption, which makes our protocol more efficient overall given a relatively small number of users.

We now consider protocols in Table 6.1 that are able to detect malicious users. Two protocols such protocols are by Corrigan-Gibbs and Boneh [19] and by Bunz et al. [14]. Both solutions rely on a custom type of zero-knowledge proof. While both solutions can be used to evaluate arbitrary logic circuits on users' inputs, we consider only the variants in which the aggregator verifies ranges. For both methods, the complexity depends, in part, on the size of the valid range. Meanwhile, the complexity of our protocol is independent of the

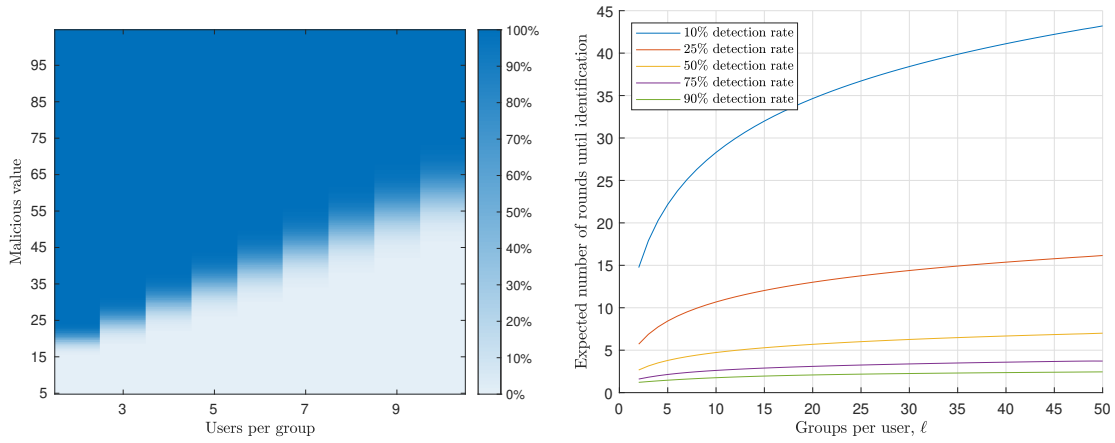
valid range, making our protocol more suitable for very large ranges. Note, however, that with our protocol, the detection rate varies depending on the valid range as we describe in Section 6.4.

With the protocol by Bunz et al., the aggregator can quickly verify large amounts of proofs using batch verification. However, this requires that all proofs are for the same range; if ranges differ between users then the complexity increases linearly in the number of users. The performance of our protocol does not change if users have different ranges to validate. Additionally, our protocol supports ranges of arbitrary size, while the protocol by Bunz et al. require ranges of the form $[0, 2^r)$ for some positive integer r.

6.4. Detection rate analysis

It may happen that the aggregator does not detect a malicious user despite that user sending an invalid value. This is because values submitted by honest-but-curious users in the same group as the malicious user may coincidentally compensate for the malicious transgression. That is, the aggregate of one invalid value and several valid values may still be valid. As a result, our detection algorithm is probabilistic. In this section we quantify the detection rate in terms of the protocol's parameters. In our analysis we model each honest-but-curious user's value as a truncated binomial distribution X with $\mu = \frac{min+max}{2}$ and a support of $[min, max]$. For the sake of illustration, we use $\sigma = 2$, $min = 5$, and $max = 15$. We model the sum of n independent honest-but-curious users' values, denoted X_n, by approximating X with a non-truncated binomial distribution, multiplying the distribution by n, and truncating this distribution to the range $[n \cdot min, n \cdot max]$. Given that these distributions are symmetrical, we investigate only malicious values larger than max, without loss of generality. Note that our analysis does not apply to the differential privacy extension from Section 5.4 because the following analysis does not consider the possibility that an honest-but-curious user sends an invalid value.

6.4.1. Detection rate of a single malicious user



(a) Probability that a group of given size transgresses the valid range given one malicious user in the group

(b) Expected number of rounds until a given number of the malicious user's groups has been detected by the aggregator, given the probability that a single group is detected in any round

Figure 6.2: Detection rates given a single malicious user

First of all, we look at the detection rate of a single malicious user i who submits the out-of-range measurement m. Recall that the aggregator detects a malicious user only once ℓ groups of that user have been added to V. The aggregate $M_{j,t}$ of group $j \in G_i$ is within range if and only if $X_{|U_j|-1} + m \leq |U_j| \cdot max$, given malicious value $m > max$. We illustrate this probability as a function of m and $|U_j|$ in Figure 6.2a. The figure shows that fixing a particular detection rate results in the corresponding malicious value growing linearly with the group size. Note that the detection rate is exactly 0% at $m = max$ (because this value is valid) and exactly 100% at $m = |U_j|(max - min) + min$ (because the aggregate is always invalid regardless of the honest-but-curious values).

We can model the detection of a group in a particular round as a Bernoulli trial with the probability of success calculated as above. Then, the expected number of rounds until the aggregator detects this group is exactly the expected value of a geometric distribution over independent instances of this Bernoulli trial. To find the expected number of rounds until all $|G_i|$ groups of malicious user i have been detected, first consider that it is irrelevant if the aggregator detects the user's groups in multiple rounds. The expected number of

rounds is thus the expected maximum number of rounds of $|G_i|$ independent geometric variables. In our case, the groups G_i of user i are independent by construction except for user i themselves, which makes the detection rate of these groups independent if we fix the value of m. In a complete regular hypermesh, all groups have the same number of users, so we can express the expected number of rounds until the aggregator detects a single malicious user as the maximum of ℓ identically distributed geometrically variables, which is [27]

$$f(\ell, p) = \sum_{k=0}^{n} \left(\binom{\ell}{k} p^k (1-p)^{\ell-k} \big(1 + f(\ell - k, p)\big) \right), \tag{6.7}$$

where ℓ is the number of groups and p is the per-round probability that the aggregator detects an individual group. Figure 6.2b shows how the expected number of rounds changes depending ℓ and p. The figure shows that more groups per user necessitates a higher per-group detection rate if the expected number of rounds before identification is to remain the same, which can be achieved by reducing the group size or by accepting that larger range transgressions may take place.

If the hypermesh is irregular or incomplete, the group sizes are different and their detection rates are no longer identically distributed. We propose that Equation 6.7 can be generalised to non-identical group detection distributions as

$$f(p) = \sum_{q \in \mathscr{P}(p)} \left(\prod_{i \in q} i \cdot \prod_{i \in p \setminus q} (1-i) \cdot \big(1 + f(p \setminus q)\big) \right), \tag{6.8}$$

where p is the set of probabilities for the malicious user's groups and $\mathscr{P}(x)$ denotes the powerset of x. Note that if all values in p are equal, our equation equals Equation 6.7. Intuitively, the function f expresses the expected number of rounds until the remaining groups in p are detected. When p is empty, the function returns 0 since no more rounds are necessary. Otherwise, the expected number of remaining rounds depends on which groups the aggregator detects in that round. Therefore, the function looks at all possible combinations of groups that can be detected, and takes for each such combination q the probability times the expected number of rounds for the remaining groups to be detected, plus one to increase the number of rounds with each recursive step.

6.4.2. Detection rate of multiple malicious users

When a group contains multiple malicious users, these users can either intensify or diminish the sum effect they have on their group's aggregate. Multiple malicious users thus either become harder to detect (if malicious users have equal reason to transgress the range in either direction) or easier to detect (if malicious users have more reason to transgress the range in one particular detection). We quantify the effect that multiple malicious users have on the detection rate, investigate the possibility of collusion, and analyse the effects of covariance.

Given a group of n users of which d are malicious, the detection rate depends on the sum m of the malicious users' values. In particular, the probability that the aggregate does not exceed the upper boundary of the group's valid range is

$$\Pr[X_{n-d} + m \leq n \cdot max] = \Pr[X_{n-d} + m - (d-1) \cdot max \leq (n-d+1) \cdot max]. \tag{6.9}$$

That is, the probability that the d malicious values do not transgress the maximum value of a group with n users is the same as the probability that a single malicious user does not transgress the maximum value of a group with $n-d+1$ users by sending the value $m-(d-1)\cdot max$. Similarly, the probability that the malicious values do not transgress the minimum value of the group is

$$\Pr[X_{n-d} + m - (d-1) \cdot min \geq (n-d+1) \cdot min]. \tag{6.10}$$

We can thus express the detection rate of a group with multiple malicious users in terms of the detection rate of a group with only one malicious user, allowing us to reuse the results from Section 6.4.1.

Malicious users may collude by coordinating the malicious values they send to minimise the risk of detection. If malicious users that share a group send values such that the sum of their values is valid even though the individual values are not, then the aggregator will not detect their shared group and will therefore not be able to pinpoint the malicious users. However, given d colluding malicious users in one group, the sum of their values must be in the range $[d \cdot min, d \cdot max]$ for their group not to be detected by the aggregator. Because the values that users send to the various groups they are in are consistent by Theorem 6.1, the sum effect that these colluding malicious users have on the total aggregate is in fact valid. Therefore, while the aggregator

does not detect the malicious users, the malicious users are unable to maliciously influence the aggregate as well. In other words, the detection rate of a coalition of malicious users is proportional to their sum effect on the aggregate.

Finally, an important observation given multiple malicious users is that those malicious users that do not share a group may still have an overlap in the users that they share groups with. In this case, the detection rates of these groups become covariant because of the common users. As shown in Figure 6.3, the impact of this covariance depends on the group size b and quickly becomes negligible.

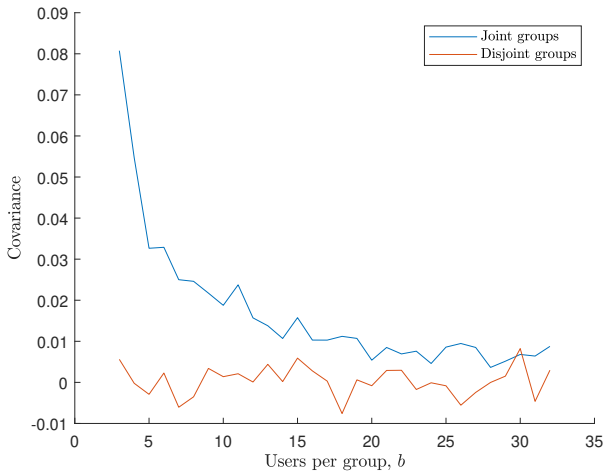


Figure 6.3: The covariance of the detection rate of two groups of equal size, each with a different malicious user and overlapping in one honest-but-curious user. The covariance is sampled from 5000 simulations per group size, with the malicious users behaving identically.

Table 6.1: Complexity analysis of several privacy-sensitive data aggregation protocols, separated by party: User (U) or Aggregator (A), given total number of users n and range size 2^r.

Protocol	[31]		[42]		[29]		[5]		[28]		[19]		[13]		[14]		Ours	
Aggregation	✓		✓		✓		✓		✓		✓		✓				✓	
Detection											✓				✓		✓	
Group	Paillier 2048 bits		DH group 256 bits		Paillier 2048 bits		HE 32 bits		Paillier 2048 bits		FFT field 87 bits		AES-GCM 128 bits		EC 256 bits		EC 256 bits	
Party	U	A	U	A	U	A	U	A	U	A	U	A	U	A	U	A	U	A
Encryptions	$\mathcal{O}(n)$	–	–	–	$\mathcal{O}(1)$	–	–	–	$\mathcal{O}(1)$	–	–	–	$\mathcal{O}(n)$	–	–	–	–	–
Decryptions	$\mathcal{O}(1)$	–	–	–	–	$\mathcal{O}(1)$	–	–	–	$\mathcal{O}(1)$	–	–	$\mathcal{O}(n)$	–	–	–	–	–
Multiplications	–	$\mathcal{O}(n^2)$	–	$\mathcal{O}(n)$	–	$\mathcal{O}(n)$	–	$\mathcal{O}(1)$	–	$\mathcal{O}(n)$	$\mathcal{O}(r \log r)$	$\mathcal{O}(r \log r)$	–	–	–	–	–	$\mathcal{O}(n \log n)$
Exponentiations	–	–	$\mathcal{O}(1)$	–	–	–	–	–	$\mathcal{O}(1)$	$\mathcal{O}(n)$	–	–	$\mathcal{O}(n)$	–	$\mathcal{O}(r)$	$\mathcal{O}(nr)$	$\mathcal{O}(\log n)$	$\mathcal{O}(n \log n)$
Additions	–	–	–	–	–	–	$\mathcal{O}(1)$	$\mathcal{O}(n)$	–	–	–	–	–	–	–	–	$\mathcal{O}(\log n)$	$\mathcal{O}(n \log n)$
Subtractions	–	–	–	–	–	–	–	$\mathcal{O}(1)$	–	–	–	–	–	–	–	–	–	–
Communications	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(\log r)$	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(r)$	$\mathcal{O}(nr)$	$\mathcal{O}(\sqrt{n} \log n)$	$\mathcal{O}(n\sqrt{n} \log n)$

7

Discussion and future work

Privacy is an important human right that is necessary to safeguard democratic values such as freedom of speech and freedom of thought. On one hand, privacy safeguards the dignity and integrity of people in their personal spheres, allowing them to develop themselves without constant scrutiny from their peers. On the other hand, privacy enforces a balance between individuals and powerful entities by creating an opportunity for defiance. When privacy is not maintained, either through malice or through incompetence, the consequences for democracy are dire. Design philosophies such as privacy by design aim to guarantee privacy by setting out clear guidelines for those working with privacy-sensitive data, requiring all involved personnel to consider privacy from the very start. Amongst others, the use of privacy-enhancing technologies (PETs) is a crucial component. Privacy-preserving data aggregation (PDA) protocols are one type of PETs that aim to provide strict privacy guarantees for aggregated data. Research into PDA protocols is thus an important step in the process of maintaining the right to privacy.

When choosing a PET to be incorporated into an application, it is essential that the PET satisfies the application's requirements. When the users that supply their data to the aggregator may be malicious, the aggregator should have a way to validate these data. Some PDA protocols can validate inputs, but come at the cost of greatly increased computational complexity, as is the case with some zero-knowledge proof-based solutions. As long as there is a lack of suitable PDA protocols, applications will continue to use non-PDA protocols. We asked whether it is possible to create a PDA protocol that performs range validation without the use of zero-knowledge proofs, and proposed and analysed a protocol that satisfies these criteria. In this chapter we discuss our results, consider some areas of future work, and conclude with an outlook on the future.

7.1. Discussion

In our protocol, the aggregator places the n users in multiple overlapping groups by choosing a b-hypermesh for bases $[b_{\ell-1}, b_{\ell-2}, \ldots, b_0]$ such that $\prod_{i=0}^{\ell-1} b_i = n$. In each round, each user submits their private value once for each group they are in, each submission with a different group-dependent mask. The aggregator asynchronously aggregates and validates user submissions per group to probabilistically identify users that sent invalid values. The aggregator finally obtains an approximation of the sum of all honest-but-curious values by taking the sum of all group sums, excluding groups that have displayed malicious behaviour.

We have shown several important properties of our protocol. Firstly, the aggregator never marks an honest-but-curious user as malicious as long as fewer than ℓ users are malicious. Secondly, users cannot submit different values to different groups under the assumption that the discrete log problem is intractable. Thirdly, the aggregator cannot find the individual private value of an honest-but-curious user i as long as fewer than $\prod_{i=0}^{\ell-1}(b_i - 1)$ users collude with the aggregator, and each group $j \in G_i$ contains at least one user other than user i that does not collude with the aggregator.

Our protocol imposes low computational requirements on its users, requiring only $\mathcal{O}(\log n)$ computations and communications per user per round. Unlike some zero-knowledge proof-based alternatives, the complexity of our protocol does not depend on the size of the valid range. As a result, our protocol also supports different valid ranges for different users without increasing computational complexity. Meanwhile, the detection rate of our probabilistic protocol depends on the valid ranges and the choice of the grouping algorithm's

bases b, and can thus be fine-tuned per specific application. Our protocol is most suitable for applications in which users are likely to transgress the same boundary of the valid range, although colluding users cannot use this property to their advantage.

We have also provided four extensions that generalise our protocol to improve upon the privacy and practicality of our protocol. With these extensions, our protocol supports temporal aggregation and the dynamic addition and removal of users without needing to restart the entire protocol. Additionally, we provide an extension for differential privacy to ensure that the aggregator cannot infer users' private data from changes in aggregates. While the differential privacy extension sacrifices the guarantee that the aggregator does not have false-positive detections, none of our extensions significantly affect the protocol's complexity or security.

Based on these results, we find that our protocol addresses a shortcoming in PDA protocols with range validation, without imposing significant new requirements on users or the aggregator. However, our protocol excels only under particular circumstances and should not be construed as a one-size-fits-all solution to all forms of privacy-preserving data aggregation, despite the flexibility in configuration it offers. More precisely, because our protocol is probabilistic in nature, its unpredictability may deter deployment in cases where immediate detection is of critical importance. Furthermore, our protocol is limited to calculating the sum of user inputs, and cannot be used or trivially adjusted to calculate other metrics such as the product or minimum of these values. Finally, our protocol continues to work correctly with up to a logarithmic proportion of malicious users, whereas zero-knowledge proofs do not suffer in detection rate when the number of malicious users increases. Therefore, the benefits of our solution and alternative solutions must be considered carefully before deploying it in an application.

7.2. Future work

Our protocol operates within specific constraints and provides well-defined guarantees. Some use cases may have requirements that do not fit within our protocol, however, and would benefit from an adjusted protocol. We envision three different ways in which our protocol could be extended or generalised.

Increasing the maximum number of malicious users Our protocol guarantees that the aggregator does not mistakenly identify an honest-but-curious user as a malicious one, but only if there are strictly fewer than ℓ malicious users. Once ℓ or more malicious users participate in the protocol, they may cause false-positive detections by the aggregator, even if these malicious users do not collude. A direction for future work would be to increase this limit so that the protocol becomes more resistant to malicious users.

Aggregating without aggregators Instead of using a single centralised aggregator, the protocol could be extended so that users themselves obtain the aggregates. With such an extension, some or all users would each perform the work the aggregator currently performs. The difficulty with implementing this extension lies in making decisions such as the eviction of malicious users since malicious users could deceive only some aggregators, and malicious aggregators could try to protect fellow malicious users.

Higher-dimensional groups Currently, the set of bases b provides a natural tradeoff between privacy, complexity, and detection rate. Instead of creating a group for each hypermesh, however, it might be possible to create groups from the smaller hypermeshes contained within the b-hypermesh. For example, every 2-dimensional hypermesh within a 4-dimensional hypermesh could be made into a group. Increasing the dimensionality of groups increases the group size and reduces the total number of groups, which could decrease overall complexity for users and the aggregator alike. However, with more users per group, the detection rate would suffer. Additionally, with two users now possibly sharing multiple groups with each other, aggregates become covariant in multiple variables, and it is not known if our privacy and security proofs apply to this generalisation.

7.3. Concluding remarks

Existing proposals for PDA protocols either ignore the problem of range validation or rely on zero-knowledge proofs. However, zero-knowledge proofs often come with limiting requirements on the setup of the protocol or impose high computational requirements on the users. We have proposed a PDA protocol that incorporates range validation into its design without the need for zero-knowledge proofs. Our privacy-preserving protocol requires no direct interaction between its users, features asynchronous validation and aggregation

by the aggregator, has low computational complexity demands for its users, and is robust to user dropouts and collusions. We additionally provide several extensions to our protocol to allow users to dynamically join and leave the protocol, and to add differential privacy to all group aggregates. Our protocol contributes to the attainability of deploying privacy-preserving protocols in the real world by addressing the need for new features. In turn, this contributes to the maintenance of dignity and integrity in people's personal spheres, and to the restoration of balance between people and powerful entities. With our protocol, it becomes yet more feasible to benefit from large-scale data collection while protecting individual privacy.

Bibliography

[1] Google Chrome Privacy Notice. Technical report, Google, 2013. URL https://www.google.com/chrome/privacy/whitepaper.html.

[2] Deceived By Design. Technical report, Forbrukerrådet, 2018. URL https://fil.forbrukerradet.no/wp-content/uploads/2018/06/2018-06-27-deceived-by-design-final.pdf.

[3] Google Privacy Policy Letter, 2020. URL http://www.attorneygeneral.gov/uploadedFiles/Press/AG-Google-Privacy-Policy-Letter.pdf.

[4] ...It Protects Marriage Equality, 2020. URL https://privacyinternational.org/case-study/3538/it-protects-marriage-equality.

[5] Gergely Ács and Claude Castelluccia. I have a DREAM! (DiffeRentially privatE smArt Metering). In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6958 LNCS, pages 118–132. Springer Berlin Heidelberg, 2011. ISBN 9783642241772. doi: 10.1007/978-3-642-24178-9_9. URL http://link.springer.com/10.1007/978-3-642-24178-9_9.

[6] Alireza Ahadipour, Mojtaba Mohammadi, and Alireza Keshavarz-Haddad. Statistical-Based Privacy-Preserving Scheme with Malicious Consumers Identification for Smart Grid. pages 1–9, apr 2019. URL http://arxiv.org/abs/1904.06576.

[7] Benny Applebaum, Haakon Ringberg, Michael J. Freedman, Matthew Caesar, and Jennifer Rexford. Collaborative, privacy-preserving data aggregation at scale. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6205 LNCS, pages 56–74. Springer Berlin Heidelberg, 2010. ISBN 3642145264. doi: 10.1007/978-3-642-14527-8_4. URL http://link.springer.com/10.1007/978-3-642-14527-8_4.

[8] Frederik Armknecht, Stefan Katzenbeisser, and Andreas Peter. Group homomorphic encryption: Characterizations, impossibility results, and applications. *Designs, Codes, and Cryptography*, 67(2):209–232, may 2013. ISSN 09251022. doi: 10.1007/s10623-011-9601-2. URL http://link.springer.com/10.1007/s10623-011-9601-2.

[9] Julia Belluz. Grindr is revealing its users' HIV status to third-party companies, 2018. URL https://www.vox.com/2018/4/2/17189078/grindr-hiv-status-data-sharing-privacy.

[10] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity, 2018. ISSN 1746-8094. URL https://eprint.iacr.org/2018/046.pdf.

[11] Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnes, and Bernhard Seefeld. PROCHLO: Strong Privacy for Analytics in the Crowd. In *SOSP 2017 - Proceedings of the 26th ACM Symposium on Operating Systems Principles*, pages 441–459, New York, New York, USA, 2017. ACM Press. ISBN 9781450350853. doi: 10.1145/3132747.3132769. URL http://dl.acm.org/citation.cfm?doid=3132747.3132769.

[12] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: The SulQ framework. *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 128–138, 2005. doi: 10.1145/1065167.1065184.

[13] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 1175–1191, New York, New York, USA, 2017. ACM Press. ISBN 9781450349468. doi: 10.1145/3133956.3133982. URL http://dl.acm.org/citation.cfm?doid=3133956.3133982.

[14] Benedikt Bunz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short Proofs for Confidential Transactions and More. In *Proceedings - IEEE Symposium on Security and Privacy*, volume 2018-May, pages 315–334. IEEE, may 2018. ISBN 9781538643525. doi: 10.1109/SP.2018.00020. URL https://ieeexplore.ieee.org/document/8418611/.

[15] Janet Burns. ACLU Lawsuit Says Facebook's Targeted Job Ads Exclude Women, Older Men, 2018. URL https://www.forbes.com/sites/janetwburns/2018/09/20/aclu-suit-says-facebook-excluded-women-older-men-from-targeted-job-ads/.

[16] Ann Cavoukian. The Foundational Principles. Technical report, Information and Privacy Commissioner of Ontario, 2009. URL https://www.ipc.on.ca/wp-content/uploads/Resources/7foundationalprinciples.pdf.

[17] T.-H. Hubert Chan, Elaine Shi, and Dawn Song. Privacy-preserving stream aggregation with fault tolerance. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 7397 LNCS, pages 200–214. 2012. ISBN 9783642329456. doi: 10.1007/978-3-642-32946-3_15. URL http://link.springer.com/10.1007/978-3-642-32946-3_15.

[18] Delphine Christin. Privacy in mobile participatory sensing: Current trends and future challenges. *Journal of Systems and Software*, 116:57–68, 2016. ISSN 01641212. doi: 10.1016/j.jss.2015.03.067. URL http://dx.doi.org/10.1016/j.jss.2015.03.067.

[19] Henry Corrigan-Gibbs and Dan Boneh. Prio: Private, robust, and scalable computation of aggregate statistics. In *Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017*, pages 259–282, mar 2017. ISBN 9781931971379. URL http://arxiv.org/abs/1703.06255.

[20] Ivan Damgård and Mads Jurik. A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 1992, pages 119–136. 2001. ISBN 3540416587. doi: 10.1007/3-540-44586-2_9. URL http://link.springer.com/10.1007/3-540-44586-2_9.

[21] Matt Day, Giles Turner, and Natalia Drozdiak. Amazon Workers Are Listening to What You Tell Alexa, 2019. URL http://files/2800/is-anyone-listening-to-you-on-alexa-a-global-team-reviews-audio.html.

[22] Mark Doman, Stephen Hutcheon, Dylan Welch, and Kyle Taylor. China's frontier of fear, 2018. URL https://www.abc.net.au/news/2018-11-01/satellite-images-expose-chinas-network-of-re-education-camps/10432924.

[23] George Duncan. Engineering: Privacy by design. *Science*, 317(5842):1178–1179, 2007. ISSN 00368075. doi: 10.1126/science.1143464. URL https://www.esat.kuleuven.be/cosic/publications/article-1542.pdf.

[24] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4004 LNCS:486–503, 2006. ISSN 16113349. doi: 10.1007/11761679_29.

[25] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3876 LNCS:265–284, 2006. ISSN 03029743. doi: 10.1007/11681878_14.

[26] Costas Efthymiou and Georgios Kalogridis. Smart Grid Privacy via Anonymization of Smart Metering Data. In *2010 First IEEE International Conference on Smart Grid Communications*, pages 238–243. IEEE, oct 2010. ISBN 978-1-4244-6510-1. doi: 10.1109/smartgrid.2010.5622050. URL http://ieeexplore.ieee.org/document/5622050/.

[27] Bennett Eisenberg. On the expectation of the maximum of IID geometric random variables. *Statistics and Probability Letters*, 78(2):135–143, 2008. ISSN 01677152. doi: 10.1016/j.spl.2007.05.011.

[28] Zekeriya Erkin. Private data aggregation with groups for smart grids in a dynamic setting using CRT. In *2015 IEEE International Workshop on Information Forensics and Security, WIFS 2015 - Proceedings*, volume 30, pages 1–6. IEEE, nov 2015. ISBN 9781467368025. doi: 10.1109/WIFS.2015.7368584. URL http://ieeexplore.ieee.org/document/7368584/.

[29] Zekeriya Erkin and Gene Tsudik. Private computation of spatial and temporal power consumption with smart meters. In Feng Bao, Pierangela Samarati, and Jianying Zhou, editors, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 7341 LNCS, pages 561–577. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 9783642312830. doi: 10.1007/978-3-642-31284-7_33. URL http://link.springer.com/10.1007/978-3-642-31284-7_33.

[30] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. RAPPOR: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 1054–1067, New York, New York, USA, 2014. ACM Press. ISBN 9781450329576. doi: 10.1145/2660267.2660348. URL http://dl.acm.org/citation.cfm?doid=2660267.2660348.

[31] Flavio D. Garcia and Bart Jacobs. Privacy-friendly energy-metering via homomorphic encryption. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6710 LNCS, pages 226–238. Springer Berlin Heidelberg, 2011. ISBN 9783642224430. doi: 10.1007/978-3-642-22444-7_15. URL http://link.springer.com/10.1007/978-3-642-22444-7_15.

[32] Craig Gentry. Fully Homomorphic Encryption Using Ideal Lattices. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, volume 26, pages 169–178, New York, New York, USA, 2009. ACM Press. ISBN 9781605585062. doi: 10.1145/1536414.1536440. URL http://portal.acm.org/citation.cfm?doid=1536414.1536440.

[33] Megan Graham. Digital ad revenue in the US surpassed $100 billion for the first time in 2018, 2019. URL https://www.cnbc.com/2019/05/07/digital-ad-revenue-in-the-us-topped-100-billion-for-the-first-time.html.

[34] Graham Greenleaf. Global Table of Countries with Data Privacy Laws and Bills. *Privacy Laws & Business*, (January):14–26, 2019. URL https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2992986.

[35] Glenn Greenwald. NSA collecting phone records of millions of Verizon customers daily, 2013. URL http://www.theguardian.com/world/2013/jun/06/nsa-phone-records-verizon-court-order.

[36] Seda Gürses, Carmela Troncoso, and Claudia Diaz. Engineering privacy by design reloaded. *Amsterdam Privacy Conference*, pages 1–21, 2015. URL https://iapp.org/resources/article/engineering-privacy-by-design-reloaded/.

[37] Hao Jiang, Wenjian Luo, and Zhenya Zhang. A privacy-preserving aggregation scheme based on immunological negative surveys for smart meters. *Applied Soft Computing Journal*, 85:105821, dec 2019. ISSN 15684946. doi: 10.1016/j.asoc.2019.105821. URL https://doi.org/10.1016/j.asoc.2019.105821.

[38] David Kaye. UN expert decries conviction of journalists, urges reversal, 2020. URL https://www.ohchr.org/EN/NewsEvents/Pages/DisplayNews.aspx?NewsID=25962{&}LangID=E.

[39] Fabian Knirsch, Dominik Engel, and Zekeriya Erkin. A fault-tolerant and efficient scheme for data aggregation over groups in the smart grid. In *2017 IEEE Workshop on Information Forensics and Security, WIFS 2017*, volume 2018-Janua, pages 1–6. IEEE, dec 2017. ISBN 9781509067695. doi: 10.1109/WIFS.2017.8267646. URL http://ieeexplore.ieee.org/document/8267646/.

[40] Ivan Krastev. Democracy Disrupted. Technical Report July, Information Commissioner's Office, 2014. URL https://ico.org.uk/media/action-weve-taken/2259369/democracy-disrupted-110718.pdf.

[41] Klaus Kursawe. Some Ideas on Privacy Preserving Meter Aggregation. *Radboud Universiteit Nijmegen, Tech. Rep. ICIS–R11002*, pages 1–15, 2010.

[42] Klaus Kursawe, George Danezis, and Markulf Kohlweiss. Privacy-friendly aggregation for the smart-grid. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6794 LNCS, pages 175–191. Springer Berlin Heidelberg, 2011. ISBN 9783642222627. doi: 10.1007/978-3-642-22263-4_10. URL http://link.springer.com/10.1007/978-3-642-22263-4_10.

[43] Ewen MacAskill, Julian Borger, Nick Hopkins, Nick Davies, and James Ball. GCHQ taps fibre-optic cables for secret access to world's communications, 2013. ISSN 0261-3077. URL http://www.theguardian.com/uk/2013/jun/21/gchq-cables-secret-world-communications-nsa.

[44] Amber Macintyre. Who's Working for Your Vote?, 2018. URL https://ourdataourselves.tacticaltech.org/posts/whos-working-for-vote/.

[45] Mimezine. WiGLE, 2001. URL https://wigle.net/.

[46] Andrés Molina-Markham, Prashant Shenoy, Kevin Fu, Emmanuel Cecchet, and David Irwin. Private memoirs of a smart meter. In *BuildSys'10 - Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, pages 61–66, New York, New York, USA, 2010. ACM Press. ISBN 9781450304580. doi: 10.1145/1878431.1878446. URL http://portal.acm.org/citation.cfm?doid=1878431.1878446.

[47] Eduardo Morais, Tommy Koens, Cees van Wijk, and Aleksei Koren. A survey on zero knowledge range proofs and applications. *SN Applied Sciences*, 1(8):1–17, 2019. ISSN 2523-3963. doi: 10.1007/s42452-019-0989-z. URL https://doi.org/10.1007/s42452-019-0989-z.

[48] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *Proceedings - IEEE Symposium on Security and Privacy*, pages 111–125. IEEE, may 2008. ISBN 9780769531687. doi: 10.1109/SP.2008.33. URL http://ieeexplore.ieee.org/document/4531148/.

[49] Netflix. Privacy Statement, 2016. URL https://help.netflix.com/legal/privacy.

[50] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 1592, pages 223–238. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999. ISBN 3540658890. doi: 10.1007/3-540-48910-X_16. URL http://link.springer.com/10.1007/3-540-48910-X_16.

[51] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 576 LNCS, pages 129–140. Springer Berlin Heidelberg, Berlin, Heidelberg, 1992. ISBN 9783540551881. doi: 10.1007/3-540-46766-1_9. URL http://link.springer.com/10.1007/3-540-46766-1_9.

[52] Privacy International. How do data companies get our data?, 2018. URL https://privacyinternational.org/.

[53] Privacy International. Your mental health for sale. Technical Report September, Privacy International, 2019. URL https://privacyinternational.org/node/3193.

[54] PwC. Consumer Intelligence Series: Protect.me. Technical report, PwC, 2017. URL https://www.pwc.com/us/en/services/consulting/library/consumer-intelligence-series/cybersecurity-protect-me.html.

[55] Cooper Quintin. The Pregnancy Panopticon. Technical report, EFF, 2017. URL https://www.eff.org/wp/pregnancy-panopticon.

[56] Vibhor Rastogi and Suman Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 735–746, New York, New York, USA, 2010. ACM Press. ISBN 9781450300322. doi: 10.1145/1807167.1807247. URL http://portal.acm.org/citation.cfm?doid=1807167.1807247.

[57] Raquel Rolnik. Report of the Special Rapporteur on adequate housing as a component of the right to an adequate standard of living, on the right to non-discimination in this context - A/HRC/22/46, 2012. URL https://www.ohchr.org/EN/NewsEvents/Pages/DisplayNews.aspx?NewsID=25033{&}LangID=E.

[58] Bruce Schneier. The Public Good Requires Private Data, 2020. URL https://www.schneier.com/essays/archives/2020/05/the_public_good_requ.html.

[59] Adi Shamir. How to Share a Secret. *Communications of the ACM*, 22(11):612–613, nov 1979. ISSN 15577317. doi: 10.1145/359168.359176. URL http://portal.acm.org/citation.cfm?doid=359168.359176.

[60] Elaine Shi, T.-H. Hubert Chan, Eleanor Rieffel, Richard Chow, and Dawn Song. Privacy-Preserving Aggregation of Time-Series Data. *Annual Network & Distributed System Security Symposium (NDSS)*, 2011.

[61] Zhiguo Shi, Ruixue Sun, Rongxing Lu, Le Chen, Jiming Chen, and Xuemin Sherman Shen. Diverse Grouping-Based Aggregation Protocol with Error Detection for Smart Grid Communications. *IEEE Transactions on Smart Grid*, 6(6):2856–2868, nov 2015. ISSN 19493053. doi: 10.1109/TSG.2015.2443011. URL http://ieeexplore.ieee.org/document/7169604/.

[62] Ruixue Sun, Zhiguo Shi, Rongxing Lu, Min Lu, and Xuemin Shen. APED: An efficient aggregation protocol with error detection for smart grid communications. *GLOBECOM - IEEE Global Telecommunications Conference*, pages 432–437, 2013. doi: 10.1109/GLOCOM.2013.6831109.

[63] Ted Szymanski. A fiber optic hypermesh for SIMD/MIMD machines. In *Proceedings SUPERCOMPUTING '90*, pages 710–719. IEEE Comput. Soc. Press, 1990. ISBN 0818620560. doi: 10.1109/superc.1990.130091. URL http://ieeexplore.ieee.org/document/130091/.

[64] Ted Szymanski. "Hypermeshes": Optical Interconnection Networks for Parallel Computing. *Journal of Parallel and Distributed Computing*, 26(1):1–23, apr 1995. ISSN 07437315. doi: 10.1006/jpdc.1995.1043. URL https://linkinghub.elsevier.com/retrieve/pii/S074373158571043X.

[65] Larry D. Wittie. Communication Structures for Large Networks of Microcomputers. *IEEE Transactions on Computers*, C-30(4):264–273, apr 1981. ISSN 00189340. doi: 10.1109/TC.1981.1675774. URL http://ieeexplore.ieee.org/document/1675774/.

[66] Tom Wright and Peter Hustinx. Privacy-Enhancing Technologies: The Path to Anonymity. Technical report, Information and Privacy Commissioner/Ontario Canada / Registratiekamer The Netherlands, Toronto, Ontario, Canada / Rijswijk, Netherlands, 1995. URL http://www.ontla.on.ca/library/repository/mon/10000/184530.pdf.

[67] Lei Yang and Fengjun Li. Detecting false data injection in smart grid in-network aggregation. In *2013 IEEE International Conference on Smart Grid Communications, SmartGridComm 2013*, pages 408–413. IEEE, oct 2013. ISBN 9781479915262. doi: 10.1109/SmartGridComm.2013.6687992. URL http://ieeexplore.ieee.org/document/6687992/.