Graph-Based Reconstruction in Summation Sequences

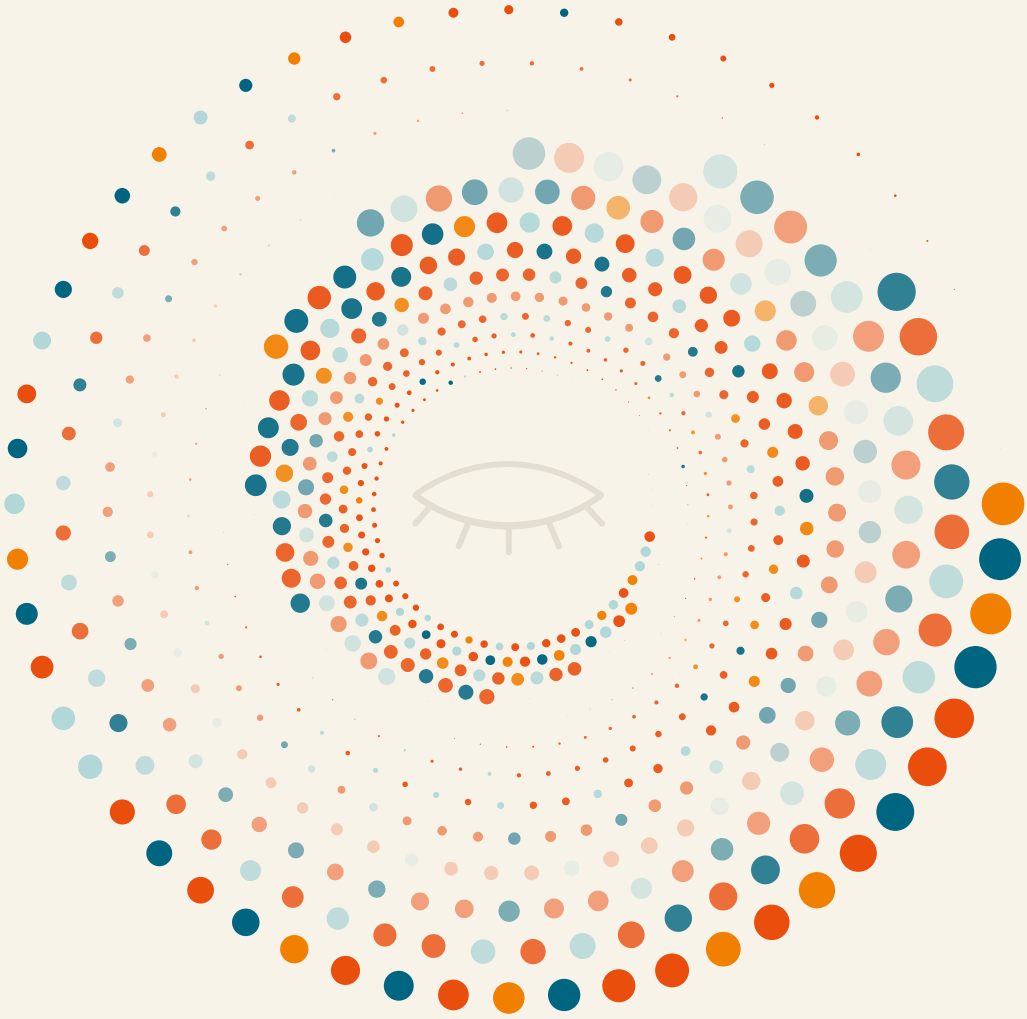
Dekker, Florine W.

DOI
10.4233/uuid:6b567512-a2f0-4588-a3c6-199d8bbb8a79

Publication date
2025

Document Version
Final published version

Citation (APA)
Dekker, F. W. (2025). *Graph-Based Reconstruction in Summation Sequences*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:6b567512-a2f0-4588-a3c6-199d8bbb8a79



GRAPH-BASED RECONSTRUCTION IN SUMMATION SEQUENCES

Florine W. Dekker

Graph-Based Reconstruction in Summation Sequences

Graph-Based Reconstruction in Summation Sequences

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus prof. dr. ir. T.H.J.J. van der Hagen,
Chair of the Board for Doctorates,
to be defended publicly on
Thursday, 18 September 2025 at 15:00

by

Florine Willemke Dekker

Master of Science in Computer Science
Delft University of Technology, the Netherlands
born in Zwijndrecht, the Netherlands

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

Rector Magnificus	*chairperson*
prof. dr. M. Conti	Delft University of Technology / University of Padua, Italy, *promotor*
dr. Z. Erkin	Delft University of Technology, *promotor*
Independent members:	
prof. dr. M. T. J. Spaan	Delft University of Technology
prof. dr. S. Roos	University of Kaiserslautern–Landau, Germany
prof. dr. ir. P. J. M. Veugen	University of Twente
dr. ir. L. A. M. Schoenmakers	Eindhoven University of Technology
prof. dr. G. Smaragdakis	Delft University of Technology
prof. dr. ir. G. J. P. M. Houben	Delft University of Technology, *substitute*

Cover design: © 2025 Marilou Maes

Printed by: Proefschriftspecialist

ISBN: 978-94-6518-090-8

The primary role of cryptography in human rights struggles is not to exit from society, but to provide a robust, temporary shield for those who would reform it.

Cory Doctorow

Contents

English Summary

In a world of increasing threats from monopolies and oligarchies, people are increasingly looking for ways to protect their privacy. Isolating oneself from the world may be tempting, but there is a collective benefit to the processing of sensitive data. For example, hospitals use patient data to improve treatments, energy companies use power consumption data to predict grid usage, and governments can address inequality only if they measure it.

Privacy-enhancing technologies promise to close the apparent gap between privacy and utility. They provide a cryptographic solution by which statistics can be calculated without exposing individual inputs. In the world envisioned here, people gain the benefits of sharing data without exposing themselves to potential abuse.

Though solutions to societal problems are rarely if ever purely technical, this dissertation is concerned only with the technical. Specifically, with privacy-preserving summation, a protocol allowing users to learn the sum of their inputs without anyone learning the individual value of anyone else. While it may sound restrictive to focus only on summation, this is sufficient to achieve complex operations including principal component analysis, singular-value decomposition, and decision tree classifications.

In this dissertation, I provide novel methods for enforcing input and output validity in privacy-preserving summation, describe how running multiple summations in parallel leads to reconstruction attacks, and propose and evaluate countermeasures based on distributed short-cycle removal.

Validation of inputs and outputs is enforced through extensions, which can be added to any privacy-preserving summation scheme without sacrificing confidentiality. The first extension ensures that the individual pieces of data being summed over each fall within a specified numeric range. The second extension allows others to ensure that the sum published by the aggregator actually corresponds to the inputs.

Reconstruction attacks are an inherent risk when multiple summations run in sequence, regardless of the implementation of the summation protocol. When users obtain the sum $A + B$, one cannot learn either A or B due to the summation's privacy-preserving guarantees. However, if users subsequently also learn $A + B + C$, then anyone can infer C from the difference of the sums.

Understanding how and when reconstruction is possible is not trivial, especially as the numbers of variables and equations grows large. In this dissertation, I show that representing summations as a graph reveals that reconstruction coincides with the graph's cycles. In other words, removing cycles prevents reconstruction attacks. Therefore, I propose a decentralised protocol for removing short cycles. Finally, I evaluate the impact that restricting valid summation has on distributed averaging, and find that though the effect is largely negative, this can mostly be ameliorated through a subsequent greedy repair algorithm.

Nederlandse Samenvatting

In een wereld van toenemende bedreigingen door monopoliën en oligarchieën zijn mensen steeds meer op zoek naar bescherming van hun privacy. Zelfisolatie klinkt verleidelijk, maar de verwerking van persoonlijke gegevens heeft wel degelijk een collectief voordeel. Ziekenhuizen gebruiken bijvoorbeeld patiëntgegevens om behandelingen te verbeteren, energiebedrijven bekijken stroomverbruik om overbelasting te voorspellen, en overheden kunnen ongelijkheid alleen bestrijden als ze het meten.

Privacyverbeterende technologieën beloven de ogenschijnlijke kloof tussen privacy en nut te dichten. Ze bieden een cryptografische oplossing om statistieken te berekenen zonder individuele waarden te onthullen. In de voorgestelde wereld profiteren mensen van het delen van gegevens zonder zichzelf bloot te stelen aan mogelijk misbruik.

Hoewel oplossingen voor maatschappelijke problemen zelden tot nooit puur technisch zijn, gaat dit proefschrift alleen over de techniek. Specifiek gaat het over privacybehoudende sommatie, een protocol om een som te berekenen zonder dat individuele invoerwaarden te achterhalen zijn. Sommatie is een krachtige berekening, voldoende om complexe berekeningen als hoofdcomponentenanalyse, singulierewaardenontbinding, en beslissingsboomclassificaties te implementeren.

In dit proefschrift geef ik nieuwe methoden voor het afdwingen van in- en uitvoervaliditeit in privacybehoudende sommatie, beschrijf ik hoe het parallel uitvoeren van meerdere sommaties leidt tot reconstructieaanvallen, en stel ik tegenmaatregelen gebaseerd op gedistribueerde verwijdering van korte cycli voor en evalueer ik deze.

Validatie van in- en uitvoer werkt door middel van uitbreidingen bovenop arbitraire bestaande privacybeschermende sommatieschema's, zonder de vertrouwelijkheid op te offeren. De ene uitbreiding verzekert dat de individuele invoerwaarden elk binnen een gespecificeerd numeriek bereik vallen. De andere uitbreiding verifiëert dat de door de aggregator gepubliceerde som daadwerkelijk overeenkomt met de invoer.

Reconstructieaanvallen zijn een inherent risico wanneer meerdere sommaties na elkaar worden uitgevoerd, ongeacht de implementatie van de sommatie. Wanneer gebruikers $A + B$ leren, kan niemand A of B te weten komen vanwege de garanties van de privacybehoudende sommatie. Als gebruikers echter vervolgens ook $A + B + C$ te weten komen, kan iedereen C afleiden uit het verschil van de sommen.

Begrijpen hoe en wanneer reconstructie mogelijk is, is niet triviaal, vooral als het aantal variabelen en vergelijkingen groeit. In dit proefschrift laat ik zien dat de graafrepresentatie van de sommen onthult dat reconstructie samenvalt met de cycli. Met andere woorden, het verwijderen van cycli voorkomt reconstructie-aanvallen. Daarom stel ik een gedecentraliseerd protocol voor om korte cycli te verwijderen. Tot slot meet ik de invloed die het beperken van sommaties heeft op gedistribueerde middeling, en ontdek ik dat dit grotendeels negatief is, maar grotendeels kan worden voorkomen met een daaropvolgend gretig reparatiealgoritme.

Part I

Prologue

Chapter 1

Introduction

Spending time alone is important. When you are alone, you are free to do what you want, and need not suffer the judgment of others. At these times, your thoughts flow more freely. You have the privilege to explore your opinions and deepen your knowledge. Occasionally being alone is vital to grow as a person.

On the other hand, collaborating is important to broaden your knowledge. When you observe others, you may see methods or solutions you had not thought of yourself. When others observe you, they may point out what they think you do well or what you do wrong, and may both learn from the experience. Moreover, by observing large groups of people, you can learn about collective or aggregate behaviours.

Unfortunately, it is not easy to maintain a balance between privacy and sharing. If you discuss intimate details with someone, they may use that information against you, or share it with others. Even if your conversation partner is trustworthy, your conversation may be overheard by a malicious eavesdropper. And even if you are sure no one can overhear you, you will have to trust your conversation partner never to write anything down, lest their notes be stolen.

The situation is not dissimilar in the digital realm: Once you share information, you no longer have any control over what happens to that data, at least not beyond the promises of those you shared it with. Even if you do not explicitly share anything, others can learn about you by seeing which websites you visit, and what you do there. Which news articles pique your interest? What kinds of products do you buy? Which routes do you use in your navigation app? What symptoms and ailments do you enter in search engines? Private information inferred from your behaviour is continuously collected and is used to target you [BN15, DT19, New21]. Even if companies promise not to sell such data, temptations of increased profit regularly lead to broken promises [FTC19, Fun21, Zia23, Fun24]. Though disengaging from the digital realm completely would prevent others from learning about you (except insofar as others share information about you), this is often not an option [Rai18, CD21].

In an ideal world, we could safeguard our data against undesired computations: For example, if we send a query to a search engine, then our computers should ensure that our query is used only to search for information, and not to infer private information about us. In other words, we want *technologically-enforced purpose limitation* [TL24]. One important technology to achieve such purpose limitation is Multi-Party Computation (MPC), a family of cryptographic techniques for computing over private data. After the participants have chosen the functions they want to compute, and agreed upon an implementation to achieve it, MPC presents participants with the desired outputs, while ensuring the inputs remain completely confidential. For example, using MPC, participants can calculate the sum of their ages, without anyone learning the age of anyone else. To prove that an MPC design guarantees confidentiality of the inputs, the implementation is compared with its *ideal functionality*: a high-level black-box description of the desired calculations. On the one hand we have the real-world MPC

implementation with all its nitty-gritty implementation details regarding synchronisation, availability, integrity, and so on; and on the other hand we have an ideal-world trusted third party (who is indeed *actually* trusted) who simply looks at everyone's inputs and gives the desired outputs. If we can prove that participants can infer the exact same information about others' inputs in both worlds, then we can conclude that the real-world MPC implementation achieves optimal levels of confidentiality.

The seminal MPC paper, from 1979, describes a method to play "mental poker", which is like regular poker, except the players can only communicate by telephone, and neither player is above cheating [SRA79, GM82]. Subsequent works have focused on improving the security definitions and models, providing evermore elaborate methods and protocols for achieving all kinds of functionalities [Lin20]. The first large-scale real-world deployment of MPC was at a beet auction in 2009 [Bog+09], where production contracts needed to be allocated to farmers. Farmers could bid on these contracts, but wanted their bids to remain private so as to not reveal their economic positions to competitors and customers. The auction was first represented as an ideal functionality, then implemented as an MPC protocol, proven to be secure, and finally executed at the auction, with the protocol completing its calculations in 25 minutes. Since then, MPC has seen trials and deployments in several other niche cases, including the calculating of statistics in vulnerable populations [Lap+18] and collaborations between government offices [vEgm+21, Wor+20]. Currently, *any* computation can be implemented with MPC [Lin20].

The cost of MPC is not negligible, however. As the number of participants increases and the computations grow in complexity, so do the computation and communication requirements. This has not stopped the increasing interest in MPC [AOdR22, BD20]. Companies see MPC as a valuable tool for retaining their business models in the face of increasing data breach risks and tightening privacy regulations, as MPC allows these companies to process encrypted privacy-sensitive data (seemingly) without violating privacy. Governments see MPC as a way to process sensitive census data, and to get different governmental departments to collaborate in ways that were previously thought impossible. Perhaps, with increased attention comes increased funding, allowing improvements of the underlying primitives, new models of interactivity, and extremely efficient MPC compilers [NSTC23].

1.1 Reconstruction Attacks

MPC is not without its flaws and pitfalls [STPO22, Can+22]. One potential pitfall surrounds the definition of the ideal functionality itself. While MPC guarantees that nothing leaks beyond the intended output, participants may still be able to infer information from multiple outputs. For example, let's say that three people—Alice, Bob, and Charlie—learn through their use of MPC that their cumulative age is 120. Afterwards, Alice and Bob run another MPC protocol again, this time without Charlie, and learn that *their* cumulative age is 80. Despite the privacy-preserving guarantees of the underlying methods, Alice and Bob can infer from the difference that Charlie must be 40 years old. Clearly, the privacy guarantees of singular collaborations do not extend to sequences of collaborations.

The type of attack described above is known as a *reconstruction attack*, and has been known since at least the early 1970s [Fel72], before the first literature on MPC.

While census offices have been publishing aggregate information about the general population for a long time, it was around that time that computers had become powerful enough to allow users to write their own queries. To preserve the privacy of the people whose information was stored in these databases, records pertaining to only a small number of individuals had to be forbidden. However, exactly as in the aforementioned case of Alice, Bob, and Charlie, the outputs of multiple non-forbidden queries may still allow a querier to infer private information. Various solutions for the problem have been proposed, including random sampling [Den80], query auditing [CÖ82], and perturbation [Dwo06]. For summation queries specifically, it is sufficient to keep a history of queries, and forbid subsequent queries if they could be combined to infer private information [Chi78, WWJ02].

As noted earlier, any algorithm can be translated to a corresponding MPC variant. Therefore, it is entirely possible to do auditing in MPC, and thereby prevent reconstruction attacks. However, this does not mean that such a protocol is practical. In a peer-to-peer system, there is no single party that can keep track of queries. Sometimes, a summation on one side of the network may allow users on the other side of the network to infer private information. Therefore, in a naive translation of the query-tracking mechanism to MPC, the entire (hidden) database of past queries must continuously be communicated to all users before any decision can be made, which would be excessively costly.

1.2 Composability and Disclosure

The problem of reconstruction attacks is superficially similar to the problem of composability in MPC. Composability is the guarantee that the implementation details of an MPC protocol do not cause issues even when the protocol is executed multiple times. A non-composable protocol will be secure when used in isolation, but when two instances run in parallel, adversaries can gain an undue advantage by using information from one instance in the other. There are numerous frameworks to model composability, including universal composability [Can01], constructive composability [Mau11], and reactive simulatability [BPW07].

However, composability is unrelated to reconstruction attacks. Even if Alice, Bob, and Charlie are given a summation protocol that is composable in every relevant way, the previously mentioned calculations will still allow Bob and Charlie to learn Alice's age. To understand this, recall that confidentiality is shown in MPC by comparing what participants learn in the real protocol with what they learn in the ideal protocol. If Alice, Bob, and Charlie learn that $A + B + C = 120$ and that $B + C = 80$, they can infer that $A = 40$, even if they ask a trusted third party to perform the summations. Thus, while composability ensures that participants learn nothing beyond the intended outputs, it cannot judge whether the intended outputs were well-chosen.

As the number of real-world MPC deployments increases, so does the chance of designers combining sub-protocols in ways that are vulnerable to reconstruction. Creating well-defined formal notions of this type of leakage is thus an important next step in ensuring the correctness of MPC systems. We note that while differential privacy [Dwo06] is sufficient to prevent information disclosure, its guarantees are probabilistic, its outputs are necessarily distorted, and its privacy-utility tradeoff is infamously difficult to understand and calibrate in real-world settings [CT13, JE19].

Several recent works investigated information-theoretic bounds on information disclosure in multi-party computation [BBZ24, AH20a, AH20b]. These works measure the amount of information that outputs reveal about the inputs, and carry over these estimates into subsequent computations to accurately estimate the total privacy loss in a pipeline of computations. However, the above works are all limited to specific computations, adversarial models, or network structures. Many interesting avenues remain unexplored, such as generalisation to other security models, translating findings to and from differential privacy, and the development of practical tools and guidelines for protocol designers to quantify leakage with.

1.3 Research Objectives

With increasing efficiency, public interesting in MPC is growing. MPC has an increasingly sound and complete framework to prove that desired ideal functionalities are implemented correctly. However, there are no comprehensive frameworks that determine whether ideal functionalities should be desired. To prevent faulty MPC deployments, it is vital that such frameworks are developed. The goal of this dissertation has been to contribute to the collective knowledge surrounding information disclosure in MPC. Specifically, the goal has been to determine the graph-theoretic conditions under which full input leakage occurs after a sequence of summation operations.

The research presented in this dissertation was conducted in a curiosity-driven manner. There has not been a singular research question to be answered; instead, a chain of research questions emerged along the way. The initial goal was to, one way or another, contribute to the developing field of federated learning by creating privacy-preserving summation protocols for various security models. After realising that the aforementioned reconstruction attacks would undo any and all privacy guarantees of these summation protocols, the research objective became to determine the exact circumstances under which reconstruction attacks pose a threat. Finally, we moved to evaluate the suitability of the developed defence mechanism for real-world protocols that rely on privacy-preserving summation.

1.4 Contributions

The scientific contributions of this dissertation have been structured into two parts, and are finalised by a discussion.

Part II: Privacy-Preserving Summation

We look at the summation primitive and provide two extensions for existing protocols.

Chapter 2: Probabilistic range validation [DE21]. We enhance privacy-preserving summation, a classical MPC protocol wherein a central party learns the sum of the participants' values (and nothing else). Note that summation alone is a sufficient primitive to implement more complex operations such as principal component analysis, singular-value decomposition, and decision tree classifications, simply by writing the inputs as aggregate-sum queries [Blu+05]. We describe a protocol by which the aggregator can validate that all users' numerical inputs are within a desired range. Unlike existing works, our protocol does not rely on complex zero-knowledge proofs, instead probabilistically detecting disallowed values with a likelihood proportional to

the value's transgression. The simplicity of our scheme means that validation requires no cryptographic operations, and does not require any hardness assumptions. We provide an in-depth analysis of our scheme's detection rate; though the scheme's probabilistic nature requires assuming a distribution to model the users' values under, our analysis should be relatively easy to extend to other distributions. Furthermore, we formally prove the scheme's correctness, argue that confidentiality holds, and calculate the asymptotic communication and computation complexity and compare this to related works.

Chapter 3: MPVAS [Pal+24]. We propose MPVAS, a protocol that allows the aggregator in (centralised) privacy-preserving summation to publicly prove that the sum it publishes was correctly calculated. MPVAS is an add-on: MPVAS creates the proof, but the summation should be done by some other (privacy-preserving) summation protocol. Our adversarial model is the strongest model yet in literature: We guarantee confidentiality and unforgeability even when an actively malicious aggregator colludes with multiple actively malicious users, and anyone (including the adversaries) can register as a verifier. We extend our protocol in several ways: MPVAS+ reduces runtime complexity at the cost of breaking the non-forgery guarantee with low probability, MPVAS-IV ensures robustness against malformed inputs, and MPVAS-UD allows users to temporarily drop out. We calculate the asymptotic communication complexity of our protocol and compare this to related works. Moreover, based on our implementations of all variants of MPVAS, we conclude that they achieve efficient runtimes, even without MPVAS+. Finally, we argue for each of the protocols that they achieve confidentiality of the inputs and that signatures cannot be forged.

Part III: Reconstruction Attacks

We show that it is feasible for non-malicious adversaries to infer private data purely from the outputs of (ideal) summations. We model the attack in detail in its algebraic form, derive the corresponding (hyper)graph-theoretic representation, and ultimately design a graph-based criterion to classify faulty sequences of summations with. We apply our criterion to peer-to-peer networks to prevent the attacks from occurring in the first place, and validate its suitability for distributed averaging protocols.

Chapter 4: Reconstruction attacks [DEC25c]. We empirically show that reconstruction attacks are feasible after running multiple privacy-preserving summation protocols, and investigate the graph-theoretic conditions that underlie these attacks. We mathematically derive the necessary (but not necessarily sufficient) condition that the graph contains cycles. More specifically, we prove that the graph's girth (which is the length of its shortest cycle) determines a lower bound on the number of adversaries required to perform a reconstruction attack. Therefore, graph stretching (which is the removal of short cycles) is sufficient to prevent reconstruction attacks. However, we also note that graph stretching is typically a drastic change to the network topology, and may significantly affect the convergence properties of distributed algorithms running on top. We investigate this impact and propose methods to reduce it in Chapter 6.

Chapter 5: Cycle detection [Jen+25]. We propose a peer-to-peer protocol for detecting nearby cycles without revealing the topology to users outside the cycle. Though we present our protocol in the context of detecting cycles in monetary transaction

networks, where cycles are indicators of money laundering, our protocol is applicable to any peer-to-peer network. Our algorithm works by flooding messages through the network, and inferring the presence of a cycle when the flood reaches back to the initiator. Messages are re-randomised each time they are forwarded to ensure that they are not linkable. However, messages have a hidden structure so that when they reach back to the initiator, the initiator recognises the message and can infer the presence of a cycle. The initiator can then start a simple sub-protocol to determine which nodes are in that cycle. The aforementioned hidden structure is a novel forwarding-based unlinkable key agreement protocol, which may be of independent interest. We calculate our protocol's asymptotic communication and computation complexities, compare these to related works, and, based on measurements of an implementation of our protocol, conclude that our protocol achieves practical runtimes for graphs without high-degree nodes. Finally, we formally model the knowledge that each node or collusion of nodes obtains, and then informally argue that nodes cannot unduly learn about the topology beyond (self-)loops.

Chapter 6: Optimal stretching [DEC25a]. We conduct a thorough empirical study of the effect that graph stretching has on the convergence speed of distributed averaging protocols. While existing literature has considered short-cycle removal, and has studied the relation between topology and convergence speed, these results are insufficient to describe the relation between girth and convergence speed. We introduce the *optimal graph stretching problem*, which is the task of modifying a graph such that convergence speed is optimal, while keeping girth above a certain minimum. We propose an algorithm that enforces the problem's constraints before heuristically optimising for the minimisation target. We empirically validate the performance of our algorithm on thousands of random graphs, comparing multiple stretching methods, various convergence speed heuristics, and several widely used graph topologies. We confirm the finding in Chapter 4 that stretching severely affects convergence speed. However, we additionally observe that this effect is mostly caused by the removal of edges, and the reintroduction of edges compensates for a large part of the lost convergence speed. We conclude that graph stretching is a feasible reconstruction prevention method, and the negative consequences for convergence speed can largely be overcome using a subsequent heuristic optimisation procedure.

***Part IV*: Discussion**

We summarise and discuss the results of this dissertation. We embed these achievements in the greater picture, and look at open problems and potential future directions.

1.5 About This Dissertation

Format. This dissertation is an anthology of the scientific works I have written during my PhD. These works have been included as-is without significant modifications. Consequently, its chapters are self-contained and may use conflicting notation or reintroduce concepts. Even so, all chapters contain minor modifications, for example to rectify errors, improve figures, or eliminate typographical widows and orphans.

Works. This dissertation includes the following scientific works. The following list additionally details my contributions to each work.

1. **Florine W. Dekker** and Zekeriya Erkin. "Privacy-preserving data aggregation with probabilistic range validation". In: *ACNS 2021: Proceedings of the 19th International Conference on Applied Cryptography and Network Security*. Volume 12727. Lecture Notes in Computer Science. 2021, pages 79–98. DOI: 10.1007/978-3-030-78375-4_4.

   Included as Chapter 2. Based on my master's thesis. Written by me, with editing guidance by my promotors.

2. Marco Palazzo, **Florine W. Dekker**, Alessandro Brighente, Mauro Conti and Zekeriya Erkin. "Privacy-preserving data aggregation with public verifiability against internal adversaries". In: *USENIX Security 2024: Proceedings of the 33rd USENIX Security Symposium*. 2024. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/palazzo.

   Included as Chapter 3. Based on the master's thesis of and in collaboration with Marco Palazzo, whom I supervised; and with the support of Alessandro Brighente. Palazzo and I independently identified and cooperatively resolved several crucial security flaws in the thesis version, and jointly designed MPVAS-UD. Additionally, I wrote or rewrote all parts of the work, performed the literature survey, performed the statistical analysis of MPVAS+, and guided the asymptotic complexity analysis. A small amount of editing guidance was provided by my promotors.

3. **Florine W. Dekker**, Zekeriya Erkin and Mauro Conti. "Topology-based reconstruction prevention for decentralised learning". In: *Proceedings on Privacy Enhancing Technologies* 2025.1 (2025), pages 553–566. DOI: 10.56553/POPETS-2025-0030.

   Included as Chapter 4. Written by me, with editing guidance by my promotors.

4. Juno Jense, **Florine W. Dekker**, Zekeriya Erkin and Mauro Conti. *Privacy-preserving peer-to-peer cycle detection*. 2025. In preparation.

   Included as Chapter 5. Based on the master's thesis of and written in collaboration with Juno Jense, whom I supervised. Jense and I subsequently jointly resolved several minor security flaws in the protocols. I wrote or rewrote all parts of the work and performed the security analysis.

5. **Florine W. Dekker**, Zekeriya Erkin and Mauro Conti. *Optimal graph stretching for distributed averaging*. 2025. arXiv: 2504.10289.

   Included as Chapter 6. Written by me.

Artifacts. The software artifacts underlying all results presented in this dissertation have been published. Some results have also been reproduced by independent parties.

1. **Florine W. Dekker** and Zekeriya Erkin. *Source code underlying the publication: Privacy-preserving data aggregation with probabilistic range validation.* 15th May 2025. DOI: 10.4121/b9db276f-5522-4986-9d98-e9710134fd71.v1.

   Supports Chapter 2. Written by me.

2. Marco Palazzo and **Florine W. Dekker**. *Source code underlying the publication: Privacy-preserving data aggregation with public verifiability against internal adversaries.* 11th June 2025. DOI: 10.4121/56552cc8-7ebf-46ce-a6e0-668dd6065eb2.v1.

   Supports Chapter 3. Written by Palazzo, and subsequently rewritten by me to ensure understandability and reproducibility.

3. **Florine W. Dekker**, Zekeriya Erkin and Mauro Conti. *Source code underlying the publication: Topology-based reconstruction prevention for decentralised learning.* 13th Jan. 2025. DOI: 10.4121/21572601.v2.

   Supports Chapter 4. Written by me. All results reproduced independently by PoPETS artifact review committee [DEC25c].

4. Juno Jense and **Florine W. Dekker**. *Source code underlying the publication: Privacy-preserving peer-to-peer cycle detection.* 17th June 2025. DOI: 10.4121/d23e6d7d-15d9-4c83-86de-5a3fc1fd5aa6.v1.

   Supports Chapter 5. Written by Jense, and subsequently edited by me to ensure reproducibility.

5. **Florine W. Dekker**. *Source code underlying the publication: Optimal graph stretching for distributed averaging.* 16th June 2025. DOI: 10.4121/e64c61d3-deb5-4aad-af60-92d92755781f.v3.

   Supports Chapter 6. Written by me. All results reproduced independently by CODECHECK [UQ25].

Part II

Privacy-Preserving Summation

Chapter 2

Privacy-Preserving Aggregation with Probabilistic Range Validation

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 these 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 a probabilistic hypergraph-based detection algorithm with which the aggregator can quickly pinpoint malicious users. Furthermore, our protocol is robust to user dropouts and, apart from the setup phase, it is non-interactive.

Based on: **Florine W. Dekker** and Zekeriya Erkin. "Privacy-preserving data aggregation with probabilistic range validation". In: *ACNS 2021: Proceedings of the 19th International Conference on Applied Cryptography and Network Security*. Volume 12727. Lecture Notes in Computer Science. 2021, pages 79–98. DOI: 10.1007/978-3-030-78375-4_4.

2.1 Introduction

Data aggregation gives us many valuable insights into the real world in the form of machine learning [Bon+17], participatory sensing [Bur+06], software telemetry [FPE16, Bit+17], and smart metering [LeM+07]. Although the usefulness of these methods depends on the amount of available data, privacy concerns make users reluctant to share their sensitive data with a third party [GJ10, Chr16]. This poses a significant threat to the viability of large-scale data analysis.

2

To overcome this problem, Privacy-preserving Data Aggregation (PDA) protocols have been proposed which allow an aggregator to calculate statistics on privacy-sensitive data without being able to determine private values. There are various ways to achieve this. For example, several proposals use techniques such as homomorphic encryption [GJ10, ET12] and secret sharing [Kur10, Erk15] to guarantee that user contributions cannot be decrypted unless they have been aggregated. Other proposals use differential privacy [RN10, ÁC11, Shi+11] to ensure that the connection between the observed value and the actual value is perturbed. Either way, PDA protocols provide the same expressiveness as non-PDA protocols, but without sacrificing user privacy. These guarantees usually come at the cost of increased computational complexity, increased bandwidth usage, or decreased accuracy.

A shortcoming of many existing proposals is that they assume that all users are honest-but-curious, for example as in [KDK11, ET12, YL13]. As a result, these proposals cannot be used to defend against dishonest users that want to invalidate the aggregate or nudge it in their favour. This means that dishonest users could tamper with their smart meter to reduce their reported electricity consumption [McL+13] or inject false data to increase their score in a ranking system [LM17]. The aggregator would have been able to detect these attacks by looking at the users' private values, but the privacy-preserving properties of the PDA protocol prevent this.

Transitioning from the honest-but-curious model to the malicious model can be achieved using zero-knowledge proofs and commitments, as suggested in proposals such as [Shi+11, KDK11, ET12]. In particular, range proofs [Bou00] can be used to prove in zero knowledge that a committed value is within a given range. However, range proofs—and zero-knowledge proofs in general—often either require a trusted setup or a significant amount of resources from the user [Mor+19]. This makes these approaches unappealing or even infeasible for resource-constrained users.

In this paper, we present an efficient PDA protocol for finding the sum of all private user values at a regular interval. The protocol lets an aggregator probabilistically identify private values that are not within a desired range without the need for zero-knowledge proofs. First, the aggregator divides all users into multiple overlapping groups such that every user is in a unique set of groups. Then, in each interval, each user sends their encrypted values to the aggregator, who determines the sum of private values per group. Finally, the aggregator pinpoints malicious users by looking at the intersection of groups that violate the range. By memorising which groups have out-of-range aggregates, the aggregator can combine detections from different rounds to further enhance its detection rate.

Our protocol boasts several important properties. Firstly, the scheme can be configured to customise the balance between privacy, complexity, and detection rate. For example, one can increase the work the aggregator needs to perform per round

to increase the protocol's resistance to user collusions. Secondly, our protocol 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 $O(\log n)$ complexity per round in the number of users n. 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 [Blu+05].

The remainder of this paper is structured as follows. In Section 2.2 we look at related work. Then, in Section 2.3 we present our protocol in detail, and in Section 2.4 we analyse its security, privacy, complexity, and detection rate. Finally, in Section 2.5 we present our conclusions.

2.2 Related Work

We discuss various protocols for range validation of malicious inputs. First, we consider PDA protocols that have range validation built in. Then, we consider several alternative approaches not inherent to PDA protocols.

Kursawe [Kur10] proposes a scheme in which the aggregator verifies that all private values are valid by checking that the sum of inputs approximates the true aggregate. However, it cannot identify which user sent the invalid value and requires knowledge of the true aggregate beforehand, which is not always feasible.

Sun et al. [Sun+13] present APED, a PDA protocol that detects defective smart meters using a method similar to ours. In APED, a trusted third party divides all users into w random sets of disjoint pairs such that each user is in w pairs at once, and creates a random key k_i for each user i. Then, for each pair of users (i, j), the third party sends $k_{i,j} = -(k_i + k_j)$ to the aggregator. In each round, each user i sends a ciphertext of their measurement, encrypted with the key k_i. After receiving the ciphertexts for that round, the aggregator decrypts the product of the ciphertexts of each pair in one of the w pairing sets of users using that pair's combined key $k_{i,j}$. If a pair cannot be decrypted, at least one of the two users must be defective, and the aggregator will use a different pairing set in the next round. Eventually, the aggregator infers from the overlap of invalid pairs which users are defective. An extension of the protocol, DG-APED [Shi+15], uses groups of arbitrary size. Both protocols have two drawbacks. Firstly, they rely on a trusted third party to create groups and generate key material. Secondly, because the protocols are tailored to defective users, the detection algorithms are unsuitable for users that do not always send invalid users.

Ahadipour, Mohammadi and Keshavarz-Haddad [AMK19] 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 the users' private values. The aggregator then looks at the collected private values to determine which users sent invalid values. While this reduces the privacy impact on its users, giving the aggregator access to even a single private value is not tolerable for sensitive data.

Yang and Li [YL13] 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 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 [Shi+11, KDK11, ET12]. With zero-knowledge proofs, users can mathematically prove that their value is within a particular range without having to reveal their value. Generic zero-knowledge proofs such as SNARKs require a trusted setup, which is often not a realistic assumption. Its cousin, the STARK [Ben+18], resolves this problem, but this comes at the cost of increased communication complexity. Corrigan-Gibbs and Boneh [CB17] 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 [Bou00] are a specific form of zero-knowledge proof specific to range checking. Even though range proofs such as Bulletproofs [Bün+18] are more efficient than generic zero-knowledge proofs, they still incur a relatively high complexity for the users (i.e. the provers) [Mor+19], and must also be used in addition to the privacy-preserving data aggregation protocol and a cryptographic link between the two such as a commitment scheme.

2.3 Probabilistic Range-Limited Private Data Aggregation

We consider a setting with n users and a single aggregator, similar to related work in Section 2.2. Users continuously submit new 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. We require setup parameters b and ℓ, both positive integers, such that there are exactly $n = b^\ell$ users. Up to ℓ users may be 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 other related works including [ET12, Erk15]. 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 behaviour were discovered. Still, we allow for collusions between at most $(b-1)^\ell$ users and the aggregator. We assume that the sets of malicious and colluding users do not change throughout the protocol. 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 2.1. Our protocol broadly works as follows.

1. *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.
2. *Submission*: Every round, each user creates a new secret share of the value zero for each group they are in. The user takes copies of their private value and blinds each copy with a different secret share. The user sends the blinded copies in addition to commitments to the secret shares to the aggregator.
3. *Aggregation*: The aggregator verifies that the secret shares of each group sum to zero 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.
4. *Detection*: The aggregator looks back at which groups have exhibited malicious activity, and derives from their overlap which users are the cause. As the protocol progresses, the aggregator is able to identify more and more malicious users.

Table 2.1 The notation used in the description of our protocol

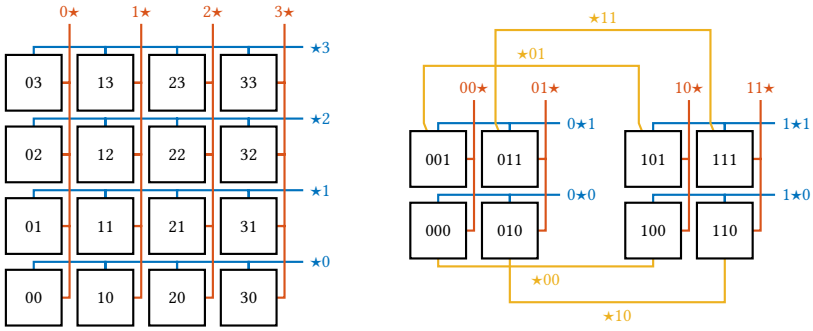
Symbol	Meaning
n	Number of users
b	Grouping base/radix = users per group
ℓ	Grouping dimensionality = groups per user
$[min, max]$	Valid range of a single private value
g	Generator for commitments
pp	Public parameters, contains all of the above
U	Set of all user identifiers
G	Set of all group identifiers
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
$m_{i,t}$	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 group identifiers aggregator marked as malicious
W	Set of user identifiers aggregator marked as malicious

2.3.1 Registration

The goal of the registration phase is to determine the parameters under which the protocol will run and to exchange the necessary information for subsequent rounds. Firstly, the honest-but-curious aggregator chooses a random generator g of an algebraic structure in which the discrete log problem is hard, such as a specific elliptic curve. Additionally, the aggregator chooses application-specific values for n and $min < max$. Then, each user sends a message to the aggregator indicating the desire to participate in the protocol. Once n users have registered, the aggregator sends the public parameters pp and some additional information to all users. The remaining public parameters and additional information are chosen based on the following grouping algorithm and secret sharing scheme.

Parameters for the Grouping Algorithm

The grouping algorithm divides users into groups such that the aggregator can pinpoint malicious users based on which groups exhibit malicious behaviour. We base our algorithm on the structure of a hypermesh [Szy95]. A b-ary ℓ-dimensional hypermesh is a hypergraph with b^ℓ nodes, where each node is assigned an ℓ-digit identifier $d_{\ell-1}d_{\ell-2}\dots d_0$ such that $d_i \in [0, b)$ for all $0 \le i < \ell$. Two nodes are neighbours if and only if their identifiers differ in exactly one digit. Nodes are connected by b-edges, i.e. edges with b endpoints. Edge identifiers have the same format as node identifiers, except that exactly one digit is replaced by the wildcard symbol $\star$. Every edge then connects the b nodes of which the identifier matches that of the edge, ignoring the digit in the wildcard's position. Identifiers can be considered coordinates in an ℓ-dimensional Euclidean space, with $b^{\ell-1}$ edges aligned along each dimension for a total of $\ell b^{\ell-1}$ edges. We give some examples of hypermeshes in Figure 2.1.



(a) A 4^2-hypermesh, which has 16 nodes and 8 hyperedges

(b) A 2^3-hypermesh (which is the graph of a regular cube), which has 8 nodes and 12 hyperedges

Figure 2.1 Examples of hypermeshes

In our protocol, the aggregator generates a b-ary ℓ-dimensional hypermesh after all n users have registered, with the requirements that $n = b^\ell$, $b \ge 2$, and $\ell \ge 2$. The edges in the hypermesh are then exactly the groups that users are in. Generating such a hypermesh constitutes choosing values for b and ℓ, and assigning to each

user a unique identifier in $[0, b^\ell)$, which can be converted to a unique ℓ-digit b-ary identifier. These three variables are sufficient for a user to reconstruct the hypermesh and determine their own position. The ℓ groups that user i is in, denoted G_i, can be found by replacing the respective ℓ digits in i by the wildcard symbol $\star$. The b users in group j, denoted U_j, can be found by replacing the wildcard symbol $\star$ with the respective values $[0, b)$. The neighbours of user i, denoted N_i, can be found by taking the union of $\{G_j \mid j \in U_i\}$, minus i.

Parameters 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 zero-sum additive secret shares used in [ET12] 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

$$\forall j \in G : \sum_{i \in U_j} s_{i,j,t} = 0. \tag{2.1}$$

While the following description assumes that users exchange random numbers each round, such excessive communication can be avoided by having users exchange seeds for random number generators 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. Then, in each round t, user i generates a random number $r_{i \to k,t}$ for each neighbour $k \in N_i$, encrypts it with pk_k, and sends this value to the aggregator, who forwards the message to user k. Once user i has obtained $r_{k \to i,t}$ for each neighbour k, user i creates the secret share

$$s_{i,j,t} = \sum_{k \in G_j} (r_{i \to k,t} - r_{k \to i,t}) \tag{2.2}$$

for each $j \in G_i$. We consider the privacy of this construction in Section 2.4.2. We present a communication diagram that includes registration in Figure 2.2.

2.3.2 Submission

In round t, each user i submits the private value $m_{i,t}$ such that the aggregator can obtain the group aggregates without seeing $m_{i,t}$. We use encryption function $c_{i,j,t} = m_{i,t} + s_{i,j,t}$ to have each user i send $\{c_{i,j,t} \mid j \in G_i\}$ to the aggregator, with the secret share $s_{i,j,t}$ as described in Section 2.3.1. To prevent malicious users from avoiding detection by using a different $m_{i,t}$ in different groups, users must additionally send commitments to their secret shares. We use a simple homomorphic commitment scheme that is computationally binding and computationally hiding: To commit to a value x, a user sends g^x. 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.

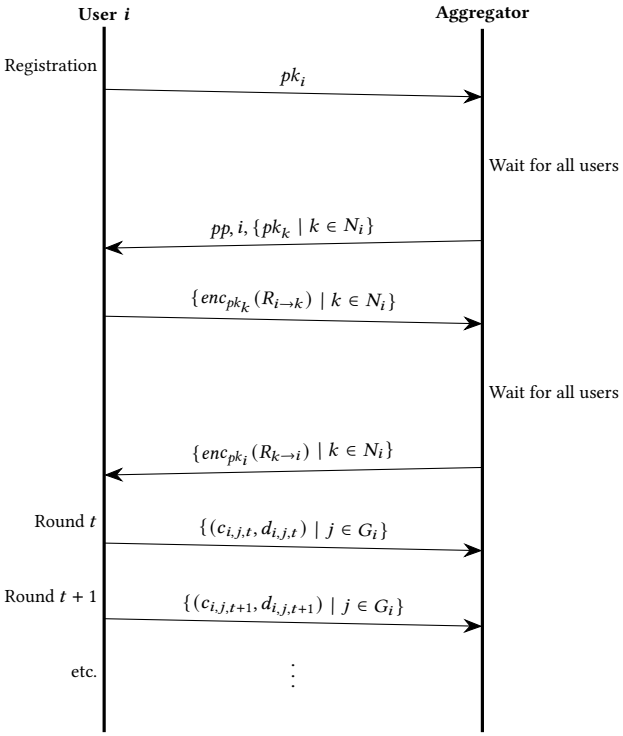


Figure 2.2 An overview of the communication in our protocol. To reduce per-round communication, users exchange seeds $R_{i\to k}$ during registration to generate $r_{i\to k,t}$ in round t.

2.3.3 Aggregation

The aggregation phase is asynchronous to user submissions and may be invoked by the aggregator at any time. Before aggregating the submissions for round t, the aggregator verifies for each group $j \in G$ of which all users have submitted their values by checking that

$$\prod_{i\in U_j} d_{i,j,t} = \prod_{i\in U_j} g^{s_{i,j,t}} = g^{\sum_{i\in U_j} s_{i,j,t}} = g^0 = 1 \tag{2.3}$$

to ensure that users committed to secret shares of the value zero. 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. Next, the aggregator constructs for each user i the set

$$\{g^{c_{i,j,t}}(d_{i,j,t})^{-1} \mid j \in G_i\} = \{g^{m_{i,t}+s_{i,j,t}} g^{-s_{i,j,t}} \mid j \in G_i\} = \{g^{m_{i,t}} \mid j \in G_i\} \tag{2.4}$$

and checks that all values in the set are equal. This ensures that each $c_{i,j,t}$ for user i uses the same $m_{i,t}$. If user i fails this check, all groups in G_i are added to V, effectively marking this user as malicious once the detection algorithm runs. Users that fail to

submit measurements similarly have their groups added to V. If desired, a level of lenience can be introduced by only adding these groups once a user fails to submit in multiple rounds.

After the aggregator has completed its verifications, aggregation can start. For each group $j \in G$, the aggregator calculates

$$M_{j,t} = \sum_{i \in U_j} c_{i,j,t} = \sum_{i \in U_j} (m_{i,t} + s_{i,j,t}) = \sum_{i \in U_j} m_{i,t}. \tag{2.5}$$

If an aggregate $M_{j,t}$ is not in the range $[b \cdot min, b \cdot max]$, 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 in which ranges differ per user or per round by checking that the aggregate is in the sum of the users' current ranges.

The sum of all private values can be calculated by taking the sum of all group aggregates. However, the aggregator should refrain from including invalid groups. Therefore, the aggregator calculates

$$M_t = \frac{\sum_{j \in G \setminus V} M_{j,t}}{\ell}, \tag{2.6}$$

which is the average of the total sums along each of the hypermesh's ℓ dimensions, excluding groups in V. This approximates the sum of only the honest-but-curious users; if all users behave honestly 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 $\{M_{j,t} \mid j \in G \setminus V\}$ in round t.

2.3.4 Detection

The detection algorithm lets the aggregator identify malicious users. Throughout the protocol and across rounds, the aggregator adds groups that exhibit malicious behaviour to the set V. In particular, the set V contains all groups in which at least one user sent a wrongly constructed secret share or sent different private values to different groups in the same round, and contains a subset of groups in which at least one user sent an out-of-bounds value. Looking at 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 are added to W. Over time, the set V becomes more and more complete until all groups containing malicious users have been detected. We prove that this method does not result in false-positive detections in Section 2.4.1, even if some malicious users collude. We analyse the detection rate in Section 2.4.4.

2.4 Analyses

2.4.1 Security Analysis

In this section we prove that the aggregator does not incorrectly identify users, we prove that malicious users cannot submit different measurements to different groups, and we analyse the impact of missing users to the aggregate.

Proof of No False Positives

It is important that the aggregator correctly identifies which users are malicious. We prove that malicious users cannot frame an honest-but-curious user, even if they coordinate the values they send.

Theorem 1. In our protocol, the aggregator will never identify an honest-but-curious user as a malicious user if there are fewer than ℓ malicious users.

Proof. For the sake of contradiction, let there be an honest-but-curious user whom the aggregator falsely identifies as malicious. Then this user must be in ℓ groups of V, so this user shares ℓ groups with malicious users. Because a group contains those users that differ by exactly one digit, two users can at most share a single group. The wrongly-identified user must therefore share groups with ℓ different malicious users. However, by assumption of the theorem's antecedent, there are strictly fewer than ℓ malicious users. Therefore, the honest-but-curious user could not have been identified as a malicious user. ❐

Proof of Aggregate Consistency

Users blind their private measurements $m_{i,t}$ using secret shares $s_{i,j,t}$ to obtain $c_{i,j,t}$. It is important that the aggregator verifies that a user's $c_{i,j,t}$ values use the same underlying $m_{i,t}$; otherwise malicious users could avoid detection by causing inconsistencies between aggregates. We show that it is infeasible for users to do this under our security model, regardless of how many users are malicious. Working in the standard model, every user i sends $(c_{i,j,t}, d_{i,j,t})$ for each $j \in G_i$ to the aggregator, constructed in any way the users want. Let $s_{i,j,t} = dlog_g(d_{i,j,t})$ and $m_{i,j,t} = c_{i,j,t} - s_{i,j,t}$ for all users i and for all $j \in G_i$, regardless of whether values are constructed honestly.

Theorem 2. In our protocol, a malicious user i cannot send messages in round t to the aggregator such that $m_{i,j,t} \neq m_{i,j',t}$ for any two groups $j, j' \in G_i$ such that the aggregator's verification does not fail, assuming that the discrete log problem is intractable in the group generated by g.

Proof. Firstly, if either user i or any neighbour $k \in N_i$ fails to send their messages, the aggregator's verification fails right away and the malicious user does not succeed. Now, it follows from the aggregator's verification of Equation 2.3 on page 22 that $\sum_{i \in U_j} s_{i,j,t} = 0$. Subsequently, we know from the verification of Equation 2.4 on page 22 that, for fixed $i \in U$ and $t \in \mathbb{N}$, all $c_{i,j,t} - s_{i,j,t}$ for $j \in G_i$ are equal. Therefore, by definition of $m_{i,j,t}$, all $m_{i,j,t}$ for fixed $i \in U$ and $t \in \mathbb{N}$ are also equal. ❐

Impact of Missing User Values

The influence of malicious users on M_t decreases as the aggregator adds more groups to V. At the same time, groups in V contain honest-but-curious users. We quantify the effect that malicious users have on the correctness of the total aggregate.

Each user effectively contributes their measurement ℓ times, and, by Theorem 2, each contribution is the same. An ideal protocol would remove only the ℓ contributions of each malicious user. Our protocol also removes the $(b-1)\ell$ contributions of each malicious user's neighbours. The total impact of any set of fewer than ℓ malicious

users is greatest when these malicious users do not share any groups, in which case V contains $(\ell - 1)\ell$ groups. The aggregator then removes $b(\ell - 1)\ell$ contributions instead of the optimal $(\ell - 1)\ell$; a factor of b more than optimal. With a total of ℓb^ℓ contributions amongst all users, the effect of malicious users on M_t therefore diminishes as ℓ increases.

2.4.2 Privacy Analysis

We argue that our protocol is a secure data summation protocol in the setting described in Section 2.3. In particular, we argue that when executing the protocol using a b^ℓ-hypermesh, both the joint view of any set of users and the joint view of the aggregator and a set of fewer than $(b - 1)^\ell$ users do not leak any information about honest-but-curious users' inputs, besides what can be inferred from the group aggregates. We should note that we assume that each group with an honest-but-curious user also contains at least one other non-colluding honest-but-curious user so as to prevent trivial attacks on the aggregates. This assumption is naturally present in many group-based aggregation schemes, including [KDK11, ET12, Erk15]. Recall that the aggregator is honest-but-curious and will therefore assign users to random positions honestly.

Firstly, we consider the joint view of any set of users $U_A \subset U$, which consists only of the public parameters pp, the users' private data, and the public keys pk_i and random seeds $r_{i\to k,t}$ other users have sent to users in U_A. Confidentiality is trivial because the view does not contain any data derived from the private values $m_{i,t}$ of any user $i \notin U_A$.

Next, we consider the joint view of the honest-but-curious aggregator and any set $U_A \subset U$ of fewer than $(b - 1)^\ell$ users. The view consists of the same data as before, now in addition to the aggregator's private information and the data that are sent to the aggregator. We proceed to dissect the implications of this view. Firstly, malicious users in U_A differ only from honest-but-curious users in U_A in that they can interact dishonestly with other users, but this does not give them an advantage. If a malicious user refuses to interact with or sends malformed data to a user, then this user halts and privacy is maintained. Otherwise, if a malicious user sends non-random data to user i, then this is no worse than an honest-but-curious user in U_A sharing their data with the aggregator. Secondly, users that are not in U_A receive sensitive information through the aggregator, but privacy is ensured by encrypting data such that the decryption key is not in the adversary's view. Thirdly, the private values $m_{i,t}$ of user $i \notin U_A$ are masked using the secret shares $s_{i,j,t}$ constructed from values $r_{i\to k,t}$. Because at least one user $k \neq i$ of each group $j \in G_i$ is not in U_A, both $r_{i\to k,t}$ and $r_{k\to i,t}$ are chosen honestly and remain unknown to the adversary. Because additive secret sharing is trivially secure, the secret shares $s_{i,j,t}$ properly mask $m_{i,j,t}$. Finally, we observe that each submission occurs in multiple linearly dependent aggregates, which is equivalent to a system of linear equations. We prove that it is infeasible for the adversary to solve this system because it is not full rank.

Theorem 3. The rank of the incidence matrix of a b^ℓ-hypermesh is $b^\ell - (b - 1)^\ell$. (Equivalently, the number of unknowns in the incidence matrix is $(b - 1)^\ell$.)

Proof. We model the incidence matrix such that each row describes a group and each column describes a user. We construct the incidence matrix recursively, similar to how the hypermesh itself can be constructed. Given a b-ary 1-dimensional hypermesh, its

incidence matrix $C_{b,1}$ is a $(1\times b)$-matrix containing only 1s. Then, a b-ary ℓ-dimensional hypermesh can be constructed from b copies of the b-ary $(\ell-1)$-dimensional hypermesh, where all nodes are additionally connected to their counterparts in the other copies using b-edges. This allows us to construct the incidence matrix $C_{b,\ell}$ for $\ell > 1$ as

$$\begin{bmatrix} C_{b,\ell-1} & 0 & \dots & 0 \\ 0 & C_{b,\ell-1} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & C_{b,\ell-1} \\ I_{b^{\ell-1}} & I_{b^{\ell-1}} & \dots & I_{b^{\ell-1}} \end{bmatrix}, \tag{2.7}$$

where each 0 represents a matrix of the same size as $C_{b,\ell-1}$ containing only zeroes, and I_x denotes an identity matrix of size $x \times x$.

We now use complete induction on ℓ to prove that $\operatorname{rank}(C_{b,\ell}) = b^\ell - (b-1)^\ell$. For the base case, we take $\ell = 1$ and find that $\operatorname{rank}(C_{b,1}) = 1$, which matches our theorem:

$$b^\ell - (b-1)^\ell = b - (b-1) = 1. \tag{2.8}$$

For the recursive case, take as our induction hypothesis that $r = \operatorname{rank}(C_{b,\ell-1}) = b^{\ell-1} - (b-1)^{\ell-1}$. We write $C_{b,\ell}$ in column echelon form as follows to determine its rank. Firstly, consider the column operations necessary to write $C_{b,\ell-1}$ in column echelon form, and apply them to each instance of $C_{b,\ell-1}$ in $C_{b,\ell}$. Note that this also transforms the $I_{b^{\ell-1}}$s located beneath the $C_{b,\ell-1}$s. After applying these steps, each instance of $C_{b,\ell-1}$ has $b^{\ell-1} - r$ empty columns on the right, while each instance of $I_{b^{\ell-1}}$ has no zero columns because it is full rank. The rightmost $b^{\ell-1} - r$ columns of each $I_{b^{\ell-1}}$ are now identical, however, and have nothing but zeroes above them. As such, we cancel out these columns except in the rightmost instance of $I_{b^{\ell-1}}$ using simple column operations. This cancels out $(b-1)(b^{\ell-1} - r)$ columns, while all other columns are non-zero. After moving these zero columns to the right of the matrix, $C_{b,\ell}$ is in column echelon form. The rank of $C_{b,\ell}$ is then the number of non-zero columns, which is

$$\begin{aligned} & b^\ell - (b-1)\left(b^{\ell-1} - r\right) && (2.9)\\ &= b^\ell - (b-1)\left(b^{\ell-1} - \left(b^{\ell-1} - (b-1)^{\ell-1}\right)\right) && (2.10)\\ &= b^\ell - (b-1)(b-1)^{\ell-1} && (2.11)\\ &= b^\ell - (b-1)^\ell, && (2.12) \end{aligned}$$

proving our theorem. ❐

With fewer than $(b-1)^\ell$ users in the view, the adversary always has at least one unknown in this system. To give an intuition into the growth of $(b-1)^\ell$, consider Figure 2.3, where we show the maximum ratio of users that may collude with the aggregator as a function of b and ℓ without breaking confidentiality. For example, a system with $b = \ell = 2$ could not tolerate a single colluding user, while a system with $b = \ell = 5$ could tolerate up to $(4/5)^5 \approx 33\%$ of all users colluding. 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 2.4.4, this decreases the detection rate.

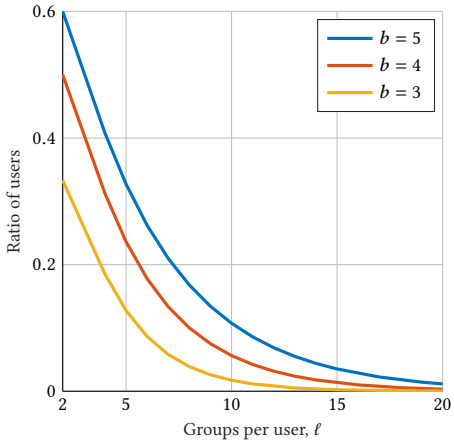


Figure 2.3 Maximum proportion of users that can collude with the aggregator as a function of b (users per group) and ℓ (groups per user)

2.4.3 Complexity Analysis

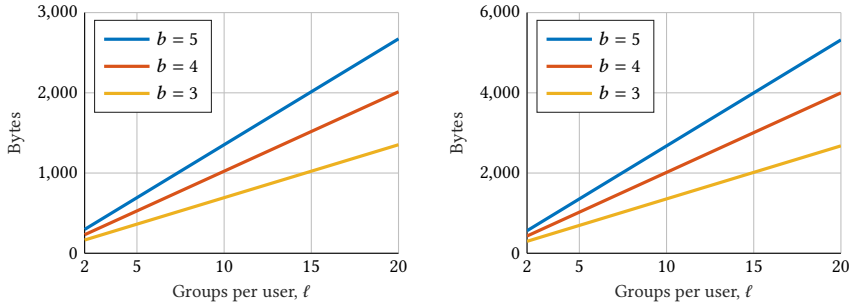
We quantify the complexity of our protocol in terms of the number of users n and compare this to several related PDA protocols. We express complexity as the number of encryptions, decryptions, multiplications, exponentiations, additions, subtractions, and outgoing messages, separately for each user and the aggregator, similar to [Erk15].

Complexity of Our Protocol

Firstly, note that in our protocol, $b = n^{\frac{1}{\ell}}$. This number is maximal when $\ell = 2$, so we say that b is $O(\sqrt{n})$. Meanwhile, $\ell = \log_b(n)$ is $O(\log n)$. Recall the time diagram of our protocol in Figure 2.2 on page 22.

During the registration, each user sends one encrypted seed for each neighbor and a fixed-size key to the aggregator, resulting in an outgoing communication complexity of $O(\sqrt{n}\log n)$ per user. Meanwhile, each user receives one key and one encrypted seed per neighbor, for an incoming communication complexity of $O(\sqrt{n}\log n)$ per user. We visualize registration communication complexity in Figure 2.4. Later, in each round, each user sends for each group it is in a constant-size message containing a masked plaintext and a commitment, for a communication complexity of $O(\log n)$. Users do not receive anything during rounds. Creating a submission requires one commitment and one masked private value for each of the user's groups, for a total of $O(\log n)$ exponentiations and $O(\log n)$ additions per user per round.

The aggregator forwards each user message during the registration, resulting in a factor of n more communication. After the registration, however, the aggregator does not need to communicate with users other than sending acknowledgements. During aggregation, the aggregator verifies user inputs, requiring one exponentiation and one multiplication for each group for each user, for a total of $n\ell b^{\ell-1}$ of either operation. The calculation of the aggregate itself requires only that the aggregator sums together all $n\ell b^{\ell-1}$ submissions. The detection phase does not require complex operations, as the aggregator need only find which users are in ℓ groups of V.



(a) Outgoing communication

(b) Incoming communication

Figure 2.4 Per-user communication during registration. We assume 4 bytes for each (masked) private value and 256-bit EC-cryptography. With point compression, this gives 33-byte keys, ciphertexts (for seeds), and commitments.

Comparison to Related Protocols

We compare our protocol to a selection of related PDA protocols in Table 2.2. Our analysis is subject to several limitations. Firstly, because our protocol is tailored to identifying malicious users, we restrict our analysis to detection protocols for malicious users, thus also excluding APED [Sun+13] and DG-APED [Shi+15]. Secondly, in our analysis of the protocol in [CB17], we assume that the number of multiplication gates is linear in the size of the range, which corresponds to the size of an integer comparison circuit. Finally, for the protocol in [YL13], we assume a binary tree topology for simplicity, and include operations related to the detection sub-protocol for fairness.

The protocol in [YL13] provides by far the lowest complexity by validating in a decentralised fashion, but requires long periods of interactivity and has the weakest security model: An honest-but-curious aggregator and any single user can collude to obtain all private values. Prio [CB17] and Bulletproofs [Bün+18] have a complexity that depends on the size r of the valid range; meanwhile, our complexity is independent of r. Additionally, with Bulletproofs, the size of the range must be of the form $[0, 2^r)$ for some natural number r, whereas our protocol supports arbitrary ranges, as does Prio. Finally, Bulletproofs can verify user submissions in bulk, but only if all users have the same range. Otherwise, the verification complexity grows linearly with the number of different ranges. While an alternative would be to verify the widest range in bulk, this is not practical. Our protocol supports different ranges for all users without an increase in complexity, instead affecting the detection rate, as we discuss in Section 2.4.4.

We conclude that the complexities of these protocols must be considered in light of the application. If users have different, personalised use cases, the computation and communication complexities of our protocol scale better than competing protocols.

2.4.4 Detection Rate Analysis

Values submitted by honest-but-curious users in the same group as a malicious user may coincidentally compensate for the malicious transgression. As a result, our detection algorithm is probabilistic. In this section we analyse how the detection rate

Table 2.2 Comparison of related PDA protocols, given total number of users n and range size 2^r

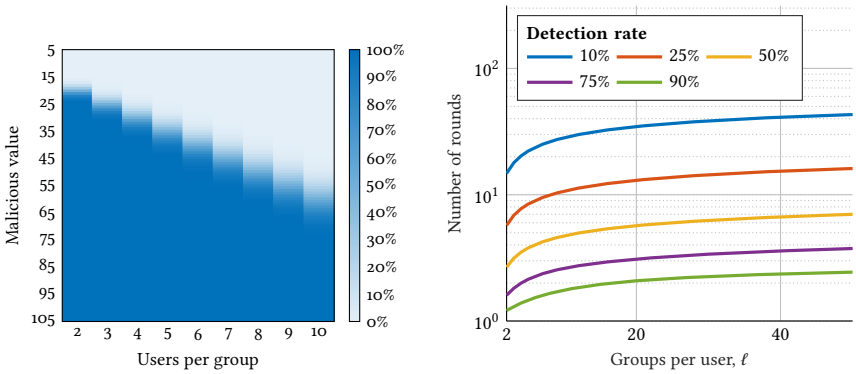
	[YL13]	[CB17]	[Bün+18]	Ours
Properties				
Topology	Tree	Arbitrary	Arbitrary	Hypermesh
Group	ElGamal	FFT field	EC	EC
Aggregation	✓	✓		✓
Detection	✓	✓	✓	✓
Robustness		✓	✓	✓
User complexity				
Enc	$O(1)$	–	–	–
Dec	$O(1)$	–	–	–
Mult	$O(1)$	$O(r \log r)$	–	–
Exp	$O(1)$	–	$O(r)$	$O(\log n)$
Add	–	–	–	$O(\log n)$
Sub	–	–	–	–
Com	$O(1)$	$O(\log r)$	$O(r)$	$O(\sqrt{n} \log n)$
Aggregator complexity				
Enc	$O(1)$	–	–	–
Dec	$O(1)$	–	–	–
Mult	$O(1)$	$O(r \log r)$	–	$O(n \log n)$
Exp	$O(1)$	–	$O(nr)$	$O(n \log n)$
Add	–	–	–	$O(n \log n)$
Sub	–	–	–	–
Com	$O(1)$	$O(1)$	$O(nr)$	$O(n\sqrt{n} \log n)$

varies as a function 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 $[n \cdot min, n \cdot max]$. Our model does not capture serial dependence in user data, which is unrealistic but ultimately does not impinge upon our conclusions. The source code underlying the figures in this section is publicly available [DE25].

Detection Rate of a Single Malicious User

Consider a system with a single malicious user i who submits the out-of-range measurement m. We assume that $m > max$, without loss of generality because X and X_{b-1} are symmetrical. Recall that user i is detected only once all ℓ groups in G_i are in V.

First, we consider the detection rate of an individual group. The aggregate of a group $j \in G_i$ does not exceed its upper bound if and only if $M_{j,t} = X_{b-1} + m \leq b \cdot max$,



(a) Probability that a group of given size transgresses the valid range, given the value of the single malicious user in that group, with each honest value distributed as X

(b) Expected number of rounds until the aggregator has detected a given number of the malicious user's groups, given the per-group detection rate for any single round

Figure 2.5 Detection rate of a single malicious user

or, equivalently, if $X_{b-1} \leq b \cdot max - m$. We illustrate the probability that this relation holds as a function of m and b in Figure 2.5a. 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 $m = max$ and exactly 100% at $m = b(max - min) + min$.

We can thus model the detection rate of a group as a geometric variable to express the expected number of rounds until it is detected. Because the groups G_i overlap only in user i, their detection rates are independent for fixed m. The expected number of rounds until all ℓ groups have been detected at least once is then the expected maximum of ℓ iid geometric variables, which is [Eiso8]

$$f(\ell, p) = \sum_{k=0}^{n} \left(\binom{\ell}{k} p^k (1-p)^{\ell-k} (1 + f(\ell - k, p)) \right), \tag{2.13}$$

where ℓ is the number of groups and p is the per-round detection probability of each group. Figure 2.5b shows $f(\ell, p)$ for various combinations of ℓ and p. We conclude that increasing the number of groups per user necessitates a higher per-group detection rate to retain the number of expected rounds, which can be done by reducing the group size, for example.

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. This means that, depending on the usage scenario, multiple malicious users 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 a

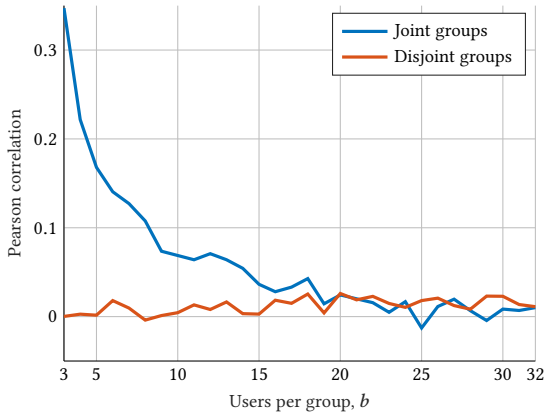


Figure 2.6 The correlation of the detection rate of two groups, each with a different malicious user and overlapping in one honest-but-curious user. Simulated in MATLAB by sampling honest values from truncated normal distribution $\mathcal{N}(10, 9)$ with support $[5, 15]$. Malicious users send $10+5b$, which ensures the groups are not always detected. Correlation was calculated with 5000 trials per group size.

particular detection). Therefore, our protocol is best suited for applications where users are most likely to transgress in a particular direction.

We can reuse our results from Section 2.4.4 to quantify the detection rate of a group with multiple malicious users. Given a group of size b with n malicious users, the detection rate of the sum of malicious values m is

$$\Pr[X_{b-n} + m \leq b \cdot max] \tag{2.14}$$
$$= \Pr[X_{b-n} + m - (n-1) \cdot max \leq (b - (n-1)) \cdot max]. \tag{2.15}$$

That is, this detection rate is the same as that of a single malicious user that sends the value $m - (n-1) \cdot max$ in a group with only $b - (n-1)$ users.

Users may coordinate the malicious values they send to avoid being detected by the aggregator in some groups. However, it follows from Equation 2.14 that complete avoidance is possible only if the sum of their values is valid. Because values are consistent between groups by Theorem 2 on page 24, this type of avoidance detection requires that the sum effect on the total aggregate is valid, so malicious users do not gain any significant advantages by working together.

An important observation regarding the interplay of group aggregates is that 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 user. As shown in Figure 2.6, the impact of this covariance depends on the group size b and quickly becomes negligible. Therefore, the expected number of rounds until detection as expressed in Figure 2.5b holds for multiple users up to covariance.

2.5 Conclusion

Data aggregation is an immensely useful tool for various applications, but introduces a number of privacy concerns. Existing privacy-preserving data aggregation protocols tend to assume that the users are honest-but-curious rather than malicious, or use zero-knowledge proofs, which impose significant computational requirements on the users. Either way, adoption of these much-needed protocols is difficult. We present a data aggregation protocol that probabilistically detects out-of-range user values without giving the aggregator access to these values. Our protocol imposes only $O(\log n)$ per-round computational complexity on its users without relying on expensive cryptography. The protocol is also robust to missing data because it can exclude any number of groups that have exhibited malicious behaviour. Furthermore, given b^ℓ users for positive integers b and ℓ, the aggregator will not misidentify an honest-but-curious user as malicious as long as there are strictly fewer than ℓ malicious users. Finally, our protocol continues to guarantee privacy even when up to $(b-1)^\ell$ users collude with the aggregator.

Chapter 3

Privacy-Preserving Aggregation with Public Verifiability Against Internal Adversaries

Abstract. We consider the problem of publicly verifiable privacy-preserving data aggregation in the presence of a malicious aggregator colluding with malicious users. State-of-the-art solutions either split the aggregator into two parties under the assumption that they do not collude, or require many rounds of interactivity and have non-constant verification time.

In this work, we propose MPVAS, the first publicly verifiable privacy-preserving data aggregation protocol that allows arbitrary collusion, without relying on trusted third parties during execution, where verification runs in constant time. We also show three extensions to MPVAS: MPVAS+, for improved communication complexity, MPVAS-IV, for the identification of malicious users, and MPVAS-UD, for graceful handling of reduced user availability without the need to redo the setup. We show that our schemes achieve the desired confidentiality, integrity, and authenticity. Finally, through both theoretical and experimental evaluations, we show that our schemes are feasible for real-world applications.

Based on: Marco Palazzo, **Florine W. Dekker**, Alessandro Brighente, Mauro Conti and Zekeriya Erkin. "Privacy-preserving data aggregation with public verifiability against internal adversaries". In: *USENIX Security 2024: Proceedings of the 33rd USENIX Security Symposium*. 2024. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/palazzo.

3.1 Introduction

Data aggregation refers to the collection of data from one or more sources and its processing by a central *aggregator* for statistical analysis. These protocols find applications in many situations concerning sensitive data, such as automated power delivery and balancing mechanisms in smart grids [Fan+12, Dhr+20], patient monitoring [Ull+21], and mobile computing [Hul+06, Shi09]. While the benefits brought by data aggregation are evident, without proper countermeasures, they may represent a threat to the privacy of users [GJ10, MEO13, Kap16]. Malicious actors could exploit the collected data to profile or track users' activities, social status, religious beliefs, and medical conditions [GJ10, MEO13, Kap16]. Therefore, data aggregation protocols should guarantee users' privacy in the presence of malicious actors.

Drawbacks of current solutions. "Classical" privacy-preserving data summation protocols [ET12, Shi+11, KDK11] assume that participants are honest-but-curious: They follow the protocol but may try to infer others' sensitive information. When users may be actively malicious by attempting to influence the correctness of the output, zero-knowledge proofs can be used to prove that users' inputs are well-formed [FLC15, KÖB21]. When aggregators may influence correctness, the problem is more difficult. The aforementioned protocols typically assume that the legal and reputational consequences of malicious behavior deter the aggregator from publishing a forged aggregate. However, without efficient methods to actually detect such tampering, these protocols may fail to prevent malicious behaviour.

A simple technique to detect tampering by the aggregator is to have each user sign their submitted value. However, this incurs high verification costs as possibly thousands of signatures must be verified in applications with many users. A more common technique adopted in the literature is to make only the result of the aggregation verifiable using privacy-preserving verifiable summation protocols [Leo+15, Ni+15, Bak+15, Guo+21, Wan+23, Hah+23]. Unfortunately, both solutions fail to guarantee unforgeability when the malicious aggregator may collude with malicious users.

To the best of our knowledge, only three works have considered privacy-preserving verifiable summation against a malicious aggregator who colludes with malicious users [MT21, LL21, Ren+22]. Leontiadis and Li [LL21] and Mouris and Tsoutsos [MT21] both assume a two-aggregator model in which at most one aggregator may collude with malicious users. Unfortunately, as we describe in Section 3.3.3, the work by Leontiadis and Li [LL21] contains a mistake that breaks unforgeability. Finally, Ren et al. [Ren+22] propose a single-server protocol in which the aggregate is hidden from the aggregator. Unfortunately, their protocol fails to provide confidentiality for small plaintext spaces, and verification time is linear in the number of users.

Contributions. We present mpvas, the first privacy-preserving publicly verifiable summation protocol that allows for arbitrary collusions between a malicious aggregator and malicious users, requiring only a single server and constant-time verification. Note that data poisoning attacks from the users are outside the scope of this paper.

Our contributions can be summarized as follows.

- We propose a publicly verifiable aggregate signature scheme considering malicious users and aggregators (mpvas), a novel signature scheme that allows users to sign their reports and compute a signature over the sum of the private values.

- We present three extensions to MPVAS. MPVAS+ reduces communication overhead in a slightly weaker adversarial model, MPVAS-IV allows the detection and removal of malicious users, and MPVAS-UD allows users to exit the protocol without necessitating a new setup phase for the other users.
- We provide theoretical evaluations of the security and performance of our protocols, as well as a practical analysis of their performance using a proof-of-concept implementation. Our results show that MPVAS and its extensions are practical for real-world scenarios.

Outline. In Section 3.2, we present the system model and our assumptions. In Section 3.3, we discuss related works. In Section 3.4, we introduce the building blocks of our schemes. In Section 3.5, we introduce MPVAS. In Section 3.6, we introduce MPVAS+, which reduces communication complexity. In Section 3.7, we introduce MPVAS-IV, which adds input validation to combat malicious users. In Section 3.8, we introduce MPVAS-UD, which adds support for user dropouts. In Section 3.9, we evaluate all four protocols. Finally, in Section 3.10, we present our conclusions.

3.2 System Model and Assumptions

The goal of our protocol is to publish the (authenticated) sum of all users' private values in round t, subject to two properties: Individual users' values remain unknown to other parties *(confidentiality)*, and the published sum is guaranteed to match the true sum *(unforgeability)*. Here, unforgeability implies both integrity and authenticity [Sma16].

We assume all adversaries are probabilistic and polynomially time-bounded. Furthermore, similar to related work [Bak+15, LL21], we assume availability: Parties do not intentionally try to make the protocol fail (denial of service), and do not unexpectedly drop out. We loosen this assumption in Section 3.7 and Section 3.8, where we provide extensions for availability.

The following parties participate in the protocol.

Aggregator. The aggregator collects users' inputs and signatures, and publishes the input sum and an aggregate signature of the sum. The aggregator is malicious and may collude with other malicious parties. That is, the aggregator may deviate from the protocol in arbitrary ways, for example to learn users' private values, tamper with signatures, or publish an incorrect aggregate.

Users. We consider a set of n users $\mathbb{U} = \{1, 2, \ldots, n\}$. In any round t, each user $i \in \mathbb{U}$ holds some private integer $x_{i,t}$. We assume at most $k \leq n - 2$ users are malicious and may collude with the aggregator. The remaining $n - k \geq 2$ users are honest-but-curious; these users follow the protocol, but may still try to obtain private data without colluding. Finally, all users have access to a synchronized clock indicating the current round t.

Verifier. Verifiers check that the aggregator's published output is correct. Any party may be a verifier; this includes external auditors, the aggregator, users, the dealer, and system administrators. We assume there is at least one verifier.

Dealer. We require a trusted dealer to set up the system, similar to nearly all related works [Leo+15, Ni+15, Bak+15, Guo+21, Wan+23, Hah+23, LL21, Ren+22]. Though a fully trusted party is a strong assumption, we argue that it is feasible in relevant

applications such as smart grids and medical data sharing, where the role can be fulfilled by a trusted institution or hardware manufacturer. The dealer is tasked with generating and distributing the public and private parameters to the other parties. After the setup, the dealer exits the protocol.

Communication. The dealer and aggregator both have direct communication channels with all users and verifiers, and with each other. These channels provide secrecy, authenticity, and integrity. Users cannot interact with each other directly but can ask the aggregator to forward messages for them.

3.3 Related Work

There is a large body of work on privacy-preserving computation. We discuss why these works cannot be trivially adapted to the adversarial model from Section 3.2. In Section 3.3.1, we discuss protocols for general verifiable computation. Then, in Section 3.3.2, we discuss protocols for non-verifiable summation. Finally, in Section 3.3.3, we discuss protocols for verifiable summation, which we also summarise in Table 3.1.

Table 3.1 Overview of privacy-preserving verifiable summation works and their properties. The symbols ○, ◑, and ● respectively denote that a property is not, is partially, or is fully achieved by a particular work. The symbol – denotes that a property is not applicable. We denote users by $\mathcal{U}$, aggregator(s) by $\mathcal{A}$, and verifier(s) by $\mathcal{V}$; and abbreviate "trusted" to "trust." and "malicious" to "mal."

	Trust. setup[1]	$\mathcal{A}$ is single	$\mathcal{U}$ is mal.	Verifiable by			Collusions	
				$\mathcal{U}$	$\mathcal{A}$	$\mathcal{V}$	$\mathcal{A}+\mathcal{U}$	$\mathcal{A}+\mathcal{V}$
[Leo+15]	●	●	○	●	●	●	○	○
[Ni+15]	●[2]	●	○	○	○	◑[3]	○	○
[Bak+15]	●	●	○	●	●	●	○	○
[Guo+21]	●	●	○	●	●	○	○	–
[Wan+23]	●	○	○	●	●	○	○	–
[Hah+23]	●	●	○	●	●	○	○	–
[LL21]	●	○	●	○	○	◑[3]	◑[4]	○
[MT21]	◑[5]	○	●	●	●	●	◑[4]	◑[4]
[Ren+22]	●	●	●	●	○	○	●	–
mPVAS	●	●	●	●	●	●	●	●
mPVAS+	●	●	●	●	●	●	●	●
mPVAS-UD	●	●	●	●	●	●	●	●
mPVAS-IV	●	●	●	●	●	●	●	●

[1] Also includes public-key infrastructure and common reference string.
[2] Requires trusted party in all phases of protocol.
[3] Exactly one (trusted) verifier.
[4] May collude with at most one aggregator.
[5] Requires public ledger in all phases of protocol.

3.3.1 General verifiable computation

In general verifiable computation [GGP10], the aggregator computes an arbitrary function over users' data, while learning neither the function nor its inputs. Users can verify that the output is correct, without learning others' values.

Gordon et al. [Gor+15] prove that general verifiable computation is impossible if the aggregator colludes with users, even when only a single user colludes, this user is honest-but-curious, and the protocol uses a trusted setup. Therefore, general verifiable computation is not a suitable solution for our adversarial model. Note that since the above impossibility result requires that the function remains private, this does not preclude verifiable privacy-preserving summation in this adversarial model.

3.3.2 Non-verifiable summation

Privacy-preserving summation [KDK11, Shi+11, Bon+17, Bel+20] ensures confidentiality and availability in a variety of adversarial models. However, these works are not verifiable. That is, if a malicious aggregator publishes an arbitrary value as the sum, this cannot be detected by other parties. Therefore, these protocols are insufficient when the aggregator has an incentive to lie.

3.3.3 Verifiable summation

With privacy-preserving verifiable summation, the aggregator's output can be proven to be the sum of users' inputs. We first discuss protocols for honest-but-curious users and then discuss protocols for malicious users. The aggregator is necessarily assumed malicious. We restrict our discussion to the verification techniques, ignoring the protocols' summation mechanisms. We summarise our results in Table 3.1.

Honest-but-curious users. Given honest-but-curious users and a malicious aggregator, the aggregator must prove that the published sum corresponds to the users' inputs.

Early works [Leo+15, Ni+15, Bak+15] rely on a shared secret between the users and the verifier to ensure only authenticated parties can sign, and rely on a signature key that is secret-shared between the users to ensure a signature is valid only if all users are included. These protocols cannot ensure unforgeability when the aggregator colludes with honest-but-curious users, because if a user sends the aggregator the shared authentication secret, the aggregator can homomorphically modify valid signatures.

Recent works [Guo+21, Wan+23, Hah+23] use the same high-level ideas, but combine this with the non-verifiable summation protocol of Bonawitz et al. [Bon+17] to achieve reliability when users unexpectedly drop out. Each of these works similarly cannot ensure unforgeability when the aggregator colludes with users. We point out to interested readers that two of the above works have received security fixes [Guo22, LWY24]. There are more works that achieve privacy-preserving verifiable summation with honest-but-curious users, but none that do not fit the above general descriptions.

Malicious aggregator and malicious users. To the best of our knowledge, only a few works tackle the problem of privacy-preserving verifiable summation with a malicious aggregator *and* malicious users.

Leontiadis and Li [LL21] propose the addition of a new honest-but-curious party, the converter. Users work with the converter to create homomorphic commitments of their data based on shares of the verifier's secret key. The aggregator then aggregates

users' private data and their commitments (respectively), and sends both to the verifier. Finally, the verifier checks that the aggregation was done correctly. Unfortunately, this protocol is not truly publicly verifiable, since the verification key cannot be shared with users of the protocol. Furthermore, if a user, aggregator, and converter collude, unforgeability no longer holds. Finally, it appears that the protocol is flawed: If a malicious user sends the converter a commitment to zero and then forwards the response to the aggregator, the aggregator can create arbitrary valid signatures.

Mouris and Tsoutsos [MT21] propose splitting the aggregator into two parties: a curator and an analyst. Both may be malicious, but they do not collude, and only the analyst has the decryption key for the aggregate. Users homomorphically encrypt their data and send it to the curator, and publish a homomorphic commitment to the ciphertext on a public ledger. The curator verifies that the received ciphertexts correspond to the commitments on the public ledger, and then publishes an aggregate ciphertext and an aggregate commitment on the public ledger. Finally, the analyst verifies the aggregate commitment, decrypts the aggregate ciphertext, and publishes the aggregate data together with a proof of correct decryption on the public ledger. The protocol requires that the curator and analyst do not collude; otherwise, the protocol cannot guarantee confidentiality and unforgeability. Furthermore, users colluding with the curator may affect correctness.

Ren et al. [Ren+22] propose a summation protocol that provides confidentiality, unforgeability, and availability against a malicious aggregator colluding with a malicious subset of clients. The protocol has four major drawbacks. First, only the users learn the obtained sum, whereas the aggregator learns nothing. Second, only participating users can verify the obtained sum, and there is no trivial extension to allow external parties to learn and verify the sum. Third, if the plaintext space is small (as in verifiable summation for smart meters [Shi+11, Leo+15]), confidentiality can be broken by brute-forcing commitments. Finally, verification time is linear in the number of users.

3.4 Preliminaries

Before we present MPVAS in Section 3.5, we introduce its basic building blocks. We follow the definitions in [KL14].

Bilinear pairings. Given cyclic groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$, each of the same prime order p, a bilinear pairing is a function $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ such that, for any $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$,

$$e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}. \tag{3.1}$$

Furthermore, $e(g_1, g_2)$ should be a generator of $\mathbb{G}_T$, and e should be efficiently computable. This excludes so-called degenerate bilinear pairings, in which $e(g_1, g_2) = 1$ for all $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$. Finally, we assume that the Symmetric External Diffie-Hellman (SXDH) assumption [Ate+05] holds, i.e. that the Decisional Diffie-Hellman (DDH) assumption holds (and thus also that the Discrete Logarithm Problem (DLP) is hard) in both $\mathbb{G}_1$ and $\mathbb{G}_2$, and that there exist no efficiently computable homomorphisms between the two.

Zero-knowledge proof of equality between commitments. We describe ZKPEQ, a zero-knowledge proof that two different Pedersen commitments share the same committed

value. Formally, given commitments $C(x, r_1)$ and $C(x, r_2)$, ZKPEQ proves the relation

$$\{(x, r_1, r_2) : S = g_1^x h_1^{r_1} \wedge T = g_2^x h_2^{r_2}\}. \tag{3.2}$$

This proof can be implemented as an EQ-composition on the common witness x of two Okamoto protocols [Oka92] running in parallel [Sch25]. ZKPEQ can be made non-interactive using the Fiat-Shamir heuristic [FS86].

3.5 MPVAS: Publicly Verifiable Aggregate Signatures with Malicious Users and a Malicious Aggregator

We present MPVAS, a novel aggregate signature scheme for summations. MPVAS can be used to verify that the output of a separate summation protocol was not tampered with by the aggregator. The core idea behind MPVAS is to create commitment-like signatures of the inputs and wrap each signature under a common secret exponent s, similar to other verifiable schemes [Bak+15, Leo+15, Li+16]. Unlike other schemes, however, we allow users to collude with the aggregator by revealing their private parameters. MPVAS guarantees unforgeability of the aggregate signature given at most k malicious users. To achieve this, we use Shamir secret sharing over s with a threshold of $k + 1$.

MPVAS runs in *four phases*: setup, signing, aggregation, and verification. During setup, the participants interactively determine the scheme's public and private parameters. During signing, users cooperatively calculate signatures of their inputs to a separate summation protocol. During aggregation, the aggregator combines users' signatures into a single signature. Finally, during verification, verifiers compare the aggregate signature with the summation protocol's output.

MPVAS provides only an aggregate signature and, for large plaintext spaces, must operate adjacent to a separate privacy-preserving summation scheme. The order of operations is that MPVAS runs up to (but excluding) verification, then the summation protocol reveals the sum and, finally, MPVAS verifies correctness. Alternatively, if the plaintext space is small enough, the sum can be extracted in polynomial time from the aggregate signature itself by repeated verification on all possible values.

Data poisoning attacks are outside the scope of this paper. Still, we note that these attacks can be partially mitigated by including range proofs in the signing phase.

3.5.1 Setup

During the setup, the trusted dealer chooses and publishes the public parameters $pp = (H, H_1, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e, p, n, k)$, generated according to a strong security parameter λ. Each $\mathbb{G}_i$ is a cyclic group of order p, where p is a large prime number. g_1 and g_2 are random generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a type-3 bilinear pairing in which the SXDH [Ate+05] assumption holds. Furthermore, $H : \{0,1\}^* \to \mathbb{G}_1$ and $H_1 : \{0,1\}^* \to \mathbb{G}_1$ are two distinct and cryptographically-secure hash functions. Finally, n is the number of users, and $k \leq n-2$ is the maximum number of malicious users.

The dealer assigns each user a unique identifier $i \in \{1 \dots n\}$, chooses a secret $s \xleftarrow{\$} \mathbb{Z}_p$, and creates n secret shares $[s]_i$ using $(k+1)$-out-of-n Shamir secret sharing. Recall that each Shamir secret share consists of a coordinate (x_i, y_i) on the 2D plane. The dealer ensures that the x-coordinates correspond exactly to the user identifiers,

and defines $[s]_i = y_i$. Next, for each $i \in \{1 \dots n\}$ and each $j \in \{1 \dots k+1\}$, the dealer chooses encryption key $ek_{i,j} \leftarrow\!\$\ \mathbb{Z}_p$, but sets the last encryption key to

$$ek_{n,k+1} = -\sum_{i=1}^{n}\sum_{j=1}^{k} ek_{i,j}. \tag{3.3}$$

It follows that the sum of all encryption keys is zero. The dealer then sends $(pp, i, [s]_i, \{ek_{i,j} \mid 1 \le j \le k+1\})$ to the corresponding user i.

Each user i generates a signature key $sk_i \leftarrow\!\$\ \mathbb{Z}_p$ and sends it to the dealer.

Once all n signature keys have been received, the dealer calculates the verification key tuple

$$vk = \left((g_2^s)^{\sum_{i=1}^{n} sk_i}, g_2^s\right), \tag{3.4}$$

sends (pp, vk) to each verifier, and then leaves the protocol.

3.5.2 Signing

User i creates a signature of their private input $x_{i,t}$ in round t of the summation protocol using the following interactive four-step procedure.

1) Create initial signature. User i computes their initial signature for round t as

$$\sigma_{i,t}^1 = H(t)^{sk_i} g_1^{x_{i,t}} \in \mathbb{G}_1. \tag{3.5}$$

and sends it to the aggregator.

2) Create partial signatures. The aggregator forwards the initial signature tuple of each user i to an arbitrary[1] set $\mathcal{U}_i$ of k users other than user i. We call $\mathcal{U}_i$ the signing set of user i. The aggregator also sends the list of identifiers $\mathcal{U}_i$ to user i. After receiving the initial signature of user i, each user $j \in \mathcal{U}_i$ computes

$$\sigma_{i,t}^{2,j} = H_1(t)^{ek_{j,i}} (\sigma_{i,t}^1)^{[s]_j^*} \tag{3.6}$$

$$= H_1(t)^{ek_{j,i}} (H(t)^{sk_i} g_1^{x_{i,t}})^{[s]_j^*} \in \mathbb{G}_1, \tag{3.7}$$

where (in a minor abuse of notation) $ek_{j,i}$ denotes the encryption key that user j chooses to uniquely associate with user i, and $[s]_j^*$ denotes the partial reconstruction of user j's Shamir secret share of s. User j then sends $\sigma_{i,t}^{2,j}$ to the aggregator.

3) Sum secret shares. Once the aggregator has received k partial signatures $\sigma_{i,t}^{2,j}$ for user i, the aggregator combines the shares in the exponent by computing

$$\sigma_{i,t}^3 = \prod_{j \in \mathcal{U}_i} (\sigma_{i,t}^{2,j}) = H_1(t)^{\sum_{j \in \mathcal{U}_i} ek_{j,i}} (\sigma_{i,t}^1)^{\sum_{j \in \mathcal{U}_i} [s]_j^*} \tag{3.8}$$

$$= H_1(t)^{\sum_{j \in \mathcal{U}_i} ek_{j,i}} (H(t)^{sk_i} g_1^{x_{i,t}})^{\sum_{j \in \mathcal{U}_i} [s]_j^*} \in \mathbb{G}_1. \tag{3.9}$$

The aggregator then sends $\sigma_{i,t}^3$ back to user i.

[1] In our evaluation, we assume that the signing set consists of the next k users after user i when ordered by their numerical identifier.

4) Compute final user signature. At this point, k secret shares of s have been added to the exponent. Adding one more secret share therefore reconstructs s in the exponent. User i computes the final user signature as

$$\sigma_{i,t} = H_1(t)^{ek_{i,i}} \cdot \sigma_{i,t}^3 \cdot (H(t)^{sk_i} g_1^{x_{i,t}})^{[s]_i^*} \tag{3.10}$$

$$= H_1(t)^{ek_{i,i} + \sum_{j \in \mathcal{U}_i} ek_{j,i}} (H(t)^{sk_i} g_1^{x_{i,t}})^s \in \mathbb{G}_1, \tag{3.11}$$

where $ek_{i,i}$ is the single remaining unused encryption key. User i submits their final user signature $\sigma_{i,t}$ to the aggregator.

Note that this signature cannot be verified using the verification key, because this key only works for aggregated signatures. This is intentional, as verifying individual user signatures would trivially allow an adversary to learn the private input of a user by brute force.

3.5.3 Signature Aggregation

After having received the final user signatures of all users for round t, the aggregator computes the aggregate signature

$$\sigma_t = \prod_{i=1}^{n} \sigma_{i,t} \tag{3.12}$$

$$= H_1(t)^{\sum_{i=1}^{n} \sum_{j=1}^{k+1} ek_{i,j}} \left(H(t)^s\right)^{\sum_{i=1}^{n} sk_i} \left(g_1^s\right)^{\sum_{i=1}^{n} x_{i,t}} \tag{3.13}$$

$$= \left(H(t)^s\right)^{\sum_{i=1}^{n} sk_i} \left(g_1^s\right)^{\sum_{i=1}^{n} x_{i,t}} \in \mathbb{G}_1. \tag{3.14}$$

The aggregator sends σ_t to each verifier.

Only at this point should the adjacent summation protocol reveal the sum of users' inputs.

3.5.4 Verification

Once the aggregator has published the sum of all $x_{i,t}$ and the aggregate signature σ_t, each verifier checks the equation

$$e(H(t), vk_1)\, e\left(g_1^{\sum_{i=1}^{n} x_{i,t}}, vk_2\right) \tag{3.15}$$

$$= e\left(H(t), (g_2^s)^{\sum_{i=1}^{n} sk_i}\right) e\left(g_1^{\sum_{i=1}^{n} x_{i,t}}, g_2^s\right) \tag{3.16}$$

$$\stackrel{?}{=} e(\sigma_t, g_2). \tag{3.17}$$

3.5.5 Security Analysis of MPVAS

We show that the verification procedure is correct, does not leak private data, and cannot be fooled into accepting an incorrect signature.

To see that verification succeeds for a correct signature, observe that

$$e(\sigma_t, g_2) = e\left((H(t)^s)^{\sum sk_i} (g_1^s)^{\sum x_{i,t}}, g_2\right) \tag{3.18}$$

$$= e\left((H(t)^s)^{\sum sk_i}, g_2\right) e\left((g_1^s)^{\sum x_{i,t}}, g_2\right) \tag{3.19}$$

$$= e\left(H(t), (g_2^s)^{\sum sk_i}\right) e\left(g_1^{\sum x_{i,t}}, g_2^s\right) \in \mathbb{G}_T. \tag{3.20}$$

Theorem 4. MPVAS is Aggregator Oblivious (AO).

Proof. See Appendix 3.A.1. □

Theorem 5. MPVAS is Aggregate Unforgeable (AU).

Proof. See Appendix 3.A.2. Intuitively, because the aggregator does not know s, they cannot create a correct signature for a sum other than the published one. □

3.6 MPVAS+: MPVAS WITH LOWER COMMUNICATION OVERHEAD

In MPVAS (see Section 3.5), communication complexity is linear in the number of malicious users k. Though we assume malicious users to be in the minority, this level of interactivity may be too high for some applications. We present MPVAS+, an extension of MPVAS to significantly reduce communication complexity. Recall that we provide a runtime analysis of MPVAS and all extensions in Section 3.9.

We show that we can significantly decrease the communication complexity using a divide-and-conquer strategy. Intuitively, our solution works by dividing users into random groups of size $c \leq k$ and providing each group with an independent set of secret shares of s. Since each group can now individually reconstruct s in the exponent, we can eliminate cross-group communication. As long as at least one user in each group is non-malicious, adversaries cannot reconstruct s. We provide a statistical analysis that this holds in Section 3.6.3.

3.6.1 Modifications in MPVAS+

We describe how MPVAS+ differs from MPVAS.

Setup. The key difference with the setup of MPVAS (see Section 3.5.1) is that in MPVAS+, instead of creating a single sharing over all users, the dealer randomly assigns users to groups of size $c \leq k$ and, for each group, generates c-out-of-c Shamir secret shares of s. If c does not divide n, then n mod c arbitrary groups should have one additional user, and the secret sharing threshold of this group is adjusted accordingly. After having chosen the random secret $s \leftarrow\!\!\$\ \mathbb{Z}_p$ (as in MPVAS), the dealer creates separate c-out-of-c Shamir secret shares of s for each group. Because a separate set of shares is created for each group, shares from different groups cannot be combined together. As in MPVAS, each share $[s]_i$ is sent to the corresponding user i. Furthermore, the dealer creates only nc encryption keys instead of $n(k+1)$. In MPVAS+, the list of other users in the group is additionally sent to user i.

Signing. In this phase, the only difference with MPVAS (see Section 3.5.2) is that the aggregator sends the initial signature $\sigma_{i,t}^1$ of each user i to the $c-1$ other users in user i's group, rather than sending them to k arbitrary other users.

Aggregation. The aggregation phase remains unchanged (see Section 3.5.3).

Verification. The verification phase remains unchanged (see Section 3.5.4).

3.6.2 Security Analysis of MPVAS+

Theorem 6. MPVAS+ is AO.

Proof. The MPVAS+ extension changes the behavior of plain MPVAS. When c malicious users end up in the same group, they can collectively reconstruct s and share it with the aggregator, thus allowing it to tamper with the signatures of honest users. However, note that even with knowledge of s, the aggregator still cannot learn the private values of individual users because they are also blinded by the secret factor $H(t)^{sk_i}$. When s is known by the aggregator, MPVAS directly reduces to PPATS, which is AO. □

Theorem 7. MPVAS+ is AU[2] if each group contains at least one honest user.

Proof. The MPVAS+ scheme can be seen as multiple instances of the regular MPVAS scheme running on multiple groups of users, but with different instances of Shamir secret sharing used to generate the secret shares of s. Thus, AU still holds following the same logic presented in the proof of Theorem 5 for MPVAS, as long as each group contains at least one honest user. We provide a statistical analysis that this requirement holds in Section 3.6.3. □

We emphasise that MPVAS+ provides AU with k malicious users only if we assume non-adaptive corruptions. Otherwise, the security of MPVAS+ is downgraded to that of an MPVAS instance with $k = c - 1$.

3.6.3 Statistical Analysis of MPVAS+

The communication complexity of MPVAS+ is better than that of MPVAS only if $c \leq k$. However, unlike MPVAS, in MPVAS+ it is possible that at least one group consists of adversaries only, who may then collude to retrieve private key material. We give an exact formula for this probability, and show that it can be made negligibly small.

Let n be the number of users, k the number of malicious users, c the group size, and $d = floor(n/c)$ the number of groups. We assume that c divides n exactly. Otherwise, $n - cd$ groups should be given one user more, and the following calculations give an upper bound rather than an exact value.

We calculate the probability using a combinatorial counting argument. We model the process of dividing users into groups as first dividing all n users into groups, and then randomly (non-adaptively) corrupting k users. We count the number of instances in which at least one group is fully corrupted, and divide this by the total number of instances.[3] The total number of instances (the denominator) is simply $\binom{n}{k}$ (i.e. the binomial coefficient "n choose k"), but the number of problematic instances (the numerator) is harder to compute.

Intuitively, the numerator is the number of combinations in which *exactly* one group is fully compromised (which is d, since there are d groups), multiplied by the number of ways in which the remaining $n - c$ users can contain $k - c$ corruptions (which is

[2] Against type-I and type-II forgeries. See Appendix 3.A.2.

[3] We calculate the probability as a combinatorial problem. Modeling this as a permutation instead would require counting all possible ways to assign identity to users after fixing a specific combination. This can be done by multiplying both the numerator and denominator by $(n-k)!\,k!$. Since this cancels out, both methods give the same result.

$\binom{n-c}{k-c}$), *seemingly* giving the probability

$$\frac{d\binom{n-c}{k-c}}{\binom{n}{k}}. \tag{3.21}$$

However, this is inaccurate, because if the remaining users also fully corrupt a group, that case is counted twice. In fact, duplicates are counted twice, triplicates are counted thrice, and, in general, r-replicates are counted r times. Luckily, by the inclusion-exclusion principle, it suffices to separately count and subtract these cases.

Let $R = \mathit{floor}\left(\frac{k}{c}\right)$ denote the "replicity", which is the highest order of replication. To determine the number of r-replicates, we first define a helper function that counts the number of r-replicates *after fixing which r groups are fully corrupted*:

$$\mathit{rep}(r) = \binom{n-rc}{k-rc} - \sum_{i=r+1}^{R} \left(\binom{d-r}{i-r} \cdot \mathit{rep}(i)\right). \tag{3.22}$$

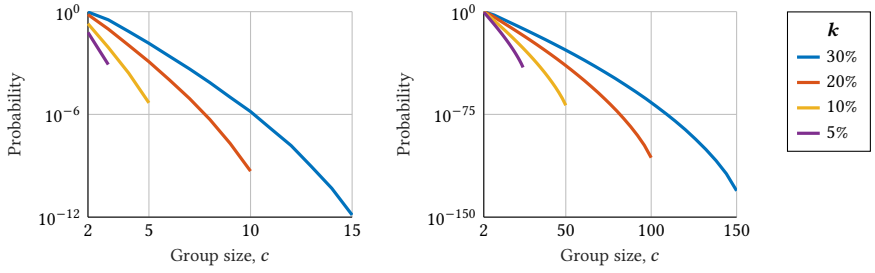
This function counts the number of ways to corrupt remaining users and then recursively subtracts higher-order replicates. The number of recursive i-replicates is found by first fixing $r - i$ additional groups and then multiplying by $\mathit{rep}(i)$.

We conclude that the probability that at least one group is fully corrupted is exactly

$$\frac{d \cdot \binom{n-c}{k-c} - \sum_{r=2}^{R}((r-1) \cdot \binom{d}{r} \cdot \mathit{rep}(r))}{\binom{n}{k}} \tag{3.23}$$

if c divides n, and is a strict upper bound otherwise. As in Equation 3.22, for each order of replication, we multiply by the number of ways to choose r groups, and additionally multiply by $r - 1$ to actually the multiply-counted items.

We visualize Equation 3.23 for various values of n, k, and c in Figure 3.1. The figure shows that, given sufficient users, the probability of accidentally assigning only malicious users to a group can be made negligible with an appropriate choice of c. For example, with 50 users, of which 20% malicious, choosing $c = 7$ gives a probability of approximately 0.000841%, and can be made even smaller.



(a) $n = 50$ (b) $n = 500$

Figure 3.1 Probability that at least one group is fully corrupted in MPVAS+. Due to the logarithmic y-axis, lines end when they reach zero.

3.7 mPVAS-IV: mPVAS with Input Validation

We have thus far assumed that parties do not try to make the protocol fail. In this section, we present mPVAS-IV, an extension to mPVAS (see Section 3.5) to allow the aggregator to identify and remove users that attempt to cause an invalid aggregate signature. mPVAS-IV is fully compatible with both mPVAS+ (see Section 3.6) and mPVAS-UD (see Section 3.8).

In terms of our adversarial model (see Section 3.2), we loosen our assumptions on users, who may now send ill-formed messages with the intent of causing verification to fail. We model the aggregator as a service provider, who may attempt to obtain users' private data or output a falsified signature, but is expected to ensure well-formed outputs so as to not disrupt users reliant on their services. As such, the malicious aggregator may still collude with malicious users against confidentiality and unforgeability, but the aggregator acts honestly with regard to availability. Furthermore, we assume that the adjacent summation protocol ensures availability in this model, including *verified* commitments to users' summation inputs.

mPVAS-IV validates that users' inputs are well-formed, and pinpoints users causing malformedness. Those users are then barred from participating in future instances of the protocol, and the protocol is restarted from scratch with the remaining users. In the worst case, all k malicious users are removed one at a time, requiring k restarts. However, since there will then be no more malicious users left, users can then continue normal operation without further interruptions. Therefore, whereas in mPVAS adversaries could prevent all output without end, mPVAS-IV reduces this adversarial capability to a linearly bounded overhead.

At its core, mPVAS-IV adds a mechanism to validate individual final user signatures when verification fails. However, since users may tamper with others' signatures, we must also add a detection mechanism there. Note that we do not need to validate the initial signature, since either the subsequent final user signature is valid and there is no problem to begin with, or the final user signature is invalid and is detected as such.

3.7.1 Modifications in mPVAS-IV

We describe how mPVAS-IV differs from mPVAS.

Setup. The setup phase of mPVAS-IV starts by running the setup phase of mPVAS (see Section 3.5.1). The dealer then generates extra information for input validation.

First, the dealer chooses g_T and h_T as random generators of $\mathbb{G}_T$, and, for each user i, generates $r_i \leftarrow\!\!\$\ \mathbb{Z}_p$ and the set of values $EK_{j,i} = g_2^{ek_{j,i}}$. The dealer sends r_i, $EK_{j,i} = g_2^{ek_{j,i}}$, and g_2^s to each user i.

Next, for each user i, the dealer generates $SK_i = g_T^{sk_i} h_T^{r_i}$, $SS_i = g_2^{[s]_i}$, $EK_{j,i} = g_2^{ek_{j,i}}$ for $j \in \{1 \dots k\}$, and $EKS_{j,i} = g_2^{ek_{j,i}/s}$ for $j \in \{1 \dots k+1\}$. The dealer sends these values to the aggregator, along with $v = g_2^{1/s}$.

Finally, the dealer sends the additional public parameters (g_T, h_T) to all participants.

Signing. We require additional operations before and after the regular signing phase of mPVAS (see Section 3.5.2).

Before the regular signing phase, we require that the adjacent summation scheme outputs Pedersen commitments of the users' inputs to that summation scheme. This

is to ensure consistency of the inputs between the two schemes. If MPVAS-IV is used without a separate summation scheme, the committed values should be validated using range proofs instead. Either way, let $C(x_{i,t})$ for each user i denote these commitments.

After the regular signing phase is complete, each user checks their final user signature for tampering. If tampering is detected, the user informs the aggregator, who then validates the corresponding partial signatures. If the aggregator also detects tampering, the aggregator marks the user(s) who sent that partial signature as malicious. Otherwise, if the aggregator does not detect tampering, the reporting user is instead marked as malicious. The protocol then restarts without the detected malicious users.

User i checks their final user signature $\sigma_{i,t}$, calculated in Equation 3.10 on page 43, for tampering by checking that

$$e(H_1(t), g_2^{\varepsilon_i})\, e(H(t), (g_2^s)^{sk_i})\, e(g_1^{x_{i,t}}, g_2^s) \stackrel{?}{=} e(\sigma_{i,t}, g_2) \tag{3.24}$$

holds, where $\varepsilon_i = \sum_{j=1}^{k+1} ek_{j,i}$. In a nutshell, the left-hand side re-calculates user i's expected final user signature under a bilinear mapping (compare with Equation 3.10 on page 43), while the right-hand side bilinearly maps the actual final user signature. If Equation 3.24 does not hold, the user reports this to the aggregator.

Since the aggregator forwards all partial signatures, the aggregator possesses the initial signature $\sigma_{i,t}^1$ as well as the partial signature $\sigma_{i,t}^{2,j}$ of each user $j \in \mathcal{U}_i$. Upon receiving user i's claim that their signature was tampered with, the aggregator computes for each user $j \in \mathcal{U}_i$ the value

$$\sigma_{i,t}^{2,j^*} = e\left(H_1(t), g_2^{ek_{j,i}}\right) e\left(\sigma_{i,t}^1, g_2^{[s]_j^*}\right), \tag{3.25}$$

where $[s]_j^*$ is as in Equation 3.6 on page 42, i.e. the partial reconstruction of user j's Shamir secret share of s, here calculated in the exponent of g_2 using the set SS. Finally, the aggregator validates the partial signature $\sigma_{i,t}^{2,j}$ by checking

$$\begin{aligned}
\sigma_{i,t}^{2,j^*} &\stackrel{?}{=} e\left(\sigma_{i,t}^{2,j}, g_2\right) && (3.26)\\
&= e\left(H_1(t)^{ek_{j,i}} \left(H(t)^{sk_i} g_1^{x_{i,t}}\right)^{[s]_j^*}, g_2\right) && (3.27)\\
&= e\left(H_1(t)^{ek_{j,i}}, g_2\right) e\left((H(t)^{sk_i} g_1^{x_{i,t}})^{[s]_j^*}, g_2\right) && (3.28)\\
&= e\left(H_1(t), g_2^{ek_{j,i}}\right) e\left(\sigma_{i,t}^1, g_2^{[s]_j^*}\right) && (3.29)
\end{aligned}$$

If this holds, then user j did not act maliciously; otherwise, user j is marked as malicious and expelled from the protocol. The aggregator repeats this process for all users in $\mathcal{U}_i$, as there can be more than one user behaving maliciously in a single signing set.

Aggregation. The aggregation phase remains unchanged (see Section 3.5.3).

We assume that the adjacent summation scheme aborts no later than this point if the Pedersen commitment $C(x_{i,t})$ of any user i does not correspond to that user's real input to the summation scheme.

Verification. The verification phase of MPVAS-IV starts by running the verification phase of MPVAS (see Section 3.5.4). If verification fails, the aggregator tries to find the culprit by verifying that the final user signature $\sigma_{i,t}$ of each user i matches Equation 3.10 on page 43. This verification entails removing the term containing the encryption keys ek and then asking user i for a zero-knowledge proof of equality between the expected and the actual value of the remaining term.

To remove the term with the encryption keys, the aggregator first computes

$$g_2^{\frac{\varepsilon_i}{s}} = \prod_{j=1}^{k+1} EK_{j,i} = \prod_{j=1}^{k+1} g_2^{\frac{ek_{j,i}}{s}} \in \mathbb{G}_2, \tag{3.30}$$

and then removes the term by computing

$$\sigma'_{i,t} = \frac{e\left(\sigma_{i,t}, g_2^{\frac{1}{s}}\right)}{e\left(H_1(t), g_2^{\frac{\varepsilon_i}{s}}\right)} \tag{3.31}$$

$$= \frac{e\left(H_1(t)^{\varepsilon_i}(H(t)^{sk'_i} g_1^{x'_{i,t}})^s, g_2^{\frac{1}{s}}\right)}{e(H_1(t), g_2)^{\frac{\varepsilon_i}{s}}} \tag{3.32}$$

$$= \frac{e\left(H_1(t)^{\varepsilon_i}, g_2^{\frac{1}{s}}\right) e\left(H(t)^{sk'_i} g_1^{x'_{i,t}}, g_2\right)}{e(H_1(t), g_2)^{\frac{\varepsilon_i}{s}}} \tag{3.33}$$

$$= e(H(t), g_2)^{sk'_i}\, e(g_1, g_2)^{x'_{i,t}} \in \mathbb{G}_T, \tag{3.34}$$

where the values of sk'_i and $x'_{i,t}$ are implied. Finally, the aggregator asks user i to prove that $sk'_i = sk_i$ and $x'_{i,t} = x_{i,t}$. User i does so by interpreting Equation 3.34 as a Pedersen commitment and providing two ZKPEQ proofs (see Section 3.4): one for proving the equality $sk'_i = sk_i$ between $\sigma'_{i,t}$ and SK_i, and another for proving equality of $x'_{i,t} = x_{i,t}$ between $\sigma'_{i,t}$ and $C(x_{i,t})$.

If MPVAS-IV is used without an adjacent summation protocol, user i must also provide a range proof (such as a Bulletproof [Bün+18]) of $x_{i,t}$ to show that their input lies in a restricted range, and that extraction of the sum from σ_t is tractable. Users that fail to send valid proofs are removed from the protocol and subsequent executions.

3.7.2 Security Analysis

Theorem 8. MPVAS-IV is AO.

Proof. The additional information received by the aggregator does not yield any advantage to breaking AO. In fact, the secret shares the aggregator receives in the set SS cannot be efficiently extracted due to the hardness of the DLP in $\mathbb{G}_2$. The commitments contained in the set SK are hiding, thus the aggregator cannot extract the signing keys either. Furthermore, as with MPVAS, all initial and final user signatures $\sigma^1_{i,t}, \sigma^{2,j}_{i,t}, \sigma^3_{i,t}, \sigma_{i,t}$ contain the secret factor $H(t)^{sk_i}$, which perfectly hides $x_{i,t}$ in $\mathbb{G}_1$ and prevents the aggregator from exploiting the verification algorithm in Equation 3.24 to find $x_{i,t}$.

The intermediate value $e(H(t), g_2)^{sk'_i}\, e(g_1, g_2)^{x'_{i,t}}$ from Equation 3.34 is also hiding under the random oracle model. Finally, the proof ZKPEQ does not leak any information about the private witness due to its zero-knowledge property. We conclude that the aggregator cannot learn the private value of honest users, and thus MPVAS-IV is AO. □

Theorem 9. MPVAS-IV is AU.

Proof. See Appendix 3.A.3. □

3.8 MPVAS-UD: MPVAS WITH USER DROPOUTS

Requiring that all users are always online is not feasible for some applications. In this section, we present MPVAS-UD, an extension to MPVAS (see Section 3.5) to allow users to choose a set of rounds in which they will not participate by sending one or more recovery keys containing the necessary material that would otherwise be missing from those rounds. As with the base MPVAS protocol, MPVAS-UD works as long as at least $k + 2$ users do not drop out of the protocol. MPVAS-UD is fully compatible with both MPVAS+ (see Section 3.6) and MPVAS-IV (see Section 3.7).

3.8.1 Modifications in MPVAS-UD

We describe how MPVAS-UD differs from MPVAS.

Setup. In addition to the regular setup of MPVAS (see Section 3.5.1), the dealer also sends $EK_j = g_2^{\sum_{i=1}^{k} ek_{j,i}}$, for each $j \in \{1 \dots n\}$, to all verifiers.

If MPVAS-UD is combined with MPVAS+, then the setup should be adjusted to use c-out-of-c' secret sharing instead of c-out-of-c secret sharing, where $c' \geq c$. This ensures that at most $c' - c$ users in each group can drop out without resulting in incomplete signatures. The statistical analysis in Section 3.6.3 still applies to c.

Signing. In any round t, before running the regular signing phase (see Section 3.5.2), each user i has the option of dropping out for a set of rounds $\mathcal{T}$, possibly including the remainder of the current round t. For each round $\tau \in \mathcal{T}$ from which user i would like to drop out, user i calculates a recovery key

$$rk_{i,\tau} = e\Big(H(\tau)^{-sk_i}, g_2^{s}\Big) \in \mathbb{G}_T. \tag{3.35}$$

User i then sends $m_i = (i \parallel \tau \parallel rk_{i,\tau})$ to the aggregator, who forwards both to the verifiers. Note that the aggregator can aggregate all recovery keys $rk_{i,t}$ together before sending them to the verifiers to save space and reduce the communication overhead.

If MPVAS-UD is used in the adversarial model of MPVAS-IV (see Section 3.7), we must additionally ensure that user i cannot invalidate signatures of rounds $\mathcal{T}$. Therefore, user i must prove that the recovery key is well-formed using a zero-knowledge proof that sk_i in $rk_{i,\tau}$ is the same as in the commitment $g_T^{sk_i} h_T^{r_i}$ from the setup of MPVAS-IV. Concretely, user i proves the relation

$$\{(x, y) : S = g_1^x h_1^y \wedge T = g_2^x\}, \tag{3.36}$$

which can be implemented and made non-interactive similar to ZKPEQ (see Section 3.4).

The signing phase then proceeds as normal, but without the users who have opted to drop out of round t.

Aggregation. The aggregation phase remains unchanged (see Section 3.5.3).

Verification. The verification phase of MPVAS-UD replaces that of regular MPVAS (see Section 3.5.4). In round t, let $\mathscr{D}_t$ be the set of users that dropped out, and let $\mathscr{R}_t$ be the set of the remaining users. Since users $\mathscr{D}_t$ do not participate in the adjacent summation protocol of round t, the published sum should be $x_t = \sum_{i \in \mathscr{R}_t} x_{i,t}$, and the aggregate signature should similarly be over that sum. To verify that the signature σ_t is correct, the verifier uses the dropped-out users' recovery keys and checks

$$e\left(g_1^{x_t}, vk_2\right) e(H(t), vk_1) \prod_{i \in \mathscr{D}_t} rk_{i,t} \stackrel{?}{=} \frac{e(\sigma_t, g_2)}{e\left(H_1(t), \prod_{i \in \mathscr{R}_t} EK_i\right)}. \tag{3.37}$$

This is essentially a modification of Equation 3.15 on page 43 wherein the verifier assumes that dropped-out users input $x_{i,t} = 0$, while compensating for missing information using the recovery material. To see that correctness holds, let $\sigma'_t = (H(t)^{\sum_{i \in \mathscr{R}_t} sk_i} g_1^{x_t})^s$ be the desired signature (see Equation 3.14 on page 43), recall the definition of vk from Equation 3.4 on page 42, and observe that on the left-hand side of Equation 3.37 we find

$$\begin{aligned}
&e\left(g_1^{x_t}, vk_2\right) e(H(t), vk_1) \prod_{i \in \mathscr{D}_t} rk_{i,t} && (3.38)\\
&= e\left(g_1^{x_t}, g_2^s\right) e(H(t), vk_1)\, e(H(t), g_2^s)^{-\sum_{i \in \mathscr{D}_t} sk_i} && (3.39)\\
&= e\left(g_1^{x_t}, g_2^s\right) e(H(t), g_2^s)^{\sum_{i \in \mathscr{R}_t} sk_i} && (3.40)\\
&= e\left((H(t)^{\sum_{i \in \mathscr{R}_t} sk_i} g_1^{x_t})^s, g_2\right) = e(\sigma'_t, g_2). && (3.41)
\end{aligned}$$

Similarly, on the right-hand side of Equation 3.37, we find

$$\begin{aligned}
&\frac{e(\sigma_t, g_2)}{e\left(H_1(t), \prod_{i \in \mathscr{R}_t} EK_i\right)} && (3.42)\\
&= \frac{e\left(H_1(t)^{\sum_{i \in \mathscr{R}_t} \sum_{j=1}^{k+1} ek_{j,i}} \left(H(t)^{\sum_{i \in \mathscr{R}_t} sk_i} g_1^{x_t}\right)^s, g_2\right)}{e\left(H_1(t), \prod_{i \in \mathscr{R}_t} EK_i\right)} && (3.43)\\
&= \frac{e(H_1(t), g_2)^{\sum_{i \in \mathscr{R}_t} \sum_{j=1}^{k+1} ek_{j,i}}\, e(\sigma'_t, g_2)}{e(H_1(t), g_2)^{\sum_{i \in \mathscr{R}_t} \sum_{j=1}^{k+1} ek_{j,i}}} && (3.44)\\
&= e(\sigma'_t, g_2). && (3.45)
\end{aligned}$$

3.8.2 Security Analysis

Theorem 10. MPVAS-UD is AO.

Proof. See Appendix 3.A.4. ❐

Theorem 11. MPVAS-UD is AU.

Proof. See Appendix 3.A.5. ❐

3.9 Complexity Analysis of the mPVAS Family

We evaluate the complexity of mPVAS and each of its extensions. In Section 3.9.1, we present the asymptotic communication complexity of our schemes, and compare this with a selection of related works. In Section 3.9.2, we describe our experimental setup for empirically determining runtime complexity. After that, we present the results of this analysis for mPVAS in Section 3.9.3, for mPVAS+ in Section 3.9.4, for mPVAS-IV in Section 3.9.5, and for mPVAS-UD in Section 3.9.6.

3.9.1 Asymptotic Communication Complexity

In Table 3.2, we summarize the asymptotic communication complexity of all proposed schemes and compare them with state-of-the-art protocols that consider malicious users. The dealer has to share information with every user, which leads to a complexity of $O(n)$ for all signature schemes. Users only communicate within their own signing set, for a complexity of $O(k)$, or $O(c)$ in the mPVAS+ scheme. The aggregator needs to relay messages between each user and their signing set, which leads to a complexity of $O(kn)$ for the mPVAS and mPVAS-IV schemes, and $O(cn)$ for the mPVAS+ scheme. Verifiers do not actively participate in the protocol.

Table 3.2 Asymptotic communication complexity per party in related works and in the mPVAS family

	Dealer	Aggregator	User	Verifier	Ledger
[LL21]	$O(n)$	$O(1)$	$O(1)$	$O(0)$	–
[MT21]	$O(1)$	$O(1)$	$O(1)$	$O(0)$	$O(n)$
[Ren+22]	$O(n)$	$O(n^2)$	$O(n)$	$O(0)$	–
mPVAS	$O(n)$	$O(kn)$	$O(k)$	$O(0)$	–
mPVAS+	$O(n)$	$O(cn)$	$O(c)$	$O(0)$	–
mPVAS-IV	$O(n)$	$O(kn)$	$O(k)$	$O(0)$	–
mPVAS-UD	$O(n)$	$O(kn)$	$O(k)$	$O(0)$	–

The mPVAS family of protocols enjoys reduced communication complexity compared to [Ren+22], but increased communication complexity compared to [LL21, MT21]. We note that similar schemes such as [Bak+15, Leo+15, LL21, MT21], work in a different system and adversarial model where there is little to no interaction between the participants except for the initial setup. As such, the communication complexities for these schemes is $O(1)$ for both the aggregator and the users. (Similarly, the computation complexity is $O(1)$ for the users and $O(n)$ for the aggregator.) While this is better than any of the mPVAS variants, the adversarial model in these related works is also weaker than those used in our work. As discussed in Section 3.3, the compared schemes either assume honest behavior from the users [Leo+15, Bak+15], no collusions between the aggregator and the users [MT21], or they rely on a semi-trusted party during protocol execution [MT21, LL21]. That said, mPVAS can trivially be generalized to these alternative scenarios. For example, honest users can be simulated by choosing $k = 0$, which leads to a non-interactive scheme with $O(1)$ communication complexity and a computation complexity nearly identical to that of the PUDA scheme [Leo+15].

Similarly, choosing $k = 1$ for MPVAS corresponds to the scheme presented in [LL21], which entrusts a semi-trusted third party with the secret signing key, and similarly leads to constant communication complexity.

3.9.2 Experimental Setup

We created a proof-of-concept implementation of MPVAS and its extensions [PD25]. We use the Charm framework [Bla79], which is widely used for the prototyping and benchmarking of cryptographic schemes [RW13, Ara+17]. All experiments were run on a Threadripper 7970X CPU with 256 GB of RAM, on Debian 12. The protocol ran sequentially on a single core in a single thread without special optimizations.

We do not model communication between nodes, measuring only the runtime of our schemes' computations. We measure wall-clock time with nanosecond precision. We repeat each experiment five times, and take the mean runtime.

The experiments are performed over the MNT224 elliptic curve, which is pairing friendly, provides 112 bits of security [Cui+18], and allows for type-3 pairings, which are necessary for the SXDH assumption [Bak+15]. This is the most secure curve provided by the Charm framework that is compatible with our schemes. While the current recommendation is to use curves that provide 128 bits of security as a conservative choice, 112 bits is the minimum security level required by NIST for the United States Federal Government [BR19]. In this curve, elements in $\mathbb{G}_1$ are 56 bytes, in $\mathbb{G}_2$ are 168 bytes, in $\mathbb{G}_T$ are 168 bytes, and in $\mathbb{Z}_p$ are 28 bytes [KR19], which we verified experimentally. The size of the elements influences the performance of the various algebraic operations performed in each group.

3.9.3 MPVAS Runtime

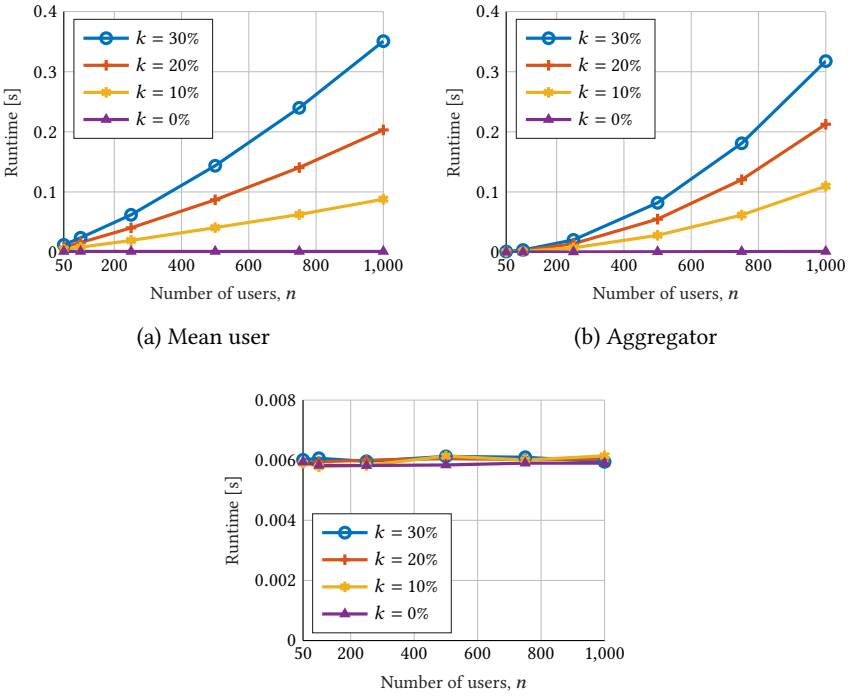
We show the runtime of MPVAS (see Section 3.5) in Figure 3.2. Firstly, Figure 3.2a shows that, even when there are 1000 users and $k = 30\%$ of all users are malicious, the runtime is only around 0.36 seconds for a single user. As expected from an asymptotic complexity $O(k)$, the runtime decreases with the number of malicious users k. Secondly, Figure 3.2b shows a similar trend for the aggregator. Moreover, when $k = 0$, the aggregator does not have to combine partial secret sharing for every user, and the runtime dips below even that of a single user. Finally, Figure 3.2c shows the runtime for verifiers. As expected, since a verifier only needs to compute three pairings regardless of the number of users, runtime is constant.

3.9.4 MPVAS+ Runtime

MPVAS+ (see Section 3.6) reduces computational complexity under well-defined probabilistic assumptions, assuming non-adaptive corruptions. We present its runtime in Figure 3.3. We note that, at $k = 0\%$, MPVAS+ reduces directly to MPVAS, and complexity is independent of the number of users. For experiments with $k > 0\%$, we choose the smallest group size c such that the probability that at least one group is fully compromised is at most 10^{-5}, using our combinatorial formula in Equation 3.23 on page 46, giving us group sizes ranging from 5 up to 14.

We see in Figure 3.3a that, compared to MPVAS, user runtime is reduced by an order of magnitude. Since the runtime depends only on the constant c, the runtime appears to become constant even as n continues to grow.

We see in Figure 3.3b that the speedup for the aggregator is similar. As in MPVAS, the main bottleneck for the aggregator is combining the partial user signatures as in Equation 3.8 on page 42. (Combining the final user signatures requires negligible runtime.) Reducing the group size c affects this bottleneck directly. For example, with 1000 users, setting $c = 14$ means the aggregator must only aggregate $c - 1 = 13$ values per user instead of $k = 300$, reducing complexity in this part by a factor of 23.

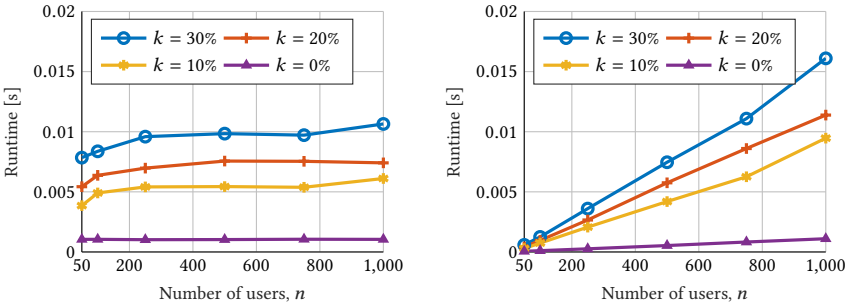


(a) Mean user

(b) Aggregator

(c) Verifier

Figure 3.2 Empirical runtime of MPVAS. Note the different y-axis scales.



(a) Mean user

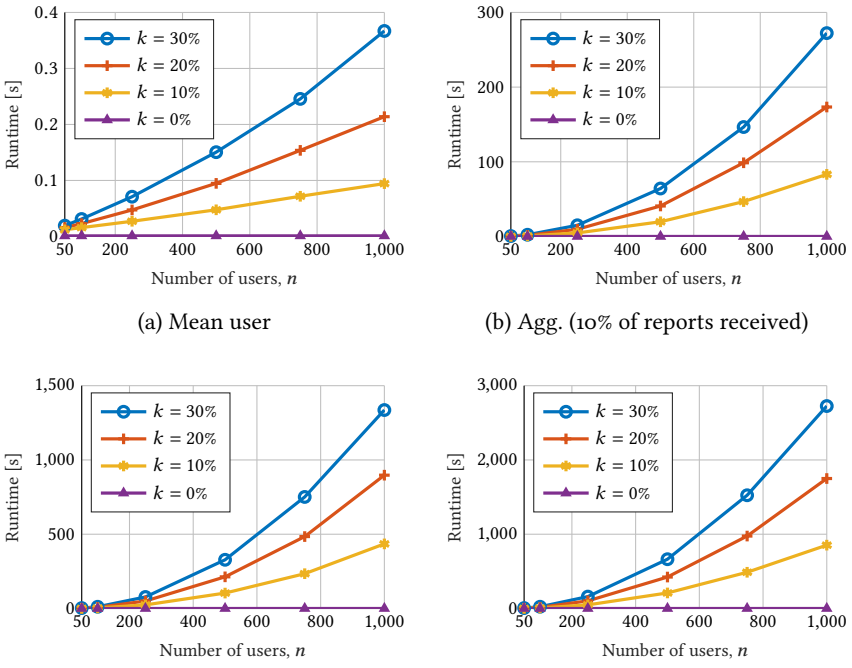
(b) Aggregator

Figure 3.3 Empirical runtime of MPVAS+

3.9.5 mPVAS-IV Runtime

mPVAS-IV (see Section 3.7) deals with malicious users causing malformed signatures. The aggregator identifies these users through several additional checks. Since the aggregator does not know the number of malicious users beforehand, they will check all signatures in order to find every user that acted maliciously during each round.

First, we consider the case in which users obtain invalid final user signatures during the signing phase and report these to the aggregator. Figure 3.4a shows the runtime for a single user. We see that the total runtime for a single user is comparable to that of the mPVAS scheme, as there are no additional steps required from users at this stage. For the aggregator, instead, the runtime is dependent on the number of reports received and the size k of each signing set. This dependence is clearly shown in Figure 3.4b, Figure 3.4c, and Figure 3.4d, in which we consider three cases where 10%, 50%, and 100% of users submit a report to the aggregator. In all cases, the runtime is noticeably higher than in the base mPVAS protocol. The reason for this steep increase is the additional exponentiations and pairings required to check whether each signature $\sigma_{j,i}^2$ is well-formed. Moreover, these checks must be repeated for every user in a signing set in order to find every possible instance of tampering or whether the report was actually false. We remark that our implementation does not include any specific optimizations, such as parallelization. Since verification is embarrassingly parallel for the aggregator, we expect this can be sped up linearly in the number of cores.



(a) Mean user

(b) Agg. (10% of reports received)

(c) Agg. (50% of reports received)

(d) Agg. (100% of reports received)

Figure 3.4 Empirical runtime of mPVAS-IV with final user signature tampering. Note the different y-axis scales.

Next, we consider the case in which malicious users submit malformed final user signatures to the aggregator. When this happens, verification of the aggregate signature fails, and the aggregator starts a procedure to identify the malicious users. The checks in this procedure must be performed on all n users, thus giving a linear complexity for the aggregator. Figure 3.5a and Figure 3.5b show the runtime for a single user and for the aggregator, respectively. We see that the runtime for the aggregator can reach up to 22 seconds on our machine. Despite the increased runtime for the aggregator, recall from Section 3.7 that this identification procedure is necessary only after malicious behavior has occurred. Our experiments represent the cumulative worst-case "denial of service" that malicious users can inflict on the aggregator and other users.

3.9.6 mPVAS-UD Runtime

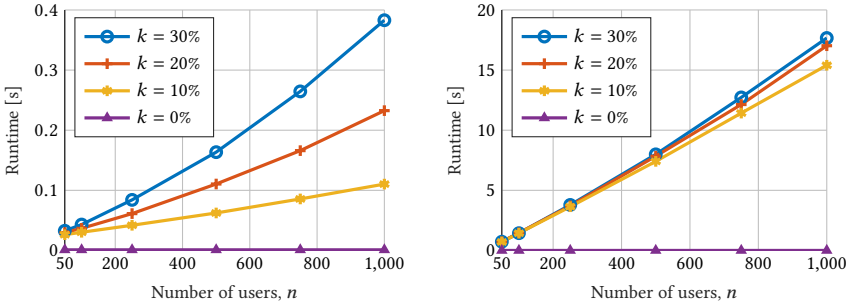
In mPVAS-UD (see Section 3.8), there are changes in the signing phase for users that drop out and in the verification phase for verifiers. We, therefore, focus on their runtimes in Figure 3.6. In our experiments, we fix the number of user dropouts to be 10%, 30%, and 50% of the total number of users.

Figure 3.6a shows the runtime for each dropped-out user. We find that the runtime is independent of the number of malicious users and the number of dropped-out users, which is expected since the protocol is non-interactive for these users and the recovery material can be computed in constant time.

From Figure 3.6b, Figure 3.6c, and Figure 3.6d we see that the runtime for the verifier is not constant anymore, unlike all other variants of mPVAS. In mPVAS-UD, the verifier's runtime is linear in the number of remaining users because it must compute the product of the masking factors EK_j for every remaining user j, as described in Equation 3.37 on page 51. Note that the product of the recovery keys $rk_{i,t}$ is precomputed by the aggregator before being sent to the verifiers to save bandwidth. Still, these multiplications not particularly expensive and, even in the worst case we consider, with 1000 users and only 10% dropouts, the total running time is below 0.0165 seconds.

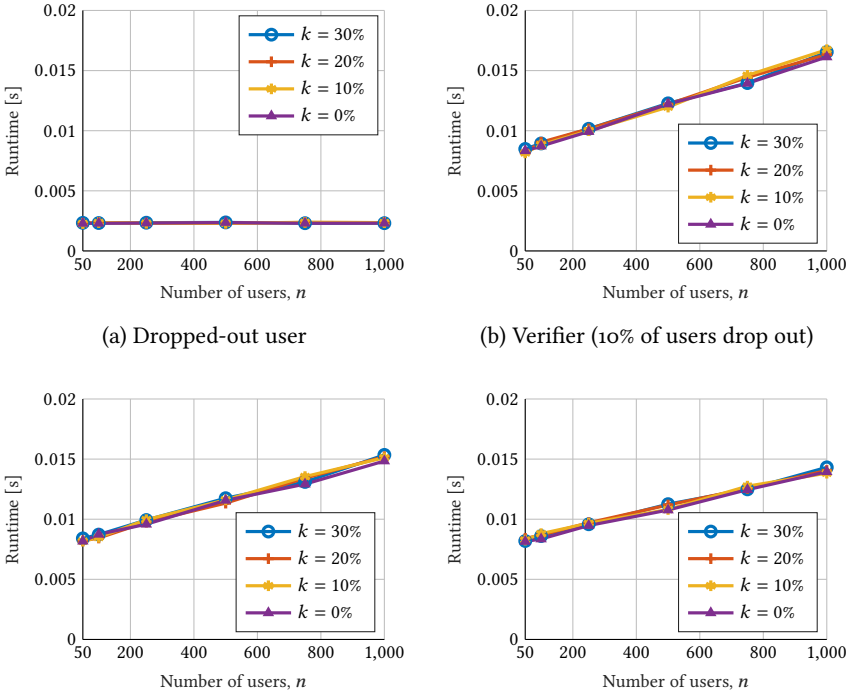
3.10 Conclusions

mPVAS and its extensions ensure the confidentiality of the input values and the integrity and authenticity of the aggregate even in the presence of a malicious aggregator and a subset of malicious users that collaborate to tamper with the result of the aggregation. Ensuring not only confidentiality but also integrity and authenticity even in the presence of malicious adversaries helps to develop more trust in the results of privacy-preserving schemes and make such schemes appealing to a wider range of scenarios.



(a) Mean user (with proof generation) (b) Agg. checks final user signatures

Figure 3.5 Empirical runtime of MPVAS-IV with aggregate signature tampering. Note the different y-axis scales.



(a) Dropped-out user (b) Verifier (10% of users drop out)

(c) Verifier (30% of users drop out) (d) Verifier (50% of users drop out)

Figure 3.6 Empirical runtime of MPVAS-UD

3.A Security Arguments for the mPVAS Family

In this appendix, we provide evidence of the unforgeability of the aggregate signature schemes of mPVAS. In order to do so, we adopt the concept of Aggregate Unforgeable (AU) [LL21, Leo+15, Emu+19], which denotes the notion that in round t, the aggregator cannot produce a valid proof of correctness σ_t for a sum that was not computed from inputs submitted by the registered users. Throughout this appendix, we regularly refer to the scheme by Shi et al. [Shi+11, Section 5], which we shall henceforth call PPATS.

Types of forgeries. We say that an adversary $\mathcal{A}$ successfully forges an aggregate signature σ_t for some round t if it outputs (sum_t, σ_t) such that $\text{Verify}(t, vk, \text{sum}_t, \sigma_t) = 1$ and $\text{sum}_t \neq \sum_{i=1}^{n} x_{i,t}$. In other words, $\mathcal{A}$ can provide a valid aggregate signature that successfully authenticates an incorrect sum. We distinguish between two types of forgeries [Bak+15, Leo+15, LL21, Emu+19, TDP16]:

- *Type-I*, when an adversary $\mathcal{A}$ forges an aggregate signature for a round t^* in which $\mathcal{A}$ did not see any signatures from the users, which implies forgeries for future rounds of the protocol, and
- *Type-II*, when an adversary $\mathcal{A}$ forges an aggregate signature for a round t^* in which $\mathcal{A}$ saw all signatures from the users, which implies forgeries for present or past rounds of the protocol.

3.A.1 Aggregator Obliviousness of mPVAS

We provide the proof of Theorem 4 on page 44. We first provide some preliminaries, and then restate the theorem as Theorem 12.

We show that if a probabilistic polynomial-time adversary has a non-negligible advantage of breaking the Aggregator Oblivious (AO) property of mPVAS, then it also has a non-negligible advantage of breaking the AO of PPATS, which is proven under the Decisional Diffie-Hellman (DDH) assumption. The proof of this property follows an indistinguishability-based game, and it provides an adversary with access to the following oracles.

- $O_{\text{Setup}}(1^\lambda)$: Performs the setup of the mPVAS scheme using the given security parameter λ and replies with the public parameters pp and the verification key vk. The secret values of each user $([s]_i, sk_i)$ are kept secret.
- $O_{\text{Compromise}^1}(i \in \mathbb{U})$: When queried on user i, the oracle replies with the secret of user i, namely $([s]_i, sk_i)$.
- $O_{\text{Sign}}(i \in \mathbb{U}, t, x_{i,t})$: Given an input $x_{i,t}$ of user i in round t, the oracle replies with $\left(\sigma^1_{i,t}, \{\sigma^{2,j}_{i,t}\}_{j \in \mathcal{U}_i}, \sigma^3_{i,t}, \sigma_{i,t}\right)$, where $\sigma^\ell_{i,t}$ is a final user signature for each $1 \leq \ell \leq 3$, and $\sigma_{i,t}$ is the final user signature.
- $O_{\text{Challenge}}(\mathcal{X}^0_{t^*}, \mathcal{X}^1_{t^*})$: Given two sets of input values $\mathcal{X}^0_{t^*}, \mathcal{X}^1_{t^*}$ of size $|\mathcal{X}^i_{t^*}| = n$, such that $\sum_{i \in U^*} x^0_{i,t^*} = \sum_{i \in U^*} x^1_{i,t^*}$, the oracle randomly flips a coin $b \leftarrow\!\!\$\ \{0,1\}$ and, for set $\mathcal{X}^b_{t^*}$, it returns all the corresponding partial and final user signatures of its inputs.

Aggregator Obliviousness security game. The AO security game is based on the game introduced by Shi et al. [Shi+11]. We derive the term AO from this game, following the definitions in [Leo+15, LL21].

Definition 1 (Aggregator Oblivious (AO))**.** Let $\Pr[\mathcal{A}^{\text{AO}}]$ denote the probability that aggregator $\mathcal{A}$ outputs $b^* = b$ in the AO game. A data aggregation protocol is said to be AO if any polynomially bounded $\mathcal{A}$ has negligible advantage $\Pr[\mathcal{A}^{\text{AO}}] \leq \frac{1}{2} + \text{negl}(\lambda)$ of winning the AO game.

Theorem 12 (Restatement of Theorem 4)**.** The MPVAS scheme is AO in the random oracle model under Symmetric External Diffie-Hellman (SXDH) in $\mathbb{G}_1$ and $\mathbb{G}_2$.

Proof. Let us assume an adversary $\mathcal{A}$ that can win the AO game with a non-negligible advantage. We show how a polynomial time algorithm $\mathcal{B}$ can break PPATS, which is provably secure under the DDH assumption, by using $\mathcal{A}$ as a subroutine. We refer to the following oracles provided by the PPATS scheme:

- $O^{\text{PPATS}}_{\text{Setup}}$ returns the public parameters.
- $O^{\text{PPATS}}_{\text{Encrypt}}$ returns the ciphertext $c_{i,t}$ of a given input $x_{i,t}$ in round t using PPATS.
- $O^{\text{PPATS}}_{\text{Compromise}}$ returns the secret encryption key sk'_i of a specified user $i \in \mathbb{U}$.
- $O^{\text{PPATS}}_{\text{Challenge}}$, only called once during the game, randomly flips a coin $b \leftarrow\!\!\$\ \{0,1\}$ and, similarly to the challenge phase described above, encrypts one of the two plaintext sets chosen by the adversary $\mathcal{X}^b_{t^*} = \{x_{i,t^*}\}_{i \in \mathbb{U}^*}$.

We follow the AO security game and show how $\mathcal{B}$ reacts to the queries of $\mathcal{A}$.

1. **Setup.** When $\mathcal{A}$ queries the $O_{\text{Setup}}(1^\lambda)$ oracle, $\mathcal{B}$ queries $O^{\text{PPATS}}_{\text{Setup}}(1^\lambda)$. The latter returns the public parameters $pp_{\text{PPATS}} = (H, \mathbb{G}_1, g_1, p)$. $\mathcal{B}$ also queries $O^{\text{PPATS}}_{\text{Compromise}}(0)$, which returns the secret key of the aggregator $sk_A = -\sum_{i=1}^{n} sk'_i$. $\mathcal{B}$ will additionally choose the remaining public parameters of MPVAS, $pp = (H, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e, k)$. Finally, $\mathcal{B}$ chooses the secret keys $\left(s, \{sk_i\}_{i \in \mathbb{U}}, \{\{ek_{i,j}\}_{1 \leq j \leq k}\}_{i \in \mathbb{U}}\right)$, creates n secret shares $[s]_i$ using $(k+1, n)$-Shamir secret sharing, and creates the verification key

$$vk = \left((g_2^s)^{-sk_A}, g_2^s\right) = \left((g_2^s)^{\sum_{i=1}^{n} sk'_i}, g_2^s\right). \tag{3.46}$$

   Finally, $\mathcal{B}$ returns pp and vk to $\mathcal{A}$.

2. **Learning.** Consists of three parts.

   Compromise. When $\mathcal{A}$ queries the $O_{\text{Compromise}^1}(i \in \mathbb{U})$ oracle, $\mathcal{B}$ will, in turn, query $O^{\text{PPATS}}_{\text{Compromise}}(i \in \mathbb{U})$ and return the corresponding secret key sk'_i of user i. Additionally, the secret share $[s]_i$ is sent to $\mathcal{A}$.

   Sign. When $\mathcal{A}$ calls $O_{\text{Sign}}(i \in \mathbb{U}, t, x_{i,t})$, $\mathcal{B}$ queries $O^{\text{PPATS}}_{\text{Encrypt}}(i \in \mathbb{U}, t, x_{i,t})$ to obtain $c^{\text{PPATS}}_{i,t} = H(t)^{sk'_i} g_1^{x_{i,t}}$. $\mathcal{B}$ then computes

$$\sigma^1_{i,t} = c^{\text{PPATS}}_{i,t} = H(t)^{sk'_i} g_1^{x_{i,t}}, \tag{3.47}$$

$$\sigma^{2,j}_{i,t} = H_1(t)^{ek_{j,i}} \left(\sigma^1_{i,t}\right)^{[s]^*_j}, \tag{3.48}$$

$$\sigma^3_{i,t} = \prod_{j \in \mathcal{U}_i} \sigma^{2,j}_{i,t}, \text{ and} \tag{3.49}$$

$$\sigma_{i,t} = H_1(t)^{ek_{i,i}} \cdot \left(\prod_{j \in \mathcal{U}_i} \sigma_{i,t}^{2,j} \right) \cdot \left(\sigma_{i,t}^{0} \right)^{[s]_i^*} \tag{3.50}$$

$$= H_1(t)^{ek_{i,i} + \sum_{j \in \mathcal{U}_i} ek_{j,i}} \cdot \left(H(t)^{sk'_i} g_1^{x_{i,t}} \right)^{s}. \tag{3.51}$$

Notice how each partial signature and the final user signature $\sigma_{i,t}$ are constructed from the ciphertext output by the encryption algorithm of PPATS but perfectly simulates a partial or final user signature of the MPVAS scheme. Finally, $\mathcal{B}$ returns $\left(\sigma_{i,t}^1, \{\sigma_{i,t}^{2,j}\}_{j \in \mathcal{U}_i}, \sigma_{i,t} \right)$ to $\mathcal{A}$.

Verify. $\mathcal{A}$ can use Equation 3.15 on page 43 to test the correctness of an aggregate sum using the verification key vk obtained during the setup.

3. **Challenge.** $\mathcal{A}$ chooses a set of uncompromised users $U^* \subseteq \mathbb{U}$, with $|U^*| \geq 2$ and an aggregation round t^* for which no sign queries were made in the learning phase. Then, $\mathcal{A}$ also chooses two sets of ciphertexts $X_{t^*}^0 = \{x_{i,t}^0\}_{i \in U^*}$ and $X_{t^*}^1 = \{x_{i,t}^1\}_{i \in U^*}$ such that $\sum_{i \in U^*} x_{i,t^*}^0 = \sum_{i \in U^*} x_{i,t^*}^1$. When $\mathcal{A}$ calls the $O_{\text{Challenge}}(X_{t^*}^0, X_{t^*}^1)$ oracle, $\mathcal{B}$ queries $O_{\text{Challenge}}^{\text{PPATS}}(X_{t^*}^0, X_{t^*}^1)$. The oracle flips a coin $b \leftarrow\$ \{0,1\}$ and returns the ciphertexts of the b^{th} set $\{c_{i,t}^{\text{PPATS}^b}\}_{i \in U^*}$. $\mathcal{B}$ computes the partial and final user signatures using the O_{Sign} oracle, returning

$$\left(\{\sigma_{i,t^*}^{1^b}\}_{i \in U^*}, \{\{\sigma_{i,t^*}^{2^b,j}\}_{j \in \mathcal{U}_i}\}_{i \in U^*}, \{\sigma_{i,t^*}^{3^b}\}_{i \in U^*}, \{\sigma_{i,t^*}^{b}\}_{i \in U^*} \right) \tag{3.52}$$

to $\mathcal{A}$. In particular, the final user signature is

$$\sigma_{i,t^*}^b = H_1(t^*)^{ek_{i,i} + \sum_{j \in \mathcal{U}_i} ek_{j,i}} \cdot \left(H(t^*)^{sk'_i} g_1^{x_{i,t^*}^b} \right)^{s} \tag{3.53}$$

for $i \in U^*$. Notice how σ_{i,t^*}^b, and all partial signatures are computed from the ciphertexts output by the encryption algorithm of the PPATS scheme and perfectly simulate the ciphertexts, partial and final user signatures of the MPVAS scheme. The aggregation of all such final user signatures is also valid and correctly verified using the verification key vk as $\sigma_{t^*}^b = \prod_{i \in U^*} \sigma_{i,t^*}^b = (H(t^*)^s)^{\sum_{i \in U^*} sk'_i} \left(g_1^s\right)^{\sum_{i \in U^*} x_{i,t^*}^b}$.

If $\mathcal{A}$ has a non-negligible advantage ε of guessing the correct bit b^* in the AO game of the MPVAS scheme, then $\mathcal{B}$ can also win the AO game of the PPATS scheme with the same non-negligible advantage ε by guessing the same bit b^*. This would contradict the DDH assumption in $\mathbb{G}_1$, because the security of the PPATS scheme relies on this assumption. Additionally, if the DDH does not hold in $\mathbb{G}_1$, then the SXDH assumption does not hold either since it requires that the DDH problem be hard in $\mathbb{G}_1$. Therefore, the MPVAS scheme is AO in the random oracle model under the SXDH assumption. ▫

3.A.2 Aggregate Unforgeability of MPVAS

We restate Theorem 5 as Theorem 13 and provide a proof. Recall Appendix 3.A.

Theorem 13. MPVAS is AU against type-I and type-II forgeries.

Proof. We prove both types of unforgeability.

Type-I unforgeability [Bak+15, Leo+15, LL21, Emu+19, TDP16]. A type-I forgery occurs when the aggregator outputs a valid aggregate signature σ_t in a round t without receiving any users' signatures. Thus, the aggregator can only use knowledge from previous rounds or by colluding with users. First, we note how signatures from different rounds are incompatible with each other. Assuming a cryptographically-secure hash function $H\colon \{0,1\}^* \to \mathbb{G}_1$ under the random oracle model its output can be considered random. As such, in each round t, each signature has a different random factor $H(t)$. Even assuming the aggregator chooses the round identifier t, because of the collision resistance property of H, it has a negligible probability of finding two different round identifiers t, t' such that $H(t) = H(t')$. Similarly, because of the second pre-image resistance property of H, given t, the aggregator has negligible probability of finding another t' such that $H(t) = H(t')$. The same arguments hold for the other hash function $H_1\colon \{0,1\}^* \to \mathbb{G}_1$ used in the protocol. It follows that the aggregator cannot reuse signatures from previous rounds. The only other option left for the aggregator is to construct new signatures itself. However, in order to do so, all secret signing keys sk_i are required but, assuming colluding users, the aggregator has only access to at most k of them. The aggregator also needs the secret exponent s to compute a valid signature, but it has only access to at most k secret shares of s, which are not enough to reconstruct s.

Type-II unforgeability [Bak+15, Leo+15, LL21, Emu+19, TDP16]. There are two pieces of information that can allow the aggregator to successfully forge an aggregate signature in a round in which it received all signatures from the users: the secret exponent s or the factor g_1^s. The exponent s is secret-shared by all users using $(k+1, n)$-Shamir secret sharing. Since we assume at most k malicious users who collude with each other and k shares leak no information about the underlying secret s, then no dishonest party can directly learn s from exchanging their shares. Additionally, if $g^s \in \mathbb{G}_1$ is known, for any $g \in \mathbb{G}_1$, recovering s is considered computationally infeasible because Discrete Logarithm Problem (DLP) is assumed to be hard in $\mathbb{G}_1$. The same argument applies to the value $g_2^s \in \mathbb{G}_2$, which is part of the verification key and known by every verifier. Note that, while a malicious actor may know g_2^s, since the Co-Computational Diffie-Hellman (CO-CDH) [BLS04] problem is assumed to be hard in $\mathbb{G}_1$ and $\mathbb{G}_2$, obtaining g_1^s is still considered hard.

In a malicious setting where users can behave arbitrarily, sending malformed signatures may allow them to gain additional information that will allow them to break the AU property.

Each signature starts with the form $\sigma_{i,t}^1 = H(t)^{sk_i} g_1^{x_{i,t}}$. Clearly, any malicious party can immediately tamper with this signature since g_1 is public. However, the aggregator is required to send a *Compute final user signature* request to every user in order for the aggregate signature to be successfully verified. From Equation 3.10, any $\sigma_{i,t}^3$ that was not computed using user i's original $\sigma_{i,t}^1$ will lead to an invalid $\sigma_{i,t}$, because the bases of the two factors will not match. This, in turn, will lead to an invalid aggregate signature σ_t.

In order to prevent malicious actors from manipulating any of the partial signatures to obtain g_1^s, each user j is required to further mask any response to a *Create partial signature* or *Compute final user signature* request with a fresh masking factor $H_1(t)^{ek_{j,i}}$. Assuming at most k malicious users, any signing request from each user i will result in

a $\sigma_{i,t}$ containing at least one such factor, since either user i itself or a user in its signing set must be honest by assumption. Therefore, all that malicious actors can learn by deviating from the protocol is of the form $H_1(t)^{\varepsilon} g^{s}$ or $H_1(t)^{\varepsilon} g^{[s]_j^*}$, where $g \in \mathbb{G}_1$ and ε indicates the sum of any non-empty subset of masking exponents $ek_{j,i}$. None of these values can be used to successfully forge a valid aggregate signature, as they would introduce extra masking exponents that would not sum up to zero anymore. As a result, the verification algorithm will fail. ❐

3.A.3 Aggregate Unforgeability of MPVAS-IV

We restate Theorem 9 as Theorem 14 and provide a proof. Recall Appendix 3.A.

Theorem 14. MPVAS-IV is AU against type-I and type-II forgeries.

Proof. In the MPVAS-IV extension, the aggregator is trusted to detect users who may attempt disrupt the normal execution of the protocol and not to disrupt the protocol itself. However, the aggregator is still considered malicious with respect to the unforgeability property of the MPVAS-IV extension. This extension provides the aggregator with additional knowledge that is not available in the main scheme. As such, in this section, we provide additional arguments to show why AU is still maintained in the MPVAS-IV extension.

Type-I forgeries. The new pieces of information that the aggregator is handed in the MPVAS-IV extension are the sets $SS_i = g_2^{[s]_i}$, $SK_i = g_T^{sk_i} h_T^{r_i}$, $EK_{j,i} = g_2^{ek_{j,i}}$, $EKS_{j,i} = g_2^{ek_{j,i}/s}$, and the value $v = g_2^{1/s}$. Since DLP is assumed to be intractable in $\mathbb{G}_2$, the aggregator has a negligible probability of obtaining the secret share $[s]_i$ or the masks $ek_{j,i}$ of a user i from $g_2^{[s]_i}$ and $g_2^{ek_{j,i}}$, respectively. Similarly, finding $\frac{1}{s}$ from v is also hard. Additionally, because of the perfect hiding property of Pedersen commitments, the aggregator cannot learn any information about the signing key sk_i from its corresponding commitment in SK_i. Hence, the additional information that is handed to the aggregator in the MPVAS-IV extension gives the aggregator no advantage of learning the necessary information to create type-I forgeries.

Type-II forgeries. There is no additional piece of information handed to the aggregator in the MPVAS-IV extension that could allow it to create type-II forgeries. Intuitively, this is because all of the additional values are members of either $\mathbb{G}_2$ or $\mathbb{G}_T$, but the signatures are elements of $\mathbb{G}_1$. As such, there is no additional information that could be used by the aggregator to tamper with the signatures in $\mathbb{G}_1$, assuming the SXDH assumption holds in the chosen pairing group. ❐

3.A.4 Aggregator Obliviousness of MPVAS-UD

We recall Theorem 10 and provide a proof. Recall Appendix 3.A.1.

Theorem 10. MPVAS-UD is AO.

Proof. In the MPVAS-UD protocol, the signing phase is identical to that of the main MPVAS protocol for all remaining users. As such, the signature of each remaining user perfectly hides the input value, as proven in Theorem 12.

The signing phase is, however, different for dropped-out users. Each user is required to send a recovery key $rk_{i,t}$ for every round t it wish to drop out of. This value could, in turn, be plugged into Equation 3.24 on page 48, using $EK_{j,i}$, to find user i's secret input value by brute force. Fortunately, the signing key sk_i is bound to the generator $H(t)$ and, thus, can only be successfully used in round t, during which dropped-out users do not submit any data. Additionally, if malicious users collaborate to create k additional recovery factors $rk_{i,t}$, then, as long as at least two honest users do not drop out during round t, only the sum of their input data can be computed, but not the individual values.

As a result, the input data of both remaining and dropped-out users remain private during every step of the protocol. ❐

3.A.5 Aggregate Unforgeability of mPVAS-UD

We restate Theorem 11 as Theorem 15 and provide a proof. Recall Appendix 3.A.

Theorem 15. mPVAS-UD is AU against type-I and type-II forgeries.

Proof. The main addition introduced by the mPVAS-UD extension is the recovery key $rk_{i,t}$ that users that wish to exit the protocol during some round t submit to the aggregator. The recovery key is computed over $\mathbb{G}_T$, so it cannot be used to directly affect the signatures, which are elements of $\mathbb{G}_1$.

The aggregator cannot lie about the set of users that drop out during any given round t and cannot publish more than one valid aggregate signature. Forcing a subset of users out of the protocol would lead to a failed verification, since the aggregator cannot provide the verifiers with valid recovery keys for the missing users on its own. Similarly, the aggregator cannot force a dropped-out user i in the protocol as it does not possess its signing key sk_i. Assuming a subset of users colludes with the aggregator and provides it with valid recovery keys, we identify two cases.

If any of these users actually engage in the protocol, then their recovery key alone would not suffice anymore, because their final user signature would contain at least one masking factor $H_1(t)^{ek_{j,i}}$, with $ek_{j,i}$ belonging to an honest user j, which has not been redistributed among the remaining users. As such, the check in Equation 3.37 would fail. Otherwise, if they do not submit anything, then the aggregator is forced to forward their recovery keys, indicating they have indeed dropped out, otherwise, the verification would fail.

As for type-I forgeries, the same arguments presented in Theorem 5 similarly apply to mPVAS-UD. ❐

Part III

Reconstruction Attacks

Chapter 4

Topology-Based Reconstruction Prevention for Decentralised Learning

Abstract. Decentralised learning has recently gained traction as an alternative to federated learning in which both data and coordination are distributed over its users. To preserve the confidentiality of users' data, decentralised learning relies on differential privacy, multi-party computation, or a combination thereof. However, running multiple privacy-preserving summations in sequence may allow adversaries to perform reconstruction attacks. Unfortunately, current reconstruction countermeasures either cannot trivially be adapted to the distributed setting, or add excessive amounts of noise.

In this work, we first show that passive honest-but-curious adversaries can infer other users' private data after several privacy-preserving summations. For example, in subgraphs with 18 users, we show that only three passive honest-but-curious adversaries succeed at reconstructing private data 11.0% of the time, requiring an average of 8.8 summations per adversary. The success rate depends only on the adversaries' direct neighbourhood, and is independent of the size of the full network. We consider weak adversaries that do not control the graph topology, cannot exploit the inner workings of the summation protocol, and do not have auxiliary knowledge; and show that these adversaries can still infer private data.

We develop a mathematical understanding of how reconstruction relates to topology and propose the first topology-based decentralised defence against reconstruction attacks. Specifically, we show that reconstruction requires a number of adversaries linear in the length of the network's shortest cycle. Consequently, exact reconstruction attacks over privacy-preserving summations are impossible in acyclic networks.

Our work is a stepping stone for a formal theory of topology-based decentralised reconstruction defences. Such a theory would generalise our countermeasure beyond summation, define confidentiality in terms of entropy, and describe the interactions with (topology-aware) differential privacy.

Based on: **Florine W. Dekker**, Zekeriya Erkin and Mauro Conti. "Topology-based reconstruction prevention for decentralised learning". In: *Proceedings on Privacy Enhancing Technologies* 2025.1 (2025), pages 553–566. DOI: 10.56553/POPETS-2025-0030.

4.1 Introduction

Machine learning is used in a wide array of systems, including malware detection [Rie+11], predictive text [Bon+17], and smartwatches [Wei+16]. These systems require access to large amounts of reliable data in order to function accurately. In practice, the necessary data usually exist, but are distributed over many data owners. The naive approach for data collection is to have the data owners send their data to a central server, which trains a machine learning model on these data before deploying it. However, sharing private data may result in misuse, for example in the form of targeted advertising or harassment. In an age of increasing privacy awareness, data owners may be reluctant to share their data, threatening the viability of data-intensive machine learning applications.

The emerging field of federated learning, first formalised in [McM+17], addresses these privacy issues by distributing the training process over the data owners. Instead of submitting their data, each data owner first trains a machine learning model on their local data and then submits this model to a central server. This central server, called the aggregator, uses a privacy-preserving summation protocol to combine the received models into a single global model. The central server then sends back the global model to the data owners, who apply another round of training, repeating the entire process until the global model has converged.

A significant drawback of classical federated learning is that communication is a bottleneck, scaling quadratically [Bon+17] or poly-logarithmically [Bel+20] in the number of users. Decentralised learning, a variant of federated learning [Kai+21], removes this bottleneck by distributing both the data and the coordination between users. Training happens in a peer-to-peer fashion, with users exchanging information only with their direct neighbours. This significantly reduces the communication complexity [Lia+17], allowing for cost-effective deployments without a central server. Furthermore, because communication is local, it becomes much harder for adversaries to observe the full network [Tro+17].

Recently, there has been increased interest in decentralised learning. Though some works do not consider privacy [Lia+17, Tan+18, ZBT20], many other works do. Some of these works [VBT17, Bel+18, ZBT20] consider algorithms in which nodes are randomly selected to calculate updates, and protect the private data underlying the models using differential privacy. That is, they apply carefully calibrated random noise to the calculated gradients before sharing them with others. A slight variation of this is to use a random walk through the graph to determine the order in which updates occur [Cyf+22]. There are also works [Che+18a, Qu+20, Sch+20] that use blockchains to facilitate the communication and coordination between nodes, and then similarly use differential privacy. Finally, instead of differential privacy, some works utilise multi-party computation [Dan+18, Kan+20, Tra+21], which does not give noisy results, but has higher computational costs.

A common thread in these works is that they apparently assume that if a single summation is secure, then the protocol remains secure after multiple summations. However, this requires further scrutiny, as combining information from multiple rounds may reveal previously hidden information. For example, given private records A, B, and C, and a privacy-preserving summation protocol, an adversary could separately query $A + B$, then $B + C$, and finally $A + C$, and use a linear algebra solver

to learn all three private records. To defend against such attacks, one must prevent sequences of queries that would reveal private data. Naive restrictions, such as requiring a minimum number of included records per query, are insufficient: The adversary could still first query the sum of all models and then query the sum of all models except one, allowing them to reconstruct the excluded model. As such, designing proper countermeasures requires a formal theory.

Extracting data from output traces is known as a reconstruction attack, which has its roots in the theory of statistical disclosure [Fel72]. Many defences have been proposed since the 1970s, including query auditing [CÖ82], perturbation [Dwo06], and random sampling [Den80]. However, these works assume either a central database, or otherwise assume a central arbiter that determines which queries are allowed. In decentralised learning, there is no clear leader who can be trusted to audit queries. Instead, decentralised learning requires a decentralised solution. Apart from works on perturbation, to the best of our knowledge, only da Silva et al. [dSil+04] have considered reconstruction attacks in peer-to-peer networks, but their work applies only to distributed clustering, and does not propose any countermeasures. When considering perturbation, naively applying user-level differential privacy in a distributed setting results in linearly-scaling noise, severely reducing the protocol's utility [DR14, ZMW17, Cyf+22]. Intuitively, utility can be increased while retaining the level of privacy by correlating noise by topology [Dwo06], but to the best of our knowledge only a few works have done this. Guo et al. [Guo+22] reduce noise based on the mutual overlaps of neighbours' neighbourhoods, but do not consider time-series correlations. Cyffers et al. [Cyf+22] observe that data sensitivity decreases as mutual node distances increase, but their solution does not scale well under collusion.

In this work, we analyse reconstruction attacks performed by colluding adversaries in peer-to-peer networks. We model the network after decentralised learning, though our analysis is sufficiently generic to describe a sequence of summations in any environment. Summation is a simple protocol, but is sufficient to implement many of the aforementioned decentralised learning protocols, in addition to smart metering [GJ10] and even principal component analysis, singular-value decomposition, and decision tree classifications [Blu+05]. We assume a set of nodes, each with a private datum that changes over time, and allow privacy-preserving summation over one's direct neighbours. We do not consider auxiliary knowledge; see Section 4.3.3 and [Cor+13, CT13] for a detailed discussion on the real-world applicability of this model. We then formalise the relation between reconstruction and network topology, and prove that exact reconstruction attacks are impossible in a specific class of topologies.

Concretely, we begin by showing that reconstruction attacks are practical, and that, in random peer-to-peer subgraphs, three honest-but-curious adversaries with 15 neighbours succeed in finding at least one neighbour's private datum with an 11.0% success rate, requiring an average of only 8.8 rounds per adversary. The success rate is independent of the size of the full network; it depends only on the adversaries' local neighbourhood. We then show that the success rate depends on the connectivity of the network rather than its size. Specifically, we show that reconstruction corresponds to cycles in the graph: If the graph's shortest cycle has length $2k$, then reconstruction never succeeds if there are fewer than k adversaries. Finally, we briefly evaluate the impact of increasing girth on the convergence of a distributed averaging protocol, and

find that while all graphs require more rounds to achieve convergence, dense graphs are affected less when "stretched" to higher girths.

To the best of our knowledge, our work is the first to propose a topology-based decentralised defence to reconstruction attacks. We show that restricting how summations may be composed makes it impossible to reconstruct private data. We assume that adversaries do not have auxiliary knowledge, as restrictions on summations cannot be guaranteed otherwise. With the ultimate goal of developing a general theory of structured composition as a distributed reconstruction countermeasure, future work may include finding a condition that is not only sufficient (as seen in this work) but also necessary for reconstruction, generalising these countermeasures to operations beyond summation, stronger notions of privacy rooted in information theory, and investigating the interactions with (topology-aware) differentially private noise.

The remainder of this paper is structured as follows. In Section 4.2, we discuss related work. In Section 4.3, we describe the preliminaries: We explain basic primitives, formalise our assumptions, and introduce our notation. In Section 4.4, we formally describe reconstruction attacks, and show that the attack is feasible. In Section 4.5, we prove that the success rate of the reconstruction attack depends on the graph's girth, and investigate how girth affects application performance. Finally, in Section 4.6, we present our conclusions.

4.2 Related Work

In this work we propose a decentralised reconstruction countermeasure for privacy-preserving summation with dynamic data. To the best of our knowledge, this exact problem has not been treated in literature before. Therefore, in this section, we consider related works from various fields, and describe their similarities and differences.

4.2.1 Reconstruction Attacks

Consider a database that users can query for statistical information. For example, in a database with employee records, users can query for the sum of salaries of all PhD students. Naturally, the database must ensure that users cannot learn individual employees' salaries. A naive defence would be to disallow queries over single records. However, a clever adversary would still be able to reconstruct private data. For example, the user could query the sum of salaries of all employees, and the sum of salaries of all employees except Jay Doe, and reconstruct Jay Doe's salary from that.

The attack described above is known under various names: *statistical disclosure*[1], the *inference problem*, and the *reconstruction attack*. It has been the subject of research since at least the 1970s [Fel72], originally in the context of releasing census statistics. Since then, many reconstruction defences have been proposed, including random sampling [Den80], query auditing [CÖ82], and perturbation [Dwo06].

Most related to our research question are those works that consider sum queries only. Chin [Chi78] studies summation query graphs to determine the exact conditions under which disclosure occurs. However, his analysis is limited to queries that are over exactly two records each, and cannot easily be generalised. Wang, Wijesekera and Jajodia

[1] Confusingly, the term "statistical disclosure attack" is also a separate attack in peer-to-peer literature [Dan03], but this is an unrelated attack on anonymity rather than confidentiality.

[WWJ02] allow queries over more than two records. The authors propose cardinality-based criteria for determining whether reconstruction is possible, and create a whitelist of summations that can be performed without allowing reconstruction.

All aforementioned solutions consider a single trusted database or auditor, making them unsuitable for peer-to-peer protocols, in which the data are spread over many users. Except for perturbation-based techniques, there are very few works that consider reconstruction defences in peer-to-peer settings. In their study on reconstruction attacks in distributed environments, Jebali, Sassi and Jemai [JSJ19] note only the work by da Silva et al. [dSil+04] when discussing peer-to-peer solutions, but the latter applies only to distributed clustering, and does not propose any countermeasures.

Perturbation, on the other hand, has been studied in more detail. Probably the most popular perturbation mechanism for the decentralised setting is local differential privacy [War65, EGS03, Kas+08], a variation of differential privacy [Dwo06]. With this technique, when a query is performed over some set of nodes, each node adds a small amount of noise such that the aggregate is relatively accurate, but reconstruction remains impossible even after multiple queries. Various fully-decentralised learning protocols use local differential privacy to allow learning a shared machine learning model without revealing users' private datasets [VBT17, Bel+18, ZBT20]. However, the perturbation is calibrated to protect individual records in users' private datasets, rather than protecting users' entire datasets. As a result, these works are potentially vulnerable to inversion attacks [HAP17, Wan+19]. The level of noise can be increased, but this severely impacts utility [ZMW17, Cyf+22]. Intuitively, noise can be made more "efficient" by exploiting correlations between users' data [DR14], which, in peer-to-peer networks, amounts to calibrating noise to the topology. To the best of our knowledge only a few works have done this. Guo et al. [Guo+22] reduce noise based on the mutual overlaps of neighbours' neighbourhoods, but do not consider time-series correlations. Cyffers et al. [Cyf+22] observe that data sensitivity decreases as mutual node distances increase, but their solution does not scale well when adversaries collude.

4.2.2 Multi-Party Computation

In secure multi-party computation, composability [Lin03] is the property of a cryptographic scheme that no additional leakage occurs when it is invoked multiple times, with varying parties, combined with other schemes, and so on. There are numerous frameworks to model composability, including universal composability [Can01], constructive composability [Mau11], and reactive simulatability [BPW07].

Composability solves a different issue than the one posed in this work. While composability ensures nothing leaks beyond what can be inferred from the outputs, our work is concerned exactly with that which can be inferred from the outputs. Composability does not help when the desired output (implicitly) reveals private data.

In secure multi-party computation literature, this difference is occasionally acknowledged. For example, Bogdanov et al. [Bog+14] note that "the composition of ideal functionalities is no longer an ideal functionality", and, before them, Yang et al. [Yan+10] made a similar observation. There are more works that consider this difference, but, to the best of our knowledge, these works all resolve the issue by removing or protecting intermediate values, but do not consider protocols which desire intermediate values, and even then do not consider that reconstruction attacks may be

possible after multiple instantiations of the protocol. An exception is the work by Dekker and Erkin [DE21], which releases intermediate values in a structured manner such that it is not possible to reconstruct all users' values. However, the authors do not prove (or disprove) that it is impossible to find a *single* user's value.

4.3 Preliminaries

We briefly explain some basics on privacy-preserving summation in Section 4.3.1 and on bipartite graphs in Section 4.3.2. After that, we formulate our assumptions and define our notation in Section 4.3.3.

4.3.1 Privacy-Preserving Summation

Privacy-preserving summation is a special case of multi-party computation in which an aggregator calculates the sum of users' private values without learning the users' individual values. In this work, we consider privacy-preserving summation to be an information-theoretically secure black-box that reveals only the identities and the sum of the variables.

4.3.2 Bipartite Graphs

A bipartite graph $H = (U, V, E)$ is a graph with nodes $U \cup V$ and edges E, subject to $U \cap V = \emptyset$ and $\forall (u, v) \in E : u \in U \Leftrightarrow v \in V$.

Furthermore, a bipartite graph $H = (U, V, E)$ can be described by a biadjacency matrix $A \in \{0, 1\}^{|U| \times |V|}$, where $\forall 0 \leq u < |U|, 0 \leq v < |V| : A_{u,v} = 1 \Leftrightarrow (U_u, V_v) \in E$.

In this work, all graphs are undirected.

4.3.3 Assumptions and Notation

The underlying models and assumptions in this work are based on those seen in the decentralised learning literature [Dan+18, Bel+18, ZBT20], but are especially close to the work by Vanhaesebrouck, Bellet and Tommasi [VBT17].

In general, we denote the first element of a vector v by v_0, the first row of a matrix A by A_0, the range of integers $\{0 \dots n-1\}$ by $[\![n]\!]$, and the cardinality of a collection S by $|S|$.

User data and objectives

Consider a system of n users V, each with a private datum. Each datum is dynamic; it changes each time the user initiates a round and incorporates new knowledge from their neighbours. (We describe the time model in Section 4.3.3.) Each datum can be a vector of values, though for simplicity we assume scalar values in our notation. Examples of dynamic data are power consumption, GPS coordinates, and machine learning models. In round t, the data of user $i \in [\![n]\!]$ is denoted $\theta_{i,t}$.

The users want to compute some function over their data without revealing their data to others. Each user regularly runs a privacy-preserving summation protocol to find the sum of their direct neighbours' private data. This sum can be used for principal component analysis, singular-value decomposition, or distributed gradient descent, for example.

Network model

Users communicate with each other in a peer-to-peer network. This can be a physical network, for example based on Bluetooth or Wi-Fi Direct, or an overlay network, in which users are connected through the Internet. We model the network as an undirected, self-loopless, static graph $G = (V, E)$ in which each node represents a user. (We consider graphs with dynamic edges in Section 4.5.4.) The direct neighbours of a node $v \in V$ are denoted $N_G(v)$, and for any set of users $U \subseteq V$ we define their shared neighbours $N_G(U) \coloneqq \bigcup_{u \in U} N_G(u) \setminus U$. The network topology is not private; in fact, users know who their direct neighbours are. Users may run a privacy-preserving summation protocol to learn the sum of their direct neighbours' private values.

Adversarial model

We assume all n users V are honest-but-curious. That is, all users honestly follow the protocol, but may attempt to obtain other users' private data by operating on the data obtained in the protocol in any way they see fit. Additionally, k users $C \subseteq V$ may collude with each other, but we require that each adversary has either zero or at least two non-adversary neighbours, as retrieving private data is trivial otherwise. We give an example of a valid set of adversaries in Figure 4.1. Colluding users are still honest-but-curious, so their collusion is limited to sharing information outside the protocol. While excluding all actively malicious behaviour is a strenuous assumption in practice, we argue that the challenges in the honest-but-curious model are already sufficiently interesting to warrant investigation. We leave stronger notions of adversarial behaviour to future work; see also Section 4.6.

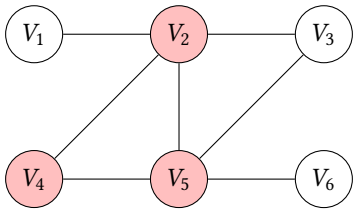


Figure 4.1 A network with 6 users V. The adversaries $C = \{V_2, V_4, V_5\}$ are shaded. Removing edge (V_2, V_3) would violate our requirements, as adversary V_2 would have exactly one non-adversary neighbour.

Finally, we assume that adversaries do not possess auxiliary knowledge. That is, we aim for syntactic privacy [CT13], of which the privacy guarantees do not compose trivially with those of other protocols using the same private data. Syntactic privacy is suitable when high utility is desired and participants have some level of mutual trust [Cor+13, CT13]. Moreover, prescribing a syntax on the data is inherent to this work's goal of establishing an interpretable relation between privacy and topology. We note that syntactic privacy does not preclude the use of semantic protections such as differential privacy, though the investigation of that combination is out of scope for this work. See [Cor+13, CT13] for a detailed discussion of the subject.

Time model

We work in the asynchronous time model [Boy+06], in which a global clock ticks whenever a user wakes up and performs some work. Equivalently, each user has their own clock ticking at the speed of a rate-1 Poisson process; when a user's clock ticks, that user wakes up. We denote the current global round number by t (for "time").

4.4 Reconstruction in Multi-party Summation

In this section we formally define reconstruction attacks in privacy-preserving multi-party dynamic-data summation, and experimentally verify that this attack is feasible. Adversaries passively record the summations they obtain throughout the protocol. Because adversaries know which users are included in which summation, they obtain a system of linear equations. Even if the system has no global solutions, adversaries may still learn the private data of some users.

In Section 4.4.1, we informally explain reconstruction attacks with examples. In Section 4.4.2, we give an exact definition of the adversaries' knowledge. In Section 4.4.3, we formally define reconstruction on multi-party dynamic-data summation. In Section 4.4.4, we experimentally verify the feasibility and success rate of reconstruction attacks on random graphs.

4.4.1 Introduction to Reconstruction Attacks

For this brief introduction, we use somewhat informal notation. We formally define our notation in Section 4.4.2.

A small example. Consider a graph $G = (V, E)$ with users V and a set of k adversaries $C \subseteq V$. If a single adversary $c \in C$ sums their neighbours' values, they learn a linear equation Θ_c over the private values θ of neighbours $N_G(c)$. If multiple adversaries C collude, they share a *system* of linear equations $A\theta = \Theta$ over the private values θ of $N_G(C)$. If the system of linear equations has a solution, then the adversaries are able to calculate all observed users' private values using linear combinations of the system's rows. For example, given adversaries A, B, and C with observations

$$\begin{aligned} \theta_1 + \theta_2 &= \Theta_A, \\ \theta_1 + \theta_3 &= \Theta_B, \text{ and} \\ \theta_2 + \theta_3 &= \Theta_C, \end{aligned} \tag{4.1}$$

this is equivalent to the system of linear equations

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \theta = \Theta. \tag{4.2}$$

Since this system is full rank, adversaries can calculate

$$\theta_1 = \frac{\Theta_A + \Theta_B + \Theta_C}{2} - \Theta_C, \tag{4.3}$$

$$\theta_2 = \Theta_A - \theta_1, \text{ and} \tag{4.4}$$

$$\theta_3 = \Theta_B - \theta_1. \tag{4.5}$$

For example, if $\Theta_A = 7$, $\Theta_B = 13$, and $\Theta_C = 8$, the adversaries know with certainty that $\theta_1 = 6$, $\theta_2 = 1$, and $\theta_3 = 7$. Observe that this works even if each individual summation in Equation 4.1 is information-theoretically secure.

Partial solutions. If the system is rank-deficient, no unique solution exists, but the system may still have partial solutions. That is, even if a system has infinitely many possible solutions, it may be the case that some variables have the same value in all solutions. Even a single user's private value being leaked is a major issue for any privacy-preserving protocol. Consider, for example, the adversarial knowledge consisting of

$$\begin{array}{ccccccc} \theta_1 & + & \theta_2 & + & \theta_3 & = & \Theta_A \text{ and} \\ \theta_1 & + & \theta_2 & & & = & \Theta_B. \end{array} \tag{4.6}$$

Even though there is no unique solution, all solutions have the same value for θ_3, calculated as $\theta_3 = \Theta_A - \Theta_B$.

The case of Equation 4.6 is trivial because Θ_B is the sum over a subset of Θ_A. However, there are also rank-deficient systems in which no summation is a subset of another:

$$\begin{array}{ccccccccc} \theta_1 & + & \theta_2 & + & \theta_3 & & & = & \Theta_A, \\ \theta_1 & + & \theta_2 & & & + & \theta_4 & = & \Theta_B, \text{ and} \\ & & & & \theta_3 & + & \theta_4 & = & \Theta_C. \end{array} \tag{4.7}$$

This system, too, has an infinite number of solutions, but each possible solution has the same values

$$\theta_3 = \frac{\Theta_A + \Theta_C - \Theta_B}{2} \text{ and} \tag{4.8}$$

$$\theta_4 = \frac{\Theta_B + \Theta_C - \Theta_A}{2}. \tag{4.9}$$

Time dimension. The above examples do not take into account that users' data change over time. To model dynamic data, first recall from Section 4.3.3 that users update their values only after initiating a summation. Since each update requires an interactive summation, users implicitly inform their neighbours whenever they update; and since each update represents the introduction of a new unknown value to θ, adversaries can represent an update by adding a new column to their adversarial knowledge. If a user updates their value multiple times before being observed by an adversary, the adversaries treat this as a single update.

To give an example, consider adversaries C and their neighbours $N_G(C)$ in Figure 4.2. If adversaries C_1 and C_2 run their summations, they learn

$$\left[\begin{array}{c|c|c} 1 & 1 & 0 \\ 1 & 0 & 1 \end{array}\right] \theta = \Theta. \tag{4.10}$$

The added vertical lines group the columns per non-adversarial user. Next, say that user N_1 updates their private value. This is noticed by the adversaries, who insert a new column into their system of equations. If user C_1 then does another summation (including user N_1's new value), the adversaries know the system

$$\overbrace{\left[\begin{array}{cc|c|c} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{array}\right]}^{N_1} \theta = \Theta. \tag{4.11}$$

The last row represents adversary C_1's new summation, and the second column represents user N_1's new value. Finally, if users N_1, N_2, and C_1 subsequently update (in that order), then users N_1 and N_2 each get a new column, and C_1's update adds a new row, giving

$$\overbrace{\left[\begin{array}{cc|c|c} 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{array}\right]}^{N_1 \qquad N_2} \theta = \Theta. \tag{4.12}$$

In the remainder of this work, to simplify notation, we will always assign the same number of columns t to each user.

Observations. Before we give a formal definition of reconstruction attacks, we make two observations:

1. Reconstruction does not rely on weaknesses in the summation algorithm; **reconstruction works even if summation is done by a trusted third party**. Instead, reconstruction relies only on the summation revealing both the identities of included variables and the sum of those variables.
2. Reconstruction is independent of how users update their private values, and works even if users update their models in random ways or multiple times. **Reconstruction works because adversaries observe multiple summations with at least one unchanged value, and know how the summations are related.**

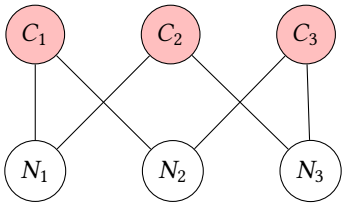


Figure 4.2 Example graph G with adversaries $C = \{C_1, C_2, C_3\}$ (shaded) and non-adversaries $N = N_G(C) = \{N_1, N_2, N_3\}$.

4.4.2 Obtained Adversarial Knowledge

We give a formal description of adversarial knowledge, which is the system of linear equations that adversaries obtain in a privacy-preserving multi-party dynamic-data summation protocol, and observe two important properties.

Let $G = (V, E)$ be an undirected graph, let $C \subseteq V$ be a collusion of k adversaries, let $n := |N_G(C)|$, and let $t \in \mathbb{N}$ be the number of summations performed by C.

Definition 2 (Adversarial knowledge)**.** The adversarial knowledge over t summations by C is a consistent system of linear equations $A\theta = \Theta$, subject to the conditions that

- $\theta \in \mathbb{R}^{nt\times 1}$ are the private values of neighbours $N_G(C)$, such that θ_{vt+i} is the $i \in [\![t]\!]$th unique private value of neighbour $v \in [\![n]\!]$ that is observed by any adversary in C,
- $\Theta \in \mathbb{R}^{t\times 1}$ are the sums obtained by the adversaries, where Θ_τ is the $\tau \in [\![t]\!]$th such sum, and
- $A \in \{0,1\}^{t\times nt}$ indicates which private values are observed in which summation, such that $A_{\tau,vt+i} = 1$ if and only if the adversaries' $\tau \in [\![t]\!]$th summation includes the $i \in [\![t]\!]$th unique private value of neighbour $v \in [\![n]\!]$.

Remark 1. In Theorem 17, we will show that it is not necessary to include adversaries' own private values in $A\theta = \Theta$.

Property 1. Let A be the adversarial knowledge over t summations by C. In each equation, each neighbour in $N_G(C)$ contributes at most one private value:

$$\forall \tau \in [\![t]\!], v \in [\![n]\!] : \sum_{i\in[\![t]\!]} A_{\tau,vt+i} \in \{0,1\}. \tag{4.13}$$

Property 2. Let A be the adversarial knowledge over t summations by C. Since each equation is over all the neighbours of an adversary in C, each row in A corresponds exactly to $N_G(c)$ for some $c \in C$:

$$\forall \tau \in [\![t]\!] : \exists c \in C : \forall v \in [\![n]\!] : \left(\sum_{i\in[\![t]\!]} A_{\tau,vt+i} = 1\right) \Leftrightarrow (c, N_G(C)_v) \in E. \tag{4.14}$$

As in Property 1, the summation merely describes whether neighbour v is included in the τth linear equation.

4.4.3 Reconstruction from Adversarial Knowledge

Finding a (partial) solution is not trivial. It is well-known that the reduced row echelon form (rref) of a system of linear equations reveals the system's unique solution, if it has one. Clearly, this unique solution is also at least a partial solution. However, if there is no unique solution, there may still be a partial solution, as in Equation 4.6 on page 75. We will show in Theorem 16 that finding the reduced row echelon form of the adversarial knowledge is both necessary and sufficient to find all partial solutions. Moreover, we will show in Theorem 17 that this is true even if adversaries' own private values are removed from the adversarial knowledge matrix.

We begin with some definitions. Let $G = (V, E)$ be an undirected graph, let $C \subseteq V$ be a set of k adversaries, let $n \coloneqq |N_G(C)|$, let $t \in \mathbb{N}$, and let $A\theta = \Theta$ be the adversarial knowledge over t summations by C; that is, $A \in \mathbb{R}^{t\times nt}$.

Definition 3 (Solution of a variable)**.** Let $y \in \mathbb{R}^{1\times t}$ and let $i \in [\![nt]\!]$. We say that "y solves θ_i in $A\theta = \Theta$" if and only if the vector yA contains exactly one non-zero value, at index i:

$$\big((yA)_i \neq 0\big) \wedge \big(\forall j \in [\![nt]\!] \setminus i : (yA)_j = 0\big). \tag{4.15}$$

Remark 2. Since Equation 4.15 is independent of θ and Θ, it is equivalent to say that "y solves θ_i in A".

Definition 4 (Partial solution)**.** Let $y \in \mathbb{R}^{1\times t}$. If y solves θ_i in A for any $i \in [\![nt]\!]$, then we say that "y is a partial solution to A".

We proceed with the central theorem of this section, which states that the reduced row echelon form of A describes all partial solutions to A. We remark that a weaker variant of this theorem was previously given without a formal proof [WWJ02].

Theorem 16. Let $i \in [\![nt]\!]$, and let $B \in \mathbb{R}^{t\times t}$ such that $BA = \operatorname{rref}(A)$. Then θ_i has a solution in A if and only if there exists $r \in [\![t]\!]$ such that B_r solves θ_i in A.

Proof. Given $i \in [\![nt]\!]$, we give a proof for both directions.

We first prove that if there exists $r \in [\![t]\!]$ such that B_r solves θ_i, then θ_i has a solution in A. Since $A\theta = \Theta$, it follows that $B_rA\theta = B_r\Theta$, and by Equation 4.15 we have that $B_rA\theta = \theta_i$. Therefore, $\theta_i = B_r\Theta$, proving the first direction of Theorem 16.

We prove the other direction of Theorem 16 by contradiction. Let $y \in \mathbb{R}^{1\times t}$ be a solution to θ_i in A, so yA has its only non-zero value at $(yA)_i$. For the sake of contradiction, assume that there is no row in B that solves θ_i in A. Because y is in the row space of A, and the row space of A is the same as the row space of $\operatorname{rref}(A)$, there exists $y' \in \mathbb{R}^{1\times t}$ such that $yA = y' \cdot \operatorname{rref}(A) = y'BA$. By associativity of matrix multiplication, $y'B$ solves θ_i in A. Furthermore, since we assumed (for the sake of contradiction) that no single row of B solves θ_i in A, it follows that y' must have multiple non-zero coefficients. Thus, let y'_r and y'_s be any two non-zero coefficients in y', and let j, k such that $(BA)_{r,j}$ and $(BA)_{s,k}$ are the leading coefficients of their respective rows; these are their columns' only non-zero values, and $j \neq k$. Therefore, $(yA)_j = (y'BA)_j = y'_r \neq 0$, and similarly $(yA)_k = y'_s \neq 0$. However, this is a contradiction, because we initially assumed that yA has its only non-zero value at $(yA)_i$. Therefore, there exists a row in B that solves θ_i in A. This proves the other direction of Theorem 16.

Therefore, it is both necessary and sufficient to check the rows of $BA = \operatorname{rref}(A)$ to learn all partial solutions to A. ▫

Note that A does not describe that adversaries know each other's private values, since $N_G(C)$ excludes adversaries themselves. We show that including this knowledge does not reveal new partial solutions. Specifically, observe that the adversarial knowledge *including self-knowledge* over t summations by k adversaries C is

$$A' = \begin{bmatrix} A & R \\ 0 & I_{tk} \end{bmatrix}, \tag{4.16}$$

where I_{tk} is the $(tk \times tk)$ identity matrix, 0 is an appropriately-sized matrix of zeroes, and R is some appropriately-sized binary matrix. The rows of I_{tk} represent that adversaries know each other's values, and R represents the edges between adversaries.

Theorem 17. Let $i < tn$. Then θ_i has a solution in A if and only if θ_i has a solution in A'.

Proof. Observe that

$$\operatorname{rref}(A') = \begin{bmatrix} \operatorname{rref}(A) & 0 \\ 0 & I_{tk} \end{bmatrix} \tag{4.17}$$

ignoring row-switching transformations. The bottom tk rows solve exactly θ_i in A for $i \geq tn$. The upper rows solve θ_i in A for $i < tn$ if and only if the rows of $\operatorname{rref}(A)$ do so. ❐

Intuitively, Theorem 17 holds because the linear dependencies that exist within A remain unaffected by R.

4.4.4 Reconstruction Attack Feasibility

We show that reconstruction is feasible for honest-but-curious adversaries. We run the attack in static graphs with randomly-placed adversaries passively collecting data. We measure both the success rate and the number of rounds until success. Our source code is publicly available [DEC25b].

Remark 3. This section pertains only to static graphs. We show a reduction from edge-dynamic graphs to static graphs in Section 4.5.4.

Experimental setup

By Theorem 16, the success rate of the attack depends only on the adversaries' direct neighbourhood. Therefore, instead of modeling large peer-to-peer networks, it suffices to model only the subgraph that is relevant for the attack. Additionally, by Theorem 17, edges between adversaries can be ignored. Therefore, given any graph $G = (V, E)$ and a set of colluding adversaries $C \subseteq V$, it suffices to model the induced subgraph $G[C]$, minus edges between adversaries. This forms a bipartite graph H. We provide an example in Figure 4.3.

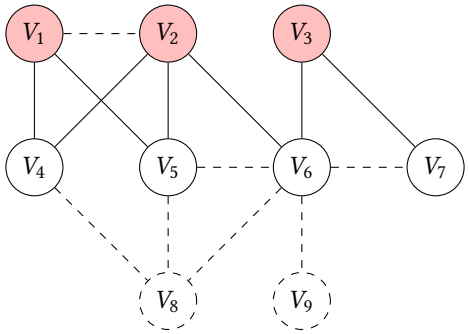


Figure 4.3 A graph G. Adversaries $C = \{V_1, V_2, V_3\}$ are shaded. The bipartite subgraph $H = G[C]$ consists of exactly the non-dotted nodes and edges.

We emphasise that reconstruction depends only on the adversaries' view, regardless of the remaining graph outside this view. However, the likelihood of obtaining any specific adversarial view *does* depend on the full graph. For example, the probability that a random adversarial view contains a cycle depends on the connectivity of the

full graph. For our experiments, we choose not to make assumptions on the graph's topology, analysing all possible adversarial views equally, so that our results are agnostic to the specific network, application, and adversary.

Bipartite graphs can be parameterised by three variables: the number of adversaries, the number of direct neighbours, and the number of edges. We generate random graphs according to these parameter, subject to some filtering:

- We exclude graphs in which there is an adversary with only one edge because this would allow trivial attacks, as described in Section 4.3.3.
- We do *not* exclude graphs in which there is an honest-but-curious user with only one edge, because this user may have more edges in G that are not in H.
- We exclude graphs in which an honest-but-curious user has no neighbours, because these cases do not accurately represent the bipartite graph's parameters.
- We do *not* exclude graphs in which an adversary has no neighbours.
- We do *not* exclude disconnected graphs.

Amount of reconstructed data

For our first experiment, we measure the amount of private data that adversaries can reconstruct. We generate a large amount of random bipartite graphs as described above, and count the number of partial solutions in the biadjacency matrices. This corresponds to the adversarial knowledge if neighbours do not update their values, and thus represents the strongest reconstruction attack that adversaries can perform. In Section 4.4.4 we also consider neighbours updating their values.

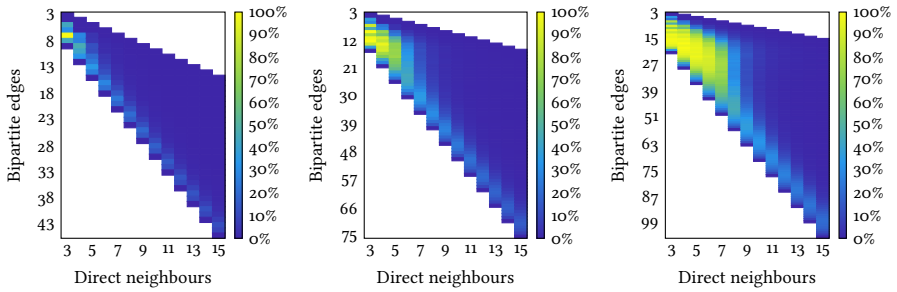
Firstly, we look at the proportion of data that can be reconstructed, shown in Figure 4.4. We see that if the number of adversaries is close to the number of neighbours, the adversary is typically able to reconstruct all neighbours' data. As the number of neighbours increases, fewer data can be reconstructed, unless compensated for by a higher connectivity. If the graph has many neighbours and few edges, adversaries share fewer neighbours, and are thus typically unable to exploit the overlaps.

Secondly, we look at the distribution of how much data can be reconstructed, shown in Figure 4.5. We see again that adversaries are more successful if they outnumber their neighbours. As the number of neighbours increases, so does the probability of being unable to reconstruct any data. However, even if three adversaries passively observe 15 neighbours, they still have an 11.0% probability of reconstructing at least one neighbour's datum, which is unacceptable for any privacy-preserving scheme.

Rounds until first reconstruction

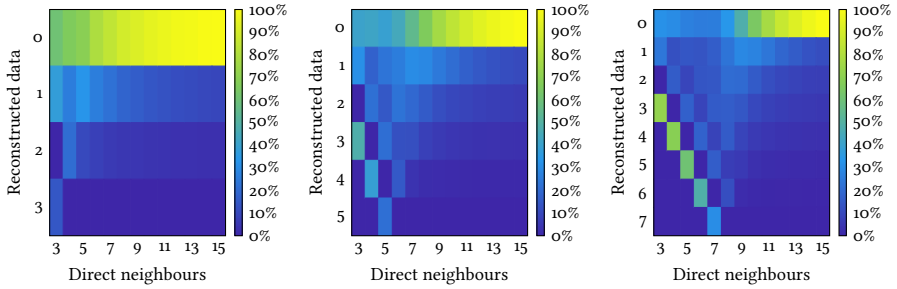
Some partial solutions are harder to obtain than others. For example, if the graph is such that users update their values faster than adversaries can collect them, adversaries may never "converge" to a (partial) solution.

In the next experiment, we measure how many rounds adversaries need before reconstruction succeeds. For each of the subgraphs in Figure 4.4 that were found to be susceptible to the attack, we simulate a multi-party summation protocol as follows. Each round, a uniformly random user in the subgraph wakes up. If the user is an adversary, they learn the sum of their neighbours' values, and adds this to the adversarial knowledge. Otherwise, if a non-adversary wakes up, we simulate



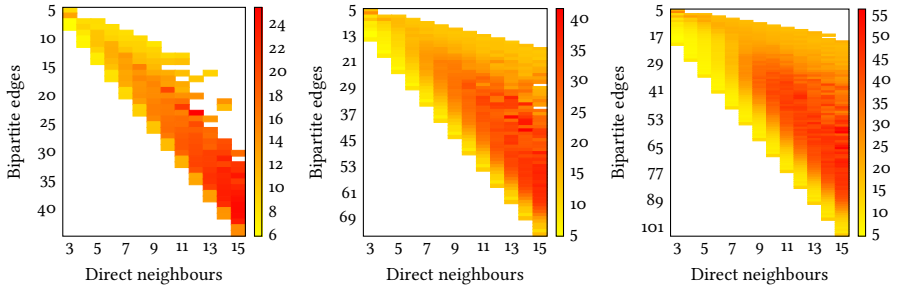
(a) Three adversaries (b) Five adversaries (c) Seven adversaries

Figure 4.4 Proportion of neighbours' private data that can be reconstructed by adversaries. Each point represents the mean over 1000 random bipartite graphs. White points indicate no valid bipartite graphs could be found.



(a) Three adversaries (b) Five adversaries (c) Seven adversaries

Figure 4.5 Probability of reconstructing a given number of neighbours' data, ignoring the number of edges. Each column adds up to 100%, and corresponds to a column in Figure 4.4.



(a) Three adversaries (b) Five adversaries (c) Seven adversaries

Figure 4.6 Mean number of adversarial summations needed to obtain private data. Each point corresponds to 100 attacks on each of the solvable graphs from Figure 4.4. White points indicate no private data was obtained.

an update: The next adversarial sum that includes this non-adversary will use a new column in the adversarial knowledge matrix. After every round, the adversaries check for a partial solution. We repeat this procedure 100 times to control for the order in which users wake up, truncate instances that have no partial solutions after 250 rounds, and take the mean number of rounds until the first partial solution is found.

We show the mean number of rounds until the reconstruction attack succeeds in Figure 4.6. We see that the attack is fastest when there are more adversaries, more edges, and fewer neighbours. Intuitively, this means that the required number of summations increases if neighbours can update their values at a higher rate than adversaries can observe them. For example, 3 adversaries against 15 neighbours require on average 8.8 rounds before they can reconstruct private data. In related works such as [VBT17, Che+18b, CBU24], users run hundreds or thousands of rounds before the protocol terminates, significantly more than required in our attack.

Conclusion of results

We sampled the set of all possible views of randomly selected adversaries in random graphs, excluding some trivial attack cases. If the reconstruction attack succeeds, the adversaries obtain other users' private inputs to the information-theoretically secure summation operation. Our results show that passive honest-but-curious adversaries are able to obtain private data in this scenario with non-negligible probability. While we note that different classes of graph topologies may have varying susceptibility to reconstruction attacks, we conclude that, in general, individually protecting each summation is insufficient for confidentiality.

4.5 Girth as a Reconstruction Countermeasure

In a centralised protocol, the single aggregator can track which summations have occurred, and refuse a subsequent summation if it would result in a partial solution. However, in a distributed computation, there is no such aggregator, and simulating the aggregator using a multi-party protocol is impractical as this would require involving all users in each summation. In this section, we show that to prevent reconstruction it is sufficient to increase the network's girth, which is the length of the network's shortest cycle. The network's girth is an established metric for peer-to-peer networks, with various peer-to-peer algorithms for measuring and increasing the girth [Cen+21, DK08, LU95, Oli+18]. Using such an algorithm before running a privacy-preserving dynamic-data multi-party summation protocol is thus sufficient to prevent reconstruction of private data by honest-but-curious adversaries.

We begin in Section 4.5.1 by showing that reconstruction requires collusion. In Section 4.5.2, we show that reconstruction does not work in acyclic graphs, regardless of the number adversaries. In Section 4.5.3, generalise results to determine an upper bound on the number of adversaries. In Section 4.5.4, consider graphs with dynamic edges. Finally, in Section 4.5.5, we briefly evaluate the impact that increasing girth has on distributed convergence.

4.5.1 Privacy in Static Graphs without Collusion

We begin by considering the special case of $k = 1$, i.e. a setting without collusion. We show that, if the graph is static, the adversary cannot obtain other users' private

values regardless of topology, barring trivial attacks.

Assuming a privacy-preserving summation protocol, it is self-evident that repeating the summation over the same set of values does not leak any private data. However, while the set of neighbours is always the same in the static no-collusion setting, neighbours still update their local values. Thus, it remains to be shown that no reconstruction is possible with this kind of composition.

Lemma 1. Given adversarial knowledge $A \in \mathbb{R}^{t \times nt}$ of a single adversary with $n \geq 2$ fixed neighbours, we have for any $y \in \mathbb{R}^{1 \times t}$

$$\forall \mu, \nu \in [\![n]\!] : \sum_{i \in [\![t]\!]} (yA)_{\mu t+i} = \sum_{i \in [\![t]\!]} (yA)_{\nu t+i}. \tag{4.18}$$

Here, $\sum_{i \in [\![t]\!]} (yA)_{\nu t+i}$ is the sum of components of yA relating to neighbour ν. The equation states that in any linear combination yA, every neighbour has the same sum of components.

Proof. Firstly, because the adversary has fixed neighbours,

$$\forall \tau \in [\![t]\!], \nu \in [\![n]\!] : \sum_{i \in [\![t]\!]} A_{\tau, \nu t+i} = 1. \tag{4.19}$$

In the linear combination yA, the rows of A are scaled according to y and then summed together. Therefore, since each row includes each neighbour exactly once,

$$\forall \nu \in [\![n]\!] : \sum_{i \in [\![t]\!]} (yA)_{\nu t+i} = \sum_{\tau \in [\![t]\!]} y_\tau. \tag{4.20}$$

□

Corollary 1. Given adversarial knowledge $A \in \mathbb{R}^{t \times nt}$ of a single adversary with $n \geq 2$ fixed neighbours, there exists no $y \in \mathbb{R}^{1 \times t}$ such that yA has exactly one non-zero value. Therefore, there exist no partial solutions for A.

4.5.2 Privacy in Static Graphs with Unbounded Collusion

The special case of $k = 1$ provides some insights into the workings of the reconstruction attack, but not allowing any collusion is not realistic, as honest-but-curious collusion in the form of secretly exchanging information is undetectable and there are no strong incentives against it. Therefore, we now proceed to consider the general case of $k \geq 1$.

Partial solutions are linear combinations of the rows of the adversarial knowledge such that all but one column cancels out, as in Equation 4.1 on page 74. We already know from Corollary 1 that a partial solution requires multiple adversaries. If two rows in the adversarial knowledge from different adversaries match in multiple columns, then these adversaries share multiple neighbours, and the graph has a cycle. Otherwise, if no two rows from different adversaries overlap in multiple columns, then, since each equation has at least two non-zero columns, each equation introduces new unknowns, taking the adversaries further from a partial solution. In this case, if the adversaries are able to find a partial solution, they must have another row that cancels out the unknowns of multiple other rows; but this, too, introduces a cycle. The intuition thus seems to be that partial solutions require a cyclic graph. We now formally prove that this intuition is correct.

Theorem 18. Let $G = (V_G, E_G)$ be an undirected graph, let $C \subseteq V_G$ be the set of adversaries, let $k := |C|$, let $n := |N_G(C)|$, let t be the number of summations performed by the adversaries C, and let $A \in \mathbb{R}^{t \times nt}$ be the adversarial knowledge.

If G is acyclic, then A does not have partial solutions.

Proof. We give a proof by contraposition: Given a partial solution to A, we show that G is cyclic. Let $y \in \mathbb{R}^{1 \times t}$ be a partial solution to A. We show how to find a bipartite subgraph H of G such that its biadjacency matrix A'' has a partial solution y''. We then show that this implies the existence of a cycle in G. Our proof works in multiple steps: (1) combine columns of A to create A', (2) remove rows from A' to create A'', (3) create the corresponding partial solution y'', and finally (4) show that G is cyclic. We show an example of this procedure in Figure 4.7.

1. *Combine columns.* We merge the t columns in A assigned to each neighbour to obtain A'. Let $y' = y$, and let $A' \in \mathbb{R}^{t \times n}$ such that

$$\forall \tau \in [\![t]\!], v \in [\![n]\!] : A'_{\tau,v} := \sum_{i \in [\![t]\!]} A_{\tau, vt+i}. \tag{4.21}$$

   It follows from Property 1 on page 77 that this is a binary matrix, and it follows from Property 2 on page 77 that no neighbour relations are removed. Furthermore, observe that

$$\forall v \in [\![n]\!] : (y'A')_v = \sum_{i \in [\![t]\!]} (yA)_{vt+i}. \tag{4.22}$$

   Since yA contains exactly one non-zero value, so does $y'A'$. Therefore, y' is a partial solution to A'.

2. *Remove rows.* We remove duplicate and unused rows from A' to obtain A''. We define A'' as a set of rows:

$$A'' := \{A'_i \mid i \in [\![t]\!] \wedge \tag{4.23}$$
$$\nexists j \in [\![i]\!] : A'_i = A'_j \wedge \tag{4.24}$$
$$\Sigma\{y'_j \mid j \in [\![t]\!] \wedge A'_i = A'_j\} \neq 0\}. \tag{4.25}$$

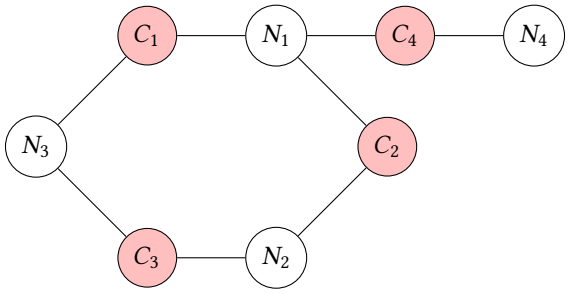
   Here, Equation 4.24 excludes duplicates by only choosing row A'_i if there is no $j < i$ such that $A'_i = A'_j$, and Equation 4.25 excludes unused rows by only picking row A'_i if the sum of y'_j over all identical rows A'_j is non-zero.

3. *Create partial solution.* We similarly combine and remove the corresponding columns from y' to obtain y''. To do so, we define a function ϕ that describes how the rows of A'' relate to the rows of A'. Let s be the number of rows in A''. Then we define $\phi : [\![s]\!] \to [\![t]\!]^*$ such that

$$\forall \tau \in [\![t]\!], \sigma \in [\![s]\!] : \tau \in \phi(\sigma) \Leftrightarrow A'_\tau = A''_\sigma. \tag{4.26}$$

   Using this function, we define $y'' \in \mathbb{R}^{1 \times s}$ as

$$\forall \sigma \in [\![s]\!] : y''_\sigma := \sum_{\tau \in \phi(\sigma)} y'_\tau. \tag{4.27}$$



(a) A graph G featuring adversaries $\{C_1, C_2, C_3, C_4\}$ and non-adversaries $\{N_1, N_2, N_3, N_4\}$.

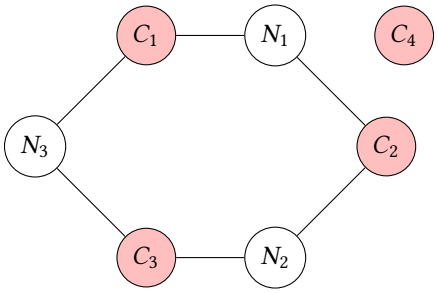
$$A = \left[\begin{array}{\overbrace{ccccc}^{N_1} \overbrace{ccccc}^{N_2} \overbrace{ccccc}^{N_3} \overbrace{ccccc}^{N_4}} \\ 1&0&0&0&0 & 0&0&0&0&0 & 1&0&0&0&0 & 0&0&0&0&0 \\ 1&0&0&0&0 & 1&0&0&0&0 & 0&0&0&0&0 & 0&0&0&0&0 \\ 0&0&0&0&0 & 1&0&0&0&0 & 1&0&0&0&0 & 0&0&0&0&0 \\ 0&0&0&0&0 & 1&0&0&0&0 & 0&1&0&0&0 & 0&0&0&0&0 \\ 1&0&0&0&0 & 0&0&0&0&0 & 0&0&0&0&0 & 1&0&0&0&0 \end{array}\right],$$

$$A' = \begin{bmatrix} 1&0&1&0 \\ 1&1&0&0 \\ 0&1&1&0 \\ 0&1&1&0 \\ 1&0&0&1 \end{bmatrix}, \quad A'' = \begin{bmatrix} 1&0&1&0 \\ 1&1&0&0 \\ 0&1&1&0 \end{bmatrix}$$

(b) The adversarial knowledge A after the users from Figure 4.7a run in the sequence $(C_1, C_2, C_3, N_3, C_3, C_4)$; the matrix A' with collapsed columns; and the matrix A'' without duplicate and unused rows.

$$y = \begin{bmatrix} 1 & 1 & -1 & 0 & 0 \end{bmatrix}, \quad y' = \begin{bmatrix} 1 & 1 & -1 & 0 & 0 \end{bmatrix}, \quad y'' = \begin{bmatrix} 1 & 1 & -1 \end{bmatrix}$$

(c) Partial solutions respectively of A, A', and A''.



(d) The bipartite graph H corresponding to biadjacency matrix A''.

Figure 4.7 Example transformation of graph and adversarial knowledge as seen in the proof of Theorem 18.

It follows that

$$\forall v \in [\![n]\!] : (y''A'')_v = \sum_{\sigma \in [\![s]\!]} (y''_\sigma A''_{\sigma,v}) \tag{4.28}$$

$$= \sum_{\sigma \in [\![s]\!]} \sum_{\tau \in \phi(\sigma)} (y'_\tau A''_{\sigma,v}) \tag{4.29}$$

$$= \sum_{\sigma \in [\![s]\!]} \sum_{\tau \in \phi(\sigma)} (y'_\tau A'_{\tau,v}) \tag{4.30}$$

$$= \sum_{\tau \in [\![t]\!]} (y'_\tau A'_{\tau,v}) \tag{4.31}$$

$$= (y'A')_v. \tag{4.32}$$

Therefore, $y''A'' = y'A'$, and y'' is a partial solution to A''.

4. *Find cycle.* Note that A'' is the biadjacency matrix of some bipartite subgraph $H = (C', N_G(C), E_H)$ of G, where $C' \subseteq C$ and $E_H \subseteq E_G$. Assume, for the sake of contradiction, that H is acyclic. Then H has two distinct nodes i, j with degree one. Since adversaries cannot have degree one in G, and $\forall c \in C' : (N_H(c) = N_G(c) \vee N_H(c) = \emptyset)$, we know that $i, j \in N_G(C)$. Consequently, the columns in A'' for i, j must each contain only one non-zero value, and y'' does not contain zeroes at all by Equation 4.25. Therefore, $(y''A'')_i \neq 0$ and $(y''A'')_j \neq 0$. However, this implies that $y''A''$ has multiple non-zero values, which contradicts the earlier observation that y'' is a partial solution to A''. Therefore, H is cyclic, and so is G. ▫

Our proof shows that partial solutions imply the existence of cycles. This does not mean that cycles imply the existence of partial solutions. Indeed, we show in Section 4.5.3 that structured cycles can be introduced without creating partial solutions.

Remark 4. Theorem 18 pertains only to *partial* solutions. Even in an acyclic topology, there may be linear relations that reveal sensitive information without leaking private values outright, such as $\theta_1 = \theta_2$ or $\theta_3 = 4 \times \theta_5$. Protecting these relations is left for future work.

4.5.3 Privacy in Static Graphs with Bounded Collusion

While acyclic graphs resist reconstruction attacks, these graphs are not well-suited for peer-to-peer networks for two reasons. Firstly, if any non-leaf node becomes unavailable, the network becomes disconnected. Secondly, leaf nodes have only one neighbour, and thus cannot initiate summations to learn from their neighbours.

We show that there are no partial solutions given an upper bound on the number of adversaries. This bound depends on the graph's girth, which is the length of its shortest cycle.

Theorem 19. Let $G = (V_G, E_G)$ be an undirected graph, let $C \subseteq V_G$ be a set of k adversaries, let $n := |N_G(C)|$, let t be the number of summations performed by C, and let $A \in \mathbb{R}^{t \times nt}$ be the adversarial knowledge.

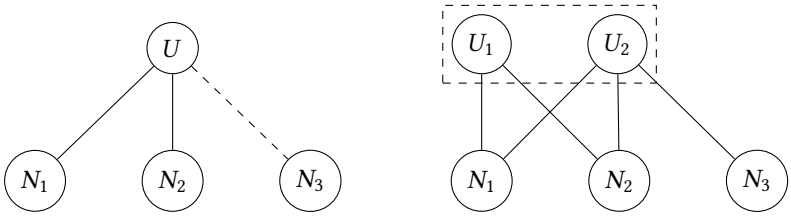
If $\text{girth}(G) > 2k$, then A does not have partial solutions.

Proof. We give a proof by contraposition: Given a partial solution to A, we show that $\text{girth}(G) \leq 2k$. Let H be as in the proof of Theorem 18. Then H is cyclic. Since H is bipartite, every edge in the cycle is between an adversary and a neighbour. Since each node in the cycle is visited at most once, the cycle length is at most $2k$. This cycle also exists in G. Therefore, $\text{girth}(G) \leq 2k$. ▫

4.5.4 Privacy in Dynamic Graphs

So far, we have assumed that graphs are static. However, this prevents users from changing their neighbours, which is unrealistic if users move through the network. We briefly show that dynamic graphs can be reduced to static graphs.

If a single user performs two summations over two sets of neighbours, they learn exactly the same information as two users would over those same sets of neighbours. We show an example in Figure 4.8. More generally, k users with static neighbours can learn the exact same information as ℓ users with k different sets of neighbours. Our results on reconstruction feasibility in static graphs from Section 4.4.4 can be translated similarly to dynamic graphs.



(a) A dynamic graph. The dotted edge is not present in all rounds.

(b) A reduction to a static graph. U has been split into U_1 and U_2.

Figure 4.8 Example of how a dynamic graph can be reduced to a static graph. U learns the same as U_1 and U_2 together.

We conclude that Theorem 19 implies the following.

Corollary 2. Let $G = (V_G, E_G)$ be a dynamic undirected graph, let $C \subseteq V_G$ be a set of adversaries, let $n := |N_G(C)|$, let t be the number of summations performed by C, let k be the number of sets of neighbours the adversaries sum over, and let $A \in \mathbb{R}^{t \times nt}$ be the adversarial knowledge.

If $\text{girth}(G) > 2k$, then A does not have partial solutions.

There are several important limitations to this result. Firstly, the upper bound on the number of adversaries depends on the girth, but the girth may not be known beforehand if users move through the network in unpredictable ways. Secondly, even if a minimum girth is guaranteed throughout the protocol, the upper bound implies a maximum number of changes that may occur during the protocol.

4.5.5 Impact on Convergence

We briefly evaluate the impact of increasing the network's girth on the convergence of a protocol running over that network. Specifically, we numerically simulate a distributed averaging protocol [XB04], which is just a non-privacy-preserving form of

distributed learning. We intentionally choose a simple, efficient, non-noisy protocol to make the impact of the girth parameter most apparent. The "numerical simulation" part of the description is because we do not actually create separate processes and communication for the nodes. Our source code is publicly available [DEC25b].

We use the system model presented in Section 4.3.3. We create a network by generating a random Erdős–Rényi graph with 50 nodes and with each edge having a probability p of being added. Each node holds a single private scalar value, sampled uniformly from the range $\{0 \dots 50\}$. Each round, one random node updates their private value to be the unweighted mean of their neighbours' values and their own value. We then measure the number of rounds until convergence, and take the mean over 1000 repetitions of this procedure. We define convergence as the moment at which any two nodes' local values differ by at most 1. Changing this threshold does not give fundamentally different results.

To measure the effect girth has on convergence, we "stretch" graphs to a given girth by iteratively removing random edges from cycles shorter than the desired girth until no such cycles remain. With 50 nodes, stretching to a girth of x ensures reconstruction attacks are impossible when less than $\frac{x/2}{50} = x\%$ of users collude.

We show our results in Figure 4.9. Since undirected graphs always have girth at least 3, no significant changes occur at these low girths. As the girth increases, so does the number of rounds required. As the girth approaches 25, the slope approaches zero. Graphs that initially have more edges (as determined by p) require more rounds at low girths, but settle at a lower number of rounds at high girths. When we look at our experiments in more detail, we see that ceilings occur once all cycles have been removed, and that graphs with high p retain more edges. This matches the intuition that information propagates more efficiently when there are more edges.

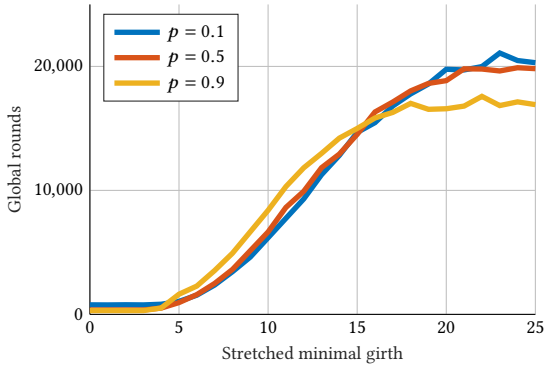


Figure 4.9 Number of rounds until convergence in distributed averaging in random Erdős–Rényi graphs with 50 nodes and varying edge probabilities p, as a function of the girth to which the graphs are "stretched".

Our results show that increasing girth affects convergence speed significantly. Though state-of-the-art distributed learning protocols typically already require several tens of thousands of rounds [VBT17, Che+18b, CBU24], the magnitude by which increasing girth increases the number of required rounds may be excessive for some

applications. More sophisticated edge removal methods may ameliorate this issue. Furthermore, though implementing the cycle removal method from our experiments as a distributed protocol is trivial,[2] this method is not communicationally efficient. To the best of our knowledge, there is no research on communication-efficient distributed "graph stretching". That said, there are distributed protocols for measuring the network's girth [Cen+21] and for removing *all* cycles [DK08, Oli+18]. We conclude that determining a network's resistance by measuring the girth is feasible in general, but increasing girth is practical only when communication efficiency is not a concern.

4.6 Conclusion

We investigated reconstruction attacks in the setting of secure multi-party computation. We observed that existing multi-party computation literature does not consider protocols in which intermediate values are intentionally exposed by the ideal functionality, and seemingly assumes that protocols are not self-composed when deployed. In our investigation, we focused on a peer-to-peer setting with privacy-preserving summation in which users' data change over time. In random subgraphs with 18 users, we found that three passive honest-but-curious adversarial users have an 11.0% success rate at recovering another user's private data using a reconstruction attack, requiring an average of 8.8 rounds per adversary. We analysed the structural dependencies of the underlying network graph that permit this attack, and proved that successful reconstruction attacks correspond to cycles in the network. More generally, we showed that the length of the graph's shortest cycle determines the minimum number of adversaries required for the attack. We conclude that removing short cycles from the network is a feasible countermeasure, albeit with considerable cost towards the convergence speed of distributed protocols.

Our work sets the first step towards preventing reconstruction in the peer-to-peer setting as seen in multi-party computation, and opens up multiple questions for future work. Firstly, and most obviously, though we have found a sufficient criterion to determine reconstruction feasibility, finding a criterion that is also necessary would allow using some graphs which our criterion currently forbids. Secondly, our work is limited to a strictly syntactic notion of privacy, and does not protect linear relations between data, which is required to protect against adaptive adversaries. Thirdly, though our restriction to the summation operation is already sufficient to analyse decentralised learning, our work could be extended to cover compositions with other operations, such as multiplication or comparison. Finally, the addition of differentially private noise may further strengthen the provided level of privacy.

[2] A node can break all cycles of at most length ℓ that they are part of as follows. The node floods a unique random message, paired with a counter starting at ℓ, through the network. Each time a node forwards the message, the counter is decreased. Once the counter reaches zero, nodes stop forwarding the message. If (and only if) the source node receives back their own message, they are part of a cycle of length at most ℓ, and remove the edge on which the message came in.

Chapter 5

Privacy-Preserving Peer-to-Peer Cycle Detection

Abstract. Money laundering has seen a surge in complexity over the past decades as a result of digitisation. This has forced financial institutions to develop more advanced anti-money laundering technologies. Among these technologies is transaction graph monitoring, which involves searching for transaction patterns that are likely indicators of fraud. To ensure effective monitoring across legislative boundaries, institutions should share their transaction information. However, these data are highly sensitive and heavily regulated, making the monitoring infeasible. To address this, we propose a decentralised privacy-preserving protocol that detects fraudulent patterns in transaction graphs. We limit our scope to detecting short cycles. Our protocol can be initiated by any node, who performs recursive key exchanges with the nodes in their neighbourhood up to a maximum depth. If a cycle exists, the initiator will eventually perform a key exchange with themselves, and will then run a separate sub-protocol to recover the nodes involved in the cycle. Our protocol is highly parallelisable, as any number of instances can run simultaneously. We prove the security by showing that a single adversary learns nothing about the topology beyond the detected cycles. When multiple adversaries collude, they may learn of the existence of short paths connecting them, but do not learn what these paths look like. We empirically show that the complexity scales with the local topology and maximum cycle length.

Based on: Juno Jense, **Florine W. Dekker**, Zekeriya Erkin and Mauro Conti. *Privacy-preserving peer-to-peer cycle detection.* 2025. In preparation.

5.1 Introduction

Criminals obfuscate illicitly obtained funds by running them through a complex series of financial transactions. Subsequently, the illegal proceeds appear to come from legal sources, disguising the true nature of the activities. The Russian invasion of Ukraine and its associated funding are a prominent example showcasing the scale and global impact of money laundering: Russia has an estimated $1 trillion hidden abroad, of which a quarter is controlled by Putin and his associates [ÅF20]. By setting up complex laundering schemes, this money can be used to exert political influence, fund illicit trade, and evade sanctions [ÅF20].

Anti-money laundering (AML) is an umbrella term for the laws, regulations and procedures designed to combat the generation, concealment, and integration of illicit funds. In recent years, tighter regulations have led to financial crime detection becoming a high priority among organisations subject to AML audits. These AML systems typically model the transaction network as a graph, with nodes representing bank accounts and edges representing transactions between them. The task of the AML system is to find patterns in the graph that may indicate money laundering. Typical patterns include cliques, stars, and cycles [DBB22]. GraphS [Qiu+18], a tool developed by the e-commerce platform Alibaba, is one such large-scale industry solution based on finding cycles in an internal dataset.

5

At the same time, digital finance has made it possible for criminals to set up accounts at different banks across multiple countries to avoid getting caught by law enforcement agencies [LJ16, EDF21]. Detection is difficult because this requires cross-organisational data analysis, but regulations require that no more information is exchanged than strictly necessary. Coordination between financial organisations is far from straightforward, and remains uncommon within the industry [Mou20]. Since organisations are required by law to report suspicious transactions to their respective national financial authority for analysis, there have been attempts at international cooperation. For example, Ma^{3}tch is a multinational European AML effort, but is lacking because taking a proactive stance imposes too much of a workload on the authority [Mou20]. At the same time, centralising financial data to an international financial authority raises concerns surrounding transparency and privacy, and is unrealistic as governments are largely independent in the creation of their legislative policies, and it is hard to imagine each government consistently reporting to a single authority. Therefore, cross-border data analysis that ensures both confidentiality and autonomy remains an unsolved problem, requiring decentralised privacy-preserving solution.

In this work, we propose a novel decentralised privacy-preserving graph cycle detection protocol. Though many different network patterns are indicative of money laundering, we focus on short cycles as they remain one of the stronger indicators [HK20, Qiu+18]. We model the network as an unweighted directed graph, where nodes and edges represent accounts and transactions, respectively. Our protocol starts at a single node, the initiator, who triggers a flood of messages throughout the network. Each node that is reached by the flood sends messages back through the flood to the initiator to performs a key exchange. If the graph contains a cycle, the flood will also reach the initiator, who will notice that they are performing a key exchange with themselves, and subsequently triggers a simple cycle recovery routine. Our protocol can be parallelised trivially to run many floods simultaneously.

We implement our protocol and validate its performance on various graphs. We find that our protocol is computationally and communicationally efficient for cycles with lengths commonly seen in anti-money laundering operations. Furthermore, we show that nodes learn nothing more about the topology beyond the detected cycles and the existence (but not contents) of paths between pairs of colluding adversaries.

The remainder of this work is structured as follows. In Section 5.2, we discuss related work. In Section 5.3, we cover notation, cryptographic fundamentals, and proposed solution. In Section 5.4, we discuss and empirically validate the complexity of our protocol. In Section 5.5, we prove the security of our protocol. Finally, in Section 5.6, we discuss our findings and provide some concluding remarks.

5.2 Related Work

Existing works have proposed detecting fraudulent transactions using subgraph detection, graph queries, statistical functions, and machine learning. These areas have been surveyed extensively [ATK15, Pou+20, ST21, Pan+20, Bal+23, IPB19, HGY22, DBB22]. However, to the best of our knowledge, there are only three privacy-preserving cycle detection algorithms, which we discuss in the remainder of this section.

The decentralised protocol proposed in [Mar23] uses a message-based approach for detecting cycles, taking as input the maximum cycle length ℓ. To the best of our knowledge, this is the only decentralised privacy-preserving solution that does not scale quadratically in the number of graph vertices. Instead, the protocol is efficient for small ℓ and scales well with network size, because it operates in the local neighbourhood. Any node can initiate the protocol by sending a nonce with a time-to-live of 2ℓ. Recipients forward the message to their outgoing neighbours, but not before applying a one-time pad to the nonce based on the outgoing edge and decrementing the time-to-live. Cycle detection relies on messages following the same path twice, causing one-time pads to be applied twice and thus cancel out, resulting in the initiating node receiving their original nonce.

Unfortunately, re-using the one-time pad has significant security implications. In Figure 5.1, we show a graph in which topological knowledge is leaked because an adversary can infer that messages belong to the same flood of messages. Here, node v_I is the initiator, and node v_A is controlled by a passive adversary. First, v_I sends the nonces r_a, r_b to neighbours v_1 and v_2, respectively. Each recipient re-randomises the nonce with a one-time pad and forwarding the messages until finally v_A receives one message for each path from v_I. However, because the one-time pads are based only on the identity of the edge, the messages that arrive at v_A abide by a certain pattern. Specifically, the difference between the respective pairs of messages received from v_A's neighbours are the same:

$$(r_a r_d)(r_a r_e r_f) = r_d r_e r_f = (r_b r_c r_d)(r_b r_c r_e r_f). \tag{5.1}$$

This implies that v_2 and v_3 share a common ancestor, which violates the protocol's privacy guarantees. Not re-using one-time pads is not a feasible remedy, because the protocol requires one-time pads to be applied twice to cancel out. Running only one flood at a time is not feasible either, because an adversary that knows that this is the case can derive the same information as in Equation 5.1 by looking at the differences in the time-to-live field of the messages.

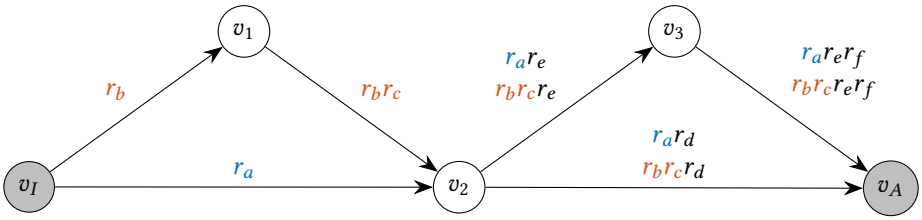


Figure 5.1 Simplified example execution of the protocol in [Mar23], with starting node v_I and passive adversary v_A. The re-use of one-time pads allows v_A to detect patterns in received messages.

The approach in [Vor23] uses secure arithmetic based on Multi-Party Computation (MPC) to implement secure variants of various graph algorithms. We briefly highlight two algorithms: one to detect cycles and another to enumerates cycles. The first algorithm is based on the iterative vertex removal algorithm. This algorithm starts with the full graph, and on each iteration removes all vertices that have no incoming edges until the remaining structure contains only cycles, or is empty if no cycles exist. Their MPC implementation makes use of a secret-shared adjacency matrix and uses an auxiliary list of decision bits to indicate which bits have been removed. The output is a boolean denoting whether the graph contains a cycle. The second algorithm finds and enumerates cycles of a fixed size α by iterating over all possible paths and keeping only those that are cycles. Work can easily be parallelised, which keeps the communication (or round) complexity reasonable. Unfortunately, their approach requires a fully-connected communication graph as the adjacency matrix and computations are shared among all nodes. Consequently, the computational complexity scales with the network size while being exponential in α.

Finally, we discuss topology-hiding computation, which is another form of MPC which considers a partial communication graph. The goal is to perform shared computation while keeping the topology private. To the best of our knowledge, the work in [ALM20] is the only topology-hiding work that covers our objectives: Their approach is round-based, is not specific to cycles, and performs computation over all nodes in the partial graph. Their main contributions are a two-phase protocol based on random walks, as well as a proof that topology-hiding computation is feasible for arbitrary network topologies under the decisional Diffie-Hellman assumption. The first protocol phase propagates a message forward using a random walk. Homomorphic encryption is used to aggregate a secret bit for each node a message passes through. Upon forwarding the message, nodes add an encryption layer. The second phase traces back the walk, such that each node removes the layer of encryption added in the first phase. The output contains the logical or of the secret bits. To obtain the complexities for general graphs, consider the input n, κ, where the input size is polynomial in the security parameter κ. The round and communication complexity are $O(\kappa n^3)$ and $O(\kappa n^5)$ respectively. This makes the random walk approach infeasible for general graphs with large n.

5.3 Our Proposal: Decentralised Cycle Detection

We propose a privacy-preserving protocol for detecting cycles in decentralised networks. Our protocol works by flooding a message through the network and inferring the presence of a cycle when that message reaches the beginning again. To ensure privacy, messages from the same flood should not be linkable, except for the initiator, who can recognise when the message has completed a cycle.

We achieve this using a key exchange protocol, where each node that is reached by the flood performs a key exchange with the initiator, communicating through the other nodes. Once the flood reaches the initiator, the initiator does not immediately know that this message corresponds to the flood they initiated. However, after completing the key exchange, the initiator notices they have performed a key agreement with themselves, proving the presence of a cycle. The initiator then sends a message to determine which nodes are in the cycle. This way, our protocol leaks nothing about the topology beyond a small set of paths.

In Section 5.3.1, we briefly present our preliminaries, including notation, network model, and security model. In Section 5.3.2, we present the protocol in detail.

5.3.1 Preliminaries

Notation. We denote the size of a set X by $|X|$. We model the transaction network as a directed simple graph $G = (V, E)$ with nodes V and edges $E \subseteq V \times V$. Each edge $(v_i, v_j) \in E$ represents a transaction from v_i to v_j. We denote the set of all direct neighbours of $v \in V$ by $N(v)$. The sets of *incoming* and *outgoing* direct neighbours of $v \in V$ are denoted by $N^-(v)$ and $N^+(v)$, respectively. The degree of $v \in V$ is given by $d(v) \coloneqq |N(v)|$, its in-degree by $d^-(v) \coloneqq |N^-(v)|$, and its out-degree by $d^+(v) \coloneqq |N^+(v)|$. A path is a sequence of nodes $(v_1, \ldots, v_k)$. A (simple) cycle is a path where $v_1 = v_k$ and $v_1 \notin \{v_2, \ldots, v_{k-1}\}$.

Network model. Nodes are autonomous and have no common storage, memory, or processing power. Each node has a unique identifier, and each node knows their direct neighbours. The network is static throughout the protocol's execution. Though the network is directed, communication may occur in both directions along each edge.

Security model. Communication between nodes is reliable and cannot be modified, but can be tapped by a global adversary. Additionally, a (non-strict) subset of nodes may be adversarial. All adversaries are honest-but-curious and may exchange information with each other. Additionally, all adversaries are probabilistic polynomial-time.

We define privacy as the inability of adversaries to learn about the topology beyond their background knowledge and the intended output of our protocol. Concretely, for any pair of nodes $u, v \in V$, the adversary cannot guess whether $(u, v) \in E$ any better after the algorithm than before, unless (u, v) is part of a cycle containing the adversary. We formalise and analyse this notion in Section 5.5.

We do not consider attacks based on side channels or network traffic analysis based on timing or volume. Some of these threats may be mitigated by using anonymous communication, dummy traffic, and parallel execution of instances.

5.3.2 Protocol Details

During the protocol, nodes send various messages, and keep routing tables to be able to send message back and forth along a specific route. We present a minimal example of the exchanged messages in Figure 5.2, and a visual example in Figure 5.4 on page 105. Our protocol uses the following messages:

- "`flood`": The initiator starts by sending "`flood`" to each neighbour. Any node that receives "`flood`" sends "`flood`" to each of their neighbours. The time-to-live field in each "`flood`" message ensures the message is forwarded a limited number of times. Each "`flood`" message contains a unique public key.
- "`echo`": Any node that receives "`flood`" combines a fresh private key with the contained public key, and sends this back to the sender in an "`echo`". When a node receive "`echo`", it forwards this *backwards* through the flood, towards the initiator.
- "`trace`": Once the initiator has established that there is a cycle, they send "`trace`" to the next node in the cycle. Each node that receives "`trace`" appends their identity to the list, and forwards it until it reaches the initiator again.

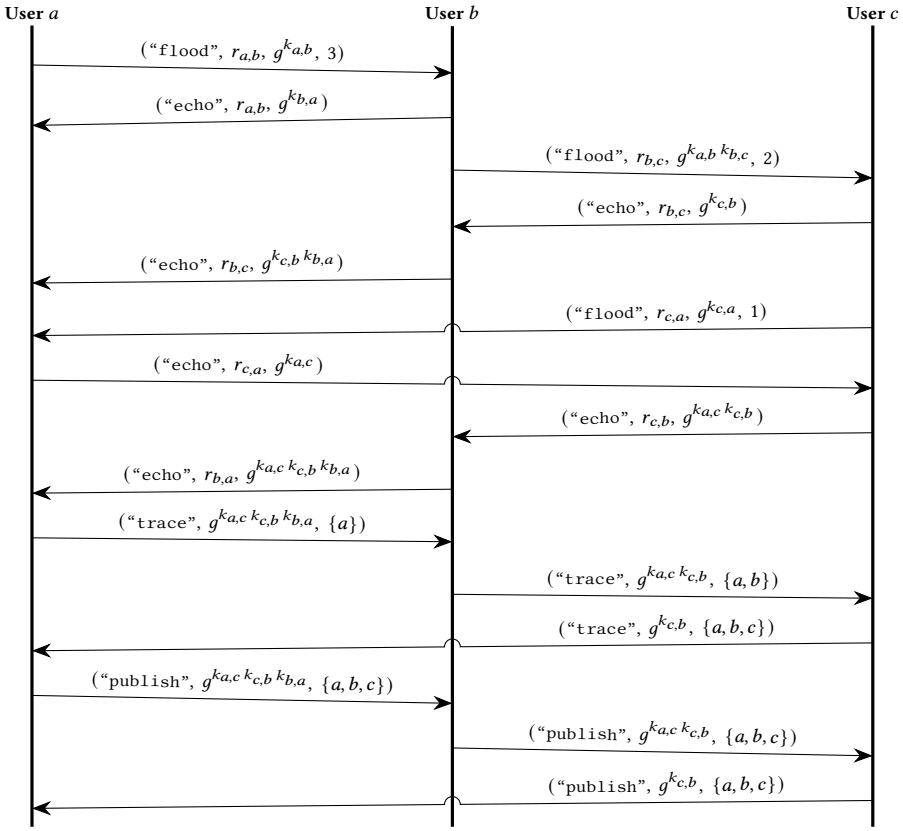


Figure 5.2 Example message exchange after node A initiates a flood of messages, with nodes A, B, and C arranged in a simple directed cycle

- "publish": When "trace" reaches the initiator again, the list of nodes is complete, and sends it in an "publish" message to the next node. This node learns about the cycle, and forwards the "publish" as well. When "publish" reaches the initiator, the protocol is complete.

For the key exchange, we use Diffie-Hellman key exchange [DH76]. We assume that all nodes agree upon a group $\mathbb{G} \subseteq \mathbb{Z}_p^*$ of order q where the decisional Diffie-Hellman problem is hard, and that generator g of $\mathbb{G}$ is publicly known. When we do arithmetic in $\mathbb{G}$, we omit the modulo for simplicity. Our protocol uses security parameters κ_p, κ_q, and κ_r: The values p and q have κ_p-bit and κ_q-bit security, respectively, while any nonces generated in our protocol have κ_r-bit security.

The message exchanges described earlier are implemented over several algorithms. It may help to imagine each algorithm running in a dedicated thread. The algorithm `Initiate` is started by a node when that node desires to detect nearby cycles, and kicks off a flood of messages. The algorithms `Flood`, `Echo`, `Trace`, and `Publish` are run continuously by each node, and describe what a node does upon receiving a message of the corresponding type.

Initiate

We present `Initiate` in Algorithm 1. This algorithm is run only by the initiating node u when they want to kick off cycle detection. Recall that the initiating node u also responds to messages as described in the other algorithms.

Node u starts by sending "flood" to each neighbour v, containing a nonce $r_{u,v}$ and a public key $g^{k_{u,v}}$. In `Flood` (Algorithm 2), each receiver v processes this message, resulting in a flood of key agreements, which are stored in the nodes' private `agreements` arrays, and "echo"'d back to the initiator (Algorithm 3). Back in `Initiate`, the initiator inspects incoming "echo"s, completes the key exchange, and if it finds the resulting key in the `agreements` array, then the exchange was apparently performed with themselves, which proves the presence of a cycle. Using the information in `agreements`, the initiator infers that the cycle starts with (u, v') and ends with (w, u). Therefore, the initiator sends "trace" to v' to create a list of nodes in the cycle (Algorithm 4), which ultimately comes back to node u through node w, containing the list of nodes C. (The function of keys K and K' is explained in subsequent algorithms.) Node u finally publishes C by sending a "publish" message (Algorithm 5).

Flood

We present `Flood` in Algorithm 2. When node v receives a "flood" from some node u, they generate fresh private key $k_{v,u}$ and send $g^{k_{v,u}}$ back to node u in an "echo", reusing the nonce. Recall from `Initiate` that the key agreement $K^{k_{v,u}}$ is stored in `agreements` to detect when self-agreement occurs. Next, node v forwards the "flood" to its neighbours. The construction of these messages is similar to the original message from `Initiate`, with one major difference: The key K is amended with the fresh key $k_{v,w}$. Additionally, the local data structures `before` and `after` are populated. These will be used in `Echo` (Algorithm 3) and `Trace` (Algorithm 4) to efficiently send messages between nodes without broadcasting. Specifically, using `before`, a message from w with nonce $r_{v,w}$ can be sent back to u with nonce $r_{u,v}$, and using `after`, a message from u with key K can be sent to w with key K_w.

Echo

We present `Echo` in Algorithm 3. When node v receives an "`echo`", they send an "`echo`" towards the initiator, using `before` to determine node u and nonce $r_{u,v}$. This repeats until the "`echo`" arrives at the initiator, who handles the message in `Initiate`. Note that the same private key $k_{v,w}$ that was applied to the "`flood`" to w is now applied to the key that will eventually reach the initiator.

Trace

We present `Trace` in Algorithm 4. The algorithm is conceptually similar to Algorithm 3, with two major differences. First, messages are forwarded in the other direction. Second, messages are not identified by nonces, but using keys in `after`. Each time the message is forwarded, the forwarded appends its identity to C. The initiator v has different behaviour (Algorithm 1) upon receiving a "`trace`"; it sees that $C_0 = v$, and stops the forwarding. The check `after[`K`]` lets the algorithm differentiate between the initiator and other nodes.

Publish

We present `Publish` in Algorithm 5. The algorithm is extremely similar to Algorithm 4. The only difference is that the cycle C is now complete and can be appended to `cycles` before being forwarded. The check `after[`K`]` ensures the initiator stops forwarding the message.

Algorithm 1: `Initiate`, from the perspective of node $u \in V$.

Data: `agreements`, `cycles`
Result: Starts flooding, waits for self-agreement, starts cycle recovery.

```
// Start key agreement flood
for v ∈ N⁺(u) do
    r_{u,v} ←_R {0,1}^{κ_r}; k_{u,v} ←_R ℤ*_q;
    send ("flood", r_{u,v}, g^{k_{u,v}}, ℓ − 1) to v;
end for

// Analyse incoming echoes until self-agreement occurs
while true do
    receive ("echo", r_{u,v'}, K) from any v' ∈ N⁺(u);
    if exists(agreements[K^{k_{u,v'}}]) then          // see Algorithm 2
        (w, r_{w,u}, K') ← agreements[K^{k_{u,v'}}];
        // Construct C
        send ("trace", K, {u}) to v';                 // see Algorithm 4
        receive ("trace", K', C) from w;
        // Publish C
        append C to cycles;
        send ("publish", K, C) to v';                 // see Algorithm 5
    end if
end while
```

Algorithm 2: `Flood`, from the perspective of node $v \in V$.

Data: `agreements`, `before`, `after`
Result: Starts back-propagation and continues flooding.

while *true* **do**
  receive (“`flood`”, $r_{u,v}$, K, ℓ) from any $u \in N^-(v)$;
  `// Respond to sender, starting back-propagation`
  $k_{v,u} \leftarrow_R \mathbb{Z}_q^*$;
  `agreements`$[K^{k_{v,u}}] \leftarrow (u, r_{u,v}, K)$; `// used in Algorithm 1`
  send (“`echo`”, $r_{u,v}$, $g^{k_{v,u}}$) to u;
  `// Continue flooding`
  if $\ell - 1 = 0$ **then continue**;
  for $w \in N^+(v)$ **do**
    $r_{v,w} \leftarrow_R \{0,1\}^{\kappa_r}$; $k_{v,w} \leftarrow_R \mathbb{Z}_q^*$; $K_w \leftarrow K^{k_{v,w}}$;
    `before`$[r_{v,w}] \leftarrow (u, r_{u,v}, k_{v,w})$; `// used in Algorithm 3`
    `after`$[K] \leftarrow (w, K_w)$; `// used in Algorithm 4`
    send (“`flood`”, $r_{v,w}$, K_w, $\ell - 1$) to w;
  end for
end while

Algorithm 3: `Echo`, from the perspective of node $v \in V$.

Data: `before`
Result: Back-propagates echo messages to the initiating node.

while *true* **do**
  receive (“`echo`”, $r_{v,w}$, K) from any $w \in N^+(v)$;
  if `exists(before`$[r_{v,w}]$`)` **then**
    $(u, r_{u,v}, k_{v,w}) \leftarrow$ `before`$[r_{v,w}]$;
    send (“`echo`”, $r_{u,v}$, $K^{k_{v,w}}$) to u;
  end if
end while

Algorithm 4: `Trace`, from the perspective of node $v \in V$.

Data: `after`
Result: Determines which nodes are in the detected cycle.

while *true* **do**
  receive (“`trace`”, K, C) from any $u \in N^-(v)$;
  if `exists(after`$[K]$`)` **then** `// see Algorithm 2`
    $(w, K_w) \leftarrow$ `after`$[K]$;
    send (“`trace`”, K_w, $C + \{v\}$) to w;
  end if
end while

Algorithm 5: `Publish`, from the perspective of node $v \in V$.

Data: `after`, `cycles`
Result: Informs nodes in the cycle of the list of nodes in the cycle.

while *true* **do**
  receive ("`publish`", K, C) from any $u \in N^-(v)$;
  if `exists(after[`K`])` **then** `// see Algorithm 2`
    append C to `cycles`;
    $(w, K_w) \leftarrow$ `after[`K`]`;
    send ("`publish`", K_w, C) to w;
  end if
end while

5.4 Performance

Firstly, in Section 5.4.1, we derive the worst-case complexity in terms of time, communication, and storage. Secondly, in Section 5.4.2, we discuss our implementation of our protocol. Finally, in Section 5.4.3, we present the empirical complexity results of our implementation.

5.4.1 Worst-Case Complexity

Communication. The number of messages in `Flood` and `Echo` each is exponential in ℓ and grows by a factor of n. The number of messages in `Trace` and `Publish` each is simply the sum of lengths of the found cycles. Overall, given the set of found cycles C, the communication complexity is $O(n^\ell + \sum_{\sigma \in C} |\sigma|)$.

Computation. Each incoming message results in a function call of the corresponding type. `Trace` and `Publish` each require a constant number of relatively trivial computations. However, `Flood` and `Echo` both require expensive exponentiations upon each invocation: `Flood` requires one exponentiation per neighbour, and `Echo` requires a constant number of exponentiations. The bit complexity of each exponentiation scales with the security parameters from Section 5.3.2. Concretely, in $\mathbb{Z}_p$, each exponentiation has a computational complexity of $O((\log \kappa_p)^3)$ [MvOV96]. The overall computational complexity is therefore $O(n^{\ell+1}(\log \kappa_p)^3 + \sum_{\sigma \in C} |\sigma|)$.

Storage. Users may receive $O(n^\ell)$ different "`flood`"s. For each such message, that user stores one κ_q-bit key plus for each outgoing neighbour one κ_q-bit key and one κ_r-bit nonce. Also, users store the cycles they find during `Publish`. Overall, given the set of found cycles C, the storage complexity is $O(n^{\ell+1}(\kappa_q + \kappa_r) + (\sum_{\sigma \in C} |\sigma|)^2)$ bits.

Observations. We conclude that our protocol exhibits its worst performance when executed over a fully-connected graph, and its best performance when executed over a fully-*dis*connected graph. For arbitrary graphs, the complexity is dependent on the search depth ℓ and the degree distribution in the local neighbourhood. In particular, the node with the highest out-degree contributes significantly to the complexity [Qiu+18], and may be used instead of n to obtain a tighter bound.

5.4.2 Method

We use our publicly available implementation of the protocol, written in C++ [JD25]. Our implementation generates synthetic graphs in the Barabási–Albert model [AB01]. The underlying principle of this model is preferential attachment: New nodes prefer to connect to high-degree nodes, resulting in a power-law distribution over the nodes' degrees. Many real-world phenomena, including financial markets [LV16], are thought to have this property While empirical analysis shows that such networks are rare in the real world [BC18], recall that the performance bottleneck in real-world cycle detection is due to a small number of high-degree nodes [Qiu+18]. Therefore, we argue that a power-law distribution is sufficiently realistic for our purposes.

The Barabási–Albert model specifies two parameters, m_0 and m_1, plus the desired number of nodes n. Generation is iterative and starts with a fully-connected graph of size m_0. Each iteration adds a node, and randomly connects it to m_1 different nodes in the existing network, with the sampling probability proportional to the existing node's degree. The direction of each new edge is decided by a fair coin toss. This process repeats until there are n nodes in total. As is commonly done, we set $\overline{m} = m_0 = m_1$. A higher value of $\overline{m}$ corresponds to a denser graph.

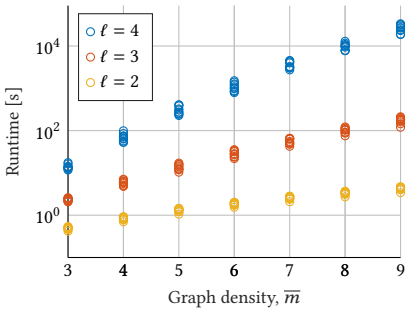
For our security parameters (recall Section 5.3.2), we recommend setting $\kappa_p = 3072$ and $\kappa_q = 384$, which is large enough to resist attacks [Bou+20, LLW24] on the discrete log problem.[1] The same paradigm applies to nonces, for which we recommend setting κ_r to be at least 128 bits. The choice of security parameters affects the efficiency of the protocol. This creates a well-known trade-off: stronger security results in worse performance of the protocol.

For each $\overline{m} \in [3, 9]$, we generate a random graph with $n = 50$ nodes. For each graph, for each $\ell \in \{2, 3, 4\}$, we sequentially run our protocol once on each node, and record the mean runtime t_{avg}, the mean number of "`flood`" and "`echo`" messages, the mean number of "`trace`" messages, and the mean number of cycles of length at most ℓ. We repeat this entire procedure 10 times, each time using a different random group with $\kappa_p = 60$ and $\kappa_q = \kappa_r = 20$.

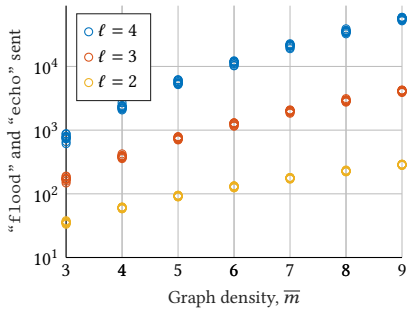
5.4.3 Results

We present our results in Figure 5.3. Note the logarithmic y-axis in each figure. In Figure 5.3a we see that the mean runtime grows exponentially as the graph becomes denser, where higher values of ℓ result in an increased growth rate. In Figure 5.3b we see a similar pattern for the communication complexity, as expected from our asymptotic complexity analysis. In Figure 5.3c we see that the number of "`trace`" messages grows slower than either the runtime or the number of "`flood`" and "`echo`" messages, but certainly does not grow linearly either. This behaviour is best understood by considering Figure 5.3d, in which we see the mean number of cycles that each user is part of, and which has a roughly similar growth rate as the number of "`trace`" messages. We note that the growth rate in these latter two figures may be different for other types of graphs.

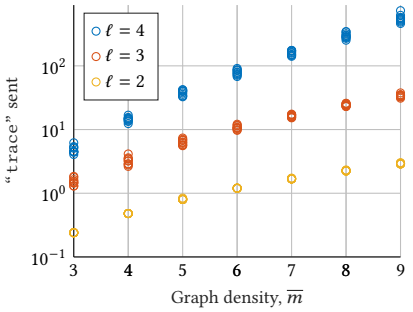
[1] The 2019 cryptographic guideline published by the National Institute of Standards and Technology [BR19] recommends picking a key size in the range of 128 to 3072 bits to mitigate security risks, and considers key lengths of under 112 bits to be insecure.



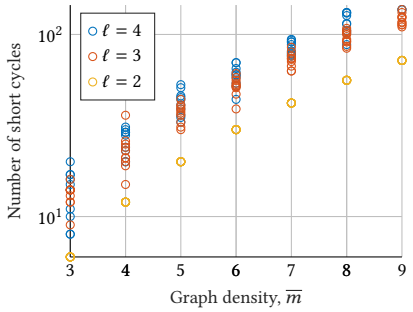
(a) Mean runtime of full protocol



(b) Mean number of floods and echoes sent



(c) Mean number of traces sent



(d) Total number of cycles shorter than ℓ

Figure 5.3 Empirical results for graphs with $n = 50$. Figures (i)–(iii) show the performance of the protocol, where each data-point is the mean over the 50 nodes. Figure (iv) shows auxiliary information resulting from the graph generation method.

5.5 Security Analysis

We show that our protocol, introduced in Section 5.3, is secure. We say our protocol achieves security (or rather: confidentiality) when running the protocol does not give the adversary an advantage in guessing the (non-)existence of an edge beyond what can be inferred from the intended outputs: cycles containing adversaries, and path existence between adversaries.

The topology is encoded into our protocol in two places: in the message contents, and in how the floods flow through the network. We can trivially show that message contents alone are not sufficient to learn about the topology: The only messages that explicitly encode the topology are "`trace`" and "`publish`", but the contained information corresponds to the intended outputs of the protocols, and therefore do not constitute a breach of confidentiality.

We now focus on analysing what adversaries can learn from the way in which messages flow through the network. Should an adversary learn that two received messages belong to the same flood, then the adversary knows that their respective senders are connected, which is more than the adversary knew before. Still, our

security model in Section 5.3.1 limits what adversaries can learn: Adversaries cannot perform side-channel attacks based on the timing and volume of messages. However, they *can* link messages together based on their contents, for example by detecting patterns in the nonces. Some linkages are trivial and do not impact confidentiality, which we capture in the following definition.

Definition 5. System of messages. Given a protocol message m and an adversarial node u, the corresponding *system of messages*, denoted $\text{system}_u(m)$, consists of all messages that u can trivially infer to be part of the same flood as m. If m is a "`flood`" received by u, then $\text{system}_u(m)$ contains m, the "`echo`" returned by u, all "`flood`"s sent by u in response to m, all "`echo`"s received in response to those forwards, and all "`echo`"s forwarded by u in response to those echoes. The function system_u represents an equivalence class: For any $m' \in \text{system}_u(m)$, we have that $\text{system}_u(m) = \text{system}_u(m')$.

Furthermore, if a message m was sent from an adversarial node u to another adversarial node v, then the set of messages that can be linked to each other consists of both users' systems of messages, $\text{system}_u(m)$ and $\text{system}_v(m)$. In fact, these systems of messages are therefore the same, large system of messages: $\text{system}_u(m) = \text{system}_v(m)$. This property is transitive.

We show that the adversary cannot determine whether two messages belong to the same flood. We consider two messages that were sent to or received by one or more adversarial nodes. We give a proof by contradiction. For the sake of contradiction, we assume that the adversaries *do* derive whether the messages belong to the same flood. If the two messages belong to the same system of messages, then they are trivially linkable, but do not reveal information that the adversaries did not already have access to. If they are not part of the same system of messages, however, we can distinguish two distinct situations, depending on whether the messages are linked sequentially, i.e. whether the path from the flood's initiator to one of the messages includes the other message. If they are linked sequentially, there clearly exists a path between the adversarial nodes; since paths between adversaries are explicitly part of the protocol's output, this situation does not leak confidential information. Otherwise, if the messages are not linked sequentially, the two messages have a common ancestor node; since the ancestor independently re-randomises the messages before forwarding them, the two messages are by definition unlinkable. Therefore, in all cases, adversaries are unable to link messages together such that they gain undue information.

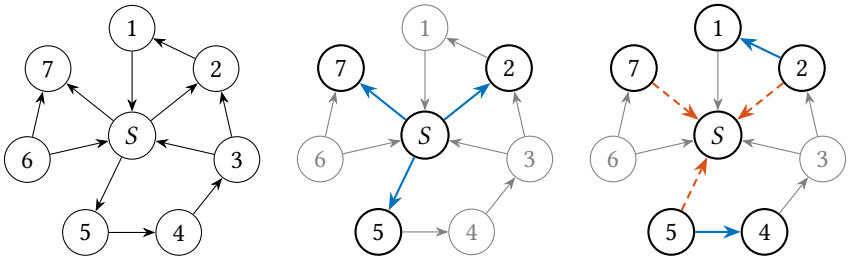
5.6 Conclusion

Anti-money laundering systems work by detecting patterns in transaction graphs. In practice, data sharing prevents these systems from being deployed across legislative boundaries. To this end, we have designed a decentralised privacy-preserving cycle detection protocol. In our protocol, the initiating node floods their local neighbourhood with messages and performs recursive key exchanges with the recipients. When the flood reaches back to the initiator, a cycle must exist, which the initiator subsequently uncovers in a sub-protocol. Our protocol can be run in parallel with many initiating nodes, allowing the participants to collectively find all cycles in the network. We

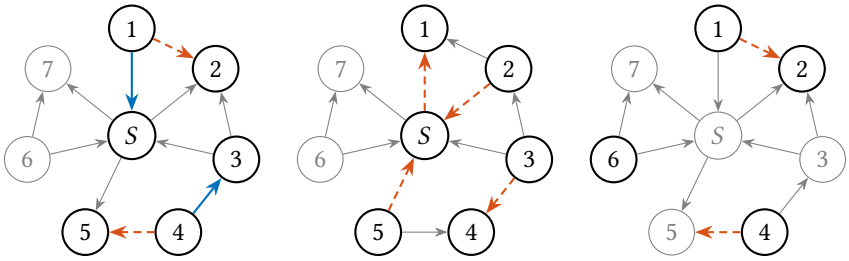
show that participants learn nothing about the topology beyond the cycles, even when participants collude.

Our protocol differs from existing works in that it is fully distributed. Previous works have typically required knowledge of the full topology, or otherwise required all nodes to be available simultaneously. These works thus cannot guarantee output delivery when nodes become unavailable, and their computational complexities typically scale with the size of the graph. Meanwhile, in our protocol, complexity scales depending on the local neighbourhood only, with nodes learning only about cycles that they themselves are part of.

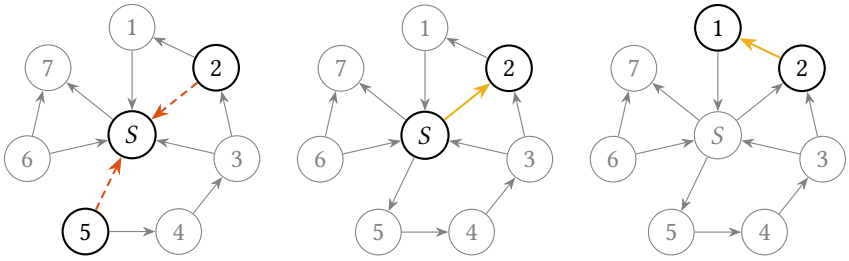
Future work may include optimisations across parallel instances, ensuring only simple cycles are found, multiple initiating nodes, and reusing the outputs of past instances, for example to detect new cycles after the topology has changed.



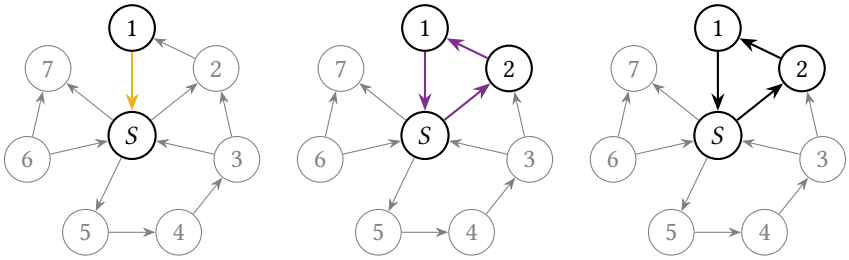
(a) Topology (b) Initiate, $\ell = 3$ (c) Forward/echo, $\ell = 2$



(d) Forward/echo, $\ell = 1$ (e) Forward/echo, $\ell = 0$. Max depth reached. (f) Echo



(g) Echo. S self-agrees. (h) Trace (i) Trace



(j) Trace (k) Publish (separately) (l) Reconstructed cycle

Figure 5.4 Visualisation of the network behaviour during an instance of our protocol, with initiator S. Note the dotted lines, representing echoes being sent opposite the edge's normal direction. Also note that Figure 5.4k actually shows three separate steps similar to the trace steps.

Chapter 6

Optimal Graph Stretching for Distributed Averaging

Abstract. The performance of distributed averaging depends heavily on the underlying topology. In various fields, including compressed sensing, multi-party computation, and abstract graph theory, graphs may be expected to be free of short cycles, i.e. to have high girth. Though extensive analyses and heuristics exist for optimising the performance of distributed averaging in general networks, these studies do not consider girth. As such, it is not clear what happens to convergence time when a graph is stretched to a higher girth.

In this work, we introduce the *optimal graph stretching problem*, wherein we are interested in finding the set of edges for a particular graph that ensures optimal convergence time under constraint of a minimal girth. We compare various methods for choosing which edges to remove, and use various convergence heuristics to speed up the searching process. We generate many graphs with varying parameters, stretch and optimise them, and measure the duration of distributed averaging. We find that stretching by itself significantly increases convergence time. This decrease can be counteracted with a subsequent repair phase, guided by a convergence time heuristic. Existing heuristics are capable, but may be suboptimal.

Based on: **Florine W. Dekker**, Zekeriya Erkin and Mauro Conti. *Optimal graph stretching for distributed averaging*. 2025. arXiv: 2504.10289.

6.1 Introduction

Distributed averaging allows nodes in a peer-to-peer network to find the global mean of the nodes' local values in a completely distributed manner. Throughout the protocol's iterative process, each node's estimate of the global mean continues to improve until a consensus is reached. Distributed averaging has applications in various fields, including gossip learning [Boy+05], fully-distributed learning [VBT17], and control systems [HDC18]. In all cases, the challenge is to find an algorithm that is efficient in terms of convergence time and communication cost.

The study of convergence in consensus algorithms is heavily tied to studies on *synchronisability* in chaos theory, which, roughly speaking, studies the ability of disjoint systems to synchronise spontaneously [PC98, BP02]. We know from chaos theory that the convergence time of distributed averaging is heavily tied to the underlying topology [Boy+05, Li+10]. Optimising a topology for convergence time is hard [XB04], and so a multitude of heuristics have been proposed, including those based on graph metrics such as degree, closeness centrality, and efficiency [HS08, SB22], and on spectral metrics such as eigenratio and algebraic connectivity [GB06, XB04].

Meanwhile, several fields study the girth of the network, which is the length of its shortest cycle. In compressed sensing, high girth positively impacts reconstruction guarantees [Kha+11, LX13]. In multi-party computation, the girth implies specific privacy guarantees [DEC25c]. Finally, in graph theory, high-girth graphs are an interesting concept per se [Mar82], and are important when studying expander graphs [Par21]. Various authors have also proposed algorithms for increasing the girth of an existing graph. Algorithms for coding theory focus on bipartite graphs [HEA05, LTT11], while algorithms for expander graphs focus on degree-regular graphs [Par21].

To the best of our knowledge, there are no works that study the intersection of these two areas. Therefore, in this work, we ask: How does "stretching" the girth of a graph to a higher value affect the convergence time of distributed averaging? Additionally, we ask how to minimise the number of leaf nodes, since these are undesirable in various applications [AHL02, DEC25c]. To answer both our questions, we formalise our optimisation problem, consider several stretching and leaf minimisation algorithms, optimisation heuristics, and graph families, and compare the results.

We find that stretching a graph to a higher girth significantly increases the convergence time, typically by an order of magnitude. Since stretching consists solely of removing edges, we find that the best algorithm prioritises the removal of those edges that are in the largest number of cycles. Additionally, lost convergence time can be recuperated partially by greedily optimising the edge set using a heuristic for convergence time. Meanwhile, minimising the number of leaves has little impact on convergence time, with little difference between the various algorithms studied. Finally, though the studied heuristics are adequate for improving convergence time, our results indicate that heuristics tailored for high-girth graphs may be able to achieve even better convergence time.

In Section 6.2, we present our notation and various preliminaries. In Section 6.3, we survey related work. In Section 6.4, we introduce the optimal graph stretching problem and our exact research questions. In Section 6.5, we explain our research method. In Section 6.6, we present our results. Finally, in Section 6.7, we offer our conclusions.

6.2 Preliminaries

In general, we denote the first element of a vector v by v_0, the absolute value of a scalar x by $|x|$, the cardinality of a collection S by $|S|$, the range of integers $\{0 \dots n-1\}$ by $[\![n]\!]$, and the Euclidian norm of a vector v by $\|v\|_2$.

6.2.1 Graph theory

Basics. A graph $G = (V, E)$ is a set of vertices V and a set of edges $E \subseteq V \times V$. In this work, we consider only simple graphs, i.e. unweighted, undirected, self-loopless graphs, where each edge may occur at most once. For any node $v \in V$, the function $\text{neigh}(v)$ gives the set of direct neighbours of v, and $\deg(v)$ gives the degree of v. The adjacency matrix A of graph G is a $|V|$-by-$|V|$-matrix where, for any $i, j \in [\![|V|]\!]$, we have $A_{i,j} = 1$ if $(V_i, V_j) \in E$ and $A_{i,j} = 0$ otherwise. The (unoriented) incidence matrix B of graph G is a $|V|$-by-$|E|$-matrix where, for any $i \in [\![|V|]\!]$, $j \in [\![|E|]\!]$, we have $B_{i,j} = 1$ if $V_i \in E_j$ and $B_{i,j} = 0$ otherwise.

Spectral theory. For any n-by-n matrix M, an eigenvector v is a vector such that $Mv = \lambda v$ for some scalar λ. This scalar λ is the eigenvalue corresponding to v. The matrix M has n (not necessarily unique) eigenvalues, collectively known as the *spectrum* of M. For any $1 \leq i \leq n$, we write $\lambda_i(M)$ to mean the ith-smallest eigenvalue of M. That is, $\lambda_1(M) \leq \lambda_2(M) \leq \ldots \leq \lambda_n(M)$. We drop the index M when the matrix is clear from context.

Spectral graph theory. The Laplacian L of a graph G is the $|V|$-by-$|V|$ matrix BB^T. For any $i, j \in [\![|V|]\!]$, we have $L_{i,j} = -A_{i,j}$ if $i \neq j$ and $L_{i,j} = \deg(V_i)$ otherwise. Some eigenvalues of L are special: $\lambda_1 = 0$; λ_2 is called the algebraic connectivity (and the associated eigenvector is called the Fiedler vector); λ_n is called the spectral radius; and $\frac{\lambda_2}{\lambda_n}$ is called the eigenratio. The algebraic connectivity $\lambda_2 = 0$ if and only if G is connected [Fie73]. All eigenvalues increase monotonically with the edge set. (This cannot be said for the eigenratio.) Formally, given graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ where $E_1 \subseteq E_2$, we have $\lambda_i(L_1) \leq \lambda_i(L_2)$ [Fie73]. In fact, the eigenvalues of the two graphs become interlaced [GMS90, Mer91]: $\lambda_i(L_1) \leq \lambda_i(L_2) \leq \lambda_{i+1}(L_1) \leq \lambda_{i+1}(L_2)$.

6.2.2 Distributed averaging

Consider a graph $G = (V, E)$ with $n \coloneqq |V|$ nodes. Each node $v \in V$ has a scalar value x_v and can communicate only with their direct neighbours $\text{neigh}(v)$. In distributed averaging, the task for each node is to find the global mean $\frac{\sum_{v \in V} x_v}{n}$.

Distributed averaging can be achieved using a distributed asynchronous push-pull algorithm: Nodes iteratively calculate the mean of their local neighbourhood and then replace their own value with that mean. Specifically, in this work, the algorithm we consider has the following properties:

- *Asynchronous [Boy+06]:* Users do not coordinate to choose which user is next. Instead, users randomly and independently "wake up" and perform their iteration.
- *Linear iterations [OM03, XB04]:* Distributed averaging algorithms differ in which neighbours are included in the averaging operation. To achieve convergence, it is sufficient that each direct neighbour is selected with a non-zero probability [HDC18].

For simplicity, in our implementation, the initiating user selects one of its neighbours at random.

- *Push-pull [Dem+88]:* The mean calculated by the initiating user is used as the new local value of both the initiating user v ("pull") and the selected neighbour w ("push").

Implementing this type of distributed averaging requires each user to simultaneously run two threads: one to initiate rounds, and one to respond. We show the corresponding algorithms respectively in Algorithm 6 and Algorithm 7. To avoid overly complex notation, these algorithms do not address issues relating to concurrency.

Algorithm 6: Active thread of each user v in distributed averaging

while *true* **do**
  `sleep();`
  $w \leftarrow_R \text{neigh}(v)$;
  `// random sample`
  send x_v to w;
  receive x_w from w;
  $x_v \leftarrow \frac{x_v + x_w}{2}$;
end while

Algorithm 7: Passive thread of each user w in distributed averaging

while *true* **do**
  receive x_v from v;
  send x_w to v;
  $x_w \leftarrow \frac{x_w + x_v}{2}$;
end while

6.3 Related work

To the best of our knowledge, there is no literature that covers the relation between distributed algorithm convergence speed and graph girth. Therefore, in this section, we survey those works that are most closely related. In Section 6.3.1, we discuss works on the relation between topology and convergence. In Section 6.3.2, we discuss works on high-girth graphs and short-cycle removal.

6.3.1 Convergence

There exists a vast body of work that analyses the relation between topology and convergence. These works have their origin in physics, aiming to predict the ability of dynamic networks to spontaneously synchronise [PC98, BP02]. Since similar dynamics occur in distributed systems, results on synchronisability were adopted into computer science, where the concept is referred to as convergence [DHM05, Li+11, Liu+14]. For simplicity, in the following overview, we will speak of convergence even if the cited work is about synchronisation.

Spectral theory. Pecora and Carroll [PC98] and Barahona and Pecora [BP02] show that the convergence speed of a graph is determined by the eigenvalues of that graph's Laplacian. Subsequent literature often uses algebraic connectivity and eigenratio as heuristics of the graph's convergence speed.

Kar, Aldosari and Moura [KAM06] show that (non-bipartite) Ramanujan graphs exhibit high convergence speeds, both as expected from their eigenratio, and as validated in numerical simulations. The authors point to various constructions of Ramanujan graphs in literature.

Donetti, Hurtado and Muñoz [DHM05] propose a new family of graphs that achieve fast convergence: entangled networks. They propose an algorithm that finds entangled networks with a desired number of nodes and average degree. The algorithm starts with an arbitrary graph and, in each iteration, chooses random pairs of edges, performs an edge exchange on each edge pair $((e_1, e_2), (e_3, e_4))$ to get $((e_1, e_4), (e_2, e_3))$, and accepts the change if the eigenratio decreases. By using simulated annealing, the algorithm avoids getting stuck in local optima. Donetti, Neri and Muñoz [DNM06] extend their analysis, and show that entangled networks correspond exactly to so-called cage graphs and Ramanujan graphs. However, the authors conclude that the aforementioned algorithm is inefficient for finding Ramanujan graphs compared to existing literature.

Wang et al. [Wan+07] improve upon the aforementioned edge exchange algorithm by using tabu search instead of simulated annealing. The authors also observe that the clustering coefficient is a good heuristic to predict convergence speed, and show that basing the search algorithm's acceptance criterion on the clustering coefficient also creates graphs with high convergence speeds.

Ghosh and Boyd [GB06] propose a greedy algorithm to optimise algebraic connectivity. At each iteration, find the Fiedler vector u, and add the edge (i, j) with largest $(u_i - u_j)^2$. Since the work focuses on optimising algebraic connectivity, it is not clear how this algorithm affects the convergence speed of distributed averaging.

Degree relations. Rad, Jalili and Hasler [RJH08] propose an algorithm that removes edges based on the sum of adjacent node degrees, and adds edges using the Fiedler vector criterion of Ghosh and Boyd [GB06], and shows that this results in a network with optimised eigenratio, which coincides with Ramanujan graphs. The authors note that many other metrics provide similar results.

In a series of works, Yang and Tang [YT11], Yeung et al. [Yeu+12], and Liu et al. [Liu+14] create increasingly performant heuristics for maximising convergence speed. Ultimately, they settle on a tabu search-based algorithm in which edges are removed and added as done by Rad, Jalili and Hasler [RJH08], and accept the resulting candidates depending on whether the eigenratio improved. The algorithm prefers adding edges between nodes that are within a short distance of each other in the underlying physical network, and ensures that the resulting graph is connected.

However, Donetti, Hurtado and Muñoz [DHM08] show that while degree-degree associations of neighbouring nodes indeed correlate negatively with the network's convergence speed, this correlation is not causative, as the mere act of introducing such heterogeneity does not by itself decrease the eigenratio.

Comparisons. Hagberg and Schult [HS08] compare a multitude of greedy edge-modifying algorithms to determine which methods achieve convergence in the fewest iterations. Overall, they conclude that methods that focus on increasing algebraic connectivity outperform those based on spectral radius and degree criteria, and that edge exchanges are not necessarily better than separate edge additions and removals. The authors do not consider eigenratio as a separate optimisation metric.

Sirocchi and Bogliolo [SB22] extensively compare metrics and find that the metrics that most strongly correlate with high convergence speed of a distributed consensus protocol are high closeness centrality, implying that information travels quickly, and

small clustering coefficient, implying that information is sent non-redundantly. However, these metrics vary in their accuracy for different graph families. Unfortunately, the authors do not investigate eigenratio as a metric.

6.3.2 Girth

We discuss works related to (increasing) girth in graphs.

Moore bound. Firstly, we note the Moore bound [Big93]. For d-regular graphs with girth g, the number of nodes must be at least

$$\begin{cases} 2\sum_{i=0}^{g/2-1}(d-1)^i, & \text{if girth is even} \\ 1+d\sum_{i=0}^{(g-1)/2-1}(d-1)^i, & \text{if girth is odd.} \end{cases} \tag{6.1}$$

Alon, Hoory and Linial [AHL02] show that if d is taken to be the graph's average degree, and each node has at least degree two, Equation 6.1 also holds for irregular graphs. Consequently, another way to interpret the Moore bound is to say that, given the number of nodes and a desired girth, there is an upper limit on the number of edges. Therefore, when a higher girth is desired, the Moore bound dictates that it may be necessary to remove some edges.

High-girth graph constructions. We note several works that present algorithms for constructing graphs with high girth. Though these works do not consider increasing girth in arbitrary existing graphs, the algorithms are interesting nonetheless.

Chandran [Cha03] provides a construction of high-girth almost-regular graphs. Briefly, this algorithm takes the number of nodes n, the desired average degree $k < \frac{n}{3}$, and outputs a graph with girth $g \geq \log_k(n) + O(1)$. The algorithm starts with n nodes and the edges being a perfect matching on those nodes, and then iteratively adds edges between the most distant pair of nodes such that at least one of the nodes in the pair is a node with the lowest degree globally. The graph is almost regular in the sense that any two nodes differ in degree by at most two.

Linial and Simkin [LS21] provide a construction of high-girth regular graphs. Their procedure is similar to that of Chandran [Cha03], but starts with a Hamiltonian cycle G on n vertices instead, and, with high probability, gives a k-regular graph with girth at least $c\log_{k-1}(n)$ for input $0 < c < 1$.

Finally, Lazebnik, Ustimenko and Woldar [LUW95] present a family of high-girth bipartite graphs, but their method cannot be adapted to non-bipartite graphs.

Short-cycle removal. Paredes [Par21] gives a polynomial-time algorithm that, given a d-regular (r,τ)-graph (that is, such that each node has at most one cycle within r hops, and has at most τ cycles of length at most r), where $r \leq \frac{2}{3}\log_{d-1}\left(\frac{n}{\tau}\right) - 5$, outputs a graph with girth $g \geq r$, while ensuring all eigenvalues remain unchanged except for a bounded factor. Briefly, the algorithm works by breaking up all short cycles by removing an arbitrary edge in each, and then adding new edges to restore the spectrum, without reintroducing short cycles. Though this work is the closest to our research question, it does not explicitly investigate the effect stretching has on the convergence speed.

Finally, Hu, Eleftheriou and Arnold [HEA05] and Lau, Tam and Tse [LTT11] both present what are effectively modifications of the aforementioned work by Chandran [Cha03] specifically for bipartite graphs.

6.4 Optimal Graph Stretching Problem

We consider the problem of increasing the girth of a connected graph $G = (V, E)$ to some $g \geq 3$ while achieving maximal distributed averaging convergence speed and ensuring that the graph has (almost) no leaves. Here, convergence speed refers to the reciprocal of the averaging time, which is the expected number of rounds until all nodes are sufficiently[1] close to the average [Boy+05]. Formally, the problem is to find

$$\begin{aligned} \max_{E' \subseteq V \times V} \quad & \text{convergence speed of } H \coloneqq (V, E') \\ \text{such that} \quad & H \text{ is a simple connected graph} \\ & |\{v \in V : \deg(v) < 2\}| = 0 \\ & \operatorname{girth}(H) \geq g \end{aligned}$$

Since this problem is non-linear, it is hard to solve efficiently. Therefore, we relax our problem definition as follows:

- Optimising the convergence speed of a graph is hard [XB04]. Therefore, we settle for a heuristic; recall Section 6.3.1.
- As seen in Moore's bound, there is a difficult-to-control interaction between girth and the number of edges. Therefore, we tolerate the presence of some leaves, as long as a best-effort attempt is made.

Given this relaxed problem formulation, we ask the following research questions:

- How does leaf minimisation affect convergence speed?
- What is the effect of different stretching methods on convergence speed?
- What heuristic achieves maximal convergence speed?

We describe our method in Section 6.5 and present our results in Section 6.6.

6.5 Method

We present our method for answering the questions posed in Section 6.4. At a high level, the way we solve the optimal graph stretching problem is to first modify the given graph to satisfy the constraints, and then greedily optimise for the convergence speed heuristic. More specifically, our approach consists of the following steps:

1. Generate a graph. (Section 6.5.1)
2. Increase the girth. (Section 6.5.2)
3. Minimise the number of leaves. (Section 6.5.3)
4. Optimise graph using a heuristic. (Section 6.5.4)
5. Run distributed averaging. (Section 6.5.5)

We repeat this procedure 500 times for each combination of parameters. We provide more details in the subsequent sections. Source code for the experiments is publicly available [Dek25]. We present the results of our method in Section 6.6.

[1] The formal definition of "sufficiently" is given in the cited works, but is not relevant for our formulation.

6.5.1 Generate Graphs

The accuracy with which heuristics predict convergence speed varies between graph types [SB22]. Therefore, we generate graphs from four families commonly used to model real-world networks. Each graph is characterised by its number of nodes n and some family-specific parameters. For all graphs, we choose n uniformly randomly from the range $\{25 \dots 100\}$. After fixing a set of parameters, we keep generating graphs until a connected graph is found. We consider the following graph families:

- (n, p) Erdős–Rényi graphs, where p determines for each possible edge the probability that it is included. We choose p uniformly random from real range $[\ln(n)/n, 1]$, ensuring an overwhelming probability of being connected [ER60].
- (n, k, p) Watts–Strogatz graphs, which have small-world properties (i.e. high clustering and low distance), which are generated by connecting each node to the previous k and next k nodes (creating a ring lattice), and then rewiring each edge with probability p. We choose k uniformly random from integer range $[1, \text{floor}\,(n/2))$ and p uniformly random from real range $[0, 1]$, which is the full range of valid parameters.
- (n, m) Barabási–Albert graphs, which have scale-free properties (i.e. asymptotic degree distribution), which are generated by starting with a star topology with $m + 1$ nodes, and then iteratively adding the remaining nodes. Each new node is connected to m random existing nodes, with probabilities proportional to nodes' degrees, without replacement. We choose m uniformly random from the integer range $[1, n)$, which is the full range of valid parameters.
- (n, r) geometric graphs, which represent physical networks, and are generated by placing the nodes uniformly random in the unit square, and connecting pairs of nodes within Euclidean distance at most r. We choose r uniformly random from real range $[1.1 \times \sqrt{\log(n)/(n\pi)}, 1)$, ensuring an overwhelming probability of being connected [Pen97].

6.5.2 Stretch Graphs

Though the underlying application we consider is a distributed protocol, the stretching algorithm itself need not be distributed. To stretch the girth of a graph to threshold g, all cycles with length below g must be removed. Trivially, it suffices to find all short cycles and remove one edge from each. However, since cycles may overlap, this naive method may disconnect the graph, and typically removes more edges than necessary.

In our experiments, we stretch graphs from girth 3 up to and including 10. Here, girth 3 represents no stretching at all (because every graph has girth at least 3), and girth 10 was chosen because preliminary experiments revealed that very little happens when stretching to even higher girths.

We consider three approaches for stretching a graph to a desired girth. All three approaches work by repeatedly removing a specific edge until the girth has reached g, but differ in how they select that edge:

- **Random:** Any edge that is part of a cycle.
- **Least-Cycles:** The edge that is part of the smallest number of shortest cycles.
- **Most-Cycles:** The edge that is part of the largest number of shortest cycles.

Each approach considers only those edges that can be removed without disconnecting the graph. When multiple edges match the criterion, one such edge is chosen randomly.

Remark 5. Note that the third method is expected to remove the most edges. We include it nonetheless because the subsequent optimiser in Section 6.5.4 may benefit from starting with fewer edges.

Remark 6. Note that the second and third method consider the "number of shortest cycles", not the "number of short cycles". If the graph currently has girth g', then only cycles with exactly length g' are counted. Eventually, the graph reaches girth $g' + 1$, and only cycles with exactly length $g' + 1$ will be counted, and so on until the graph reaches girth g. The reason for this is that the "number of short cycles" quickly becomes infeasibly large, while the "number of shortest cycles" remains much smaller. For example, the complete graph with 25 nodes has 2300 length-3 cycles, 10 626 000 length-6 cycles, and 41 186 376 000 length-9 cycles. After most-cycles stretching to girth 4, only 157 392 length-6 cycles and 8 015 760 length-9 cycles remain, and, after subsequently stretching to girth 5, 84 length-6 cycles and zero length-9 cycles remain.

Finding all cycles with length equal to the graph's girth can be done using a simple depth-first search. We perform this search once at the start, and once again whenever the girth increases. We store the results in a sparse matrix with a row for each cycle and a column for each edge, similar to an incidence matrix. (If cycles are hyperedges, then this is the incidence matrix of that hypergraph.) When an edge is removed, its column is removed from the matrix, and so are all rows that contained that edge. This way, rows always correspond exactly to eligible cycles, and columns to edges that can be removed without disconnecting the graph. Finding the edge that is in the largest number of cycles is simply a case of finding the column with the largest number of non-zero values. Columns can be mapped to edges by keeping track in a map.

6.5.3 Minimise Leaves

We minimise the number of leaves in the graph without removing nodes and without reducing girth below the threshold g. We present three methods, which are variations of one algorithm. We repeat each experiment four times: once for each method, and once without leaf minimisation.

The high-level algorithm works by iteratively adding edges between pairs of nodes. To ensure the girth does not sink below g, pairs with distance strictly less than $g - 1$ are excluded. Initially, the algorithm only connects leaves to other leaves, but when no suitable pairs remain, the algorithm moves on to connect leaves to non-leaves. The algorithm terminates when no suitable pairs remain.

The three leaf minimisation methods we propose all use the above algorithm but differ in how they choose which pair to connect from the list of candidates:

- **Random:** Connect a random pair of nodes.
- **Closest:** Connect the pair of nodes with the shortest distance.
- **Furthest:** Connect the pair of nodes with the largest distance.

This method may fail to remove all leaves in some cases. For example, when girth is stretched to $g = 4$, this may create a star topology, after which adding an edge will always reduce the girth to $g = 3$. In this case, our method will not add any edges. As noted in Section 6.4, this is acceptable.

6.5.4 Optimise Convergence

After minimising the number of leaves and stretching the graph to the desired girth, we optimise the graph's convergence speed for distributed averaging. We run a greedy algorithm that adds or removes edges until any such change would worsen the convergence speed. To estimate the convergence speed, we employ a heuristic. We do not allow the addition or removal of edges that would disconnect the graph, add leaves, or decrease girth below the desired value.

Our choice of heuristics is based on Section 6.3.1: We choose two graph metrics that are known the correlate well with convergence speed [SB22], and two spectral metrics known to provide bounds on convergence speed [GB06]. We repeat each experiment several times, once for each heuristic:

- **Eigenratio.** Equals $\frac{\lambda_2}{\lambda_n}$. Maximised.
- **Algebraic connectivity.** Equals λ_2. Maximised.
- **Closeness centrality.** Equals $\sum_{u \in V} \left(\frac{|V|-1}{\sum_{v \in V} d_{u,v}} \right) / |V|$, given pairwise distances d. Maximised.
- **Global efficiency.** Equals $\frac{1}{|V|(|V|-1)} \sum_{u,v \in V, u \neq v} \frac{1}{d_{u,v}}$, given pairwise distances d. Maximised.

Remark 7. The choice for maximisation (rather than minimisation) is based on preliminary results that show that, in our setting, each of these heuristics correlates positively with convergence speed.

Remark 8. We do not consider clustering coefficient as a metric because, for girth larger than four, the clustering coefficient is always zero by definition.

We efficiently choose edge removal candidates by finding a cycle basis of the current candidate graph. This reveals the list of all edges which are in any cycle of any length. These are exactly the edges that can be removed without disconnecting the graph, since an edge is part of a cycle if and only if the two end nodes have at least two different paths to each other.

We efficiently choose edge addition candidates by finding all pairs of nodes with distance at least $g - 1$. Adding an edge anywhere else would create a short cycle.

The above operations and heuristics require knowing at each iteration the adjacency matrix, degree matrix, and pairwise distances. Instead of constantly recalculating these, we calculate these for the initial graph and "patch" them when an edge is added or removed. These patches all take constant time, except for patching the pairwise distances when an edge is added, which requires a complete recalculation.

6.5.5 Run Distributed Averaging

We use the asynchronous push-pull model with single-neighbour selection, as described in Section 6.2.2. At any point in time, given the vector of initial values x and the vector of intermediate values $\bar{x}$, we define the error norm as $\frac{\|\bar{x}-x\|_2}{\|x\|_2}$.

Each node is assigned an integer value uniformly random from the range $[0, 50]$. We continue the protocol until the error norm is less than 0.01. The convergence time is then the number of rounds taken until convergence is achieved. For each experiment,

to control for randomness, we run 10 instances of distributed averaging, and take the mean convergence time.

The range of starting values does not affect the output; only the variance does. Similarly, the exact error norm bound does not qualitatively affect our results.

6.6 Results

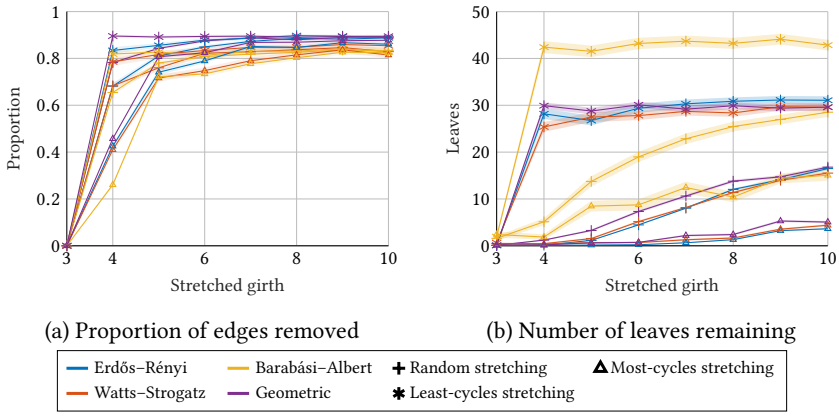
We present the results obtained through the method in Section 6.5. Firstly, we look at the impact that girth stretching has on convergence time, without considering leaf removal and optimisation in Section 6.6.1. Secondly, we consider the impact of leaf removal in Section 6.6.2. Finally, we look at the real meat of this work, which is the comparison of various heuristics, specifically when combined with stretching and leaf removal in Section 6.6.3.

In all figures, we use shaded areas to show the 95% confidence interval, calculated over 500 independent samples per plot point.

6.6.1 Stretching

We compare the stretching methods by the number of edges removed, number of leaves created, optimisation heuristics, and convergence time.

Edges and leaves. In Figure 6.1a, we show the proportion of edges removed by stretching for each combination of graph family and stretching method. Note that a girth of three implies that no stretching has taken place. Though the proportion quickly increases for all experiments, it also immediately flattens out. This shows that, at least in these graph families, removing all short cycles is typically sufficient to remove the majority of longer cycles (recall Remark 6 on page 115). As expected, the most-cycles stretching method removes the smallest proportion of edges, followed by random stretching, and then least-cycles stretching. Watts–Strogatz graphs and Barabási–Albert graphs require removing the smallest proportion of edges; their being highly clustered means that most cycles are centred around just a few edges, which are quickly removed. However, as girth increases, differences between graph types and stretching methods diminish significantly.



(a) Proportion of edges removed (b) Number of leaves remaining

Figure 6.1 Analysis of edges after stretching to a desired girth

In Figure 6.1b, we show the number of leaves in stretched graphs. All graphs have (nearly) no leaves at girth 3, which is before any stretching takes place. The number of leaves quickly goes up when the graph is stretched, with major differences between graph types and stretching methods. Among graph types, we observe that Barabási–Albert graphs have significantly more leaves than all other graph types regardless of which stretching method is used. This is because these graphs have many low-degree nodes, so the removal of any edge is likely to create a new leaf. When we compare stretching methods, we see that, regardless of graph type, most-cycles stretching creates very few new leaves even when stretching to girth 10, random stretching performs approximately three times as badly, and least-cycles stretching shoots up so quickly that it hits a ceiling because the stretched graph is (nearly) a tree.

Convergence heuristics. In Figure 6.2, we show the convergence time heuristics for stretched graphs. In all cases, higher is better. The four heuristics behave quite similarly, predicting worse convergence time as girth increases, but predicted performance flattens out at higher girths. Across graph types, all heuristics predict that Barabási–Albert graphs and geometric graphs perform worse when stretched to low girths, but joins up with the rest once stretched to girth 10. Across stretching methods, least-cycles stretching typically drops down immediately before flooring out, while random stretching and most-cycles stretching approach this floor gradually with increased girth, with the latter keeping higher predicted convergence times.

We note that the most-cycles stretching method exhibits a "sawtooth" pattern, where heuristics drop harder at odd values than at even values. When we inspect cycle counts in individual graphs, we find that stretching to an even girth typically also removes a disproportionate amount of odd-length cycles, even those longer than the desired girth. For example, after stretching to girth 4 with the most-cycles method, the resulting graphs often end up having fewer length-7 cycles than length-6 cycles, even though this is not true for any of the unstretched graphs. This holds even if we use a variant of the most-cycles stretching method that counts *all* cycles, not just the shortest ones (see Remark 6 on page 115). This effect is most pronounced in Barabási–Albert graphs.

Convergence time. In Figure 6.3, we show the empirical convergence time for stretched graphs. Lower is better. It is immediately clear that least-cycles stretching performs terribly, presenting a fourfold increase compared to random stretching, and a sevenfold increase in convergence time compared to most-cycles stretching. We see from Figure 6.2 that the heuristics are decent predictors of convergence time, though the predicted divide between graph types is not present in the empirical measurements. We note, however, that an even better predictor of performance is the number of leaves removed (see Figure 6.1b), or rather, the number of nodes removed.

We conclude that convergence time is seriously impacted by stretching, but that this is not due to cycle removal per se, but due to the removal of many edges. Therefore, most-cycles stretching is the optimal method, despite its sawtooth behaviour.

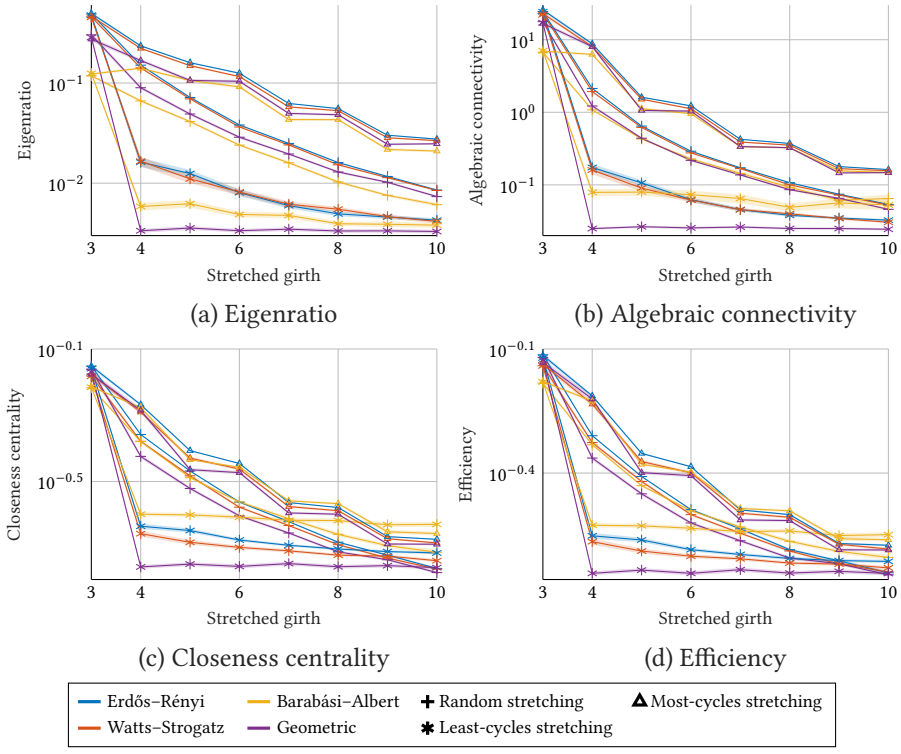


Figure 6.2 Convergence heuristics after stretching to a desired girth

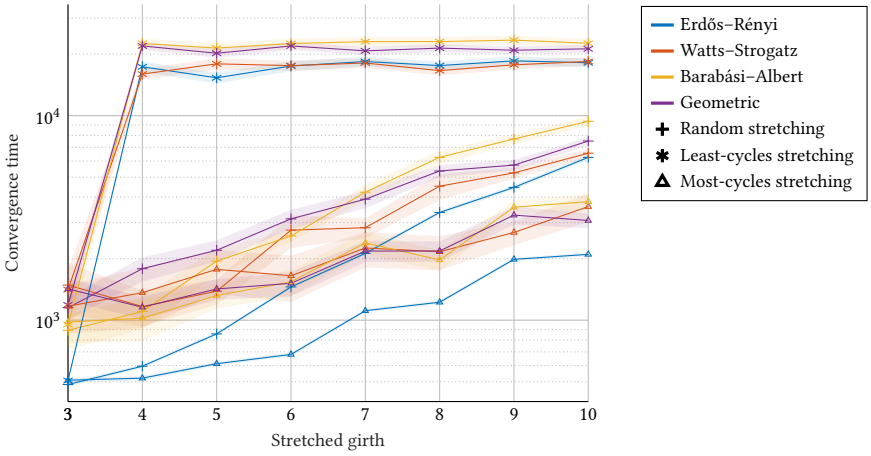


Figure 6.3 Convergence time after stretching

6.6.2 Leaf Minimisation

We look at how effective leaf minimisation is at removing leaves, and at its effect on convergence time.

Leaves and edges. In Figure 6.4, we show the number of leaves that remain after leaf minimisation. Note that, unlike previous graphs, colours indicate leaf minimisation method, not graph family. The lines representing no leaf minimisation correspond exactly to Figure 6.1b. When we compare stretching methods, we see that least-cycles stretching creates the largest number of leaves, followed by random stretching, and then most-cycles stretching, though the latter two are close. When we compare leaf minimisation methods, we see only small differences, with closest leaf minimisation most effectively eliminating leaves, followed by random leaf minimisation, and finally furthest leaf minimisation. There are no significant differences between graph types.

In Figure 6.5, we show the number of edges added by leaf minimisation. Recall that our minimisation method starts by connecting leaves to each other before connecting leaves to non-leaves, and thus the number of edges added is not necessarily linear in the number of leaves eliminated. The lines for most-cycles stretching and random stretching are similar to their counterparts in Figure 6.4, whereas the least-cycles stretching line goes down when girth goes up. The latter result is visible in Figure 6.4: The number of leaves before minimisation hits a ceiling and stays the same, while the number of leaves after minimisation increases. Thus, fewer leaves have been eliminated, and therefore fewer edges must have been added. Overall, this implies that the graph's diameter (the length of the longest shortest path) resulting from least-cycles stretching is too small to allow leaf minimisation without reducing girth.

Convergence time. In Figure 6.6, we show the convergence time after leaf minimisation. There are no significant differences between leaf minimisation methods. Though the sawtooth pattern with most-cycles stretching complicates the graphs, it is clear that leaf minimisation improves convergence time for all stretching methods, especially least-cycles stretching. However, we argue that it is not the leaf minimisation itself that improves the convergence, but simply the fact that *any* edges are added to the graph. This is apparent from the lack of similarity to Figure 6.4 and Figure 6.5. We conclude that leaf minimisation is neither detrimental nor beneficial to performance.

6.6.3 Optimisation

Finally, we look at the effect of optimising convergence time with heuristics.

Number of edges. In Figure 6.7, we show the number of edges added or removed during optimisation, without considering leaf optimisation. Intuitively, this is a measure of how many steps stretched graphs are removed from the optimum. On average, graphs have 238 edges before optimisation and 380 edges after optimisation, with significantly more edges added than removed. However, the number of changes decreases as girth increases. Though greedy algorithms may get stuck in local optima, additional experiments using simulated annealing based on the method by Jalili and Rad [JR09] show that even search algorithms without this drawback require a decreasing number of changes to the edge set. The downwards trend thus appears to be inherent to the optimal graph stretching problem itself.

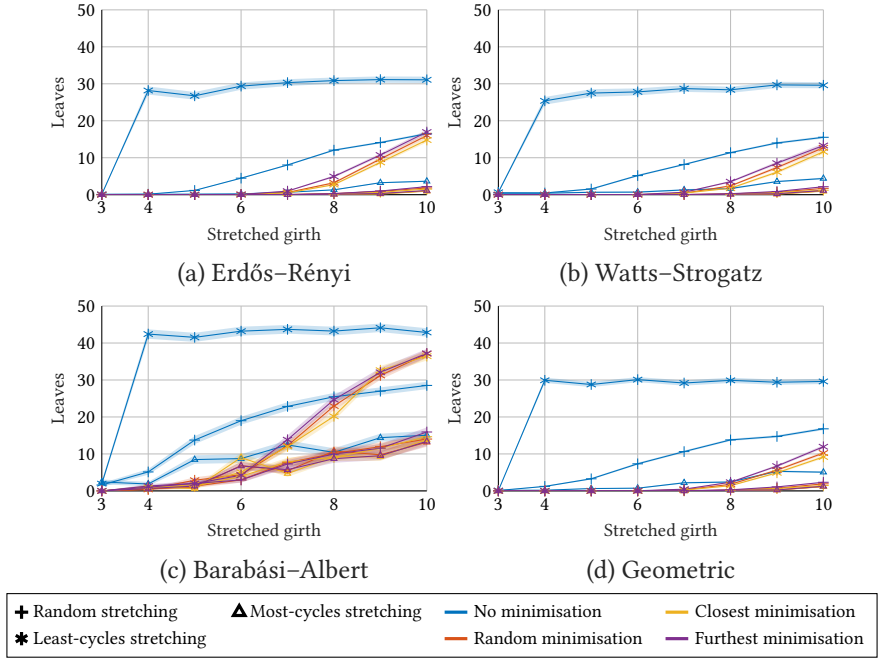


Figure 6.4 Number of leaves remaining after leaf minimisation

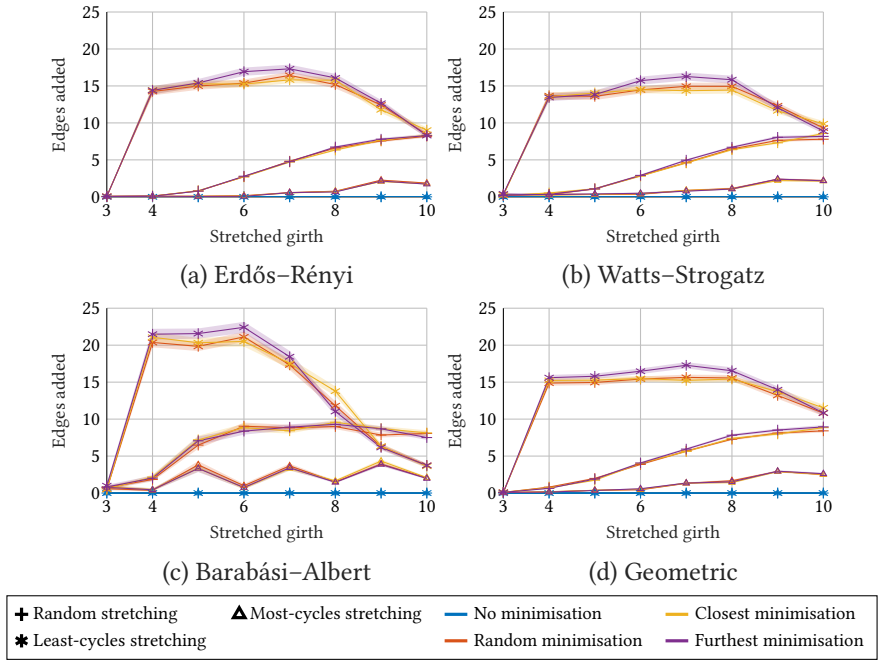


Figure 6.5 Number of edges added during leaf minimisation

When we compare graph types, we see that they differ only in scale, with Barabási–Albert graphs requiring the most changes. In all four graph types, stretched graphs require the fewest changes after most-cycles stretching, followed by random stretching, and then least-cycles stretching. The only exception is low-girth graphs optimised by eigenratio, where all stretching methods perform similarly.

Convergence time. In Figure 6.8, we show the effect of heuristic optimisation on convergence time per graph family. Note the different y-axis scale per column. All sixteen graphs have many similarities. When we compare stretching methods, most-cycles stretching and random stretching achieve the lowest convergence time, followed by least-cycles stretching, defeating the hypothesis that the optimiser may benefit from fewer edges being removed. When we consider leaf minimisation, we see that there is little difference between the various methods, and confirm that leaf minimisation by itself is not responsible for improved convergence time. When we compare heuristics, we also do not see a clear winner. Though graphs stretched with the least-cycles method appear to benefit from choosing the right heuristic for the graph type, differences are much smaller for the other stretching methods. Finally, several figures, especially those describing Barabási–Albert graphs, contain the aforementioned sawtooth pattern.

6.7 Conclusion

We investigated the relation between a graph's girth and the convergence time of distributed averaging. We introduced the *optimal graph stretching problem*, which is the task of increasing the girth of a graph while keeping the convergence time and number of leaves minimal, and the graph connected. We proposed and implemented a sequence of algorithms to solve this problem, which we applied to hundreds of thousands of graphs, after which we measured the results.

We find that stretching the girth of a graph increases convergence time proportional to the number of edges removed. Consequently, stretching by iteratively removing the edge that is simultaneously in the largest number of cycles results in the smallest convergence time cost. Furthermore, convergence time can be recuperated using a greedy algorithm to add edges without decreasing girth. Finally, minimising the number of leaves does not affect convergence time.

We note a few possible avenues for future work. Firstly, the aforementioned stretching method creates a sawtooth pattern in the distribution of cycle lengths, which may be of independent interest. Secondly, the studied heuristics correlate worse with convergence time than in related work; we postulate that our results may be improved by developing high-girth-specific heuristics. Finally, our solution to the optimal graph stretching problem requires global knowledge of the graph, but for ad-hoc networks it may be useful to create a distributed solution.

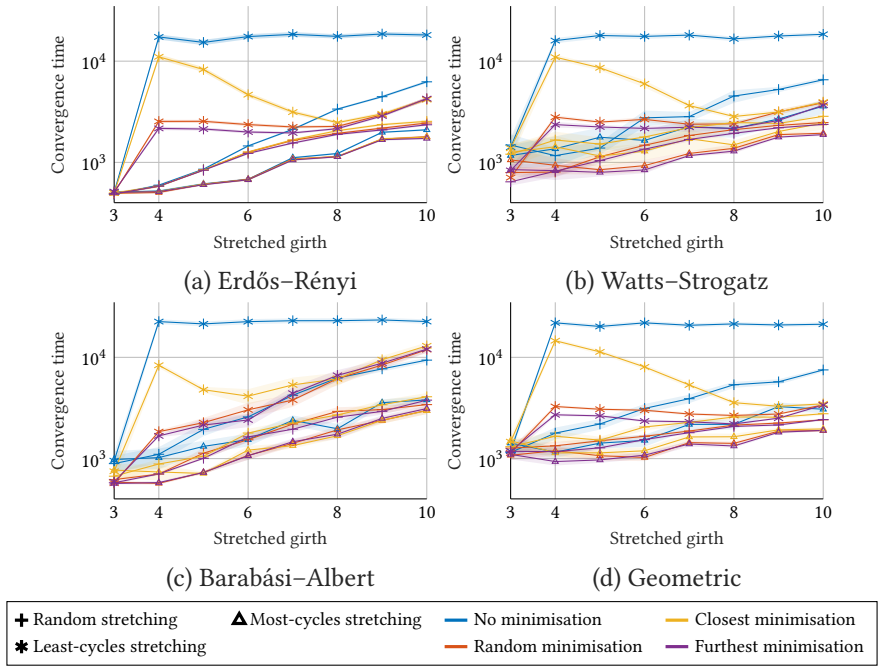


Figure 6.6 Convergence time after leaf minimisation

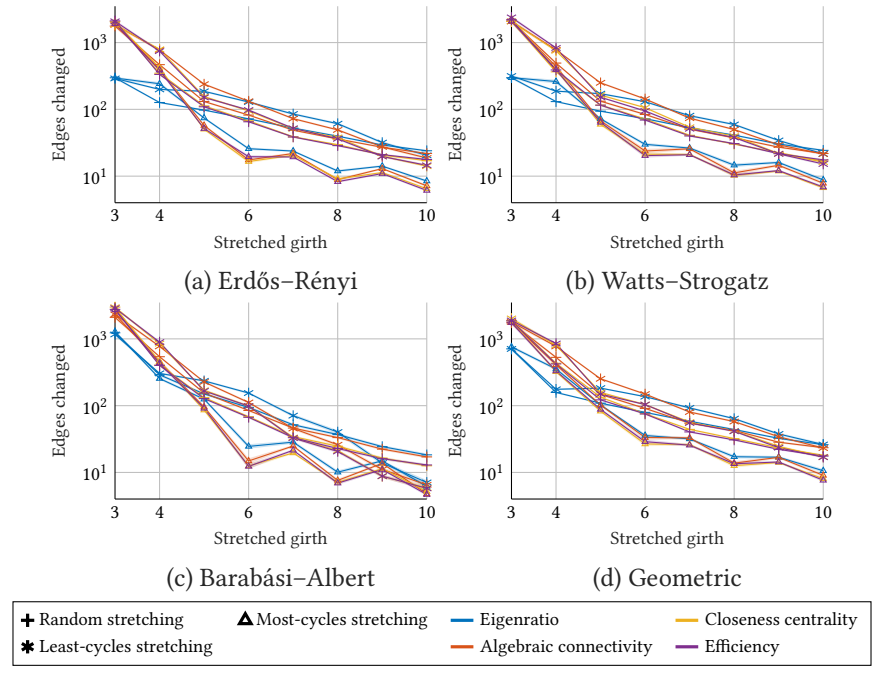


Figure 6.7 Number of edges added or removed during optimisation

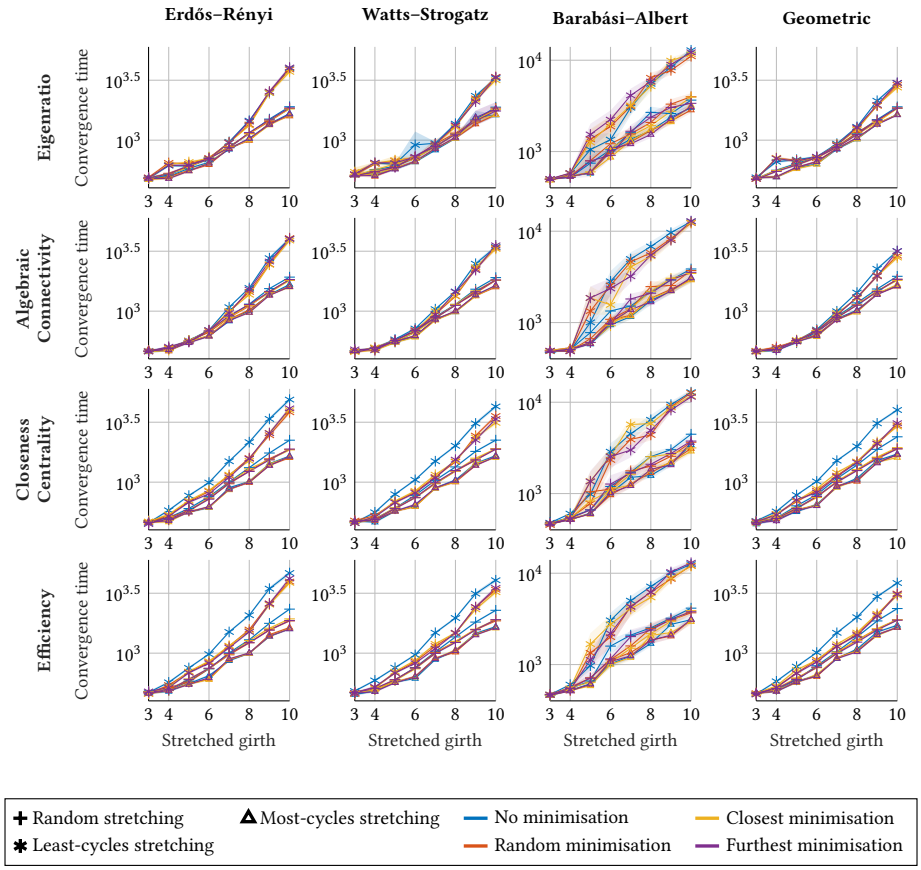


Figure 6.8 Convergence time after heurisical optimisation, with columns indicating the graph type, and rows indicating the heuristic that was optimised

Part IV

Epilogue

Chapter 7

Discussion

If Multi-Party Computation (MPC) is to see mass adoption in the coming years, we require a deeper understanding of reconstruction attacks. Current security frameworks allow proving that nothing leaks beyond what can be inferred from the outputs, but determining what can be inferred from the outputs is itself an open question. While several recent works have studied reconstruction attacks in MPC, this direction of research is young, and there are no comprehensive theories yet. This dissertation contributes to our understanding of the privacy guarantees of MPC by developing a graph-theoretic model of reconstruction attacks on summations.

We have enhanced the privacy-preserving summation primitive by proposing two extensions to respectively achieve input and output validation, showed the feasibility of reconstruction attacks, described these attacks in graph-theoretic terms, and empirically validated the applicability of our defences to distributed averaging. Though our results on reconstruction attacks apply to all sequence of sums in all contexts and all architectures, the exemplifying use case in this dissertation has been privacy-preserving distributed averaging, which can be implemented as a sequence of local summations in a distributed network. Beyond distributed averaging, summation alone is a sufficient primitive to implement more complex operations such as principal component analysis, singular-value decomposition, and decision tree classifications, simply by writing the inputs as aggregate-sum queries [Blu+05].

7.1 Achievements

Summation validation. We enhanced the privacy-preserving summation primitive by creating extensions to respectively valid inputs and outputs. In Chapter 2, we introduced a probabilistic alternative to zero-knowledge proofs, ensuring inputs are within a particular range without requiring expensive cryptographic operations. Our main achievement has been the specific construction of the topology and the analysis of its efficiency and resistance. In Chapter 3, we introduce the MPVAS family of protocols, which allow verifiers to check that the aggregator's output is truthful. Unlike existing works, MPVAS works even when all adversaries are actively malicious, when the aggregator, users, and verifiers collude, and requires only a single server.

Reconstruction attack formalisation. In Chapter 4, we formalised reconstruction attacks on summations algebraically. Our notation is generic to the method by which sums are obtained, sufficient to allow adversaries to include observations made about themselves, and allows users to update their values at any time.

Attack feasibility. In Chapter 4, we showed that reconstruction attacks are feasible. We generated a large number of random graphs, and showed that *passively* malicious adversaries will typically stumble upon sequences of summations that allow them to infer at least some private information. Our results apply even when users choose independently random new values while adversaries' are still collecting data. We

conclude that any system in which a sufficiently dynamic set of summations occurs will eventually leak private data; in line with the Fundamental Law of Information Recovery [DR14].

Girth criterion. In Chapter 4, we showed that reconstruction attacks require cycles. Specifically, if ℓ adversaries successfully reconstruct private data, then the graph must have at least one cycle of length at most 2ℓ, i.e. the graph's girth is at most 2ℓ. We conclude that, if there are at most ℓ adversaries, it is sufficient to remove all cycles with length below 2ℓ. We term the process of increasing the girth of a graph "stretching", analogous to the action of stretching the holes in a set of plastic six-pack rings. Since cycle detection and removal can be performed locally in distributed networks, graph stretching is the first fully-distributed reconstruction prevention method.

Cycle detection. In Chapter 5, we proposed a novel peer-to-peer cycle detection protocol. Previous works are either centralised or leak significant amounts of information. Our protocol relies on flooding, and uses a novel key exchange algorithm to recognise when cycles occur, after which a simple cycle recovery sub-protocol follows. Due to the high communication complexity, our protocol is best-suited for low-degree graphs.

Cycle removal. In Chapter 6, we compared various methods for removing cycles from graphs in terms of their impact on the performance of distributed averaging protocols. We find that removing the edges that are in the largest number of short cycles is the best amongst the compared methods, but random edge removal is a close second. Since determining the number of short cycles that each edge is in is a complex task, we instead count only the number of *shortest* cycles that each edge is in, and show that the resulting effect is nearly identical.

Distributed averaging. In Chapter 6, apart from the cycle removal method, we also investigated other relations between girth and distributed averaging performance. Based on measurements over millions of graphs, we found that while removing cycles decreases performance significantly, the main cause is not the removal of cycles themselves, but the removal of edges. We thus proposed to counteract the effects of graph stretching by a subsequent edge addition phase, which heuristically selects which edges to add to the graph to improve performance. All in all, edge addition compensates for a significant portion of the performance deterioration due to stretching.

7.2 Limitations

The works presented in this dissertation do not provide a full answer to the question of how to prevent reconstruction attacks. Additionally, our works are subject to various assumptions, and can thus yet be generalised.

Trivial attacks. The graph girth criterion from Chapter 4 works only to defend against non-trivial attacks. In the most trivial attack, an adversary calculates a "privacy-preserving" summation of only one neighbour. Despite the total absence of cycles in the resulting graph, the adversary clearly learns the other user's private value. Luckily, this trivial attack is easily defended against, since the user can simply reject participation if no other users take part in the summation. However, in general, adversaries have access to colluders whom they can include in summations, and whose values they can subtract from the total to uncover the honest user's value. Again,

despite the absence of cycles, the adversary still learns private data.

We do not consider this a flaw in our work. In our work, we assume that individual summations are secure, and determine the additional leakage that occurs as a result of composition. In the trivial attacks outlined above, the leakage occurs not because of composition, but because there is a single faulty summation. For completeness' sake, however, we note that we can extend our graph criterion to include these trivial attacks: To protect against m adversaries, the graph must have at least girth $2m + 1$ *and* at least degree $m + 2$. Unfortunately, real-world networks typically do not have high minimum degrees [BA99, LO14, BSZ15].

Partial leakage. Our work focuses on detecting situations in which users' values leak in their entirety. We did not consider other types of leakage, such as partial leakage and relative linear combinations. Partial leakage occurs when a sequence of summations narrows down the range that a private value is in. For example, when working with non-negative numbers, the sum $A + B + C = 100$ implies that each value is at most 100; after subsequently also learning that $A + D + E = 15$, the adversaries additionally learn that $A \leq 15$ and $85 \leq B + C \leq 100$. Relative linear combinations occur when adversaries learn to express private values in terms of other private values. For example, after learning that $A + B = 80$ and $A + C = 90$, adversaries know that $C - B = 10$, which may be private information in its own right.

Auxiliary knowledge. Adversaries may have an advantage if they obtain auxiliary knowledge from outside the protocol. For example, if one dataset publishes $A+B+C = 100$, and another publishes $A + B = 50$, then combining the two reveals $C = 50$, even though neither individual dataset contains any cycles. Our core results are independent of the existence of auxiliary knowledge: We simply model the additional leakage that occurs as a result of sequences of summations, regardless of where those summations come from.

If there is no way of knowing whether auxiliary knowledge may be obtained, and no way to model the worst-case scope of what such knowledge could entail, then our results are not applicable. Indeed, inherent to the kind of syntactic requirements we impose, our results are mostly applicable to situations in which it is feasible to exert some amount of control over which summations take place. Modeling and tracking the exact leakage that occurs over time, as noted earlier, may provide a partial remedy.

7.3 Future Work

Entropy-based analysis. Perhaps the most interesting potential avenue for future work would be to combine our results with those presented by Baccarini, Blanton and Zou [BBZ24]. While our work shows the family of graphs that fully leaks users' private inputs, Baccarini, Blanton and Zou provide a more granular method of determining leakage, at least under the assumption that users' values follow a given distribution. Combining their methodology with ours could provide us with heuristics (or exact functions) that assign leakage scores based only on (sub)structures of the full network. Overall, future work could provide improved guidance on privacy-preserving network designs.

Formal security. Several of our contributions would benefit from deeper scrutiny into the exact privacy guarantees provided. Amongst others, MPVAS (see Chapter 3) would

benefit from formal security proofs, including proofs in composition frameworks, and our cycle detection protocol (see Chapter 5) would benefit from a complete proof.

Differential privacy. Differential privacy aims to reduce information loss by adding random noise to outputs. The added noise is calibrated in a precise way to make privacy loss measurable. The privacy loss guarantees of multiple operations can be composed elegantly, while the design of the noise mechanism in each operation is agnostic of the remaining design. Meanwhile, in our work, we show that restricting which summations may take place prevents data from leaking. We argue that allowing such restrictions to be made in differentially private pipelines allows one to reduce the amount of noise required for each operation. As long as it is feasible to assume that some operations will not take place, this hybrid approach could significantly increase the accuracy of differentially private outputs.

Bibliography

[AB01] Réka Albert and Albert-László Barabási. *Statistical mechanics of complex networks*. 2001. arXiv: cond-mat/0106096 (cited on page 101).

[ÁC11] Gergely Ács and Claude Castelluccia. "I have a DREAM! (DiffeRentially privatE smArt Metering)". In: *IH 2011: Proceedings of the 13th International Conference on Information Hiding*. Volume 6958. Lecture Notes in Computer Science. 2011, pages 118–132. DOI: 10.1007/978-3-642-24178-9_9 (cited on page 16).

[ÅF20] Anders Åslund and Julia Friedlander. *Defending the United States against Russian dark money*. Nov. 2020. URL: https://www.atlanticcouncil.org/in-depth-research-reports/report/defending-the-united-states-against-russian-dark-money/ (visited on 16th June 2025) (cited on page 92).

[AH20a] Patrick Ah-Fat and Michael Huth. "Protecting private inputs: Bounded distortion guarantees with randomised approximations". In: *Proceedings on Privacy Enhancing Technologies*. Volume 2020.3. 2020, pages 284–303. DOI: 10.2478/POPETS-2020-0053 (cited on page 6).

[AH20b] Patrick Ah-Fat and Michael Huth. *Two and three-party digital goods auctions: Scalable privacy analysis*. 2020. arXiv: 2009.09524 (cited on page 6).

[AHL02] Noga Alon, Shlomo Hoory and Nathan Linial. "The Moore bound for irregular graphs". In: *Graphs and Combinatorics* 18.1 (2002), pages 53–57. DOI: 10.1007/S003730200002 (cited on pages 108, 112).

[ALM20] Adi Akavia, Rio LaVigne and Tal Moran. "Topology-hiding computation on all graphs". In: *Journal of Cryptology* 33.1 (2020), pages 176–227. DOI: 10.1007/S00145-019-09318-Y (cited on page 94).

[AMK19] Alireza Ahadipour, Mojtaba Mohammadi and Alireza Keshavarz-Haddad. *Statistical-based privacy-preserving scheme with malicious consumers identification for smart grid*. 2019. arXiv: 1904.06576 (cited on page 17).

[AOdR22] Wirawan Agahari, Hosea Ofe and Mark de Reuver. "It is not (only) about privacy: How multi-party computation redefines control, trust, and risk in data sharing". In: *Electronic Markets* 32.3 (2022), pages 1577–1602. DOI: 10.1007/S12525-022-00572-W (cited on page 4).

[Ara+17] Anees Ara, Mznah Al-Rodhaan, Yuan Tian and Abdullah Al-Dhelaan. "A secure privacy-preserving data aggregation scheme based on bilinear ElGamal cryptosystem for remote health monitoring systems". In: *IEEE Access* 5 (2017), pages 12601–12617. DOI: 10.1109/ACCESS.2017.2716439 (cited on page 53).

[Ate+05] Giuseppe Ateniese, Jan Camenisch, Susan Hohenberger and Breno de Medeiros. “Practical group signatures without random oracles”. In: *IACR Cryptology ePrint Archive* (2005). URL: https://ia.cr/2005/385 (cited on pages 40–41).

[ATK15] Leman Akoglu, Hanghang Tong and Danai Koutra. “Graph based anomaly detection and description: A survey”. In: *Data Mining and Knowledge Discovery* 29.3 (2015), pages 626–688. DOI: 10.1007/S10618-014-0365-Y (cited on page 93).

[BA99] Albert-László Barabási and Réka Albert. “Emergence of scaling in random networks”. In: *Science* 286.5439 (1999), pages 509–512. ISSN: 0036-8075,1095-9203. DOI: 10.1126/science.286.5439.509 (cited on page 131).

[Bak+15] Bence Gabor Bakondi, Andreas Peter, Maarten H. Everts, Pieter H. Hartel and Willem Jonker. “Publicly verifiable private aggregation of time-series data”. In: *ARES 2015: Proceedings of the 10th International Conference on Availability, Reliability and Security*. 2015, pages 50–59. DOI: 10.1109/ARES.2015.82 (cited on pages 36–39, 41, 52–53, 58, 61).

[Bal+23] Marshall Ball, Elette Boyle, Ran Cohen, Lisa Kohl, Tal Malkin, Pierre Meyer and Tal Moran. “Topology-hiding communication from minimal assumptions”. In: *Journal of Cryptology* 36.4, 39 (2023). DOI: 10.1007/S00145-023-09473-3 (cited on page 93).

[BBZ24] Alessandro N. Baccarini, Marina Blanton and Shaofeng Zou. “Understanding information disclosure from secure computation output: A study of average salary computation”. In: *CODASPY 2024: Proceedings of the 14th ACM Conference on Data and Application Security and Privacy*. 2024, pages 187–198. DOI: 10.1145/3626232.3653280 (cited on pages 6, 131).

[BC18] Anna D. Broido and Aaron Clauset. *Scale-free networks are rare*. 2018. arXiv: 1801.03400 (cited on page 101).

[BD20] David Balson and William Dixon. *Cyber information sharing: Building collective security*. 2020. URL: https://www.weforum.org/publications/cyber-information-sharing-building-collective-security/ (visited on 16th June 2025) (cited on page 4).

[Bel+18] Aurélien Bellet, Rachid Guerraoui, Mahsa Taziki and Marc Tommasi. “Personalized and private peer-to-peer machine learning”. In: *AISTATS 2018: Proceedings of the 21st International Conference on Artificial Intelligence and Statistics*. Volume 84. Proceedings of Machine Learning Research. 2018, pages 473–481. URL: https://proceedings.mlr.press/v84/bellet18a.html (cited on pages 68, 71–72).

[Bel+20] James Henry Bell, Kallista A. Bonawitz, Adrià Gascón, Tancrède Lepoint and Mariana Raykova. “Secure single-server aggregation with (poly)logarithmic overhead”. In: *CCS 2020: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 2020, pages 1253–1269. DOI: 10.1145/3372297.3417885 (cited on pages 39, 68).

[Ben+18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh and Michael Riabzev. "Scalable, transparent, and post-quantum secure computational integrity". In: *IACR Cryptology ePrint Archive* (2018). URL: https://ia.cr/2018/046 (cited on page 18).

[Big93] Norman Biggs. *Algebraic graph theory*. 2nd edition. Cambridge Mathematical Library. 1993. ISBN: 0-521-45897-8 (cited on page 112).

[Bit+17] Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnés and Bernhard Seefeld. "Prochlo: Strong privacy for analytics in the crowd". In: *SOSP 2017: Proceedings of the 26th Symposium on Operating Systems Principles*. 2017, pages 441–459. DOI: 10.1145/3132747.3132769 (cited on page 16).

[Bla79] G. R. Blakley. "Safeguarding cryptographic keys". In: *1979 International Workshop on Managing Requirements Knowledge, MARK*. 1979, pages 313–318. DOI: 10.1109/MARK.1979.8817296 (cited on page 53).

[BLS04] Dan Boneh, Ben Lynn and Hovav Shacham. "Short signatures from the Weil pairing". In: *Journal of Cryptology* 17.4 (2004), pages 297–319. DOI: 10.1007/S00145-004-0314-9 (cited on page 61).

[Blu+05] Avrim Blum, Cynthia Dwork, Frank McSherry and Kobbi Nissim. "Practical privacy: The SuLQ framework". In: *PODS 2005: Proceedings of the 24th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. 2005, pages 128–138. DOI: 10.1145/1065167.1065184 (cited on pages 6, 17, 69, 129).

[BN15] Finn Brunton and Helen Nissenbaum. *Obfuscation: A user's guide for privacy and protest*. 2015. DOI: 10.7551/mitpress/9780262029735.001.0001 (cited on page 3).

[Bog+09] Peter Bogetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas P. Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, Michael I. Schwartzbach and Tomas Toft. "Secure multiparty computation goes live". In: *FC 2009: Proceedings of the 13th International Conference on Financial Cryptography and Data Security*. Volume 5628. Lecture Notes in Computer Science. 2009, pages 325–343. DOI: 10.1007/978-3-642-03549-4_20 (cited on page 4).

[Bog+14] Dan Bogdanov, Peeter Laud, Sven Laur and Pille Pullonen. "From input private to universally composable secure multi-party computation primitives". In: *CSF 2014: Proceedings of the 27th IEEE Computer Security Foundations Symposium*. 2014, pages 184–198. DOI: 10.1109/CSF.2014.21 (cited on page 71).

[Bon+17] Kallista A. 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: *CCS 2017: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017, pages 1175–1191. DOI: 10.1145/3133956.3133982 (cited on pages 16, 39, 68).

[Bou+20] Fabrice Boudot, Pierrick Gaudry, Aurore Guillevic, Nadia Heninger, Emmanuel Thomé and Paul Zimmermann. "Comparing the difficulty of factorization and discrete logarithm: A 240-digit experiment". In: *Advances in Cryptology – CRYPTO 2020: Proceedings of the 40th Annual International Cryptology Conference*. Volume 12171. Lecture Notes in Computer Science. 2020, pages 62–91. DOI: 10.1007/978-3-030-56880-1_3 (cited on page 101).

[Bou00] Fabrice Boudot. "Efficient proofs that a committed number lies in an interval". In: *Advances in Cryptology – EUROCRYPT 2000: Proceedings of the 2000 International Conference on the Theory and Application of Cryptographic Techniques*. Volume 1807. Lecture Notes in Computer Science. 2000, pages 431–444. DOI: 10.1007/3-540-45539-6_31 (cited on pages 16, 18).

[Boy+05] Stephen P. Boyd, Arpita Ghosh, Balaji Prabhakar and Devavrat Shah. "Gossip algorithms: Design, analysis and applications". In: *INFOCOM 2005: Proceedings of the 24th IEEE Annual Joint Conference of the IEEE Computer and Communications Societies*. 2005, pages 1653–1664. DOI: 10.1109/INFCOM.2005.1498447 (cited on pages 108, 113).

[Boy+06] Stephen P. Boyd, Arpita Ghosh, Balaji Prabhakar and Devavrat Shah. "Randomized gossip algorithms". In: *IEEE Transactions on Information Theory* 52.6 (2006), pages 2508–2530. DOI: 10.1109/TIT.2006.874516 (cited on pages 74, 109).

[BP02] Mauricio Barahona and Louis M. Pecora. "Synchronization in small-world systems". In: *Physical Review Letters* 89, 054101 (5 July 2002). DOI: 10.1103/PhysRevLett.89.054101 (cited on pages 108, 110).

[BPW07] Michael Backes, Birgit Pfitzmann and Michael Waidner. "The reactive simulatability (RSIM) framework for asynchronous systems". In: *Information and Computation* 205.12 (2007), pages 1685–1720. DOI: 10.1016/j.ic.2007.05.002 (cited on pages 5, 71).

[BR19] Elaine Barker and Allen Roginsky. *Transitioning the use of cryptographic algorithms and key lengths*. Mar. 2019. DOI: 10.6028/NIST.SP.800-131Ar2 (cited on pages 53, 101).

[BSZ15] Shankar Bhamidi, J. Michael Steele and Tauhid Zaman. "Twitter event networks and the superstar model". In: *The Annals of Applied Probability* 25.5 (2015), pages 2462–2502. ISSN: 1050-5164,2168-8737. DOI: 10.1214/14-AAP1053 (cited on page 131).

[Bün+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille and Gregory Maxwell. "Bulletproofs: Short proofs for confidential transactions and more". In: *S&P 2018: Proceedings of the 2018 IEEE Symposium on Security and Privacy*. 2018, pages 315–334. DOI: 10.1109/SP.2018.00020 (cited on pages 18, 28–29, 49).

[Bur+06] Jeff Burke, Deborah Estrin, Mark Hansen, Andrew Parker, Nithya Ramanathan, Sasank Reddy and Mani B. Srivastava. "Participatory sensing". In: *WSW 2006: Workshop on World-Sensor-Web*. 2006. URL: https://escholarship.org/uc/item/19h777qd (cited on page 16).

[Can+22] Ran Canetti, Gabe Kaptchuk, Leonid Reyzin, Adam Smith and Mayank Varia. *Response to the RFI on advancing privacy-enhancing technologies*. 8th July 2022. URL: https://www.nitrd.gov/rfi/2022/87-fr-35250/Canetti-Kaptchuk-Reyzin-Smith-Varia-PET-RFI-Response-2022.pdf (visited on 16th June 2025) (cited on page 4).

[Can01] Ran Canetti. "Universally composable security: A new paradigm for cryptographic protocols". In: *FOCS 2001: Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*. 2001, pages 136–145. DOI: 10.1109/SFCS.2001.959888 (cited on pages 5, 71).

[CB17] Henry Corrigan-Gibbs and Dan Boneh. "Prio: Private, robust, and scalable computation of aggregate statistics". In: *NSDI 2017: Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation*. 2017, pages 259–282. URL: https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/corrigan-gibbs (cited on pages 18, 28–29).

[CBU24] Edwige Cyffers, Aurélien Bellet and Jalaj Upadhyay. *Differentially private decentralized learning with random walks*. 2024. arXiv: 2402.07471 (cited on pages 82, 88).

[CD21] Bennett Cyphers and Cory Doctorow. *Privacy without monopoly: Data protection and interoperability*. 12th Feb. 2021. URL: https://www.eff.org/wp/interoperability-and-privacy (visited on 16th June 2025) (cited on page 3).

[Cen+21] Keren Censor-Hillel, Orr Fischer, Tzlil Gonen, François Le Gall, Dean Leitersdorf and Rotem Oshman. *Fast distributed algorithms for girth, cycles and small subgraphs*. 2021. arXiv: 2101.07590 (cited on pages 82, 89).

[Cha03] L. Sunil Chandran. "A high girth graph construction". In: *SIAM Journal on Discrete Mathematics* 16.3 (2003), pages 366–370. DOI: 10.1137/S0895480101387893 (cited on page 112).

[Che+18a] Xuhui Chen, Jinlong Ji, Changqing Luo, Weixian Liao and Pan Li. "When machine learning meets blockchain: A decentralized, privacy-preserving and secure design". In: *2018 IEEE International Conference on Big Data*. 2018, pages 1178–1187. DOI: 10.1109/BigData.2018.8622598 (cited on page 68).

[Che+18b] Hsin-Pai Cheng, Patrick Yu, Haojing Hu, Feng Yan, Shiyu Li, Hai Li and Yiran Chen. *LEASGD: An efficient and privacy-preserving decentralized algorithm for distributed learning*. 2018. arXiv: 1811.11124 (cited on pages 82, 88).

[Chi78] Francis Y. L. Chin. "Security in statistical databases for queries with small counts". In: *ACM Transactions on Database Systems* 3.1 (1978), pages 92–104. DOI: 10.1145/320241.320250 (cited on pages 5, 70).

[Chr16] Delphine Christin. "Privacy in mobile participatory sensing: Current trends and future challenges". In: *Journal of Systems and Software* 116 (2016), pages 57–68. DOI: 10.1016/J.JSS.2015.03.067 (cited on page 16).

[CÖ82] Francis Y. L. Chin and Gultekin Özsoyoglu. "Auditing and inference control in statistical databases". In: *IEEE Transactions on Software Engineering* 8.6 (1982), pages 574–582. DOI: 10.1109/TSE.1982.236161 (cited on pages 5, 69–70).

[Cor+13] Graham Cormode, Cecilia M. Procopiuc, Entong Shen, Divesh Srivastava and Ting Yu. "Empirical privacy and empirical utility of anonymized data". In: *ICDEW 2013: Workshops Proceedings of the 29th IEEE International Conference on Data Engineering*. 2013, pages 77–82. DOI: 10.1109/ICDEW.2013.6547431 (cited on pages 69, 73).

[CT13] Chris Clifton and Tamir Tassa. "On syntactic anonymity and differential privacy". In: *Transactions on Data Privacy* 6.2 (2013), pages 161–183. URL: https://www.tdp.cat/issues11/abs.a124a13.php (cited on pages 5, 69, 73).

[Cui+18] Hui Cui, Zhiguo Wan, Robert H. Deng, Guilin Wang and Yingjiu Li. "Efficient and expressive keyword search over encrypted data in cloud". In: *IEEE Transactions on Dependable and Secure Computing* 15.3 (2018), pages 409–422. DOI: 10.1109/TDSC.2016.2599883 (cited on page 53).

[Cyf+22] Edwige Cyffers, Mathieu Even, Aurélien Bellet and Laurent Massoulié. "Muffliato: Peer-to-peer privacy amplification for decentralized optimization and averaging". In: *NeurIPS 2022: Advances in Neural Information Processing Systems*. 2022. URL: https://proceedings.neurips.cc/paper/2022/hash/65d32185f73cbf4535449a792c63926f-Abstract-Conference.html (cited on pages 68–69, 71).

[Dan+18] Gábor Danner, Árpád Berta, István Hegedüs and Márk Jelasity. "Robust fully distributed minibatch gradient descent with privacy preservation". In: *Security and Communication Networks* 2018 (2018). DOI: 10.1155/2018/6728020 (cited on pages 68, 72).

[Dan03] George Danezis. "Statistical disclosure attacks: Traffic confirmation in open environments". In: *SEC2003: Proceedings of the 18th IFIP TC11 International Conference on Information Security: Security and Privacy in the Age of Uncertainty*. Volume 250. IFIP Conference Proceedings. 2003, pages 421–426 (cited on page 70).

[DBB22] Bogdan Dumitrescu, Andra Baltoiu and Stefania Budulan. "Anomaly detection in graphs of bank transactions for anti money laundering applications". In: *IEEE Access* 10 (2022), pages 47699–47714. DOI: 10.1109/ACCESS.2022.3170467 (cited on pages 92–93).

[DE21] Florine W. Dekker and Zekeriya Erkin. "Privacy-preserving data aggregation with probabilistic range validation". In: *ACNS 2021: Proceedings of the 19th International Conference on Applied Cryptography and Network Security*. Volume 12727. Lecture Notes in Computer Science. 2021, pages 79–98. DOI: 10.1007/978-3-030-78375-4_4 (cited on pages 6, 9, 15, 72).

[DE25] Florine W. Dekker and Zekeriya Erkin. *Source code underlying the publication: Privacy-preserving data aggregation with probabilistic range validation*. 15th May 2025. DOI: 10.4121/b9db276f-5522-4986-9d98-e9710134fd71.v1 (cited on pages 10, 29).

[DEC25a] Florine W. Dekker, Zekeriya Erkin and Mauro Conti. *Optimal graph stretching for distributed averaging*. 2025. arXiv: 2504.10289 (cited on pages 8–9, 107).

[DEC25b] Florine W. Dekker, Zekeriya Erkin and Mauro Conti. *Source code underlying the publication: Topology-based reconstruction prevention for decentralised learning*. 13th Jan. 2025. DOI: 10.4121/21572601.v2 (cited on pages 10, 79, 88).

[DEC25c] Florine W. Dekker, Zekeriya Erkin and Mauro Conti. "Topology-based reconstruction prevention for decentralised learning". In: *Proceedings on Privacy Enhancing Technologies* 2025.1 (2025), pages 553–566. DOI: 10.56553/POPETS-2025-0030 (cited on pages 7, 9–10, 67, 108).

[Dek25] Florine W. Dekker. *Source code underlying the publication: Optimal graph stretching for distributed averaging*. 16th June 2025. DOI: 10.4121/e64c61d3-deb5-4aad-af60-92d92755781f.v3 (cited on pages 10, 113).

[Dem+88] Alan J. Demers, Daniel H. Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard E. Sturgis, Daniel C. Swinehart and Douglas B. Terry. "Epidemic algorithms for replicated database maintenance". In: *ACM SIGOPS Operating Systems Review* 22.1 (1988), pages 8–32. DOI: 10.1145/43921.43922 (cited on page 110).

[Den80] Dorothy E. Denning. "Secure statistical databases with random sample queries". In: *ACM Transactions on Database Systems* 5.3 (1980), pages 291–315. DOI: 10.1145/320613.320616 (cited on pages 5, 69–70).

[DH76] Whitfield Diffie and Martin E. Hellman. "New directions in cryptography". In: *IEEE Transactions on Information Theory* 22.6 (1976), pages 644–654. DOI: 10.1109/TIT.1976.1055638 (cited on page 97).

[DHM05] Luca Donetti, Pablo I. Hurtado and Miguel A. Muñoz. "Entangled networks, synchronization, and optimal network topology". In: *Physical Review Letters* 95, 188701 (18 Oct. 2005). DOI: 10.1103/PhysRevLett.95.188701 (cited on pages 110–111).

[DHM08] Luca Donetti, Pablo I. Hurtado and Miguel A. Muñoz. "Network synchronization: Optimal and pessimal scale-free topologies". In: *Journal of Physics A: Mathematical and Theoretical* 41.22, 224008 (2008). ISSN: 1751-8113,1751-8121. DOI: 10.1088/1751-8113/41/22/224008 (cited on page 111).

[Dhr+20] Sanket S. Dhruva, Joseph S. Ross, Joseph G. Akar, Brittany Caldwell, Karla Childers, Wing Chow, Laura Ciaccio, Paul Coplan, Jun Dong, Hayley J. Dykhoff, Stephen Johnston, Todd Kellogg, Cynthia Long, Peter A. Noseworthy, Kurt Roberts, Anindita Saha, Andrew Yoo and Nilay D. Shah. "Aggregating multiple real-world data sources using a patient-centered health-data-sharing platform". In: *npj Digital Medicine* 3 (2020). DOI: 10.1038/s41746-020-0265-z (cited on page 36).

[DK08] Shlomi Dolev and Ronen I. Kat. "HyperTree for self-stabilizing peer-to-peer systems". In: *Distributed Computing* 20.5 (2008), pages 375–388. DOI: 10.1007/s00446-007-0038-9 (cited on pages 82, 89).

[DNM06] Luca Donetti, Franco Neri and Miguel A. Muñoz. "Optimal network topologies: Expanders, cages, Ramanujan graphs, entangled networks and all that". In: *Journal of Statistical Mechanics: Theory and Experiment* 2006.08, P08007 (2006). DOI: 10.1088/1742-5468/2006/08/P08007 (cited on page 111).

[DR14] Cynthia Dwork and Aaron Roth. "The algorithmic foundations of differential privacy". In: *Foundations and Trends in Theoretical Computer Science* 9.3-4 (2014), pages 211–407. DOI: 10.1561/0400000042 (cited on pages 69, 71, 130).

[dSil+04] Josenildo Costa da Silva, Matthias Klusch, Stefano Lodi and Gianluca Moro. "Inference attacks in peer-to-peer homogeneous distributed data mining". In: *ECAI'2004: Proceedings of the 16th European Conference on Artificial Intelligence*. 2004, pages 450–454 (cited on pages 69, 71).

[DT19] Nora A. Draper and Joseph Turow. "The corporate cultivation of digital resignation". In: *New Media & Society* 21.8 (2019), pages 1824–1839. DOI: 10.1177/1461444819833331 (cited on page 3).

[Dwo06] Cynthia Dwork. "Differential privacy". In: *ICALP 2006: Proceedings on the 33rd International Colloquium on Automata, Languages and Programming*. Volume 4052. Lecture Notes in Computer Science. 2006, pages 1–12. DOI: 10.1007/11787006_1 (cited on pages 5, 69–71).

[EDF21] European Commission, Directorate-General for Financial Stability and Financial Services and Capital Markets Union. *Impact assessment accompanying the anti-money laundering package*. CELEX number 52021SC0190. July 2021. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52021SC0190 (visited on 16th June 2025) (cited on page 92).

[EGS03] Alexandre V. Evfimievski, Johannes Gehrke and Ramakrishnan Srikant. "Limiting privacy breaches in privacy preserving data mining". In: *PODS 2003: Proceedings of the 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. 2003, pages 211–222. DOI: 10.1145/773153.773174 (cited on page 71).

[Eis08] Bennett Eisenberg. "On the expectation of the maximum of iid geometric random variables". In: *Statistics and Probability Letters* 78.2 (2008), pages 135–143. ISSN: 01677152. DOI: 10.1016/j.spl.2007.05.011 (cited on page 30).

[Emu+19] Keita Emura, Hayato Kimura, Toshihiro Ohigashi and Tatsuya Suzuki. "Privacy-preserving aggregation of time-series data with public verifiability from simple assumptions and its implementations". In: *The Computer Journal* 62.4 (2019), pages 614–630. DOI: 10.1093/comjnl/bxy135 (cited on pages 58, 61).

[ER60] Paul Erdős and Alfréd Rényi. "On the evolution of random graphs". In: *A Magyar Tudományos Akadémia. Matematikai Kutató Intézetének Közleményei* 5 (1960), pages 17–61. ISSN: 0541-9514 (cited on page 114).

[Erk15] Zekeriya Erkin. "Private data aggregation with groups for smart grids in a dynamic setting using CRT". In: *WIFS 2015: Proceedings of the 2015 IEEE International Workshop on Information Forensics and Security*. 2015. DOI: 10.1109/WIFS.2015.7368584 (cited on pages 16, 18, 25, 27).

[ET12] Zekeriya Erkin and Gene Tsudik. "Private computation of spatial and temporal power consumption with smart meters". In: *ACNS 2012: Proceedings of the 10th International Conference on Applied Cryptography and Network Security*. Volume 7341. Lecture Notes in Computer Science. 2012, pages 561–577. DOI: 10.1007/978-3-642-31284-7_33 (cited on pages 16, 18, 21, 25, 36).

[Fan+12] Xi Fang, Satyajayant Misra, Guoliang Xue and Dejun Yang. "Smart grid – The new and improved power grid: A survey". In: *IEEE Communications Surveys & Tutorials* 14.4 (2012), pages 944–980. DOI: 10.1109/SURV.2011.101911.00087 (cited on page 36).

[Fel72] Ivan P. Fellegi. "On the question of statistical confidentiality". In: *Journal of the American Statistical Association* 67.337 (1972), pages 7–18. DOI: 10.1080/01621459.1972.10481199 (cited on pages 4, 69–70).

[Fie73] Miroslav Fiedler. "Algebraic connectivity of graphs". In: *Czechoslovak Mathematical Journal* 23(98) (1973), pages 298–305. ISSN: 0011-4642 (cited on page 109).

[FLC15] Jingyao Fan, Qinghua Li and Guohong Cao. "Privacy-aware and trustworthy data aggregation in mobile sensing". In: *CNS 2015: Proceedings of the IEEE Conference on Communications and Network Security*. 2015, pages 31–39. DOI: 10.1109/CNS.2015.7346807 (cited on page 36).

[FPE16] Giulia Fanti, Vasyl Pihur and Úlfar Erlingsson. "Building a RAPPOR with the unknown: Privacy-preserving learning of associations and data dictionaries". In: *Proceedings on Privacy Enhancing Technologies* 2016.3 (2016), pages 41–61. DOI: 10.1515/POPETS-2016-0015 (cited on page 16).

[FS86] Amos Fiat and Adi Shamir. "How to prove yourself: Practical solutions to identification and signature problems". In: *Advanced in Cryptology – CRYPTO 1986: Proceedings*. Volume 263. Lecture Notes in Computer Science. 1986, pages 186–194. DOI: 10.1007/3-540-47721-7_12 (cited on page 41).

[FTC19] Federal Trade Commission. "FTC imposes $5 billion penalty and sweeping new privacy restrictions on Facebook". In: *Federal Trade Commission* (24th July 2019). URL: https://www.ftc.gov/news-events/news/press-releases/2019/07/ftc-imposes-5-billion-penalty-sweeping-new-privacy-restrictions-facebook (visited on 16th June 2025) (cited on page 3).

[Fun21] Brian Fung. "Amazon hit by record $887 million EU privacy fine". In: *CNN* (30th July 2021). URL: https://edition.cnn.com/2021/07/30/tech/amazon-eu-privacy-fine/ (visited on 16th June 2025) (cited on page 3).

[Fun24] Brian Fung. "FCC fines wireless carriers millions for sharing user locations without consent". In: *CNN* (29th Apr. 2024). URL: https://edition.cnn.com/2024/04/29/tech/fcc-fines-att-verizon-200-million/ (visited on 16th June 2025) (cited on page 3).

[GB06] Arpita Ghosh and Stephen P. Boyd. "Growing well-connected graphs". In: *CDC 2006: Proceedings of the 45th IEEE Conference on Decision and Control*. 2006, pages 6605–6611. DOI: 10.1109/CDC.2006.377282 (cited on pages 108, 111, 116).

[GGP10] Rosario Gennaro, Craig Gentry and Bryan Parno. "Non-interactive verifiable computing: Outsourcing computation to untrusted workers". In: *Advances in Cryptology – CRYPTO 2010: Proceedings of the 30th Annual Cryptology Conference*. Volume 6223. Lecture Notes in Computer Science. 2010, pages 465–482. DOI: 10.1007/978-3-642-14623-7_25 (cited on page 39).

[GJ10] Flavio D. Garcia and Bart Jacobs. "Privacy-friendly energy-metering via homomorphic encryption". In: *STM 2010: Proceedings of the 6th International Workshop on Security and Trust Management*. Volume 6710. Lecture Notes in Computer Science. 2010, pages 226–238. DOI: 10.1007/978-3-642-22444-7_15 (cited on pages 16, 36, 69).

[GM82] Shafi Goldwasser and Silvio Micali. "Probabilistic encryption and how to play mental poker keeping secret all partial information". In: *STOC 1982: Proceedings of the 14th Annual ACM Symposium on Theory of Computing*. 1982, pages 365–377. DOI: 10.1145/800070.802212 (cited on page 4).

[GMS90] Robert Grone, Russell Merris and V. S. Sunder. "The Laplacian spectrum of a graph". In: *SIAM Journal on Matrix Analysis and Applications* 11.2 (1990), pages 218–238. ISSN: 0895-4798. DOI: 10.1137/0611016 (cited on page 109).

[Gor+15] S. Dov Gordon, Jonathan Katz, Feng-Hao Liu, Elaine Shi and Hong-Sheng Zhou. "Multi-client verifiable computation with stronger security guarantees". In: *TCC 2015: Proceedings of the 12th International Conference on Theory of Cryptography*. Volume 9015. Lecture Notes in Computer Science. 2015, pages 144–168. DOI: 10.1007/978-3-662-46497-7_6 (cited on page 39).

[Guo+21] Xiaojie Guo, Zheli Liu, Jin Li, Jiqiang Gao, Boyu Hou, Changyu Dong and Thar Baker. "VeriFL: Communication-efficient and fast verifiable aggregation for federated learning". In: *IEEE Transactions on Information Forensics and Security* 16 (2021), pages 1736–1751. DOI: 10.1109/TIFS.2020.3043139 (cited on pages 36–39).

[Guo+22] Shangwei Guo, Tianwei Zhang, Guowen Xu, Han Yu, Tao Xiang and Yang Liu. "Topology-aware differential privacy for decentralized image classification". In: *IEEE Transactions on Circuits and Systems for Video Technology* 32.6 (2022), pages 4016–4027. DOI: 10.1109/TCSVT.2021.3105723 (cited on pages 69, 71).

[Guo22] Xiaojie Guo. "Fixing issues and achieving maliciously secure verifiable aggregation in 'VeriFL: Communication-efficient and fast verifiable aggregation for federated learning'". In: *IACR Cryptology ePrint Archive* (2022). URL: https://ia.cr/2022/1073 (cited on page 39).

[Hah+23] Changhee Hahn, Hodong Kim, Minjae Kim and Junbeom Hur. "VerSA: Verifiable secure aggregation for cross-device federated learning". In: *IEEE Transactions on Dependable and Secure Computing* 20.1 (2023), pages 36–52. DOI: 10.1109/TDSC.2021.3126323 (cited on pages 36–39).

[HAP17] Briland Hitaj, Giuseppe Ateniese and Fernando Pérez-Cruz. "Deep models under the GAN: Information leakage from collaborative deep learning". In: *CCS 2017: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017, pages 603–618. DOI: 10.1145/3133956.3134012 (cited on page 71).

[HDC18] Christoforos N. Hadjicostis, Alejandro D. Domínguez-García and Themistoklis Charalambous. "Distributed averaging and balancing in network systems: With applications to coordination and control". In: *Foundations and Trends in Systems and Control* 5.2-3 (2018), pages 99–292. DOI: 10.1561/2600000016 (cited on pages 108–109).

[HEA05] Xiao-Yu Hu, Evangelos Eleftheriou and Dieter-Michael Arnold. "Regular and irregular progressive edge-growth Tanner graphs". In: *IEEE Transactions on Information Theory* 51.1 (2005), pages 386–398. DOI: 10.1109/TIT.2004.839541 (cited on pages 108, 112).

[HGY22] Waleed Hilal, S. Andrew Gadsden and John Yawney. “Financial fraud: A review of anomaly detection techniques and recent advances”. In: *Expert Systems with Applications* 193, 116429 (2022). DOI: 10.1016/J.ESWA.2021.116429 (cited on page 93).

[HK20] László Hajdu and Miklós Krész. “Temporal network analytics for fraud detection in the banking sector”. In: *ADBIS, TPDL and EDA 2020 Common Workshops and Doctoral Consortium*. Volume 1260. Commun. in Comput. and Inf. Science. 2020, pages 145–157. DOI: 10.1007/978-3-030-55814-7_12 (cited on page 92).

[HS08] Aric Hagberg and Daniel A. Schult. “Rewiring networks for synchronization”. In: *Chaos* 18.3, 037105 (2008). ISSN: 1054-1500,1089-7682. DOI: 10.1063/1.2975842 (cited on pages 108, 111).

[Hul+06] Bret Hull, Vladimir Bychkovsky, Yang Zhang, Kevin Chen, Michel Goraczko, Allen Miu, Eugene Shih, Hari Balakrishnan and Samuel Madden. “CarTel: A distributed mobile sensor computing system”. In: *SenSys 2006: Proceedings of the 4th ACM Conference on Embedded Networked Sensor Systems*. 2006, pages 125–138. DOI: 10.1145/1182807.1182821 (cited on page 36).

[IPB19] Paul Irofti, Andrei Patrascu and Andra Baltoiu. *Fraud detection in networks: State-of-the-art*. 2019. arXiv: 1910.11299 (cited on page 93).

[JD25] Juno Jense and Florine W. Dekker. *Source code underlying the publication: Privacy-preserving peer-to-peer cycle detection*. 17th June 2025. DOI: 10.4121/d23e6d7d-15d9-4c83-86de-5a3fc1fd5aa6.v1 (cited on pages 10, 101).

[JE19] Bargav Jayaraman and David Evans. “Evaluating differentially private machine learning in practice”. In: *USENIX Security 2019: Proceedings of the 28th USENIX Security Symposium*. 2019, pages 1895–1912. URL: https://www.usenix.org/conference/usenixsecurity19/presentation/jayaraman (cited on page 5).

[Jen+25] Juno Jense, Florine W. Dekker, Zekeriya Erkin and Mauro Conti. *Privacy-preserving peer-to-peer cycle detection*. 2025. In preparation (cited on pages 7, 9, 91).

[JR09] Mahdi Jalili and Ali Ajdari Rad. “Comment on ‘Rewiring networks for synchronization’”. In: *Chaos* 19.2, 028101 (2009). ISSN: 1054-1500,1089-7682. DOI: 10.1063/1.3130929 (cited on page 120).

[JSJ19] Adel Jebali, Salma Sassi and Abderrazak Jemai. “Inference control in distributed environment: A comparison study”. In: *CRiSIS 2019: Proceedings of the 14th International Conference on Risks and Security of Internet and Systems*. Volume 12026. Lecture Notes in Computer Science. 2019, pages 69–83. DOI: 10.1007/978-3-030-41568-6_5 (cited on page 71).

[Kai+21] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista A. Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaïd Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu and Sen Zhao. "Advances and open problems in federated learning". In: *Foundations and Trends in Machine Learning* 14.1–2 (2021), pages 1–210. DOI: 10.1561/2200000083 (cited on page 68).

[KAM06] Soummya Kar, Saeed A. Aldosari and José M. F. Moura. *Topology for distributed inference on graphs*. 2006. arXiv: cs/0606052 (cited on page 110).

[Kan+20] Renuga Kanagavelu, Zengxiang Li, Juniarto Samsudin, Yechao Yang, Feng Yang, Rick Siow Mong Goh, Mervyn Cheah, Praewpiraya Wiwatphonthana, Khajonpong Akkarajitsakul and Shangguang Wang. "Two-phase multi-party computation enabled privacy-preserving federated learning". In: *CCGRID 2020: Proceedings of the 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing*. 2020, pages 410–419. DOI: 10.1109/CCGrid49817.2020.00-52 (cited on page 68).

[Kap16] Bonnie Kaplan. "How should health data be used?" In: *Cambridge Quarterly of Healthcare Ethics* 25.2 (2016), pages 312–329. ISSN: 1469-2147. DOI: 10.1017/S0963180115000614 (cited on page 36).

[Kas+08] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova and Adam D. Smith. "What can we learn privately?" In: *FOCS 2008: Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science*. 2008, pages 531–540. DOI: 10.1109/FOCS.2008.27 (cited on page 71).

[KDK11] Klaus Kursawe, George Danezis and Markulf Kohlweiss. "Privacy-friendly aggregation for the smart-grid". In: *PETS 2011: Proceedings of the 11th International Symposium on Privacy Enhancing Technologies*. Volume 6794. Lecture Notes in Computer Science. 2011, pages 175–191. DOI: 10.1007/978-3-642-22263-4_10 (cited on pages 16, 18, 25, 36, 39).

[Kha+11] M. Amin Khajehnejad, Arash Saber Tehrani, Alexandros G. Dimakis and Babak Hassibi. "Explicit matrices for sparse approximation". In: *ISIT 2011: Proceedings of the 2011 IEEE International Symposium on Information Theory Proceedings*. 2011, pages 469–473. DOI: 10.1109/ISIT.2011.6034170 (cited on page 108).

[KL14] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. 2nd edition. 2014. ISBN: 9781466570269 (cited on page 40).

[KÖB21] Ferhat Karakoç, Melek Önen and Zeki Bilgin. "Secure aggregation against malicious users". In: *SACMAT 2021: Proceedings of the 26th ACM Symposium on Access Control Models and Technologies*. 2021, pages 115–124. DOI: 10.1145/3450569.3463572 (cited on page 36).

[KR19] Diptendu Mohan Kar and Indrajit Ray. *Systematization of knowledge and implementation: Short identity-based signatures*. 2019. arXiv: 1908.05366 (cited on page 53).

[Kur10] Klaus Kursawe. "Some ideas on privacy preserving meter aggregation". In: *Radboud Universiteit Nijmegen, Technical Report ICIS–R11002* (2010). URL: https://hdl.handle.net/2066/290003 (cited on pages 16–17).

[Lap+18] Andrei Lapets, Frederick Jansen, Kinan Dak Albab, Rawane Issa, Lucy Qin, Mayank Varia and Azer Bestavros. "Accessible privacy-preserving web-based data analysis for assessing and addressing economic inequalities". In: *COMPASS 2018: Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies*. 2018, 48. DOI: 10.1145/3209811.3212701 (cited on page 4).

[LeM+07] Michael LeMay, George Gross, Carl A. Gunter and Sanjam Garg. "Unified architecture for large-scale attested metering". In: *HICCS-40: Proceedings of the 40th Hawaii International International Conference on Systems Science*. 2007, pages 115–124. DOI: 10.1109/HICSS.2007.586 (cited on page 16).

[Leo+15] Iraklis Leontiadis, Kaoutar Elkhiyaoui, Melek Önen and Refik Molva. "PUDA – Privacy and unforgeability for data aggregation". In: *CANS 2015: Proceedings of the 14th International Conference on Cryptology and Network Security*. Volume 9476. Lecture Notes in Computer Science. 2015, pages 3–18. DOI: 10.1007/978-3-319-26823-1_1 (cited on pages 36–41, 52, 58, 61).

[Li+10] Zhongkui Li, Zhisheng Duan, Guanrong Chen and Lin Huang. "Consensus of multiagent systems and synchronization of complex networks: A unified viewpoint". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 57-I.1 (2010), pages 213–224. DOI: 10.1109/TCSI.2009.2023937 (cited on page 108).

[Li+11] Tao Li, Minyue Fu, Lihua Xie and Ji-Feng Zhang. "Distributed consensus with limited communication data rate". In: *IEEE Transactions on Automatic Control* 56.2 (2011), pages 279–292. DOI: 10.1109/TAC.2010.2052384 (cited on page 110).

[Li+16] Yongkai Li, Shubo Liu, Jun Wang and Mengjun Liu. "Collusion-tolerable and efficient privacy-preserving time-series data aggregation protocol". In: *International Journal of Distributed Sensor Networks* 12.9, 1341606 (2016). DOI: 10.1177/155014771341606 (cited on page 41).

[Lia+17] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang and Ji Liu. "Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent". In: *NIPS 2017: Advances in Neural Information Processing Systems*. Volume 30. 2017, pages 5330–5340. URL: https://proceedings.neurips.cc/paper/2017/hash/f75526659f31040afeb61cb7133e4e6d-Abstract.html (cited on page 68).

[Lin03] Yehuda Lindell. *Composition of secure multi-party protocols: A comprehensive study*. Volume 2815. Lecture Notes in Computer Science. 2003. DOI: 10.1007/b13246 (cited on page 71).

[Lin20] Yehuda Lindell. "Secure multiparty computation (MPC)". In: *IACR Cryptology ePrint Archive* (2020). URL: https://ia.cr/2020/300 (cited on page 4).

[Liu+14] Ying Liu, Cuili Yang, Wallace Kit-Sang Tang and Chunguang Li. "Optimal topological design for distributed estimation over sensor networks". In: *Information Sciences* 254 (2014), pages 83–97. DOI: 10.1016/J.INS.2013.07.012 (cited on pages 110–111).

[LJ16] Rutger Leukfeldt and Jurjen Jansen. "Cyber criminal networks and money mules: An analysis of low-tech and high-tech fraud attacks in the Netherlands". In: *International Journal of Cyber Criminology* 9.2 (2016), pages 173–184. DOI: 10.5281/zenodo.56210 (cited on page 92).

[LL21] Iraklis Leontiadis and Ming Li. "Secure and collusion-resistant data aggregation from convertible tags". In: *International Journal of Information Security* 20.1 (2021), pages 1–20. DOI: 10.1007/s10207-019-00485-4 (cited on pages 36–39, 52–53, 58, 61).

[LLW24] Yingxin Li, Fukang Liu and Gaoli Wang. "New records in collision attacks on SHA-2". In: *Advances in Cryptology – EUROCRYPT 2024: Proceedings of the 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Volume 14651. Lecture Notes in Computer Science. 2024, pages 158–186. DOI: 10.1007/978-3-031-58716-0_6 (cited on page 101).

[LM17] David Lie and Petros Maniatis. "Glimmers: Resolving the privacy/trust quagmire". In: *HotOS 2017: Proceedings of the 16th Workshop on Hot Topics in Operating Systems*. 2017, pages 94–99. DOI: 10.1145/3102980.3102996 (cited on page 16).

[LO14] JooYoung Lee and Jae C. Oh. "Estimating the degrees of neighboring nodes in online social networks". In: *PRIMA 2014: Proceedings of the 17th International Conference on Principles and Practice of Multi-Agent Systems*. Volume 8861. Lecture Notes in Computer Science. 2014, pages 42–56. DOI: 10.1007/978-3-319-13191-7_4 (cited on page 131).

[LS21] Nati Linial and Michael Simkin. "A randomized construction of high girth regular graphs". In: *Random Structures & Algorithms* 58.2 (2021), pages 345–369. DOI: 10.1002/RSA.20976 (cited on page 112).

[LTT11] Francis Chung-Ming Lau, Wai Man Tam and Chi Kong Tse. "Increasing the local girth of irregular low-density parity-check codes based on degree-spectrum analysis". In: *IET Communications* 5.11 (2011), pages 1506–1511. DOI: 10.1049/IET-COM.2010.0366 (cited on pages 108, 112).

[LU95] Felix Lazebnik and Vasiliy A. Ustimenko. "Explicit construction of graphs with an arbitrary large girth and of large size". In: *Discrete Applied Mathematics* 60.1-3 (1995), pages 275–284. DOI: 10.1016/0166-218X(94)00058-L (cited on page 82).

[LUW95] Felix Lazebnik, Vasiliy A. Ustimenko and Andrew J. Woldar. "A new series of dense graphs of high girth". In: *Bulletin of the American Mathematical Society* 32.1 (1995), pages 73–79. ISSN: 0273-0979,1088-9485. DOI: 10.1090/S0273-0979-1995-00569-0 (cited on page 112).

[LV16] Felipe Lillo and Rodrigo Valdés. "Dynamics of financial markets and transaction costs: A graph-based study". In: *Research in International Business and Finance* 38 (Sept. 2016), pages 455–465. ISSN: 0275-5319. DOI: 10.1016/j.ribaf.2016.07.024 (cited on page 101).

[LWY24] Fucai Luo, Haiyan Wang and Xingfu Yan. "Comments on 'VERSA: Verifiable secure aggregation for cross-device federated learning'". In: *IEEE Transactions on Dependable and Secure Computing* 21.1 (2024), pages 499–500. DOI: 10.1109/TDSC.2023.3253082 (cited on page 39).

[LX13] Xin-Ji Liu and Shu-Tao Xia. "Reconstruction guarantee analysis of binary measurement matrices based on girth". In: *ISIT 2013: Proceedings of the 2013 IEEE International Symposium on Information Theory*. 2013, pages 474–478. DOI: 10.1109/ISIT.2013.6620271 (cited on page 108).

[Mar23] Célio Porsius Martins. "Private cycle detection in financial transactions". English. Master's thesis. Delft University of Technology, Jan. 2023. URL: https://resolver.tudelft.nl/uuid:1ebcedc4-85cc-42e8-912e-2b3e8b9603cc (visited on 16th June 2025) (cited on pages 93–94).

[Mar82] G. A. Margulis. "Explicit constructions of graphs without short cycles and low density codes". In: *Combinatorica* 2.1 (1982), pages 71–78. DOI: 10.1007/BF02579283 (cited on page 108).

[Mau11] Ueli Maurer. "Constructive cryptography – A new paradigm for security definitions and proofs". In: *TOSCA 2011: Proceedings of the 2011 Joint Workshop on Theory of Security and Applications*. Volume 6993. Lecture Notes in Computer Science. 2011, pages 33–56. DOI: 10.1007/978-3-642-27375-9_3 (cited on pages 5, 71).

[McL+13] Stephen E. McLaughlin, Brett Holbert, Ahmed M. Fawaz, Robin Berthier and Saman A. Zonouz. "A multi-sensor energy theft detection framework for advanced metering infrastructures". In: *IEEE Journal on Selected Areas in Communications* 31.7 (2013), pages 1319–1330. DOI: 10.1109/JSAC.2013.130714 (cited on page 16).

[McM+17] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson and Blaise Agüera y Arcas. "Communication-efficient learning of deep networks from decentralized data". In: *AISTATS 2017: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. Volume 54. Proceedings of Machine Learning Research. 2017, pages 1273–1282. URL: https://proceedings.mlr.press/v54/mcmahan17a.html (cited on page 68).

[MEO13] Bradley A. Malin, Khaled El Emam and Christine M. O'Keefe. "Biomedical data privacy: Problems, perspectives, and recent advances". In: *Journal of the American Medical Informatics Association* 20.1 (2013), pages 2–6. DOI: 10.1136/amiajnl-2012-001509 (cited on page 36).

[Mer91] Russell Merris. "The number of eigenvalues greater than two in the Laplacian spectrum of a graph". In: *Portugaliae Mathematica* 48.3 (1991), pages 345–349. ISSN: 0032-5155,1662-2758 (cited on page 109).

[Mor+19] Eduardo Morais, Tommy Koens, Cees van Wijk and Aleksei Koren. *A survey on zero knowledge range proofs and applications*. 2019. arXiv: 1907.06381 (cited on pages 16, 18).

[Mou20] Foivi Mouzakiti. "Cooperation between financial intelligence units in the European Union: Stuck in the middle between the General Data Protection Regulation and the Police Data Protection Directive". In: *New Journal of European Criminal Law* 11.3 (2020), pages 351–374. DOI: 10.1177/2032284420943303 (cited on page 92).

[MT21] Dimitris Mouris and Nektarios Georgios Tsoutsos. "Masquerade: Verifiable multi-party aggregation with secure multiplicative commitments". In: *IACR Cryptology ePrint Archive* (2021). URL: https://ia.cr/2021/1370 (cited on pages 36, 38, 40, 52).

[MvOV96] Alfred Menezes, Paul C. van Oorschot and Scott A. Vanstone. *Handbook of applied cryptography*. 1996. DOI: 10.1201/9781439821916 (cited on page 100).

[New21] Gemma Newlands. "Algorithmic surveillance in the gig economy: The organization of work through Lefebvrian conceived space". In: *Organization Studies* 42.5 (2021), pages 719–737. DOI: 10.1177/0170840620937900 (cited on page 3).

[Ni+15] Jianbing Ni, Khalid Nawaf Alharbi, Xiaodong Lin and Xuemin Shen. "Security-enhanced data aggregation against malicious gateways in smart grid". In: *GLOBECOM 2015: Proceedings of the 2015 IEEE Global Communications Conference*. 2015. DOI: 10.1109/GLOCOM.2014.7417140 (cited on pages 36–39).

[NSTC23] Fast-Track Action Committee on Advancing Privacy-Preserving Data Sharing and Analytics and Networking and Information Technology Research and Development Subcommittee. *National strategy to advance privacy-preserving data sharing and analytics*. Mar. 2023. URL: https://bidenwhitehouse.archives.gov/wp-content/uploads/2023/03/National-Strategy-to-Advance-Privacy-Preserving-Data-Sharing-and-Analytics.pdf (visited on 16th June 2025) (cited on page 4).

[Oka92] Tatsuaki Okamoto. "Provably secure and practical identification schemes and corresponding signature schemes". In: *Advances in Cryptology – CRYPTO 1992: Proceedings of the 12th Annual International Cryptology Conference*. Volume 740. Lecture Notes in Computer Science. 1992, pages 31–53. DOI: 10.1007/3-540-48071-4_3 (cited on page 41).

[Oli+18] Gabriele Oliva, Roberto Setola, Luigi Glielmo and Christoforos N. Hadjicostis. "Distributed cycle detection and removal". In: *IEEE Transactions on Control of Network Systems* 5.1 (2018), pages 194–204. DOI: 10.1109/TCNS.2016.2593264 (cited on pages 82, 89).

[OM03] Reza Olfati-Saber and Richard M. Murray. "Consensus protocols for networks of dynamic agents". In: *ACC 2003: Proceedings of the 2003 American Control Conference*. 2003, pages 951–956. DOI: 10.1109/ACC.2003.1239709 (cited on page 109).

[Pal+24] Marco Palazzo, Florine W. Dekker, Alessandro Brighente, Mauro Conti and Zekeriya Erkin. "Privacy-preserving data aggregation with public verifiability against internal adversaries". In: *USENIX Security 2024: Proceedings of the 33rd USENIX Security Symposium*. 2024. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/palazzo (cited on pages 7, 9, 35).

[Pan+20] Guansong Pang, Chunhua Shen, Longbing Cao and Anton van den Hengel. *Deep learning for anomaly detection: A review*. 2020. arXiv: 2007.02500 (cited on page 93).

[Par21] Pedro Paredes. "Spectrum preserving short cycle removal on regular graphs". In: *STACS 2021: Proceedings of the 38th International Symposium on Theoretical Aspects of Computer Science*. Volume 187. Leibniz International Proceedings in Informatics. 2021, 55. DOI: 10.4230/LIPICS.STACS.2021.55 (cited on pages 108, 112).

[PC98] Louis M. Pecora and Thomas L. Carroll. "Master stability functions for synchronized coupled systems". In: *Physical Review Letters* 80 (10 Mar. 1998), pages 2109–2112. DOI: 10.1103/PhysRevLett.80.2109 (cited on pages 108, 110).

[PD25] Marco Palazzo and Florine W. Dekker. *Source code underlying the publication: Privacy-preserving data aggregation with public verifiability against internal adversaries*. 11th June 2025. DOI: 10.4121/56552cc8-7ebf-46ce-a6e0-668dd6065eb2.v1 (cited on pages 10, 53).

[Pen97] Mathew D. Penrose. "The longest edge of the random minimal spanning tree". In: *The Annals of Applied Probability* 7.2 (1997), pages 340–361. ISSN: 1050-5164,2168-8737. DOI: 10.1214/aoap/1034625335 (cited on page 114).

[Pou+20] Tahereh Pourhabibi, Kok-Leong Ong, Booi Kam and Yee Ling Boo. "Fraud detection: A systematic literature review of graph-based anomaly detection approaches". In: *Decision Support Systems* 133, 113303 (2020). DOI: 10.1016/J.DSS.2020.113303 (cited on page 93).

[Qiu+18] Xiafei Qiu, Wubin Cen, Zhengping Qian, You Peng, Ying Zhang, Xuemin Lin and Jingren Zhou. "Real-time constrained cycle detection in large dynamic graphs". In: *Proceedings of the VLDB Endowment* 11.12 (2018), pages 1876–1888. DOI: 10.14778/3229863.3229874 (cited on pages 92, 100–101).

[Qu+20] Youyang Qu, Longxiang Gao, Tom H. Luan, Yong Xiang, Shui Yu, Bai Li and Gavin Zheng. "Decentralized privacy using blockchain-enabled federated learning in fog computing". In: *IEEE Internet of Things Journal* 7.6 (2020), pages 5171–5183. DOI: 10.1109/JIOT.2020.2977383 (cited on page 68).

[Rai18] Lee Rainie. "Americans' complicated feelings about social media in an era of privacy concerns". In: *Pew Research Center* (27th Mar. 2018). URL: https://www.pewresearch.org/short-reads/2018/03/27/americans-complicated-feelings-about-social-media-in-an-era-of-privacy-concerns/ (visited on 16th June 2025) (cited on page 3).

[Ren+22] Yanli Ren, Yerong Li, Guorui Feng and Xinpeng Zhang. "Privacy-enhanced and verification-traceable aggregation for federated learning". In: *IEEE Internet of Things Journal* 9.24 (2022), pages 24933–24948. DOI: 10.1109/JIOT.2022.3194930 (cited on pages 36–38, 40, 52).

[Rie+11] Konrad Rieck, Philipp Trinius, Carsten Willems and Thorsten Holz. "Automatic analysis of malware behavior using machine learning". In: *Journal of Computer Security* 19.4 (2011), pages 639–668. DOI: 10.3233/JCS-2010-0410 (cited on page 68).

[RJH08] Ali Ajdari Rad, Mahdi Jalili and Martin Hasler. "Efficient rewirings for enhancing synchronizability of dynamical networks". In: *Chaos* 18.3, 037104 (2008). ISSN: 1054-1500,1089-7682. DOI: 10.1063/1.2967738 (cited on page 111).

[RN10] Vibhor Rastogi and Suman Nath. "Differentially private aggregation of distributed time-series with transformation and encryption". In: *SIGMOD 2010: Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. 2010, pages 735–746. DOI: 10.1145/1807167.1807247 (cited on page 16).

[RW13] Yannis Rouselakis and Brent Waters. “Practical constructions and new proof methods for large universe attribute-based encryption”. In: *CCS 2013: Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security*. 2013, pages 463–474. DOI: 10.1145/2508859.2516672 (cited on page 53).

[SB22] Christel Sirocchi and Alessandro Bogliolo. “Topological network features determine convergence rate of distributed average algorithms”. In: *Scientific Reports* 12.1, 21831 (Dec. 2022). ISSN: 2045-2322. DOI: 10.1038/s41598-022-25974-w (cited on pages 108, 111, 114, 116).

[Sch+20] Robert Schmid, Bjarne Pfitzner, Jossekin Beilharz, Bert Arnrich and Andreas Polze. “Tangle ledger for decentralized learning”. In: *IPDPSW 2020: Proceedings of the 2020 IEEE International Parallel and Distributed Processing Symposium Workshops*. 2020, pages 852–859. DOI: 10.1109/IPDPSW50202.2020.00144 (cited on page 68).

[Sch25] Berry Schoenmakers. *Lecture notes cryptographic protocols, version 1.10*. Feb. 2025. URL: https://www.win.tue.nl/~berry/CryptographicProtocols/LectureNotes.pdf (visited on 16th June 2025) (cited on page 41).

[Shi+11] Elaine Shi, T.-H. Hubert Chan, Eleanor Gilbert Rieffel, Richard Chow and Dawn Song. “Privacy-preserving aggregation of time-series data”. In: *NDSS 2011: Proceedings of the 2011 Network and Distributed System Security Symposium*. 2011. URL: https://www.ndss-symposium.org/ndss2011/privacy-preserving-aggregation-of-time-series-data (cited on pages 16, 18, 36, 39–40, 58).

[Shi+15] 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”. In: *IEEE Transactions on Smart Grid* 6.6 (2015), pages 2856–2868. DOI: 10.1109/TSG.2015.2443011 (cited on pages 17, 28).

[Shi09] Katie Shilton. “Four billion little brothers?: Privacy, mobile phones, and ubiquitous data collection”. In: *Communications of the ACM* 52.11 (2009), pages 48–53. DOI: 10.1145/1592761.1592778 (cited on page 36).

[Sma16] Nigel P. Smart. *Cryptography made simple*. Information Security and Cryptography. 2016. DOI: 10.1007/978-3-319-21936-3 (cited on page 37).

[SRA79] Adi Shamir, Ronald L. Rivest and Leonard M. Adleman. *Mental poker*. Massachusetts Institute of Technology, Laboratory for Computer Science, 29th Jan. 1979. DOI: 1721.1/148953 (cited on page 4).

[ST21] Durgesh Samariya and Amit Thakkar. “A comprehensive survey of anomaly detection algorithms”. In: *Annals of Data Science* 10 (2021), pages 829–850. DOI: 10.1007/s40745-021-00362-9 (cited on page 93).

[STPO22] Office of Science and Technology Policy. *Request for information on advancing privacy-enhancing technologies*. 9th June 2022. URL: https://www.federalregister.gov/documents/2022/06/09/2022-12432/request-for-information-on-advancing-privacy-enhancing-technologies (visited on 16th June 2025) (cited on page 4).

[Sun+13] Ruixue Sun, Zhiguo Shi, Rongxing Lu, Min Lu and Xuemin Shen. "APED: An efficient aggregation protocol with error detection for smart grid communications". In: *GLOBECOM 2013: Proceedings of the 2013 IEEE Global Communications Conference*. 2013, pages 432–437. DOI: 10.1109/GLOCOM.2013.6831109 (cited on pages 17, 28).

[Szy95] Ted H. Szymanski. "'Hypermeshes': Optical intercomnnection network for parallel computing". In: *Journal of Parallel and Distributed Computing* 26.1 (1995), pages 1–23. DOI: 10.1006/JPDC.1995.1043 (cited on page 20).

[Tan+18] Hanlin Tang, Xiangru Lian, Ming Yan, Ce Zhang and Ji Liu. "D^2: Decentralized training over decentralized data". In: *ICML 2018: Proceedings of the 35th International Conference on Machine Learning*. Volume 80. Proceedings of Machine Learning Research. 2018, pages 4855–4863. URL: https://proceedings.mlr.press/v80/tang18a.html (cited on page 68).

[TDP16] Ngoc Hieu Tran, Robert H. Deng and HweeHwa Pang. "Privacy-preserving and verifiable data aggregation". In: *SG-CRC 2016: Proceedings of the Singapore Cyber-Security Conference*. Volume 14. Cryptology and Information Security Series. 2016, pages 115–122. DOI: 10.3233/978-1-61499-617-0-115 (cited on pages 58, 61).

[TL24] Carmela Troncoso and Wouter Lueks. "Designing for data protection". In: *Handbook on Data Protection in Humanitarian Action*. 2024. Chapter 6, pages 76–95. DOI: 10.1017/9781009414630.011 (cited on page 3).

[Tra+21] Anh-Tu Tran, The-Dung Luong, Jessada Karnjana and Van-Nam Huynh. "An efficient approach for privacy preserving decentralized deep learning models based on secure multi-party computation". In: *Neurocomputing* 422 (2021), pages 245–262. DOI: 10.1016/j.neucom.2020.10.014 (cited on page 68).

[Tro+17] Carmela Troncoso, Marios Isaakidis, George Danezis and Harry Halpin. "Systematizing decentralization and privacy: Lessons from 15 years of research and deployments". In: *Proceedings on Privacy Enhancing Technologies* 2017.4 (2017), pages 404–426. DOI: 10.1515/popets-2017-0056 (cited on page 68).

[Ull+21] Ata Ullah, Muhammad Azeem, Humaira Ashraf, Abdulellah A. Alaboudi, Mamoona Humayun and N. Z. Jhanjhi. "Secure healthcare data aggregation and transmission in IoT – A survey". In: *IEEE Access* 9 (2021), pages 16849–16865. DOI: 10.1109/ACCESS.2021.3052850 (cited on page 36).

[UQ25] Aysun Urhan and Yasel Quintero. *CODECHECK certificate 2025-003*. May 2025. DOI: 10.5281/zenodo.15333601 (cited on page 10).

[VBT17] Paul Vanhaesebrouck, Aurélien Bellet and Marc Tommasi. "Decentralized collaborative learning of personalized models over networks". In: *AISTATS 2017: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. Volume 54. Proceedings of Machine Learning Research. 2017, pages 509–517. URL: https://proceedings.mlr.press/v54/vanhaesebrouck17a.html (cited on pages 68, 71–72, 82, 88, 108).

[vEgm+21] Marie Beth van Egmond, Gabriele Spini, Onno van der Galiën, Arne IJpma, Thijs Veugen, Wessel Kraaij, Alex Sangers, Thomas Rooijakkers, Peter Langenkamp, Bart Kamphorst, Natasja van de L'Isle and Milena Kooij-Janic. "Privacy-preserving dataset combination and Lasso regression for healthcare predictions". In: *BMC Medical Informatics and Decision Making* 21.1, 266 (2021). DOI: 10.1186/S12911-021-01582-Y (cited on page 4).

[Vor23] Marc H.L. Vorstermans. "Secure graph algorithms and oblivious data structures for multiparty computation". English. Master's thesis. Eindhoven University of Technology, Mar. 2023. URL: https://research.tue.nl/en/studentTheses/d171ddd4-aba0-4063-96dd-46f396de2369 (cited on page 94).

[Wan+07] Bing Wang, Tao Zhou, Zhilong Xiu and Beom Jun Kim. "Optimal synchronizability of networks". In: *The European Physical Journal B* 60.1 (Nov. 2007), pages 89–95. ISSN: 1434-6036. DOI: 10.1140/epjb/e2007-00324-y (cited on page 111).

[Wan+19] Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang and Hairong Qi. "Beyond inferring class representatives: User-level privacy leakage from federated learning". In: *INFOCOM 2019: Proceedings of the 2019 IEEE Conference on Computer Communications*. 2019, pages 2512–2520. DOI: 10.1109/INFOCOM.2019.8737416 (cited on page 71).

[Wan+23] Yong Wang, Aiqing Zhang, Shu Wu and Shui Yu. "VOSA: Verifiable and oblivious secure aggregation for privacy-preserving federated learning". In: *IEEE Transactions on Dependable and Secure Computing* 20.5 (2023), pages 3601–3616. DOI: 10.1109/TDSC.2022.3226508 (cited on pages 36–39).

[War65] Stanley L. Warner. "Randomized response: A survey technique for eliminating evasive answer bias". In: *Journal of the American Statistical Association* 60.309 (1965), pages 63–69. DOI: 10.1080/01621459.1965.10480775 (cited on page 71).

[Wei+16] Gary M. Weiss, Jessica L. Timko, Catherine M. Gallagher, Kenichi Yoneda and Andrew J. Schreiber. "Smartwatch-based activity recognition: A machine learning approach". In: *BHI 2016: Proceedings of the 2016 IEEE-EMBS International Conference on Biomedical and Health Informatics*. 2016, pages 426–429. DOI: 10.1109/BHI.2016.7455925 (cited on page 68).

[Wor+20] Daniël Worm, Bart Kamphorst, Thomas Rooijakkers, Thijs Veugen, Matteo Cellamare, Gijs Geleijnse, Daan Knoors and Frank Martin. *CONVINCED – Enabling privacy-preserving survival analyses using multi-party computation*. 2020. URL: https://resolver.tno.nl/uuid:1c4885d6-8cf3-4443-b952-e887e1b41207 (visited on 16th June 2025) (cited on page 4).

[WWJ02] Lingyu Wang, Duminda Wijesekera and Sushil Jajodia. "Cardinality-based inference control in sum-only data cubes". In: *ESORICS 2002: Proceedings of the 7th European Symposium on Research in Computer Security*. Volume 2502. Lecture Notes in Computer Science. 2002, pages 55–71. DOI: 10.1007/3-540-45853-0_4 (cited on pages 5, 70, 78).

[XB04] Lin Xiao and Stephen P. Boyd. "Fast linear iterations for distributed averaging". In: *Systems & Control Letters* 53.1 (2004), pages 65–78. DOI: 10.1016/J.SYSCONLE.2004.02.022 (cited on pages 87, 108–109, 113).

[Yan+10] Bin Yang, Hiroshi Nakagawa, Issei Sato and Jun Sakuma. "Collusion-resistant privacy-preserving data mining". In: *KDD 2010: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2010, pages 483–492. DOI: 10.1145/1835804.1835867 (cited on page 71).

[Yeu+12] Ka Wai Yeung, Cuili Yang, Wallace Kit-Sang Tang and Ying Liu. "A meta-heuristic algorithm for enhancing the synchronizability of complex networks". In: *ISIE 2012: Proceedings of the 21st IEEE International Symposium on Industrial Electronics*. 2012, pages 792–796. DOI: 10.1109/ISIE.2012.6237189 (cited on page 111).

[YL13] Lei Yang and Fengjun Li. "Detecting false data injection in smart grid in-network aggregation". In: *SmartGridComm 2013: Proceedings of the 4th IEEE International Conference on Smart Grid Communications*. 2013, pages 408–413. DOI: 10.1109/SMARTGRIDCOMM.2013.6687992 (cited on pages 16, 18, 28–29).

[YT11] Cuili Yang and Wallace Kit-Sang Tang. "Enhancing the synchronizability of networks by rewiring based on tabu search and a local greedy algorithm". In: *Chinese Physics B* 20.12, 128901 (Dec. 2011). DOI: 10.1088/1674-1056/20/12/128901 (cited on page 111).

[ZBT20] Valentina Zantedeschi, Aurélien Bellet and Marc Tommasi. "Fully decentralized joint learning of personalized models and collaboration graphs". In: *AISTATS 2020: Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*. Volume 108. Proceedings of Machine Learning Research. 2020, pages 864–874. URL: https://proceedings.mlr.press/v108/zantedeschi20a.html (cited on pages 68, 71–72).

[Zia23] Hanna Ziady. "Meta slapped with record $1.3 billion EU fine over data privacy". In: *CNN* (22nd May 2023). URL: https://edition.cnn.com/2023/05/22/tech/meta-facebook-data-privacy-eu-fine/ (visited on 16th June 2025) (cited on page 3).

[ZMW17] Kai Zheng, Wenlong Mou and Liwei Wang. “Collect at once, use effectively: Making non-interactive locally private learning possible”. In: *ICML 2017: Proceedings of the 34th International Conference on Machine Learning*. Volume 70. Proceedings of Machine Learning Research. 2017, pages 4130–4139. URL: https://proceedings.mlr.press/v70/zheng17c.html (cited on pages 69, 71).

Acknowledgements

An endless enumeration of thank yous can be a bit dry at times, but I expect most readers won't do much more than search for their names before moving on. Personally, I've always enjoyed reading the acknowledgements in others' dissertations. They provide a wholesome glimpse into the author's life, and sometimes reveal that you may not have known your colleague as well as you thought. Having said that, here follows a dry enumeration of heartfelt acknowledgements. I'm not a woman of many words, but that's just because I choose my words very carefully.

I will start with my two promotors. Thank you, **Zeki**, for providing me with the space to work in the way that best suit me. Thank you, **Mauro**, for nudging the steering wheel whenever it was required.

I want to thank the **defence committee** for taking the time to read this entire dissertation, for giving valuable feedback, and for traveling to Delft to wear silly hats and ask tough questions.

I've seen office mates come and go, but loved them all. The evenings spent together were the favourite days of my PhD, and never failed to invigorate me. Thank you, **Jelle** for your selflessness and thoughtfulness. Thank you, **Jesús** for sharing your delicious food. Thank you, **Jorrit** for your straightforwardness and assertiveness. Thank you, **Miray**, for your comical incredulity. Thank you, **Ozzy**, for showing me the ropes. Thank you, **Tianyu** for your determination and for lending me your FIFA skills. Thank you, **Tjitske** for your humour and positivity.

I also want to thank my many colleagues, who always created an incredibly lively atmosphere on the work floor and made me feel welcome. I know that's a platitude, but I'm genuine! Thank you, **Alexios**, for your unabashedly loud laughter. Thank you, **Giovane**, for your honest words. Thank you, **Georgios**, for your unreserved generosity. Thank you, **Huimin**, for the nice conversations in Darmstadt. Thank you, **Lilika**, for hearing out my outlandish talk. Thank you, **Sandra**, for being the glue that holds this group together. And I also want to thank you, **Alin**, **Clinton**, **Daniël**, **Dario**[1], **Dazhuang**, **Harm**, **Huanhuan**, **Jing**, **Kaitai**, **Lichao**, **Maarten**, **Marina**, **Martin**[2], **Murtuza**, **Roland**, **Stefanos**, **Stjepan**, **Yu**, and **Yuqian**, for being such lovely people.

I also want to thank **Bart** and **Ruud** for their technical assistance.

My dissertation would (literally) not have been the same without my collaborators. I have learnt a lot from working with each of you. Thank you, **Alessandro**, for making sure the MPVAS paper was in order and tying up some of the loose ends. Thank you, **Juno**, for your infinite stream of incredible stories. Thank you, **Marco**, for your humble genius. And, while our project didn't bear fruit, I want to thank you, **Ankit**, **Dario**[1], **Giovanni**, and **Giuseppe**, for working with me on the GBoard project.

There are also several people who have fulfilled somewhat of a mentorship role, however briefly. Each of you has inspired me, professionally or personally, and are part

[1] These are different Darios.

[2] These are different Martins.

of the reason that I am where I am, and am heading where I am heading. Without you, I might have given up on my PhD long ago. Thank you, **Dion**, for your welcoming smile, for your genuine interest, and for pointing me in the right direction when I was lost in algebra. Thank you, **Seda**, for teaching me so much about privacy, about the real world, and for your many kind and encouraging words. Thank you, **VP**, for helping me collect myself at the end of the third year, and for making me realise how tremendous my progress had actually already been. And thank you, **Wouter**, for asking the right questions when I was in despair.

Thank you, **Amber**, for the motivation and inspiration, which may be more mutual than you think. Thank you, **Jeff**, for providing me with the code of your beautiful dissertation (and of course also for many a good conversation). I also want to thank the **many people**, whose names are too many to list, but have asked great questions during my presentations (or vice versa), inspired research questions, or were just jolly good to talk with at conferences and dinners.

There are also many students I want to think. Firstly, the students whom I supervised. Thank you, **Andrei**, **Célio**, **Dāvis**, **Juno**, **Marco**, and **Martin**[2], for teaching me about teaching, for your patience, and for your gifts. Secondly, I want to thank the many **other master students** that I did not supervise, but have given feedback to, lunched with, and always had many stories to tell. Finally, I want to thank the **students** who attended my lectures or had to suffer through one of my assignments.

I want to thank the anonymous **reviewers** of my scientific works. Those who provided *constructive* feedback, anyway.

I also want to thank the many hardware devices that have supported me in my endeavours. First of all, thank you, **Lauren**, my laptop, for suffering through so much computing hours that I had to put you upside down on my desk to stop you from overheating. Thank you, **Helmut**, my now-retired desktop PC, for loyally running experiments for many years. Thank you, **Jonna**, my current desktop PC, for your reliability, and for the fun playing VR games. Thank you, **Susumu**, my old phone, and **Joel**, my current phone, for keeping me sane between long working hours. Thank you, **Theodore**, for heating our office in the winter and for running some last-minute experiments. Finally, thank you, everyone at **DAIC** (formerly INSY-HPC), for running the high-performance computing servers that enabled me to scale up my experiments several orders of magnitude.

All work and no play makes Florine a dull girl. Therefore, I want to thank my awesome lifelong friends. The time spent together playing video games, going on holidays, spilling butter on stairs, and watching movies and shows, I could not have done without. I could fill pages praising you as friends, but words do not do justice to how much I love each of you. Thank you, **Christa**, for radiating your deeply chaotic pevil energies. Thank you, **Daryl**, for your perpetual calmness. Thank you, **Luc**, for the mystery brews. Thank you, **Romano**, for your boundless enthusiasm. Thank you, **Tim**, for going on adventures with me. Beyond that, I want to thank my many other **friends** who've frequently cheered me up.

Thank you, **papa en mama**, for your boundless love and for always being there to fall back on. When I don't know what to do, I ask myself what *you* would do in my place, and thus far the results have been excellent. Thank you for inviting yourself over for coffee: Talking with you always helps me see the forest despite the trees. And

I thank **my family** for accepting me for who I am; that means a lot to me.

As the very last but *absolutely not* the very least, I want to thank **Marija**. I forget we've only known each other a few years, because it feels like we've known each other a lifetime. Your unconditional love and acceptance of me has shown me how to love myself, too. Thank you for your help when I was working late, for your understanding when I was frustrated, for your comfort when I was demotivated, and for your love always. When I fear for the future, I think of you and feel at rest again.

Florine W. Dekker
16th June 2025
Delft, the Netherlands

Curriculum Vitae

Florine W. Dekker

1997	Born June 17th in Zwijndrecht, the Netherlands
2009–2015	Gymnasium diploma *Johan de Witt-gymnasium* in Dordrecht
2015–2018	BSc in Computer Science & Engineering *Delft University of Technology* in Delft Minor Computer Science at *University of Waterloo*, Canada
2018–2020	MSc Computer Science *Delft University of Technology* in Delft Specialising in Cybersecurity
2020–2025	PhD Computer Science *Delft University of Technology* in Delft Cybersecurity research group