

# Bridging the gap between theoretical and practical privacy technologies for at-risk populations

Présentée le 27 octobre 2023

Faculté informatique et communications  
Laboratoire d'ingénierie de sécurité et privacy  
Programme doctoral en informatique et communications

pour l'obtention du grade de Docteur ès Sciences

par

## **Kasra EDALATNEJADKHAMENE**

Acceptée sur proposition du jury

Prof. J. R. Larus, président du jury  
Prof. C. González Troncoso, directrice de thèse  
N. Sullivan, rapporteur  
Prof. S. Meiklejohn, rapporteuse  
Prof. B. Ford, rapporteur

Thesis jury members:

Prof. Dr. James Larus, EPFL, jury president

Prof. Dr. Carmela Troncoso, EPFL, advisor

Prof. Dr. Bryan Ford, EPFL

Prof. Dr. Sarah Meiklejohn, University College London

Nick Sullivan, Cloudflare

“Saruman believes it is only great power that can hold evil in check, but that is not what I have found. I found it is the small everyday deeds of ordinary folk that keep the darkness at bay. Small acts of kindness and love.”  
— Gandalf the Grey



## Acknowledgements

Reflecting on my 6-year Ph.D. journey at EPFL, I've evolved immensely. The one constant has been my incredible luck, which led me to start this journey and introduced me to remarkable people and experiences.

I am deeply grateful to my advisor, Prof. Carmela Troncoso. For being a great mentor, inspiration, and a support in my life. For being family. Not only Carmela is a brilliant researcher, but also a great mentor who thought us how to become better researchers, writers, and better human beings. Carmela gave us the freedom to chase our passion, yet she provided us with her unwavering support in achieving our goals. Carmela sets a near-impossible bar for achievements and how to act as a teacher, mentor, and researcher, and watching her inspires us to aim for the same impossible goal, yet she is one of the most wonderful and fun people to know. Joining the Spring lab at its inception and watching it flourish has been rewarding. The moments of brainstorming, deadline frenzies (and always breaking the system in the deadline week), TA preparation meetings, and laughing at funny answers in our 'grading parties' are all etched in my mind. I have been the luckiest EPFL student to have you as my advisor; words fall short of describing how you impacted my life and how grateful I am.

Equally integral to my journey has been Wouter Lueks. You have been my mentor, my friend, and a reliable source of support and advice in my life. You have thought me a lot from how to research or do 'crypto-magic', to soft skills like how to write and present better and care about every aspect of a work to the minor typographic details, to valuable life lessons. Despite the notorious pressures of deadline crunches, having both you and Carmela by my side made them unexpectedly enjoyable. Wouter, you have epitomized the ideal of a 'perfect gentleman'; an image that I have strived for. I cannot imagine my journey without you, or tell you how grateful I am for having you there. But I can tell you that my mantra for facing difficult or delicate situations is checking WWW, asking myself "What Would Wouter do?"

I am grateful to my thesis committee: Prof. Sarah Meiklejohn, Nick Sullivan, Prof. Bryan Ford, and jury president Prof. James Larus. Thank you for reviewing my extensive thesis, making the journey to Lausanne, engaging

deeply with my work, and offering invaluable advice. Like every other Ph.D. student, I dreaded my defense; till I realized that I rather be in a meeting room with you, even in a defensive manner, than enjoy a calming tea outside. Thanks for this elusive peace of mind.

During the initial semester of my Ph.D., I received guidance from Prof. Bryan Ford. Bryan, your energy and unwavering enthusiasm though me that the joy of research can be everlasting. Later, I had the chance to visit IMDEA software. Prof. Dario Fiore and Dr. Claudio Soriente, thank you for your kind reception and belief in my potential. To Nicolas Mazzocchi, Lúbia Jančová, Lydia Garms, Peter Chvojka, and Emanuele Giunta, your friendship and board game nights made Madrid feel like home. Thank you.

Laurent Girod, our engineer, my office mate, and my dear friend. Your technical support and help in transforming our prototypes into usable products cannot be overstated. Thanks for being as reliable as clockwork and for your sense of programming humor, but most importantly for being a great friend. My academic journey began alongside Sandra Siby, one of the kindest souls I know. Your presence always provided a comforting sense of home. Bharath Narayanan, your unadulterated sense of humor has always brightened my day. Early in my Ph.D., I supervised a young bright student, who later became my collaborator, labmate, and more importantly my friend. Mathilde Raynal, I am glad that you stayed with us and I got to know you better. Our discussions and shared meals have been enlightening. Klim Kireev, our lunches and unique discussions will be dearly missed. I am grateful to find a friend who shares the same sense of dark humor as me. While older students typically guide the newer ones, Boya Wangs, when you joined our lab in my 5th year, you redefined my approach to life. You taught me it is cool to make healthier decisions and to live freely. For that, I am grateful.

I was lucky to find many friends and amazing colleagues in the Spring lab and our sister labs. Theresa Stadler, your candidness taught me much. Bogdan Kulynych, you have shown us all the essence of sociability. Sinem Sav, Sylvain Chatel, and Christian Mouchet, thanks for the stimulating morning coffee chats. Christian and Bogdan, defending our thesis together on the same day has been an honor and the dinner after the defence will always have a special place in my memory. Linus Gasser, your early advice at EPFL was invaluable, and your consistent positivity was comforting. I offer my sincere gratitude to other colleagues who helped me, gave me advice, and shared triumphs and rejections, especially to Rebekah Overdorf, Ceyhun Alp, Apostolos Pyrgelis, Giovanni Cherubin, Dario Pasquini, Kirill Nikitin, Eleftherios Kokoris Kogias,

Cristina Băescu, Georgia Fragkouli, Jeff Allen, and Pasindu Tennage.

Navigating the bureaucracy and admin work of a Ph.D. was daunting for me. Isabelle Coke, I am grateful for your cheerful assistance, making these processes more manageable and less intimidating with your reassuring smile.

I am profoundly grateful to the International Consortium of Investigative Journalists for welcoming me into the DatashareNetwork project and entrusting us with their crucial mission and journalist safety. A heartfelt thanks to the ICIJ's Research and Data team: Soline Ledésert, Anne L'Hôte, Bruno Thomas, and Pierre Romera. Over the past 5 years, you provided invaluable insights into investigative journalism, guiding system usability, and your enthusiasm actualized our design. Soline, I have always been amazed at how you transform our system into something beautiful, intuitive, and usable. Bruno, our engineering discussions, sometimes intense, always pushing for the best system, though me a lot. My heartfelt thanks to the Data Protection Office of the International Committee of the Red Cross especially to Justinas Sukaitis, Vincent Graf Narbel, and Massimo Marelli for giving us the opportunity to work on systems aiming for a brighter future and for teaching me about humanitarian principles. Additionally, I wish to thank Martin Strohmeier and Vincent Lender from the Armasuisse for all the fun times working on aviation ciphers and their never-ending patience; and a special thanks to Martin for helping me navigate the chaotic world of aircraft communication.

I owe a debt of gratitude to my friends in Lausanne, notably Matteo Monti, Jakab Tardos, and Paritosh Garg. Our shared moments, from dinner parties, hikes, gardening, to movie nights, have been invaluable. I fondly recall our attempt at gardening and plowing the ground with 12 Ph.D. students; unsurprisingly, our academic expertise did not translate to poor seeds' survival. Your presence has always uplifted my spirit. My experience in Lausanne would have been incomplete without you. Additionally, I am grateful to my close friend, Pouya Esmaili Dokht. You have always been like a brother to me, and my Ph.D. life would not have been this enjoyable without you.

During my Ph.D. I had the chance to supervise many amazing students: Valentyna, Bradley, Sacha, Jodok, Lorenzo, Eva, and Pierugo. Thanks for your tireless work, for enriching my Ph.D. life, and for all the fun idea sessions that we had together.

Last but not least, I want to thank my family. For always believing in me, for pushing me to achieve my potential, for always being there for me, for always picking me up after I fall even from afar, and for unconditionally loving

## Acknowledgements

---

and supporting me. This journey would have never begun if you were not filling my mind with dreams and the belief that I can achieve them before even being able to walk or talk. I always imagined myself as someone independent who happily travels the world in pursuit of knowledge, till I came here and realized that I would have not even survived the first month without you being there for me and answering my calls no matter when.

With inspiration from J. R. R. Tolkien: “Well, here at last, dear friends, on the shores of the Sea comes the end of our fellowship. Go in peace! I will not say: I shall not weep; for not all tears are an evil.”

*Lausanne, August, 2023*

Kasra EdalatNejad



## Abstract

With the pervasive digitalization of modern life, we benefit from efficient access to information and services. Yet, this digitalization poses severe privacy challenges, especially for special-needs individuals. Beyond being a fundamental human right, privacy is crucial for roles sensitive in nature, including investigative journalists exposing corruption and humanitarian organizations supporting refugees or survivors of violence. This thesis leverages privacy-enhancing technologies to mitigate the risks of digitalization while retaining its advantages.

Recent breakthroughs in cryptography, such as fully homomorphic encryption and secure multiparty computation, provide robust tools for privacy. However, there is still no silver bullet solution that can achieve efficient privacy out of the box. We observe that there often is a gap between theoretical cryptographic solutions and real-world problems. Identifying and bridging these gaps enables us to design pragmatic privacy-enhancing technologies tailored for real-world deployment. In this thesis, we identify and solve four real-world problems.

We first present the problem of searching sensitive documents among a network of investigative journalists. In collaboration with the International Consortium of Investigative Journalists, we design a decentralized peer-to-peer privacy-preserving search engine called DatashareNetwork. Our solution enables journalists to find colleagues who have relevant documents for their topic of investigation and anonymously discuss the possibility of collaboration. We develop a prototype of DatashareNetwork and demonstrate that it scales to thousands of journalists and millions of documents.

We introduce a new class of problems called private collection matching in which a client aims to determine whether a collection of sets owned by a server matches their interests such as searching confidential chemical compound databases. We design a framework based on fully homomorphic encryption to solve these problems. Our solution, takes the data minimization principle to the maximum and shows the possibility of satisfying clients' needs by only revealing a single bit. We evaluate our framework and show that it significantly improves the latency, client computation cost, and communication

cost with respect to generic solutions that offer the same privacy guarantee.

We examine the problem of preventing double registration in humanitarian aid distribution with a focus on the needs of the International Committee of Red Cross. In response, we design Janus, a privacy-preserving biometric deduplication system that is compatible with fingerprints, irises, and face recognition; and supports both biometric alignment and fusion. We design and develop three instantiations of Janus based on secure multiparty computation, somewhat homomorphic encryption, and trusted execution environments. We evaluate Janus to show it satisfies the privacy, accuracy, and performance needs of humanitarian organizations.

Finally, we study the problem of detecting insecure ciphers in aircraft communication at scale. We design and develop a decision support system that helps human analysts to detect new ciphertexts in aircraft communication. We evaluate our system by applying it to real-world data and asking our analyst to use our support system to find new ciphers. Our analysis led to uncovering of 9 previously unknown (and potentially insecure) ciphers which we disclose to various stakeholders.

**Keywords:** Privacy enhancing technologies, applied cryptography, privacy engineering, homomorphic encryption, private set intersection, private computation.

## Résumé

La numérisation croissante de notre vie nous apporte de nombreux avantages, tels que la possibilité de rechercher instantanément de grandes quantités de documents. Dans le même temps, cette numérisation suscite des inquiétudes et des risques pour la vie privée, en particulier pour les utilisateurs ayant des besoins particuliers. Non seulement la vie privée est un droit humain fondamental, mais elle est également nécessaire à l'exercice de certaines professions sensibles, tels que le journalisme d'investigation où des journalistes travaillent sur des affaires de corruption ou les organisations humanitaires où des collaborateurs aidant les réfugiés ou les survivants d'incidents violents. Cette thèse s'appuie sur les technologies de renforcement de la vie privée pour atténuer les inconvénients de la numérisation tout en conservant ses avantages.

Des progrès considérables ont été réalisés dans le domaine de la cryptographie, ce qui a permis de mettre au point des outils très polyvalents tels que le chiffrement homomorphe complet et le calcul multipartite sécurisé. Toutefois, il n'existe toujours pas de solution miracle permettant d'assurer une protection efficace de la vie privée dès le départ. Nous constatons qu'il existe souvent un fossé entre les solutions cryptographiques théoriques et les problèmes du monde réel. Identifier et combler ces lacunes nous permet de concevoir des technologies pratiques d'amélioration de la confidentialité qui peuvent être déployées dans des scénarios concrets. Dans cette thèse, nous identifions et résolvons quatre problèmes du monde réel.

Nous présentons tout d'abord le problème de la recherche de documents sensibles au sein d'un réseau de journalistes d'investigation. En collaboration avec le Consortium international des journalistes d'investigation, nous concevons un moteur de recherche décentralisé peer-to-peer préservant la vie privée, appelé DatashareNetwork. Notre solution permet aux journalistes de trouver des collègues possédant des documents pertinents pour leur sujet d'enquête et de discuter anonymement de la possibilité d'une collaboration. Nous développons un prototype de DatashareNetwork et démontrons qu'il peut s'adapter à des milliers de journalistes et à des millions de documents.

Nous introduisons une nouvelle classe de problèmes appelés "private collection matching" dans lesquels un client cherche à déterminer si une collec-

tion d'ensembles appartenant à un serveur correspond à ses intérêts, comme la recherche de bases de données confidentielles de composés chimiques. Nous concevons un cadre basé sur le chiffrement homomorphe complet pour résoudre ces problèmes. Notre solution exploite au maximum le principe de minimisation des données et montre qu'il est possible de satisfaire les besoins du client en ne révélant qu'un seul bit. Nous évaluons notre cadre et montrons qu'il améliore considérablement la latence, le coût de calcul du client et le coût de communication par rapport aux solutions génériques qui offrent la même garantie de confidentialité.

Nous examinons le problème de double enregistrement dans la distribution de l'aide humanitaire en se concentrant sur les besoins du Comité international de la Croix Rouge. En réponse, nous concevons Janus, un système de déduplication biométrique préservant la vie privée qui est compatible avec les empreintes digitales, iris, et reconnaissance faciale; Janus est aussi capable de fusionner de multiples échantillons biométriques pour réduire le taux d'erreur et de se mettre à l'échelle de plus grande populations

Finalement, nous étudions le problème de la détection des chiffrement non sécurisés dans les communications aériennes à grande échelle. Nous concevons et développons un système d'aide à la décision qui aide les analystes humains à détecter de nouveaux cryptogrammes dans les communications aériennes. Nous évaluons notre système en l'appliquant à des données réelles et en demandant à notre analyste d'utiliser notre système d'aide pour trouver de nouveaux chiffrements. Notre analyse a permis de découvrir neuf algorithmes de chiffrement inconnus jusqu'alors (et potentiellement peu sûrs), que nous divulguons à diverses parties prenantes.

**Mots clés:** Technologies d'amélioration de la vie privée, cryptographie appliquée, ingénierie de la vie privée, chiffrement homomorphe, intersection d'ensembles privés, calcul privé.

# Contents

<b>Acknowledgements</b>	<b>v</b>
<b>Abstract</b>	<b>ix</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 DATASHARENETWORK: A Decentralized Privacy-Preserving Search Engine for Investigative Journalists</b>	<b>9</b>
2.1 Introduction . . . . .	9
2.2 Towards building DATASHARE . . . . .	11
2.2.1 Requirements gathering . . . . .	12
2.2.2 Sketching DATASHARE . . . . .	14
2.3 Multi-set PSI . . . . .	16
2.4 Privacy-preserving messaging . . . . .	19
2.4.1 Messaging system construction . . . . .	19
2.4.2 Messaging service privacy . . . . .	23
2.4.3 Cost evaluation . . . . .	25
2.5 The DATASHARE system . . . . .	27
2.5.1 Preliminaries . . . . .	28
2.5.2 DATASHARE protocols and design . . . . .	30
2.5.3 DATASHARE security analysis . . . . .	34
2.5.4 Cost evaluation . . . . .	38
2.6 Related work . . . . .	42
2.7 Future steps: better protection . . . . .	43
<b>3 Private Collection Matching Protocols</b>	<b>45</b>
3.1 Introduction . . . . .	45
3.2 Private Collection Matching . . . . .	47
3.2.1 Case studies . . . . .	47
3.2.2 PCM requirements . . . . .	49
3.2.3 Formal PCM definition . . . . .	50
3.3 Related work . . . . .	51
3.4 A framework for PCM schemes . . . . .	55
3.5 Technical background . . . . .	57
3.5.1 Homomorphic encryption . . . . .	57

3.5.2	Core functions . . . . .	58
3.6	PSI layer . . . . .	59
3.6.1	Small constant-size client set . . . . .	60
3.6.2	Small input domain . . . . .	62
3.6.3	Ensuring well-formed queries . . . . .	63
3.7	Matching layer . . . . .	64
3.8	Aggregation layer . . . . .	67
3.9	Security and privacy . . . . .	69
3.10	From theory to practice . . . . .	70
3.10.1	Asymptotic cost . . . . .	70
3.10.2	Implementation . . . . .	72
3.10.3	Optimizations . . . . .	73
3.11	PCM in practice . . . . .	74
3.11.1	Chemical similarity . . . . .	74
3.11.2	Peer-to-Peer document search . . . . .	76
3.11.3	Comparison with generic solutions . . . . .	78
3.12	Takeaways and future work . . . . .	80
<b>4</b>	<b>Janus: Safe Biometric Deduplication for Humanitarian Aid Distribution</b>	<b>83</b>
4.1	Introduction . . . . .	83
4.2	Deduplication for aid distribution . . . . .	85
4.2.1	Deduplication requirements . . . . .	86
4.3	Towards a safe deduplication system . . . . .	89
4.4	Janus' architecture . . . . .	91
4.4.1	Janus-enabled registration workflow . . . . .	91
4.4.2	Requirements achieved by design . . . . .	92
4.5	Biometrics . . . . .	93
4.6	Instantiating Janus . . . . .	95
4.6.1	SMC-Janus . . . . .	95
4.6.2	SHE-Janus . . . . .	99
4.6.3	TEE-Janus . . . . .	102
4.7	Biometrics in practice . . . . .	105
4.7.1	Membership with a single sample . . . . .	106
4.7.2	Membership at scale . . . . .	108
4.8	Evaluation . . . . .	109
4.8.1	Performance of Janus . . . . .	111
4.8.2	Comparison with Closely Related Work . . . . .	112
4.9	Related work . . . . .	114
4.10	Conclusion . . . . .	115

<b>5</b>	<b>Brutus: A Decision Support System to Prevent the Use of Insecure Communication in Aircraft</b>	<b>117</b>
5.1	Introduction . . . . .	117
5.2	Aircraft communications addressing and reporting system . . . . .	119
5.2.1	ACARS system model . . . . .	119
5.2.2	ACARS messages . . . . .	120
5.2.3	Usage . . . . .	121
5.2.4	Security . . . . .	122
5.2.5	Ethics . . . . .	122
5.3	Related work . . . . .	122
5.4	Detection methods . . . . .	123
5.4.1	Cipher detection based on text compression techniques . . . . .	124
5.4.2	Supervised text classification with a CNN model . . . . .	126
5.4.3	Supervised classification with a Random Forests model . . . . .	127
5.5	Cipher detection on generated datasets . . . . .	129
5.5.1	Synthetic data generation . . . . .	130
5.5.2	Experiment setup . . . . .	131
5.5.3	Baseline . . . . .	132
5.5.4	Noisy data labels . . . . .	133
5.5.5	Heterogeneous plaintext sources . . . . .	134
5.5.6	Heterogeneous cipher suites . . . . .	138
5.5.7	Conclusions . . . . .	140
5.6	A pipeline for labeling ACARS messages . . . . .	141
5.6.1	Synthetic data generation based on noisy samples . . . . .	142
5.6.2	Manual labeling . . . . .	143
5.6.3	Bootstrapping manual labels . . . . .	144
5.6.4	Conclusion . . . . .	145
5.7	Insecure ciphers in aerospace . . . . .	145
5.7.1	Setting-up detectors . . . . .	146
5.7.2	Automated detection of known ciphers . . . . .	147
5.7.3	Exploring unknown insecure ciphers in ACARS . . . . .	147
5.7.4	Analysis overhead . . . . .	149
5.7.5	Generalization over time and geographic regions . . . . .	151
5.7.6	Conclusion . . . . .	152
5.8	Discussion and conclusion . . . . .	152
<b>6</b>	<b>Conclusion</b>	<b>153</b>
6.1	Future work . . . . .	155
<b>A</b>	<b>Appendix for DATASHARENWORK</b>	<b>157</b>

A.1	Security of MS-PSI . . . . .	157
A.2	The limits of document search . . . . .	164
A.2.1	One-bit search extraction . . . . .	164
A.2.2	#doc search extraction . . . . .	167
<b>B</b>	<b>Appendix for Private Collection Matching Protocols</b>	<b>169</b>
B.1	Extra material . . . . .	169
B.2	Sum of random $\mathbb{Z}_q^*$ elements . . . . .	170
B.3	Privacy proof . . . . .	172
B.3.1	Security properties of HE schemes . . . . .	173
B.3.2	Semi-honest security . . . . .	176
B.3.3	Malicious server . . . . .	179
B.3.4	Malicious clients . . . . .	180
B.4	Extra benchmarks . . . . .	185
B.4.1	Small-domain protocols . . . . .	185
B.4.2	Circuit-based protocols . . . . .	186
B.4.3	OT-based protocol . . . . .	188
B.5	Solving matching in mobile apps . . . . .	190
B.6	PSI-SUM . . . . .	190
<b>C</b>	<b>Appendix for Janus</b>	<b>193</b>
C.1	Extended evaluation . . . . .	193
C.2	Normalized Hamming distance . . . . .	194
<b>D</b>	<b>Appendix for Brutus</b>	<b>197</b>
D.1	Random Forrest . . . . .	197
D.2	Additional results on generated datasets . . . . .	197
D.3	Decision support system . . . . .	200
	<b>Bibliography</b>	<b>203</b>
	<b>Curriculum Vitae</b>	<b>223</b>



# CHAPTER 1

## Introduction

Digitalization is becoming a daily norm, offering improved efficiency and capabilities. For example, the ease and effectiveness of searching through digital documents far surpasses searching printed ones. However, it also raises privacy concerns, particularly for minority or special-needs users. This thesis leverages privacy-enhancing technologies to balance these challenges and benefits.

Privacy is a fundamental human right. While some may not feel the impact of privacy breaches unless severe (e.g., identity theft), for others, like those who have sexuality, religion, or beliefs that deviate from the cultural norm, leaks can lead to discrimination and harassment. If a journalist's investigation gets leaked ahead of publishing their story, it endangers journalists, their sources, and stories. For survivors of violence or those under protection authorities or humanitarian organizations, exposure can be life-threatening. System designs should prioritize user needs to ensure safety and prevent harm.

Yet, despite advancements in cryptography suggesting that “privacy is a solved problem,” the reality differs. Indeed, we have tools like fully homomorphic encryption and secure multiparty computation theoretically offering broad solutions. However, in practice, these primitives are not silver bullets. An off-the-shelf use of these tools to add privacy to existing systems often does not lead to practical solutions. These translations from theory to practice regularly reveal gaps, whether in system scalability, mismatched functionalities or privacy guarantees, or the assumptions simply not being applicable in all situations.

In this thesis, we identify the gap that hinders the deployment of privacy-preserving solutions in 4 societal challenges. Following the identification, we design systems to bridge these gaps to create deployable solutions.

One form of gap is disparity between theoretical and practical resource

and constraint assumptions. Take, for instance, the challenge of searching for sensitive documents among a network of investigative journalists. Here, bandwidth is limited; journalists are not always online and the communication round trip time between journalists can take days. At the same time, journalists are patient and willing to wait a long time for their search results. Practical solutions must be limited to one round of communication and they should emphasize reducing transfer costs over speeding up computation. This contrasts sharply with traditional search objectives in the academic literature that aim for real-time, high-throughput systems that benefit from negligible transfer delays.

Potential oversharing or overlooking a privacy requirement is another form of gap. Consider the example of privately searching confidential chemical compound databases. A server owning this confidential data can face a dilemma: (1) deny all search access, thereby limiting the database's functionality and value, or (2) allow limited search access, risking the extraction of sensitive compound information by clients. Current private search methods often provide answers for every item in the collection, inadvertently letting clients accumulate details about each item over time. To bridge this gap, we introduce the PCM framework. This framework only reveals a single bit of information about the entire collection, significantly reducing the potential for data extraction.

Another common gap is mismatching functionality and security properties. We consider the challenge of protecting against double registration in humanitarian aid distribution. The most suitable option seems to be relying on biometrics and a large body of work on privacy-preserving biometrics exists. Unfortunately, the majority of existing work focuses on authentication and identification problems where users want to be recognized by the system. In contrast, users may want to evade the recognition in our context. This distinction leads to the impracticality of some protection mechanisms and requiring different goals when handling biometric error rates.

Occasionally, ubiquitous assumptions do not hold in practice leading to the creation of gaps. For instance, almost all cryptographers believe that classical ciphers are historical relics that are not used in modern systems. Yet, archaic insecure ciphers such as monoalphabetic substitution cipher can be found in infrastructures like aircraft communication. Addressing this gap involves proactive detection of such insecure ciphers. One way to bridge this gap is by automatically detecting when insecure ciphers are in use. We design a decision support system to aid experts to detect and monitor the use of

insecure ciphers in aircraft communication at scale.

Upon identifying a gap and designing a solution, we develop a proof of concept. We then evaluate its performance across different operational points and scenarios derived from our requirement analysis. By comparing our prototype’s efficiency with both prior privacy-preserving systems and those offering fewer privacy safeguards, we demonstrate that our systems achieve a reasonable trade-off between privacy and performance.

## Contributions

We study and solve four privacy problems in this thesis:

### Chapter 2: DatashareNetwork

Investigative journalists collect large numbers of digital documents during their investigations. Gathering documents is a major hurdle for investigative journalists and requires immense effort, and other journalists can greatly benefit from accessing them. However, many of these documents contain sensitive information; and revealing them can endanger reporters, their stories, and their sources. Consequently, countless opportunities for international collaboration are lost as many documents are used only for single, local, investigations.

In Chapter 2, we introduce DatashareNetwork a decentralized and privacy-preserving search system that enables journalists worldwide to find documents via a dedicated network of peers. This system is the result of a long-running collaboration, over 5 years, between EPFL and the International Consortium of Investigative Journalists (ICIJ). ICIJ had already developed a local search platform called Datashare [1] and contacted us to help extend the local search platform to a peer-to-peer search engine that ensures the privacy of journalists.

We study the risks and practical constraints that investigative journalists face. We introduce and design a new cryptographic primitive called ‘multi-set private set intersection’ (MS-PSI) and use it to design a private search engine. Next, we design a new anonymous messaging tailored for journalists’ constraints to enable them to anonymously discuss collaboration with their colleagues. Finally, we combine our privacy-preserving search and messaging systems with existing tools such as anonymous credentials and Tor to design an end-to-end system for journalists. We perform a system-wide analysis at

the end to prove the security and privacy of the system as a whole.

We develop a prototype of our system and show that it can scale to thousands of journalists and millions of documents. Currently, DatashareNetwork software is in the beta test stage and soon it will be fully deployed by ICIJ to enable peer-to-peer document search between their journalists.

### Chapter 3: Private collection matching

We introduce *Private Collection Matching (PCM)* problems, in which a client aims to determine whether a collection of sets owned by a server matches their interests. We systematize the privacy requirements for solving PCM problems, and we show that existing privacy-preserving cryptographic primitives cannot solve PCM problems efficiently without harming privacy.

We propose a modular framework that enables protocol designers to build privacy-preserving PCM solutions. Our framework enables the design of systems that output a single bit determining whether a collection of server sets matches the client’s interest. In our framework, the server computes per-server-set *binary* answers to “Is this set of interest to the client?”, then aggregates  $N$  per server-set responses into a single collection-wide response. The communication cost of our framework scales linearly with the size of the client’s set and is independent of the number of server elements.

We demonstrate the potential of our framework by designing and implementing novel solutions for two real-world PCM problems: determining whether a dataset has chemical compounds of interest, and determining whether a document collection has relevant documents. We show that our framework offers improved privacy with competitive cost compared to custom-made solutions, and significantly improves the latency (2–96x), client’s computation cost (75–70,000x), and communication cost (93–2800x) with respect to generic solutions that offer the same privacy guarantee.

### Chapter 4: Janus

Humanitarian organizations provide aid to people in need. To use their limited budget efficiently, their distribution processes must ensure that legitimate recipients cannot receive more aid than they are entitled to. Thus, it is essential that recipients can register at most once per aid program.

Taking the International Committee of the Red Cross’s aid distribution registration process as a use case, we identify the functional, safety, and deployment requirements to detect double registration. The traditional approaches to solving this problem, such as relying on government-issued IDs or local trusted actors (e.g. community representatives), suffer from availability, accuracy, and efficiency issues. One natural path to solving this problem that coincides with digitalization is the use of biometrics to detect duplicate registration requests. However, if not handled properly, the use of biometrics can put both the humanitarian organization and its aid recipients at risk.

In Chapter 4, we present Janus, a system that combines privacy-enhancing technologies with biometrics to prevent double registration in a safe manner. Janus does not create plaintext biometric databases and reveals only *one bit* of information at registration time (whether the user registering is present in the database or not). We design three instantiations of Janus based on secure multiparty computation, somewhat homomorphic encryption, and trusted execution environments. Janus supports all major biometric sources (i.e., fingerprint, iris, and face recognition) and enables biometric alignment and fusion to lower the error rates of the system. We implement and evaluate Janus to demonstrate that it satisfies the functional, safety, accuracy, and performance needs of humanitarian organizations. We compare Janus with existing alternatives and show it is the first system that provides the low error rates necessary for operating in the context of humanitarian aid distribution while providing strong protection.

## Chapter 5: Brutus

Aircraft and ground counterparts have been communicating via the ACARS data-link protocol for more than five decades. Like many legacy protocols, ACARS was not designed with security in mind and nowadays can easily be accessed with modern consumer technology. Aircraft manufacturers and airlines have noticed the need for the security and confidentiality of ACARS messages and started to create proprietary solutions to encrypt ACARS messages. Recent work discovered, through *manual inspection*, that some parties “encrypt” messages using an insecure monoalphabetic substitution cipher. To increase the security of aircraft communication, it is important to identify all actors that use such insecure ciphers.

We design Brutus a decision support system to help authorities monitor ACARS messages and detect actors who use (insecure) encryption schemes.

We propose three methods to *automatically detect* ciphertexts in aircraft communication at scale based on text compression techniques, convolutional neural network, and random forest classification. This is particularly challenging because none of the existing ACARS datasets mark whether a message is a plaintext or ciphertext and ACARS messages are heterogenous, mixing human and machine-generated messages. We propose the use of synthetic data to evaluate our detectors when no real-world labeled data is available. We design a new methodology to gather labeled ACARS messages at scale and create a pipeline for training detectors.

We design and develop a decision support dashboard that helps a human analyst to detect new ciphertexts in ACARS communication. We evaluate our decision support system by applying it to real-world data and asking our analyst to use the dashboard to find new unknown encryptions in ACARS messages. Our analysis led to *identifying 9 previously unknown (and potentially insecure) ciphers*, which we disclose to various stakeholders.

## Bibliographic notes

The contents of this thesis are based on the following:

**Chapter 2.** Kasra EdalatNejad, Wouter Lueks, Julien Pierre Martin, Soline Ledésert, Anne L'Hôte, Bruno Thomas, Laurent Girod, Carmela Troncoso: “DatashareNetwork: A Decentralized Privacy-Preserving Search Engine for Investigative Journalists”. USENIX Security Symposium 2020.

**Chapter 3.** Kasra EdalatNejad, Mathilde Raynal, Wouter Lueks, Carmela Troncoso: “Private Collection Matching Protocols”. Proceedings on Privacy Enhancing Technologies (PoPETs). 2023.

**Chapter 4.** Kasra EdalatNejad, Wouter Lueks, Justinas Sukaitis, Vincent Graf Narbel, Massimo Marelli, Carmela Troncoso: “Janus: Safe Biometric Deduplication for Humanitarian Aid Distribution” Under submission.

**Chapter 5.** Kasra EdalatNejad\*, Theresa Stadler\*, Martin Strohmeier, Vincent Lenders, Wouter Lueks, Carmela Troncoso: “Brutus: A Decision Support

System to Prevent the Use of Insecure Communication in Aircraft”. Under submission.

\* denotes equal contribution.





# CHAPTER 2

## **DATASHARENETWORK: A Decentralized Privacy-Preserving Search Engine for Investigative Journalists**

This chapter is based on the following article:

Kasra EdalatNejad, Wouter Lueks, Julien Pierre Martin, Soline Ledésert, Anne L’Hôte, Bruno Thomas, Laurent Girod, Carmela Troncoso: “DatashareNetwork: A Decentralized Privacy-Preserving Search Engine for Investigative Journalists”. USENIX Security Symposium 2020.

### **2.1 Introduction**

Investigative journalists research topics such as corruption, crime, and corporate misbehavior. Two well-known examples of investigative projects are the Panama Papers that resulted in several politicians’ resignations and sovereign states recovering hundreds of millions of dollars hidden in offshore accounts [2], and the Boston Globe investigation on child abuse that resulted in a global crisis for the Catholic Church [3]. Investigative journalists’ investigations are essential for a healthy democracy [4]. They provide the public with information kept secret by governments and corporations. Thus, effectively holding these institutions accountable to society at large.

In order to obtain significant, fact-checked, and impactful results, journalists require large amounts of documents. In a globalized world, local issues are increasingly connected to global phenomena. Hence, journalists’ collections can be relevant for other colleagues working on related investigations. However, documents often contain sensitive and/or confidential information and possessing them puts journalists and their sources increasingly at risk of

identification, prosecution, and persecution [5, 6]. As a result journalists go to great lengths to protect both their documents and their interactions with other journalists [7]. With these risks in mind, the International Consortium of Investigative Journalists (ICIJ) approached us with this question: *Can a global community of journalists search each other's documents while minimizing the risk for them and their sources?*

Building a practical system that addresses this question entails solving five key challenges:

- 1) *Avoid centralizing information.* A party with access to all the documents and journalists' interaction would become a very tempting target for attacks by hackers or national agencies, and for legal cases and subpoenas by governments.
- 2) *Avoid reliance on powerful infrastructure.* Although ICIJ has journalists worldwide, it does not have highly available servers in different jurisdictions.
- 3) *Deal with asynchrony and heterogeneity.* Journalists are spread around the world. There is no guarantee that they are online at the same time, or that they have the same resources.
- 4) *Practical on commodity hardware.* Journalists must be able to search documents and communicate with other journalists without this affecting their day-to-day work. The system must be efficient both computationally and in communication costs.
- 5) *Enable data sovereignty.* Journalists are willing to share but not unconditionally. They should be able to make informed decisions on revealing documents, on a case-by-case basis.

The first four requirements preclude the use of existing advanced privacy-preserving search technologies, whereas the fifth requirement precludes the use of automatic and rule-based document retrieval. More concretely, the first requirement prevents the use of central databases and private information retrieval (PIR) [8–10] between journalists, as standard PIR requires a central list of all searchable (potentially sensitive) keywords. The second requirement rules out multi-party computation (MPC) between distributed servers [11–13].

The third and fourth requirement exclude technologies that require many round trips or high bandwidth between journalists such as custom private set intersection [11, 14–17], keyword-based PIR [18, 19], and generic MPC protocols [11–13, 20], as well as the use of privacy-preserving communication systems that require all users to be online [21, 22].

We introduce DATASHARENETWORK, a decentralized document search engine for journalists to be integrated within ICIJ’s open source tool for organizing information called Datashare [1]. DATASHARENETWORK addresses the challenges as follows. First, journalists keep their collections in their computers. Thus, if a journalist is hacked, coerced, or corrupted, only her collection is compromised. Second, we introduce a new multi-set private set intersection (MS-PSI) protocol that enables asynchronous search and multiplexed queries to reduce computation and communication costs. Third, we combine existing privacy-preserving technologies [23, 24] to build a pigeonhole-like communication mechanism that enables journalists to anonymously converse with each other in an unobservable manner. These components ensure that even if an adversary gains the ability to search others’ documents, she cannot extract all documents nor all users in the system. In the rest of the document, for simplicity, we refer to DATASHARENETWORK as DATASHARE.

Our contributions are as follows:

- ✓ We elicit the security and privacy requirements of a document search system for investigative journalists.
- ✓ We introduce MS-PSI, a private set intersection protocol to efficiently search in multiple databases without incurring extra leakage with respect to traditional PSI with pre-computation.
- ✓ We propose an asynchronous messaging system that enables journalists to search and converse in a privacy-preserving way.
- ✓ We design DATASHARE, a secure and privacy-preserving decentralized document search system that protects from malicious users and third parties the identity of its users, the content of the queries and, to a large extent, the journalists’ collections themselves. We show that DATASHARE provides the privacy properties required by journalists, and that the system can easily scale to more than 1000 participants, even if their document collections have more than 1000 documents.

## 2.2 Towards building DATASHARE

We build DATASHARE at the request of the International Consortium of Investigative Journalists, ICIJ. When unambiguous from the context, we refer to ICIJ simply as the organization.

### 2.2.1 Requirements gathering

In order to understand the needs of investigative journalists, ICIJ ran a survey among 70 of their members and provided us with aggregate statistics, reported below. We used the survey results as starting point for the system’s requirements, and we refined these requirements in weekly meetings held for more than one year with the members of ICIJ’s Data & Research Unit who are in charge of the development and deployment of the local tool Datashare [1].

*User base.* ICIJ consists of roughly 250 permanent journalist members in 84 countries. These members occasionally collaborate with external reporting partners. The maximum number of reporters working simultaneously on an investigation has reached 400. The organization estimates that each member is willing to make approximately one thousand of their documents available for searching. To accommodate growth, we consider that DATASHARE needs to scale to (at least) 1000 users, and (at least) 1 million documents.

Journalists work and live all over the globe, ranging from Sydney to San Francisco, including Nairobi and Kathmandu; this results in large timezone differences. Around 38% of the journalists have a computer permanently connected to the Internet, and another 53% of them are connected during work hours: eight hours a day, five days a week. The rest are connected only during a few hours per day. As it is unlikely that journalists are online at the same time, *the search system needs to enable asynchronous requests and responses.* Furthermore, many journalists live in regions with low-quality networks: only half of the journalists report having a fast connection. Thus, *DATASHARE cannot require high bandwidth.*

*Waiting time.* As the system must be asynchronous, the survey asked journalists how much they are willing to wait to obtain a the result of a query. 21% of the surveyees are willing to wait for hours, whereas another 56% can wait for one or more days. Hence, *DATASHARE does not need to enable real-time search.* Yet, given the delivery times of up to 24 hours, to keep search latency within a few days, DATASHARE must *use protocols that can operate with just one communication round.* Therefore, we discard multi-round techniques such as multi-party computation [11–13, 20].

*Queries nature.* The queries made by journalists are in a vast majority formed by *keywords* called *named entities*: names of organizations, people, or locations of interest. Therefore, journalists do not require a very expressive querying language: DATASHARE must *support queries made of conjunctions of keywords.*

Journalists are interested in a small set of these entities at a time: only those related to their current project. Consequently, *queries are not expected to include more than 10 terms at a time*, and *journalists are not expected to issue a large number of queries in parallel*.

During the design phase, we also learned that as most terms of interest are investigation-specific (e.g., XKeyScore in the Snowden leaks, or Mossack Fonseca in the Panama Papers), *a pre-defined list of terms cannot cover all potentially relevant keywords for journalists*. Therefore, techniques based on fixed lists such as private information retrieval (PIR) [8–10] are not suitable for building DATASHARE.

*Security and privacy.* Regarding security and privacy concerns, journalists identify four types of principals: the journalists themselves, their sources, the people mentioned in the documents, and the ICIJ. They identify three assets: the named entities in documents, the documents themselves, and the conversations they have during an investigation. The disclosure of named entities could leak information about the investigation, or could harm the cited entities (which could in turn could trigger a lawsuit). Whole documents are considered the most sensitive aP involved, their sources, the organization, and the whole investigation.

Journalists mostly worry about third party adversaries such as corporations, governments (intelligence agencies), and organized crime. Sources and other journalists are in general considered non-adversarial. Similarly, journalists *trust ICIJ to be an authority for membership and to run their infrastructure*. However, to prevent coercion and external pressures, ICIJ does *not* want to be trusted for privacy.

The main requirement for DATASHARE is *to protect the confidentiality of assets from third parties that are not in the system*. This implies that DATASHARE cannot require journalists to send their data to third parties for analysis, storage, indexing, or search. Journalists are concerned about only subsets of these adversaries at a time. Therefore, DATASHARE *does not need to defend against global adversaries*.

Journalists initially did not consider their colleagues as adversaries. However, after a threat analysis, we concluded that there is a non-negligible risk that powerful adversaries can bribe or compromise honest journalists, in particular when those journalists live in jurisdictions with less protection for civil rights. Therefore, we require that DATASHARE *must minimize the amount of information that journalists, or ICIJ, learn about others: searched keywords,*

*collections, and conversations.* More concretely, we require that *searches be anonymous and that the searched terms be kept confidential, with respect to both journalists and the organization.* This way neither journalists nor the organization become a profitable target for adversaries.

With respect to conversations, 64% of the surveyees report that they would prefer to remain anonymous in some cases. Furthermore, 60% of the respondents declare that they prefer to have a screening conversation before deciding to share documents. This means that *search and sharing features need to be separated to enable screening.* DATASHARE must *provide anonymous means for journalists to discuss document sharing to ensure safety.* We expect conversations within DATASHARE to be short, as their only goal is to agree on whether to proceed with sharing. After journalists agree, we assume they will switch to an alternative secure communication channel and DATASHARE does not need to support document retrieval.

### 2.2.2 Sketching DATASHARE

DATASHARE is run by ICIJ. Access to the system is exclusive to ICIJ members and authorized collaborators. Journalists trust ICIJ to act as a token issuer and only give tokens to authorized journalists. To enable journalists to remain anonymous, tokens are implemented using blind signatures. Journalists use these tokens demonstrate membership without revealing their identities.

DATASHARE provides the following infrastructure to facilitate asynchronous communication between journalists: a *bulletin board* that journalists use to broadcast information, and a *pigeonhole* for one-to-one communication. All communications between journalists and the infrastructure (pigeonhole or bulletin board) are end-to-end encrypted (i.e., from journalist to journalist) and anonymous. Hence, the infrastructure needs to be trusted for availability, but not to protect the privacy of the journalists and their documents.

Each authorized journalist in DATASHARE owns a corpus of documents that they make available for search. Journalists can take two roles: (i) *querier*, to search for documents of interest, and (ii) *document owner*, to have their corpus searched. Journalists first search for matching documents then (anonymously) converse with the corresponding document owners to request the document.

Figure 2.1 sketches DATASHARE’s architecture. First, journalists upload privacy-preserving representations of their collections and contact informa-

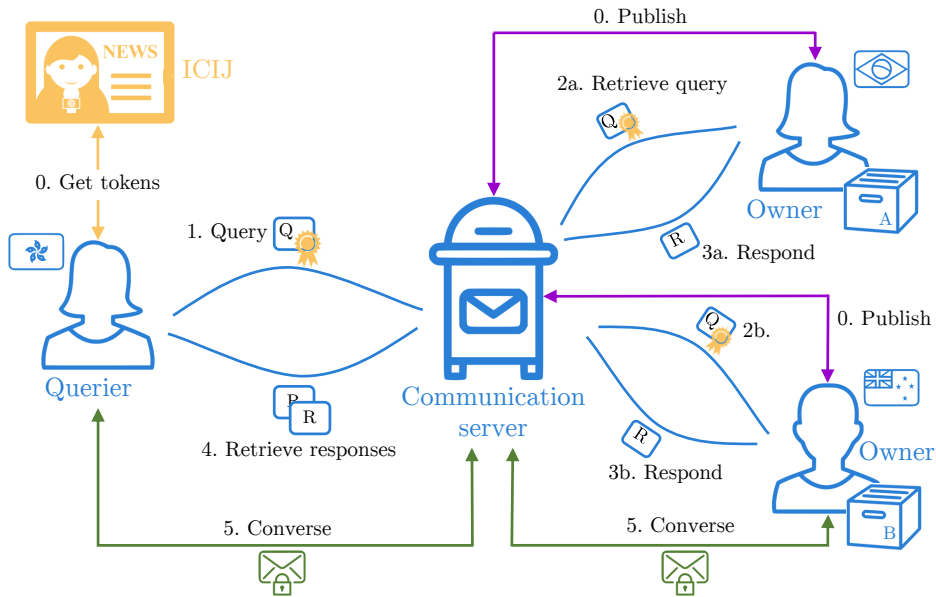


Figure 2.1: DATASHARE architecture overview.

tion to the bulletin board. To issue a query, journalists construct a privacy-preserving representation of their keywords and broadcast it together with an authorization token through the bulletin board. Owners periodically retrieve new queries from the bulletin board. If the authorization is valid, they send a response to the querier using the pigeonhole. The querier uses this response to identify matches with the documents in the owner’s collection.

When journalists find a *match* in a collection, i.e., a document that contains all the keywords in the query, they can start a conversation with the document owner to request sharing. Document owners append a public contact key to their collection to enable queriers to carry out this conversation in an anonymous way via the pigeonhole.

*Instantiation.* DATASHARE uses four main privacy-preserving building blocks: a multi-collection search mechanism, a messaging system, an anonymous communication channel, and an authorization mechanism.

We implement the privacy-preserving search mechanism by using a novel primitive that we call multi-set private set intersection (MS-PSI) described in Section 2.3. We design a privacy-preserving messaging system in Section 2.4; it provides both the bulletin board and pigeonhole functionality. We rely on the Tor [23] network as anonymous communication channel, and we use blind signatures to implement privacy-preserving authorization (see Section 2.5.1). In Section 2.5.2, we explain how DATASHARE combines these building blocks.

Table 2.1: Notation.

$\mathbb{G}, g, p$	A cyclic group, its generator and the group's order
$n$	The security parameter
$x \leftarrow^{\$} X$	Draw $x$ uniformly at random from the set $X$
$\mathbf{H}, \hat{H}$	Hash functions mapping into $\{0, 1\}^n$ resp. group $\mathbb{G}$ .
$[n]$	The set $\{1, \dots, n\}$
$s, c$	The server's and client's secret keys
$Y_i$	The server's $i$ th set $Y_i = \{y_{i,1}, \dots, y_{i,n_i}\}$
$N, n_i$	Nr. of server sets, resp. nr. of elements in set $Y_i$
$X$	The client's set $X = \{x_1, \dots, x_m\}$
$m$	The number of elements in the client's set
$\tau, \tau^{(i)}$	Pretags for client ( $\tau$ ) resp. the server's $i$ th set $Y_i$ ( $\tau^{(i)}$ )
$\mathbf{TC}$	The server's tag collection

## 2.3 Multi-set PSI

Private set intersection (PSI) protocols enable two parties holding sets  $X$  and  $Y$  to compute the intersection  $X \cap Y$ , without revealing information about the individual elements in the sets. In this section, we introduce a multi-set private set intersection (MS-PSI) protocol that simultaneously computes intersections of set  $X$  with  $N$  sets  $\{Y_1, \dots, Y_N\}$  at the server. In Section 2.6, we review existing PSI variants.

*Notation.* (See Table 2.1) We use a cyclic group  $\mathbb{G}$  of prime order  $p$  generated by  $g$ . We write  $x \leftarrow^{\$} X$  to denote that  $x$  is drawn uniformly at random from the set  $X$ . Let  $n$  be a security parameter. We define two hash functions  $\mathbf{H} : \{0, 1\}^* \rightarrow \{0, 1\}^n$  and  $\hat{H} : \{0, 1\}^* \rightarrow \mathbb{G}$ . Finally, we write  $[n]$  to denote the set  $\{1, \dots, n\}$ .

*Related PSI schemes.* We build on the single-set PSI protocol by De Cristofaro et al. [25], see Figure 2.2. In this protocol the client blinds her elements  $x_i \in \mathbb{G}$  as  $\tilde{x}_i = x_i^c$  using a blinding factor  $c$  before sending them to the server. The server applies its own secret to the blinded elements,  $\hat{x}_i = \tilde{x}_i^s$ , and sends them back to the client in the same order, together with a tag collection of her own blinded elements:  $\mathbf{TC} = \{\mathbf{H}(y^s) \mid y \in Y\}$ . The client unblinds her elements, obtaining a list of  $x_i^s$ s. Then, the client computes a tag  $\mathbf{H}(x_i^s)$  for each of them and compares it to the server's tags  $\mathbf{TC}$  to find matching elements.

To increase efficiency when the server set is large, client-server PSI (C-PSI) schemes in the literature [15, 16, 26] introduce optimizations to avoid that the server has to compute and send a large fresh set of tags every execution.



Client	Server
$X = \{x_1, \dots, x_m\} \subset \mathbb{G}$	$Y = \{y_1, \dots, y_n\} \subset \mathbb{G}$
$c \xleftarrow{\$} \mathbb{Z}_p$	$s \xleftarrow{\$} \mathbb{Z}_p$
$\tilde{x}_i = x_i^c$	$\hat{x}_i = \tilde{x}_i^s$
$T_i = H(\hat{x}_i^{c^{-1}})$	$TC = \{H(y^s) \mid y \in Y\}$
Return $\{x_i \mid T_i \in TC\}$	

Figure 2.2: Vanilla PSI protocol by De Cristofaro et al. [25].

Instead, the server *precomputes* the tag collection with a long-term secret key  $s$  and sends it to the client once. In subsequent *online* phases, the server answers clients' queries by using the long-term key  $s$ . This significantly improves the communication and computation cost, as the server does not compute or send the tag collection every time.

*A new Multi-set PSI protocol.* Our *multi-set* private set intersection protocol (MS-PSI) intersects a client set  $X = \{x_1, \dots, x_m\} \subset \{0, 1\}^*$  with  $N$  sets  $Y_i = \{y_{i,1}, \dots, y_{i,n_i}\} \subset \{0, 1\}^*$  at the server to obtain the intersections  $X \cap Y_i$ . Our protocol computes all intersections *simultaneously*, lowering the computation and communication cost with respect to running  $N$  parallel PSI protocols. In DATASHARE,  $X$  contains the query (a conjunction of search keywords) and  $Y_i$  represents document  $i$ 's keywords, as described in Section 2.5.2. We use  $\hat{H}$  to map keywords to group elements.

A naive approach to building MS-PSI would be to mimic the client-server protocols and to reuse the long-term key  $s$  for all sets  $Y_i$ . This approach maps identical elements in sets  $Y_i, Y_j$  to the same tag revealing intersection cardinalities  $|Y_i \cap Y_j|$ . We remove the link between tags across sets by adding a tag diversifying step to the precomputation phase of client-server PSI (see Figure 2.3). We first compute pretags  $\tau^{(i)}$  for each set  $Y_i$  by raising each element to the power of the long-term secret  $s$ . Then, we compute per-set tags by hashing the pretags  $\tau$  with the set index  $i$  to obtain  $H(i \parallel \tau)$ . The hash-function ensures that the tags of each set are independent. The server publishes the tag collection  $TC$  and the number of sets  $N$ .

During the online phase, the client blinds its set as in the scheme of De Cristofaro et al. and sends it to the server. The server re-blinds the set with its secret  $s$  and sends it back to the client in the same order. The client unblinds the result to obtain the pretags for her elements. The client then computes the corresponding tags  $T^{(d)}$ , for each document  $d \in [N]$ , and obtains the intersection.

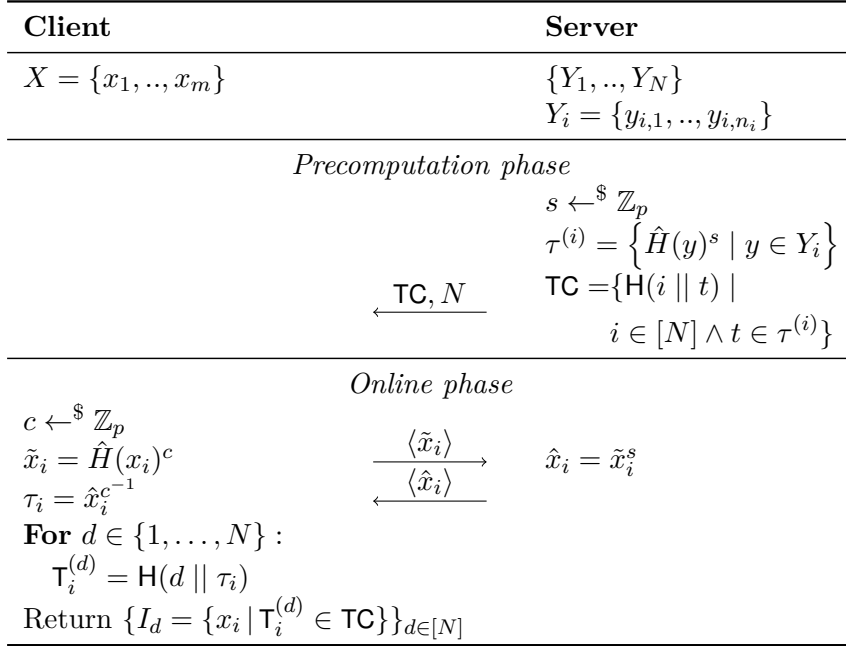


Figure 2.3: Our MS-PSI protocol.

In Appendix A.1, we prove the following theorem to show that the server learns nothing about the client’s set, and that the client learns nothing more than the intersections  $X \cap Y_i$ .

**Theorem 1.** The MS-PSI protocol is private against malicious adversaries in the random oracle model for  $\mathbf{H}$  and  $\hat{H}$ , assuming the one-more-gap Diffie-Hellman assumption holds.

The MS-PSI protocol does not provide correctness against a malicious server, who can respond arbitrarily leading the client to compute an incorrect intersection. However, from Theorem 1 we know that, even then, the malicious server cannot gain any information about the client’s set.

*Performance.* Table 2.2 compares the performance of our MS-PSI protocol with the vanilla and the client-server PSI protocols in the multi-set setting. We show the computation and communication cost for a server with  $N$  sets and a client set with  $m$  elements. MS-PSI reduces the server’s online communication and computation by a factor  $N$ . The client can replace expensive group operations by inexpensive hash computations, significantly reducing her online cost. The example costs for  $N = 1000$  (in square brackets) illustrate this reduction showing an improvement of 3 orders of magnitude.

Table 2.2: Performance of PSI variants in a multi-set scenario:  $N$  is the number of server sets;  $S$  is the total number of server elements;  $m$  is the size of the client set; and  $\tau_e$  and  $\tau_H$  denote the cost of an exponentiation and a hash computation ( $\tau_{H+e} = \tau_H + \tau_e$ ). We report in square brackets the cost estimation when  $m = 10$ ,  $N = 1000$ ,  $S = 100,000$  (i.e., server sets have 100 elements). We assume that group elements require 32 bytes,  $\tau_e = 100 \mu\text{s}$ , and  $\tau_H = 1 \mu\text{s}$ .

	Vanilla	C-PSI	MS-PSI
<i>Precomputation phase</i>			
Server	—	$S\tau_{H+e}$	$S\tau_{H+e}$
Comms	—	$S$	$S$
<i>Online phase</i>			
Client	$2mN\tau_{H+e}$ [2 s]	$2mN\tau_{H+e}$ [2 s]	$2m\tau_e + mN\tau_H$ [12 ms]
Server	$S\tau_{H+e} + mN\tau_e$ [11 s]	$mN\tau_e$ [1 s]	$m\tau_e$ [1 ms]
Comms	$S + 2mN$ [3.84 MB]	$2mN$ [640 KB]	$2m$ [640 B]

## 2.4 Privacy-preserving messaging

In this section, we introduce DATASHARE’s communication system (CS). Journalists use the CS to support MS-PSI-based search and to converse anonymously after they find a match. The CS respects the organization’s limitations (see Section 2.2.1). The communication costs do not hinder the day-to-day operation of journalists, and the system supports asynchronous communication. As the organization cannot deploy non-colluding nodes, the CS uses one server. This server is trusted for availability, but not for privacy.

DATASHARE’s communication system is designed to host short conversations for discussing the sharing of documents. We anticipate that journalists will migrate to using encrypted email or secure messengers if they need to communicate over a long period or if they need to send documents.

### 2.4.1 Messaging system construction

The server provides two components: a *bulletin board* for broadcast messages, and a *pigeonhole* for point-to-point messages. We use communication server to refer to the entity that operates both components. To hide their network identifiers from the server and network observers, journalists always use Tor

[23] for communication. To ensure unlinkability, DATASHARE creates a new Tor circuit for every request.

*Bulletin board.* The bulletin board implements a database that stores broadcast messages. Journalists interact with the bulletin board by using two protocols: **BB.broadcast**( $m$ ), which adds a message  $m$  to the database to broadcasts it to all journalists, and  $m \leftarrow$  **BB.read**() to retrieve unseen messages.

*Pigeonhole.* The pigeonhole consists of a large number of one-time-use mailboxes. Journalists use the pigeonhole to send and receive replies to search queries and to conversation messages. Journalists use the method **PH.SendRaw** (Protocol 1) to send query replies; and the asynchronous process **PH.RecvProcess** (Protocol 2) to retrieve incoming query replies and conversation messages. Journalists use **PH.Monitor** (Protocol 3) to receive notifications of new messages from the pigeonhole and to trigger **PH.RecvProcess**. Journalists are expected to connect to the system several times a week (see Section 2.2.1). In agreement with ICIJ, we decided that the pigeonhole will delete messages older than 7 days.

Journalists may initiate a conversation after receiving a successful match. To hide this event, we ensure that the *sending of conversation messages is unobservable*: the server cannot determine whether a journalist sends a conversation message or not (see Definition 1). This hides whether a conversation occurred, and therefore whether the search revealed a match or not. To ensure unobservability of conversation messages, journalists run **PH.Cover** (Protocol 4) to send cover messages at a constant Poisson rate to *every* journalist. To send a conversation message, it suffices to replace one of the cover messages with the real message (see **PH.HiddenSend**, Protocol 5).

Journalists use the Diffie-Hellman key exchange to compute mailbox addresses and message encryption keys, and an authenticated encryption scheme **AE** to encrypt messages. Queriers generate a fresh key for every query and use that key to receive query replies and to send conversation messages associated with that query. Document owners use a medium-term key to send query replies and to receive conversation messages from queriers (see Section 2.5.2). When exchanging cover traffic, journalists use fresh cover keys to send and their medium-term keys to receive.

**Protocol 1** (**PH.SendRaw**( $\mathbf{sk}_S, \mathbf{pk}_R, m$ )). To send message  $m$  to recipient  $R$  with public key  $\mathbf{pk}_R$ , a sender with private key  $\mathbf{sk}_S$  proceeds as follows. Let  $n_s$  be the number of times  $S$  called **PH.SendRaw** to send a message to  $R$  before. The sender

1. computes the Diffie-Hellman key  $k' = \text{DH}(\text{sk}_S, \text{pk}_R)$ ;
2. computes the random rendezvous mailbox  $\text{addr} = \text{H}(\text{'addr'} \parallel k' \parallel \text{pk}_S \parallel n_s)$  and a symmetric key  $k = \text{H}(\text{'key'} \parallel k' \parallel \text{pk}_S \parallel n_s)$ ;
3. pads the message  $m$  to obtain  $m'$  of length  $\text{mlen}$ , and computes the ciphertext  $c = \text{AE.enc}(k, m')$ ;
4. opens an anonymous connection to the pigeonhole and uploads  $c$  to mailbox  $\text{addr}$ .

For every upload, the pigeonhole notifies all monitoring receivers (see **PH.Monitor** below) that a message arrived at  $\text{addr}$ .

**Protocol 2** (**PH.RecvProcess**( $\text{sk}_R, \text{pk}_S$ )). To receive a message from sender  $S$  with public key  $\text{pk}_S$ , a receiver  $R$  with private key  $\text{sk}_R$  runs the following asynchronous process. Let  $n_r$  be the number of times  $R$  successfully received a message from  $S$ . The receiver

1. computes the Diffie-Hellman key  $k' = \text{DH}(\text{sk}_R, \text{pk}_S)$ ;
2. uses  $k'$  to compute a random rendezvous mailbox  $\text{addr} = \text{H}(\text{'addr'} \parallel k' \parallel \text{pk}_S \parallel n_r)$  and a symmetric key  $k = \text{H}(\text{'key'} \parallel k' \parallel \text{pk}_S \parallel n_r)$ ;
3. waits until **PH.Monitor** (see below) receives a notification of a new message on address  $\text{addr}$ . If no message is posted to  $\text{addr}$  in seven days, the process terminates;
4. opens an anonymous connection to the pigeonhole and downloads the ciphertext  $c$  at address  $\text{addr}$  (if there was no message due to a false positive, the process continues at step 3); and
5. decrypts the message  $m' = \text{AE.dec}(k, c)$  and returns the unpadded message  $m$  or  $\perp$  if decryption failed.

When the receiver goes offline, this process is paused and resumed when the receiver comes online again.

A sender may send multiple messages without receiving a response. The receiver calls **PH.RecvProcess** repeatedly to receive all messages ( $n_r$  increases every time). To ensure that the participants derive the correct addresses and decryption keys, participants keep track of the message counters  $n_s, n_r$  for each pair of keys  $(\text{sk}_S, \text{pk}_R)$  and  $(\text{sk}_R, \text{pk}_S)$ , respectively.

**Protocol 3** (**PH.Monitor**). Journalists run the **PH.Monitor** process to monitor for incoming messages. The receiver

1. opens an anonymous monitoring connection to the pigeonhole and requests a list of addresses  $\text{addr}$  that received a message since she was last online

2. via the same anonymous connection, receives notifications of addresses **addr** with new messages.

Addresses **addr** received in step 1 or 2 can cause the **PH.RecvProcess** processes to continue past step 3. To save bandwidth, the pigeonhole sends a cuckoo filter [27] that contains the addresses in step 1. Moreover, the pigeonhole only sends the first two bytes of the address in step 2 (**PH.RecvProcess** handles false positives).

The **PH.Cover** and **PH.HiddenSend** protocols ensure conversation messages are unobservable. Senders store a queue of outgoing conversation messages for each recipient.

**Protocol 4** (**PH.Cover**( $\mathbf{sk}_R$ )). As soon as the journalists come online, they start the **PH.Cover** process. Let  $\mathbf{sk}_R$  be the medium-term private key, and  $\mathbf{pk}_1, \dots, \mathbf{pk}_{n-1}$  be the medium-term public keys of the other journalists. The process runs the following concurrently:

- *Cover keys.* Draw an exponential delay  $t_k \leftarrow \text{Exp}(1/\lambda_k)$ , and wait for time  $t_k$ . Generate a fresh cover key-pair  $(\mathbf{sk}_c, \mathbf{pk}_c)$  and upload  $\mathbf{pk}_c$  to the bulletin board by calling **BB.broadcast**( $\mathbf{pk}_c$ ). Repeat.
- *Sending messages.* Wait until the first cover key has been uploaded. For each recipient  $\mathbf{pk}_i$ , proceed as follows:
  1. Draw  $t_i \leftarrow \text{Exp}(1/\lambda_c)$  and wait for time  $t_i$ .
  2. If the send queue for  $\mathbf{pk}_i$  is not empty, let  $m_i$  be the first message in the queue and  $\mathbf{sk}_q$  the corresponding query key. Send the message by calling **PH.SendRaw**( $\mathbf{sk}_q, \mathbf{pk}_i, m_i$ ) and remove  $m_i$  from the queue. Otherwise, let  $\mathbf{sk}_c$  be the most recent private cover key and  $m_i$  be a dummy message. Send the message by calling **PH.SendRaw**( $\mathbf{sk}_c, \mathbf{pk}_i, m_i$ ).
  3. Repeat.
- *Receiving cover messages.* For each of the non-expired cover keys  $\mathbf{pk}'_c$  on the bulletin board, call the process  $m \leftarrow \text{PH.RecvProcess}(\mathbf{sk}_R, \mathbf{pk}'_c)$ . If  $m$  is a real message (see Section 2.5.2) forward the message to DATASHARE, otherwise discard. Repeat.

This process stops when the user goes offline, and **PH.RecvProcess** processes started by **PH.Cover** are canceled.

**Protocol 5** (**PH.HiddenSend**( $\mathbf{sk}_S, \mathbf{pk}_R, m$ )). To send a message  $m$  to recipient  $R$  with public key  $\mathbf{pk}_R$ , sender  $S$  with private key  $\mathbf{sk}_S$  places  $m$  in the send queue for  $\mathbf{pk}_R$ .

### 2.4.2 Messaging service privacy

We first define unobservability then prove that conversation messages sent using `PH.HiddenSend` are unobservable.

**Definition 1** (Unobservability). A conversation message is unobservable if all PPT adversaries have a negligible advantage in distinguishing a scenario in which the sender  $S$  sends a conversation message to the receiver  $R$ , from a scenario where  $S$  does not send a conversation message to  $R$ .

**Theorem 2.** Messages sent using `PH.HiddenSend` are unobservable towards any adversary that controls the communication server but does not control the sender or the receiver, assuming the receiver awaits both conversation and cover messages. This statement is also true when the adversary can break the network anonymity Tor provides.

*Proof.* To show that conversation messages are unobservable, we must prove that the following two scenarios are indistinguishable: the scenario in which the sender sends a conversation message (sent by `PH.Cover` after a conversation message has been queued using `PH.HiddenSend`), and the scenario in which the sender sends a cover message (sent by `PH.Cover` when no conversation message has been queued). The intuition behind this proof is that the conversation and cover messages are indistinguishable: (1) both are encrypted so that the adversary cannot distinguish them based on content; and (2) conversation messages replace cover messages, so they are sent using the same schedule.

All messages go through the pigeonhole. For each message, the adversary observes (1) the pigeonhole address, (2) the content, (3) the length, (4) the timestamps at which the message was posted and retrieved, and – in the worst case scenario in which the adversary can break the anonymity Tor provides – (5) the sender and the receiver.

The content and pigeonhole address of messages are cryptographically indistinguishable. Senders and receivers compute rendezvous mailbox addresses by using a Diffie-Hellman key exchange based on either the query public key and the owner’s public key (when the message is a conversation messages) or the sender and receiver’s cover keys (when the message is a cover message). As the adversary does not control the sender or the receiver, it does not know the corresponding private keys in either scenario. Under the decisional Diffie-Hellman assumption, the adversary cannot distinguish between mailbox addresses for conversation messages and mailbox addresses for cover messages. Under the same DH assumption, the adversary cannot learn the symmetric

key  $k$  that is used to encrypt the message either. Moreover, all messages are padded to a fixed length of  $m\text{len}$ . Hence, the adversary cannot distinguish between the two situations based on message content or length. As a result, all messages sent between sender  $S$  and receiver  $R$  are indistinguishable to the adversary on the cryptographic layer.

We now show that the post and retrieve times of the messages are also independent of whether the message is a cover message or a conversation message:

*Sender.* The “cover keys” and “sending messages” processes of **PH.Cover** are, by design, independent of whether a conversation message should be sent or not. The sender sends (real or cover) messages to the recipient at a constant rate  $\lambda_c$ . The send times are independent of whether the sender has a real message for the receiver.

*Receiver.* The receiver is listening to both conversation and cover messages from the sender. As soon as it a new message notification arrives, **PH.Recv-Process** will retrieve this message. Therefore, the retrieval time does *not* depend on the type of message.  $\square$

As a corollary of the unobservability proof, we have the following theorem.

**Theorem 3.** The pigeonhole protects the secrecy of messages from non-participants including the communication server.

To hide their (network) identities from the communication server, users of DATASHARE communicate with the communication server via Tor. Sender anonymity hides queriers’ identities from document owners, and receiver anonymity hides document owners’ identities from queriers. Using Tor ensures these properties, even when journalists collude with the communication server. Formally, we define sender and receiver anonymity as follows:

**Definition 2** (Sender anonymity). A communication system provides sender anonymity if any PPT adversary has a negligible advantage in guessing the sender of a message.

**Definition 3** (Receiver anonymity). A communication system provides receiver anonymity if any PPT adversary has a negligible advantage in guessing the receiver of a message.

**Theorem 4.** Assuming that Tor provides sender and receiver anonymity with respect to the communication server, the communication system pro-



vides sender and receiver anonymity at the network layer against adversaries who control the communication server and a subset of journalists.

*Proof.* All messages go through the communication system and journalists never directly connect with each other. We study separately the anonymity provided by the bulletin board and the pigeonhole.

To publish an encrypted message (the query) to the bulletin board, senders run the **BB.broadcast** protocol over a fresh Tor circuit. Sender anonymity is guaranteed by Tor. The bulletin board broadcasts all messages to all journalists. As these messages do not have an intended receiver, receiver anonymity is not relevant.

Both senders and receivers use fresh Tor circuits when communicating with the communication servers. This ensures that communications are unlinkable at the network layer, and that the adversary cannot identify the journalist from network artifacts. As shown in the unobservability proof, the pigeonhole cannot distinguish senders' or receivers' given addresses or encrypted messages.  $\square$

This theorem only addresses the anonymity at the network layer. We discuss anonymity at the application layer, i.e., based on the content of messages, in Section 2.5.3.

Tor does not provide sender or receiver anonymity against global passive adversaries. To protect against global passive adversaries, DATASHARE will migrate to stronger network layer anonymity systems (e.g., the Nym system [28], based on Loopix [29])

### 2.4.3 Cost evaluation

To guarantee unobservability, we schedule the traffic according to a Poisson distribution. However, such strong protection comes at a cost [30]: Regardless of whether they have zero, one, or many conversations, every journalist sends messages at a rate  $\lambda_c$  to the other  $N$  journalists, i.e., sends  $\lambda_c N$  messages per day. Consequently, every journalist also receives  $\lambda_c N$  messages a day.

Figure 2.4, left, illustrates the trade-off between bandwidth overhead and latency for a given cover traffic rate. When journalists send few messages a day, the bandwidth requirements are very low. For instance, setting  $\lambda_c$  to be 4

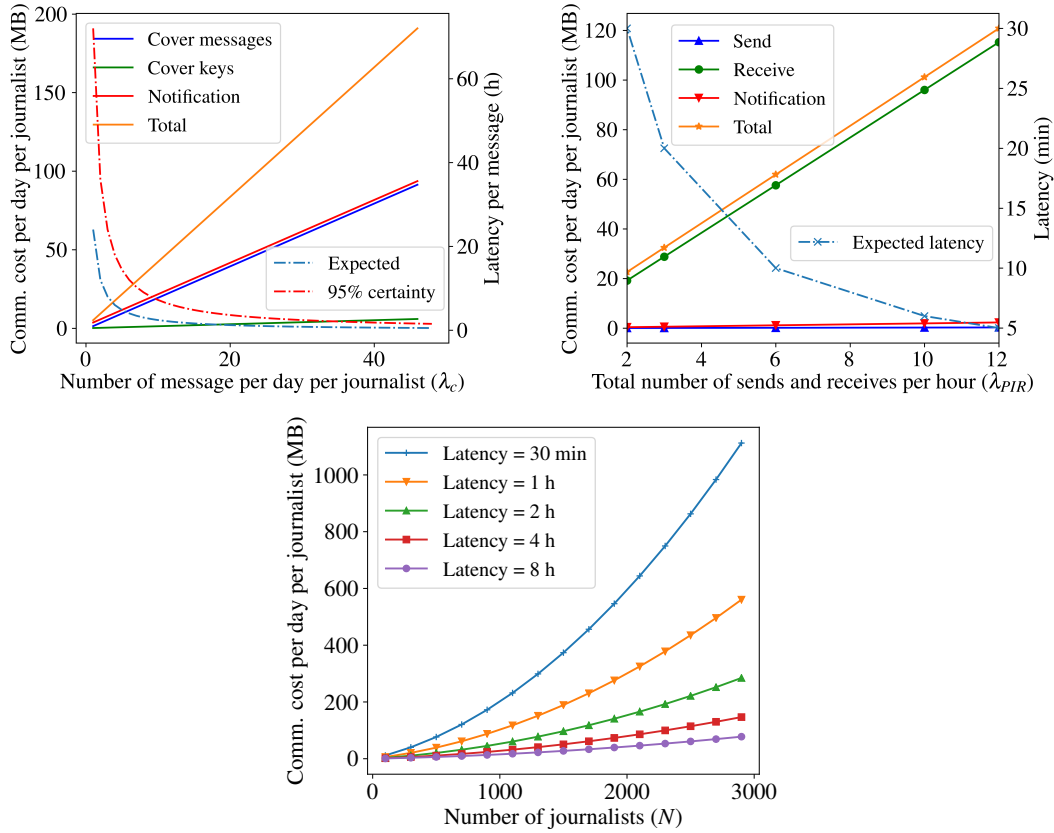


Figure 2.4: Left: bandwidth (left axis) and latency (right axis) for running the communication system (CS) with 1000 journalist for given rate  $\lambda_c$ . Bottom: varying the number of journalists and average latency in the CS. Right: bandwidth (left axis) and latency (right axis) for running the PIR system with 1000 journalists.

messages per day requires every journalist to use 16.5 MB per day, including the sending of notifications and the updating of cover keys. For these messages to be unobservable, however, journalists have to wait on average six hours between messages (less than 18 hours in 95% of the cases). If journalists require higher throughput they must consume more bandwidth. For example, setting  $\lambda_c = 48$  messages a day ensures that messages are sent within half an hour on average (and within 90 minutes with probability 95%). Storing messages from the last seven days on the pigeonhole for 1000 journalists and send rate of  $\lambda_c = 48$  requires 390 GB, which is manageable for a server.

The latency we report in Figure 2.4 assumes that journalists are online. If they disconnect from the system before a message is sent, journalists must, after coming online again, first upload a new cover key then draw a new sample from  $\text{Exp}(\lambda_c)$  to decide when to send their message. We propose to set the update latency  $\lambda_k$  to  $\lambda_c/4$ , so that the initial latency is at most 25% more than

the latency under normal circumstances.

For the current size of the population that will use DATASHARE, 250 journalists (see Section 2.2.1), the bandwidth can be kept reasonable at the cost of latency. However, as journalists send cover traffic to everyone, the bandwidth cost increases quadratically with the size of the population, and becomes pretty heavy after reaching 2000 journalists, see Figure 2.4, center.

*An alternative construction.* If the traffic requirements become too heavy for the organization members, bandwidth can be reduced by increasing the computation cost at the pigeonhole server. Instead of using cover traffic to *all* journalists to hide the mailboxes that contain real messages, journalists can retrieve messages using computational private information retrieval (PIR) [10, 24].

In this approach, senders send cover messages at a rate  $\lambda_{\text{PIR}}$ , *independent* of the number of journalists, to random mailboxes. When they have a real message, they send it instead of a cover message. They use the same rate to retrieve messages using PIR. This approach hides which messages are getting retrieved from the pigeonhole and breaks the link between the send and receive time. As a result, the server’s observation of the system is independent of whether journalists send a real message or not.

We illustrate the trade-off associated with this approach in Figure 2.4, right. We use SealPIR [24] to retrieve cover and conversation messages. Responding to a PIR request in a scenario of 1000 journalists and a send rate of 6 messages per hour takes 12 seconds. Therefore, we assume a server with 24 cores (approx 1300 USD/month in AWS) can handle this scenario. We see that this approach enables the system to send conversation messages at a higher rate and a lower cost. For example, sending 6 messages per hour (144 messages a day) requires around 59 MB. However, as opposed to the Poisson cover approach described in the previous section, this rate limits the total number of messages per day *regardless of recipient*. As a result, depending on the number of receivers journalists want to communicate with on average, one or the other method could be more advantageous.

## 2.5 The DATASHARE system

We now present DATASHARE, an asynchronous decentralized peer-to-peer document search engine. DATASHARE combines the multi-set private set inter-

section protocol (Section 2.3), the privacy-preserving communication system (Section 2.4), and an anonymous authentication mechanism.

### 2.5.1 Preliminaries

*Processing documents.* The primary interests of investigative journalists are named entities, such as people, locations, and organizations (see Section 2.2.1). ICIJ has already developed a tool [1] that uses natural language processing to extract named entities from documents. After the extraction, the tool transforms named entities into a canonical form to reduce the impact of spelling variation in names. We employ this tool to canonicalize queries. An advantage of using this tool over simply listing all words in a document is that it reduces the number of keywords per document: the majority of documents have less than 100 named entities.

*Search.* DATASHARE uses the MS-PSI protocol as a pairwise search primitive between journalists. The querier acts as MS-PSI client, and the client’s set represents the querier’s search keywords. The document owners act as MS-PSI servers, where the server’s  $N$  sets represent the keywords in each of the owner’s  $N$  documents. Each document owner has their own *different* corpus and secret key. We say a document is a match if it contains *all* query keywords (i.e., the conjunction of the query keywords, see Section 2.2.1). MS-PSI speeds up the computation and reduces the communication cost by a factor of  $N$  compared to the naive approach of running one PSI protocol per document.

*Authenticating journalists.* Only authorized journalists, such as members of the organization or collaborators, are allowed to make queries and send conversation messages. DATASHARE’s authentication mechanism operates in epochs. In each epoch journalists obtain a limited number of anonymous tokens. Tokens can be used only once, which limits the number of queries that journalists can make per epoch. Compromised journalists, therefore, can extract limited information from the system by making search queries. We considered using identity-escrow mechanisms to mitigate damage by misbehaving journalists but in agreement with the organization, we decided against this approach as such mechanisms could too easily be abused to identify honest journalists.

Recall from Section 2.2.1 that journalists trust the organization as an authority for membership and already have means to authenticate themselves to the organization. Therefore, the organization is the natural design choice for issuing anonymous tokens. We note that, even if the organization is com-

promised, it can do limited damage as it cannot link queries or conversations to journalists (because of token anonymity). However, it can ignore the rate limit. This would enable malicious queriers to extract more information than allowed. To mitigate this risk, DATASHARE could also work with several token issuers and require a threshold of valid tokens.

For the epoch duration, ICIJ proposes one month to provide a good balance between protection and ease of key management. Rate-limits are flexible. The organization can decide to provide additional one-time-use tokens to journalists who can motivate their need for extra tokens. Although this reveals to the organization which journalists are more active, it does not reveal what they use the tokens for.

*Instantiation.* Tokens take the form of a blind signature on an ephemeral signing key. We use Abe’s blind signature (BS) scheme [31]. The organization runs  $\text{BS.Setup}(1^n)$  to generate a signing key  $\text{msk}$  and a public verification key  $\text{mpk}$ . To sign an ephemeral key  $\text{pk}_T$ , the journalist and the organization jointly run the  $\text{BS.Sign}()$  protocol. The user takes as private input the key  $\text{pk}_T$ , and the organization takes as private input its signing key  $\text{msk}$ . The user obtains a signature  $C$  on  $\text{pk}_T$ . The verification algorithm  $\text{BS.Verify}(\text{mpk}, C, \text{pk}_T)$  returns  $\top$  if  $C$  is a valid for  $\text{pk}_T$  and  $\perp$  otherwise. These blind signatures are anonymous. The blindness property of  $\text{BS}$  ensures that the signer cannot link the signature  $C$  or the key  $\text{pk}_T$  to the journalist that ran the corresponding signing protocol.

Let  $\text{sk}_T$  be the private key corresponding to  $\text{pk}_T$ . We call the tuple  $T = (\text{sk}_T, C)$  an authentication token. Journalists use tokens to authenticate themselves before issuing a query or sending a message. To authenticate themselves, journalists create a signature  $\sigma$  on the message using  $\text{sk}_T$  and append the signature  $\sigma$  and blind signature  $C$  on  $\text{pk}_T$ . Non-authenticated messages and queries are dropped by other journalists.

Anonymous authentication with rate limiting could have been instantiated alternatively with  $n$ -times anonymous credentials [32], single show anonymous credentials [33,34], or regular anonymous credentials [35,36] made single-show. We opted for the simplest approach.

*Cuckoo filter.* DATASHARE uses cuckoo filters [27] to represent tag collections in a space-efficient manner. The space efficiency comes at the price of having false positives when answering membership queries. The false negative ratio is always zero. The false positive ratio is a parameter chosen when instantiating the filter. Depending on the configuration, a cuckoo filter can compress a set to less than two bytes per element regardless of the elements’ original size.

Users call  $\text{CF.compress}(S, \text{params})$  to compute a cuckoo filter  $\text{CF}$  of the input set  $S$  using the parameters specified in  $\text{params}$ . Then,  $\text{CF.membership}(\text{CF}, x)$  returns  $\top$  if  $x$  was added to the cuckoo filter, and  $\perp$  otherwise. For convenience, we write  $\text{CF.intersection}(\text{CF}, S')$  to compute the intersection  $S' \cap S$  with the elements  $S$  contained in the cuckoo filter. The function  $\text{CF.intersection}$  can be implemented by running  $\text{CF.membership}$  on each element of  $S'$ .

### 2.5.2 DATASHARE protocols and design

The journalists' organization sets up the DATASHARE system by running **SystemSetup** (Protocol 6). Thereafter, journalists join DATASHARE by running **JournalistSetup** (Protocol 7). Journalists periodically call **GetToken** (Protocol 8) to get new authentication tokens, and **Publish** (Protocol 9) to make their documents searchable. DATASHARE does not support multiple devices, and the software running on journalists' machines automatically handles key management without requiring human interaction. If a journalist's key is compromised, she contacts the organization to revoke it.

**Protocol 6 (SystemSetup)**. The journalist organization runs **SystemSetup** to set up the DATASHARE system:

1. The organization generates a cyclic group  $\mathbb{G}$  of prime order  $p$  with generator  $g$ , and hash functions  $\text{H} : \{0, 1\}^* \rightarrow \{0, 1\}^n$  and  $\hat{\text{H}} : \{0, 1\}^* \rightarrow \mathbb{G}$  for use in the MS-PSI protocol. It selects parameters  $\text{params}$  for the cuckoo filter and sets the maximum number of query keywords  $\text{lim}$  (we use  $\text{lim} = 10$ ). The organization publishes these.
2. The organization sets up a token issuer by running  $(\text{msk}, \text{mpk}) = \text{BS.Setup}(1^n)$  and publishes  $\text{mpk}$ .
3. The organization sets up a communication server, which provides a bulletin board and a pigeonhole.

**Protocol 7 (JournalistSetup)**. Journalists run **JournalistSetup** to join the network: The journalist authenticates to the organization and registers for DATASHARE.

**Protocol 8 (GetToken)**. Journalists run **GetToken** to obtain one-time-use authentication tokens from the organization.

1. The journalist  $J$  connects to the organization and authenticates herself. The organization verifies that  $J$  is allowed to obtain an extra token and, if not, aborts.
2. The journalist generates an ephemeral signing key  $(\text{sk}_T, \text{pk}_T)$ ; runs the  $\text{BS.Sign}()$  protocol with the organization to obtain the organization's sig-

nature  $C$  on the message  $\mathbf{pk}_T$  (without the organization learning  $\mathbf{pk}_T$ ); and stores the token  $T = (\mathbf{sk}_T, C)$ .

To obtain tokens for the new epoch, journalists repeatedly run the **GetToken** protocol at the beginning of each epoch.

**Protocol 9 (Publish).** Journalists run **Publish** to make their documents searchable. **Publish** takes as input a token  $T = (\mathbf{sk}_T, C)$  and a set  $\mathbf{Docs} = \{d_1, \dots, d_N\}$  of  $N$  documents such that each document  $d_i$  is a set of keywords in  $\{0, 1\}^*$ . This protocol includes the pre-computation phase of MS-PSI.

1. The journalist chooses a secret key  $s \xleftarrow{\$} \mathbb{Z}_p$  and computes her tag collection for the MS-PSI protocol as

$$\mathbf{TC} = \{\mathbf{H}(i \parallel \hat{H}(y)^s) \mid i \in [N], y \in d_i\},$$

and compresses it into a cuckoo filter  $\mathbf{CF} = \mathbf{CF.compress}(\mathbf{TC}, \mathbf{params})$ .

2. The journalist generates a long-term pseudonym  $\mathbf{nym}$ , and a medium-term contact key pair  $(\mathbf{sk}, \mathbf{pk})$ .
3. The journalist encodes her pseudonym  $\mathbf{nym}$ , public key  $\mathbf{pk}$ , compressed tag collection  $\mathbf{CF}$ , and the number of documents  $N$  as her public record

$$\mathbf{Rec} = (\mathbf{nym}, \mathbf{pk}, \mathbf{CF}, N).$$

4. The journalist signs her record  $\sigma = \mathbf{Sign}(\mathbf{sk}_T, \mathbf{Rec})$  and runs  $\mathbf{BB.broadcast}(\mathbf{Rec} \parallel \sigma \parallel \mathbf{pk}_T \parallel C)$  to publish it.

DATASHARE automatically rotates (e.g., every week) the medium-term contact key of journalists  $(\mathbf{sk}, \mathbf{pk})$  to ensure forward secrecy. This prevents that an attacker that obtains a journalist's medium-term private key can recompute the mailbox addresses and encryption key of messages sent and received by the compromised journalist.

Journalists retrieve all public records from the bulletin board. They run  $\mathbf{Verify}(\mathbf{pk}_T, \sigma, \mathbf{Rec})$  to verify the records against the ephemeral signing key, check that they have not seen  $\mathbf{pk}_T$  before to enforce the one-time use, and run  $\mathbf{BS.Verify}(\mathbf{pk}_T, C, \mathbf{mpk})$  to validate the blind signature. Journalists discard invalid records.

DATASHARE incorporates MS-PSI into its protocols to enable document search. Querying works as follows (Fig. 2.5): (1) The querier posts a query together with a fresh key  $\mathbf{pk}_q$  to the bulletin board (Protocol 10); (2) Document owners retrieve these queries from the bulletin board (2a), they compute the

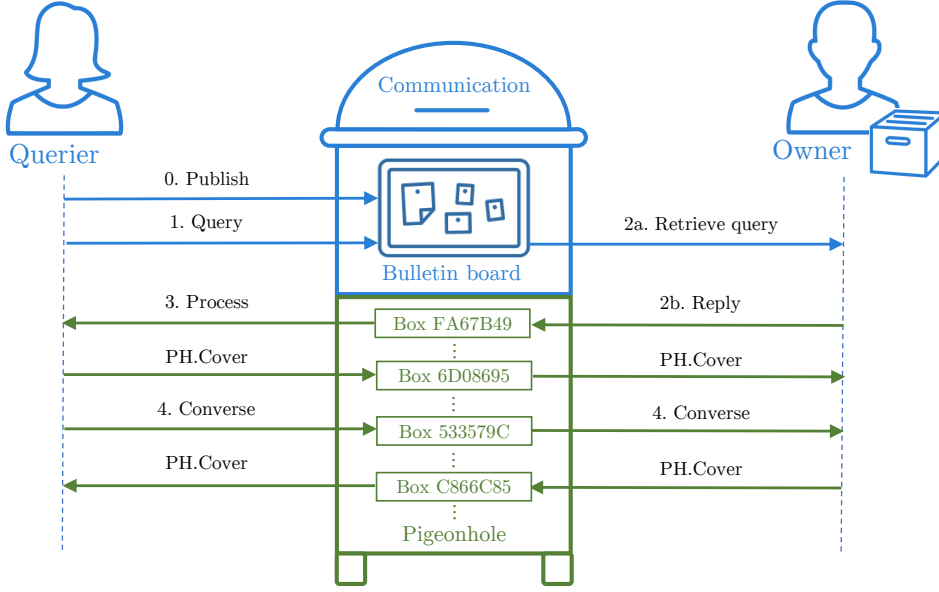


Figure 2.5: An overview of DATASHARE protocols.

reply address, and they send the reply to a pigeonhole mailbox (2b, see Protocol 11); (3) The querier monitors the reply addresses for all document owners, retrieves the replies, and computes the intersection to determine matches (Protocol 12).

**Protocol 10 (Query).** Queriers run **Query** to search for keywords  $X$ . The protocol takes as input a token  $T = (\mathbf{sk}_T, C)$ .

1. The querier generates a key pair  $(\mathbf{sk}_q, \mathbf{pk}_q)$  for the query and pads  $X$  to  $\text{lim}$  keywords by adding random elements.
2. As in the MS-PSI protocol, the querier picks a fresh blinding factor  $c \leftarrow^{\$} \mathbb{Z}_p$ , and computes:

$$Q = \{\hat{H}(x)^c \mid x \in X\}.$$

3. The querier signs the query  $Q$  and her public key  $\mathbf{pk}_q$  as  $\sigma = \text{Sign}(\mathbf{sk}_T, Q \parallel \mathbf{pk}_q)$ , and broadcasts the query  $Q$ , public key  $\mathbf{pk}_q$ , signature  $\sigma$ , ephemeral token key  $\mathbf{pk}_T$ , and token  $C$  by running  $\text{BB.broadcast}(Q \parallel \mathbf{pk}_q \parallel \sigma \parallel \mathbf{pk}_T \parallel C)$ .

Recall that MS-PSI perfectly hides the keywords inside queries. As a result, these queries can be safely broadcasted.

**Protocol 11 (Reply).** Document owners run **Reply** to answer a query  $(Q, \mathbf{pk}_q, \sigma, \mathbf{pk}_T, C)$  retrieved from the bulletin board.

1. The owner verifies the query by checking  $\text{Verify}(\mathbf{pk}_T, \sigma, Q \parallel \mathbf{pk}_q)$ ,  $\text{BS.Verify}(\mathbf{pk}_T, C)$ .



- $\text{mpk}, C, \text{pk}_T$ ), and that she did not see  $\text{pk}_T$  before. If any verification fails, she aborts.
2. The owner uses her secret key  $s$  to compute the MS-PSI response  $R = \{\tilde{x}^s \mid \tilde{x} \in Q\}$  to the query.
  3. Let  $\text{sk}$  be the owner's medium-term private key. She runs  $\text{PH.SendRaw}(\text{sk}, \text{pk}_q, R)$  to post the result to the pigeonhole, and starts the process  $\text{PH.RecvProcess}(\text{sk}, \text{pk}_q)$  to await conversation messages from the querier (see **Converse** below).

**Protocol 12 (Process).** Queriers run the **Process** protocol for every journalist  $J$  with record  $\text{Rec} = (\text{nym}, \text{pk}, \text{CF}, N)$  to retrieve and process responses to their query  $(X, \text{sk}_q, c)$ , where  $X$  is the unpadding set of query keywords.

1. The querier runs the asynchronous protocol  $R \leftarrow \text{PH.RecvProcess}(\text{sk}_q, \text{pk})$  to get the new response.
2. Similar to MS-PSI, the querier computes the size of the intersection  $I_i$  for each document  $d_i$ ,  $1 \leq i \leq N$ , as

$$I_i = \left| \text{CF.intersection} \left( \text{CF}, \{H(i \parallel \hat{x}^{c^{-1}}) \mid \hat{x} \in R\} \right) \right|.$$

3. Let  $q = |X|$  be the number of query keywords. The querier learns that the owner  $\text{nym}$  has  $t = |\{i \mid I_i = q\}|$  matching documents.

After finding a match, the querier and owner can converse via the pigeonhole to discuss the sharing of documents using the **Converse** protocol.

**Protocol 13 (Converse).** Let  $(\text{sk}_q, \text{pk}_q)$  be the query's key pair, and  $(\text{sk}_O, \text{pk}_O)$  the owner's medium-term key pair at the time of sending the query.

- The querier sends messages  $m$  to the owner by calling  $\text{PH.HiddenSend}(\text{sk}_q, \text{pk}_O, m)$ , and awaits replies by calling  $\text{PH.RecvProcess}(\text{sk}_q, \text{pk}_O)$ .
- The owner sends messages  $m$  to the querier by calling  $\text{PH.HiddenSend}(\text{sk}_O, \text{pk}_q, m)$ , and awaits replies by calling  $\text{PH.RecvProcess}(\text{sk}_O, \text{pk}_q)$ .
- After receiving a message, the receiving party calls  $\text{PH.RecvProcess}$  again, to await further messages.

Both the query's key  $\text{pk}_q$  and the owner's key  $\text{pk}_O$  are signed using a one-time-use token. Thus, querier and owner know they communicate with legitimate journalists.

### 2.5.3 DATASHARE security analysis

DATASHARE provides the following guarantees:

**Protecting queries.** The requirements established in Section 2.2.1 state that DATASHARE must protect the searched keywords and identity of the querier from adversaries that control the communication server and a subset of document owners. The **Query** protocol, which handles sending queries, is based on MS-PSI. It represents searched keywords as the client’s set in MS-PSI. Theorem 1 states that MS-PSI perfectly hides the client’s set from malicious servers. Therefore, DATASHARE protects the content of queries from owners.

DATASHARE does not reveal any information about the identity of queriers at the network and application layer. Theorem 4 ensures that the communication system provides sender and receiver anonymity and protects the querier’s identity at the network layer. At the application layer, the querier sends  $(Q \parallel \mathbf{pk}_q \parallel \sigma \parallel \mathbf{pk}_T \parallel C)$  as part of the **Query** protocol to the bulletin board. The values  $\sigma$ ,  $\mathbf{pk}_T$ , and  $C$  form an anonymous authentication token based on Abe’s blind signature [31]. Anonymous tokens are independent of the querier’s identity. The value  $\mathbf{pk}_q$  is an ephemeral public key, and  $Q$  is a MS-PSI query which uses an ephemeral secret for the client. Hence, both  $\mathbf{pk}_q$  and  $Q$  are independent of the querier’s identity too. Therefore, the content of the query does not leak the querier’s identity at the application layer.

**Protecting conversations.** According to the requirements stated in Section 2.2.1, DATASHARE must protect (1) the content, and (2) the identity of participants in a conversation from non-participants. (3) DATASHARE must protect the identities of journalists (who are in a conversation) from each other.

First, DATASHARE protects the content of conversation messages from non-participants: Theorem 3 proves that only the sender and receiver can read their conversation messages.

Second, DATASHARE protects the identity of participants in a conversation from non-participants. Theorem 2 proves that communication is unobservable, as long as participants are awaiting both conversation and cover messages. DATASHARE enforces the conditions by construction. Immediately after answering a query (see **Reply**, Protocol 11), the owner starts **PH.RecvProcess** to listen for messages from the querier. Similarly, the querier starts to listen for conversation messages from the owner right after sending him a conversation

message (see **Converse**, Protocol 13). Moreover, the “cover keys” and “receiving cover messages” processes in the **PH.Cover** protocol ensure that all journalists broadcast their cover keys and start **PH.RecvProcess** after receiving a new cover key. Therefore, DATASHARE satisfies the requirements on the communication systems in Theorem 2. As a result, non-participants cannot detect whether users communicate. Thus, protecting the identity of participants as required.

Third, DATASHARE aims to hide the identity of journalists from their counterparts in a conversation. Theorem 4 shows that the communication system does not reveal the identity of journalists at the network layer. DATASHARE also ensures protection at the cryptographic layer: as we argued above, queries are unlinkable. However, DATASHARE cannot provide unconditional protection for conversations. Queriers or document owners could identify themselves as part of the conversation. Moreover, by their very nature, messages in a conversation are linkable. Also, as we discuss below, insiders can use extra information to identify communication partners.

**Protecting document collections.** *Any* functional search system inherently reveals information about the documents that it makes available for search: To be useful it must return at least one bit of information. An attacker can learn more information by making additional queries. We show that DATASHARE provides comparable document owner’s privacy to that of ideal theoretical search systems. We use as a security metric the number of queries an attacker has to make to achieve each of the following goals:

*Document recovery.* Given a target set of keywords (e.g. “XKeyscore” and “Snowden”), an adversary aims to learn which of these target keywords are contained in a document for which some keywords are already known.

*Corpus extraction.* Given a set of target keywords, an adversary aims to learn which documents in a corpus contain which target keywords. If the target set contains all possible keywords, the adversary effectively recovers the full corpus.

Any functional search system is also susceptible to confirmation attacks. An adversary interested in knowing whether a document in a collection contains a keyword (e.g., “XKeyscore” to learn whether the collection contains the Snowden documents) can always directly query for the keyword of interest.

We compare the number of queries an adversary needs to extract the corpus or recover a document in the following three settings: when using

Table 2.3: Privacy and scalability of the hypothetical and DATASHARE’s MS-PSI based search protocols. The table shows the number of queries necessary to achieve document recovery and corpus extraction, when interacting with a corpus of  $d$  documents over a set  $n$  keywords. The document extraction bound for the 1-bit system extracts up to uniqueness bound  $u$ .

	Doc	Extract	Scale
1-bit	$n$	$n^u + nd$	--
#doc	$n$	$nd$	-
<b>DATASHARE</b>	<b><math>n/\text{lim}</math></b>	<b><math>n/\text{lim}</math></b>	<b>+</b>

DATASHARE, and when using one of two hypothetical systems. The first hypothetical system, called *1-bit*, is an ideal search system. In this system, given a query, the querier learns *only* one bit of information: whether the owner has a matching document. The second hypothetical system, called *#doc*, is an ideal search system where the querier learns how many matching documents the owner has.

Table 2.3 compares these hypothetical systems with DATASHARE’s use of MS-PSI, where  $d$  is the number of documents and  $n$  the number of relevant keywords. We show that extracting all the keywords from a document requires at most  $n$  queries in the 1-bit and #docs search systems in Appendices A.2.1 and A.2.2.

Extracting the full corpus using the 1-bit search system is not always possible. Let the *uniqueness number*  $u_D$  be the smallest number of keywords that uniquely identify a document  $D$ . If  $D$  is a strict subset of another document  $D'$ , the document cannot be uniquely identified, and we set  $u_D = \infty$ . However, as corpora are small, we expect that most documents can be identified by a few well-chosen keywords, resulting in small uniqueness numbers.

In Appendix A.2.1, we show that extracting all documents with uniqueness number less or equal to  $u$  takes  $O(n^u + nd)$  queries in the 1-bit search system. In Appendix A.2.2 we show that extracting all documents (regardless of uniqueness number) takes  $O(nd)$  queries in the #doc search system.

In DATASHARE, we limit MS-PSI queries to  $\text{lim}$  keywords per query. Hence, any document extraction attack must make at least  $n/\text{lim}$  queries to ensure all keywords are queried at least once. In fact, this bound is tight for both document recovery and corpus extraction for MS-PSI: By making  $n/\text{lim}$  queries with  $\text{lim}$  keywords each, the attacker learns which keywords are contained in which documents.

In summary, DATASHARE offers similar protection against corpus extraction as the #doc ideal system. For document recovery, not even the ideal 1-bit-search system offers much better protection. At the same time, MS-PSI is much more efficient than their ideal counterparts.

**Internal adversaries.** We now discuss how an adversary may use auxiliary information about a journalist’s behavior or corpus to gain an advantage in identifying the journalist. Some of these attacks are inherent to all systems that provide search or messaging capabilities. These attacks, however, *do not permit* the adversary to extract additional information from journalists’ corpora.

*Intersection attacks.* A malicious sender (respectively, receiver) who has access to the online/offline status of journalists can use this information to reduce the anonymity set of the receiver (respectively, sender) to only those users that are online. As more messages are exchanged, this anonymity set becomes unavoidably smaller [37]. This attack is inherent to all low-delay asynchronous messaging systems, including the one provided by the communication server. In the context of DATASHARE, we note that once document owners and queriers are having a conversation, it is likely that they reveal their identity to each other. Yet, we stress that preserving anonymity and, in general, that minimizing the digital traces left by the journalists in the system is very important to reducing the risk that journalists become profitable targets for subpoenas or hacking attempts.

*Stylometry.* A malicious receiver can use stylometry, i.e., linguistic style, to guess the identity of the sender of a message. The effectiveness of this attack depends on the volume of conversation [38, 39]. This attack is inherent to all messaging systems, as revealing the content of the messages is required to provide utility.

*Partial knowledge of corpus.* Adversaries who have prior knowledge about a journalist’s corpus can use this knowledge to identify this journalist in the system. However, due to MS-PSI’s privacy property (see Theorem 1), learning *more* about the documents in this journalist’s corpus requires making search queries.

In particular, if an adversary convinces a journalist to add a document with a unique keyword pattern to his corpus, then the adversary can detect this journalist’s corpus by searching for the pattern. DATASHARE cannot prevent

such out-of-band watermarking. However, the adversary still needs to make further queries to learn anything about non-watermarked documents in the collection.

**Non-goals.** Finally, we discuss security properties that are not required in DATASHARE.

*Query unlinkability.* DATASHARE does not necessarily hide which queries are made by the same querier. Even though anonymity is ensured at the network and application layers, queriers that have made multiple queries may retrieve responses for all these queries in quick succession after coming online. Document owners know the corresponding query of their messages, and if they collude with the communication server, then they can infer that the same person made these queries. As no adversary can learn any information about the queries themselves, we consider this leakage to be irrelevant.

*Owner Unlinkability.* DATASHARE also reveals *which* pseudonymous document owner created a MS-PSI response, making responses linkable. DATASHARE cannot provide unlinkability for document owners when using MS-PSI. Although MS-PSI itself could be modified to work without knowing the document owner’s pseudonym, an adversary could simply repeat a specific rare keyword (for example, “one-word-to-link-them-all”) and identify the document owners based on the corresponding pretag that they produce for the rare keyword. We believe that revealing the document owner’s pseudonym is an acceptable leakage for the performance gain it provides.

#### 2.5.4 Cost evaluation

At the time of writing, ICIJ has implemented the local search and indexing component of DATASHARE [1]. In addition, we have implemented a Python prototype of the cryptographic building blocks underlying search (Section 2.3) and authentication (Section 2.5.1).<sup>1</sup> We did not implement the messaging service (Section 2.4), as it relies on standard building blocks and cryptographic operations.

To agree on the final configuration of the system, we are currently running a user study among the organization members. The goal is to familiarize journalists with a type of search and messaging system that is different than those

---

<sup>1</sup>The code is open source and available at: <https://github.com/spring-epfl/datashare-network-crypto>

they typically use in their daily activities (Google and email or instant messaging, respectively), as well as with the threat model within which DATASHARE provides protection. We recall that DATASHARE hides all key management and cryptography from the users, hence we do not study those aspects.

In this section, we evaluate the performance of the cryptographic operations involved in search and authentication. Our prototype uses the `petlib` [40] binding to OpenSSL on the fast NIST P-256 curve for the elliptic curve cryptography in MS-PSI. We implement the Cuckoo filter using `cuckoopy` [41]. We ran all experiments on an Intel i3-8100 processor running at 3.60GHz using a single core. We note that operations could be easily parallelized to improve performance.

We focus our evaluation on the computational cost and bandwidth cost of the authentication and search primitives to ensure that DATASHARE fulfills the requirements in Section 2.2.1 without journalists needing fast hardware or fast connections. When reporting bandwidth cost, we omit the overhead of the meta-protocol that carries messages between system parties. We do not consider any one-time setup cost or the standard cryptography used for messaging. We also do not measure network delay as the latency the Tor network introduces – around one second [42] – is negligible compared to the waiting time imposed by connection asynchrony; and it is orders of magnitude less than the journalists waiting limits (see Section 2.2.1).

We provide performance measurements for different system work loads. We consider the *base scenario* to be 1000 journalists, each of whom makes 1000 documents available for search. There is no requirement for the number of keywords per document or keywords per query. For a *conservative* estimate, we assume that each document contains 100 keywords, and that each query contains 10 keywords.

*Authenticating Journalists.* We implement the **BS** scheme using Abe’s blind signatures [31]. Running **BS.Sign** requires transferring 413 bytes and takes 0.32 ms and 0.62 ms, respectively, for the organization and the journalist. Each blind signature is 360 bytes, and verifying it using **BS.Verify** takes 0.4 ms. We include these costs in the respective protocols.

*Publishing Documents.* Data owners run **Publish** to make their documents searchable. For the base scenario, this *one-time* operation takes 14 seconds and results in a cuckoo filter of size 400 KB for a FPR of 0.004%. For a conservative estimation, we assume all keywords are different. When documents contain duplicate elements  $y$ , the precomputation can be amortized: the pretag  $\hat{H}(y)^s$

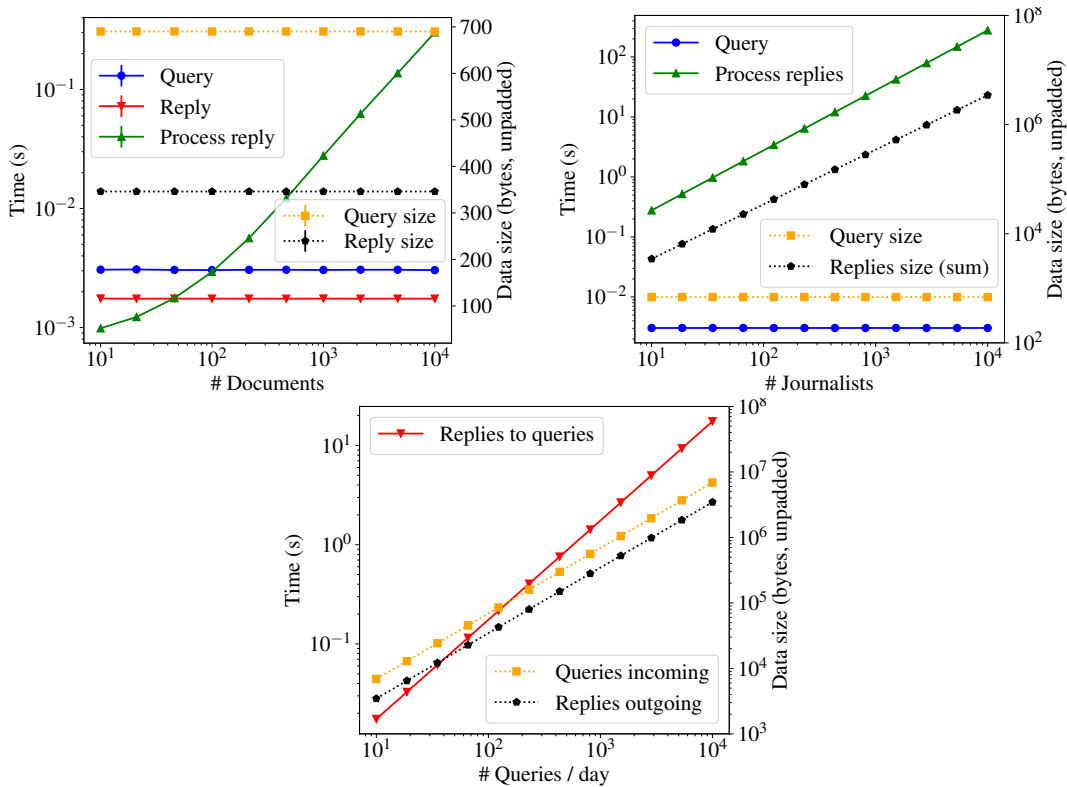


Figure 2.6: Time (left axis) and bandwidth (right axis, unpadded) for single query on one journalist (left), single query on all journalists (right), answering several queries (below).

has to be computed only once.

*Querying a Single Journalist.* Figure 2.6, left, shows the time and bandwidth required to issue one query on one collection, depending on the collection size. The querier constructs the query using **Query** and sends it to the document owner (the querier’s computation cost includes the cost of obtaining the one-time-use token using **GetToken**). The document owner responds using **Reply**. These operations are independent of the number of documents. The querier runs **Process** to retrieve the responses, and to compute the intersection of query and collection. This takes 27 ms in the base scenario. Bandwidth cost reflects the raw content size. But recall that, in practice, the messaging system pads messages to 1 KB.

*Querying All Journalists.* As expected, the processing time and bandwidth of **Query** are independent of the population size, whereas the cost of processing the responses grows linearly with the number of queried journalists (Figure 2.6, right). For the baseline scenario, processing all 999 responses takes about 27



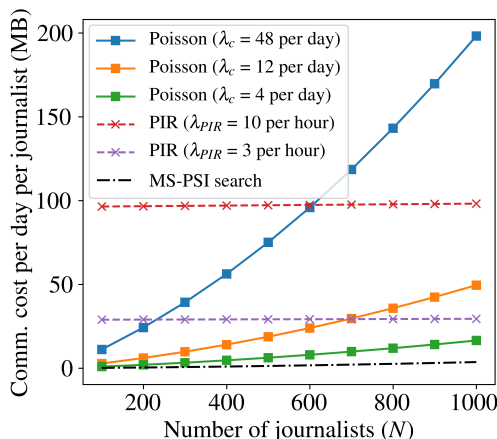


Figure 2.7: Communication cost for different communication strategies, depending on the number of journalists. We assume 1 query per journalist per day in the search component.

seconds *in total* and requires retrieving 1 MB of padded responses. We note that this cost is only paid by the querier, and does not impact the document owners (see below). Moreover, as replies are unlikely to arrive all at once, processing can be spread out over time; thus reducing the burden on the querier’s machine.

This computation assumes that each journalist has the same number of documents. In practice, this might not hold. However, as we see in Figure 2.6, left, as soon as collections have more than 50 documents the computation time grows linearly with the collection size. Hence, as long as journalists have collections with at least 50 documents, the measurements in Figure 2.6, right, are largely independent of how these documents are distributed among journalists.

*The Cost for Document Owners.* Document owners spend time and bandwidth to answer queries from other journalists. Figure 2.6, bottom, shows how these costs depend on the total number of queries an owner receives per day. Even when all journalists make 10 queries of 10 keywords each day (unlikely in practice) the *total* computation time for document owners is less than 20 seconds; and they send and receive less than 7 megabytes (10 MB when padded).

*Overall Cost of DATASHARE.* Finally, we plot in Figure 2.7 the *total* bandwidth a journalist needs per day to run DATASHARE, depending on the number of journalists in the system and the strategy implemented by the communication system. Regardless of the size of the system, the cost associated to hide communications dominates the cost stemming from searches. Regarding

the communication cost, as explained in Section 2.4.3, for small organizations Poisson-rate cover traffic provides a better trade-off with respect to throughput, but as more journalists join the system, the PIR-based system starts performing better.

## 2.6 Related work

Many PSI protocols [14, 16, 43, 44] differ from that of De Cristofaro et al. [25], but only in how they instantiate the oblivious pseudorandom functions (OPRFs). Our MS-PSI protocols can easily be adjusted to use alternative OPRFs to compute the pretags. As bandwidth is at a premium in our scenario, we base our MS-PSI protocols on the scheme of De Cristofaro et al. as it has the lowest communication cost.

The restrictions on computational power and bandwidth rule out many other PSI schemes. Protocols based on oblivious polynomial evaluation [45] have very high computational cost. Hash-based PSI protocols [12, 17, 46] have low computational cost, but require much communication. Finally, PSI protocols can be built from generic secure multi-party computation directly [11–13]. However, this approach also suffers from a high communication cost and requires more than one communication round.

Secure multi-party computation based PSI protocols can be extended to provide better privacy than MS-PSI: The underlying circuits can be extended to implement either the ideal 1-bit search or the #doc search system. However, their high communication and round complexity rule out their use in our document search system. Recently, Zhao and Chow proposed a threshold PSI protocol based on polynomial evaluation [47] that can implement the #doc search system (by setting the threshold equal to the number of keywords). But its communication and computation complexity rule it out.

A document search engine could also be implemented using private information retrieval (PIR): Queriers use PIR to privately query keywords in the document owner’s database. Computational PIR protocols [10, 24, 48] (IT-PIR protocols [8, 9] do not apply) place a high computational burden on the database owner. More importantly, PIR requires a fixed set of keywords, that cannot exist for the journalists’ use case. Keyword-based PIR approaches [18, 19] sidestep this issue, but instead require multiple communication rounds. Therefore, PIR cannot be used in our scenario.

Encrypted databases hide the queries of data owners from an untrusted database server [49–52]. Although DATASHARE could operate such a central encrypted database, this system would not be secure. On the one hand, if the encrypted database is used as a central service for all collections, then a collusion between a journalist and the database server would leak the entire database. This would violate document privacy. On the other hand, if each journalist operates a personal database, then collusion between the database server and the document owner (acting as the ‘data owner’ in the terminology used in the encrypted database literature) might leak search queries, as these systems are not designed to hide queries from a database server that colludes with the data owner. This would violate query privacy.

## 2.7 Future steps: better protection

We have introduced DATASHARE, a decentralized privacy-preserving search engine that enables journalists to find and request information held by their peers. DATASHARE has great potential to help journalists collaborate in uncovering cross-border crimes, corruption, or abuse of power.

Our collaboration with a large organization of investigative journalists (ICIJ) provided us with a novel set of requirements that, despite being deeply grounded in practicality, are rarely considered in academic publications. These requirements led us to design new building blocks that we optimized for security trade-offs different than previous work. We combined these building blocks into an efficient and low-risk decentralized search system.

Yet, DATASHARE’s protections are not perfect. Both the search primitive, and the availability of timestamps of actions in the system, leak information. At the time of writing, the high cost in bandwidth and/or computation of state-of-the-art techniques that could prevent this leakage – e.g., PIR to hide access patterns and efficient garbled circuits to implement one-bit search – precludes their deployment.

We hope that this chapter fosters new research that addresses these problems. We believe that the new set of requirements opens an interesting new design space with much potential to produce results that have a high impact, not only by helping investigative journalism to support democratic societies, but also in other domains.



# CHAPTER 3

## Private Collection Matching Protocols

This chapter is based on the following article:

Kasra EdalatNejad, Mathilde Raynal, Wouter Lueks, Carmela Troncoso: “Private Collection Matching Protocols”. Proceedings on Privacy Enhancing Technologies (PoPETs), 2023.

### 3.1 Introduction

In many scenarios, a server holds a collection of sets and clients wish to determine whether these server sets *match* their own set, while both client and server keep their privacy. We call these *Private Collection Matching (PCM)* problems. In this chapter, we study for the first time the requirements of PCM problems.

We identify the privacy and efficiency requirements of PCM problems by analyzing three real-world use cases: determining whether a pharmaceutical database contains compounds that are chemically similar to the client’s [53–55], determining whether an investigative journalist holds relevant documents [56] (or how many), and matching a user’s profile to items or other users in mobile apps [57–59]. We find that PCM problems have three common characteristics: (1) Clients want to compare their *one* set with *all* sets at the server. (2) Clients do not need per-server set results, only an aggregated output (e.g., whether any server set matches). (3) Clients and server want privacy: the server should learn nothing about the clients’ set, and the clients should only learn the aggregated output. However, PCM problems differ in their definition of *when sets match* and *how to combine individual matching responses*. Now, we discuss these two aspects in more detail:

*Set matching.* Typically, set matching is defined as a function of the intersec-

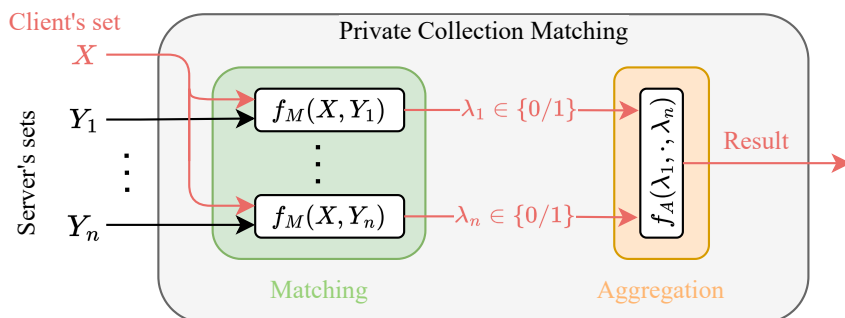


Figure 3.1: Structure of our PCM framework. Red arrows show values encrypted under the client’s key.  $f_M$  designates a matching function: it outputs a binary value  $\lambda$  indicating whether two sets match.  $f_A$  is an aggregation function that combines  $n$  matching statuses into a collection-wide result.

tion of two sets. Hence, clients could detect a matching server set by using private set intersection (PSI) protocols [12, 17, 25, 46, 60–62, 62, 63] to privately compute the intersection, then post-process the intersection to determine interest locally. PCM applications differ in their matching criteria and may decide interest using measures such as a cardinality threshold, containment, or set similarity. This local processing approach, unfortunately, *reduces privacy of the server’s sets* by leaking information beyond the set’s matching status to the client. Such leakage could, for instance, reveal secret chemical properties of compounds, or the content of journalists’ sensitive documents.

*Many-set.* In PCM problems, the server holds *a collection of  $N$  sets*. This creates two challenges. First, running one matching (or PSI) interaction per server set is inefficient. Second, revealing individual set-matching statuses harms server privacy. While servers may be interested in selling data to or collaborating with clients, they want to ensure that clients cannot use the ‘PCM solution’ to extract information about sets. Clients, meanwhile, often only need an aggregated response summarizing the utility of a collection. Servers therefore enact application-dependent aggregation policies ensuring that clients can, e.g., determine only whether at least one set matches or learn only the the number of matching sets.

We construct a framework that leverages computation *in the encrypted domain* to solve PCM problems. In our framework, shown in Fig. 3.1, given an encrypted client set, the server uses a matching criteria  $f_M$  to compute per-server-set *binary* answers to “is this set of interest to the client?”. Next, the server uses an aggregation policy  $f_A$  to combine per server-set responses into a collection-wide response. Finally, the client decrypts the aggregated result.

Our work makes the following contributions:

✓ We introduce Private Collection Matching (PCM) problems. We derive their requirements from three real-world problems.

✓ We design single-set protocols where the client learns a *one-bit output* – whether one server set is of interest to the client – and many-set protocols where the client learns a *collection-wide output* that aggregates individual matching responses. The communication cost of our protocols scales linearly with the size of the client’s set and is *independent* of the number of server sets and their total size.

✓ We propose a modular design that separates flexible set matching criteria from many-set aggregation. Our modularity enables extending our design with new matching or aggregation policies and simplifies building privacy-preserving PCM solutions.

✓ We demonstrate our framework’s capability by solving chemical similarity and document search problems. We show that our framework offers improved privacy with competitive cost compared to custom-made solutions, and significantly improves the latency, client’s computation cost, and communication cost with respect to generic solutions that offer the same privacy guarantee.

## 3.2 Private Collection Matching

In this section, we define the Private Collection Matching (PCM) problem. We derive its basic requirements from three real-world matching problems. We also explain why existing PSI solutions cannot satisfy the privacy requirements of PCM problems.

### 3.2.1 Case studies

We study three cases that can benefit from PCM.

*Chemical similarity.* Chemical research and development is a multi-billion dollar industry. When studying a new chemical compound, knowing the properties of similar compounds can speed up the research. In an effort to monetize research, companies sell datasets describing thousands to millions of compounds and their properties. Chemical R&D teams are willing to pay high prices for these datasets but *only if* they include compounds similar to their

research target. Determining whether this is the case is tricky: buyers want to hide the compound they are currently investigating [55], and sellers want to hide information about the compounds in their dataset before the sale is finalized.

Chemical similarity of compounds is determined by comparing molecular fingerprints of compounds [53, 54, 64–67]. Fingerprints are based on the substructure of compounds and are represented as fixed-size bit vectors – these vectors are between a few hundred and few thousand bits long. Measures such as Tversky [68] and Jaccard [69] determine the similarity of these fingerprints, and thus of the compounds.

Revealing pair-wise intersection cardinalities or even similarity scores between the fingerprints of a target compound and the seller’s compounds results in unacceptable leakage. A buyer can reconstruct an  $n$ -bit molecular fingerprint  $F$  by learning similarity values between  $F$  and  $n + 1$  known compounds [55]. To prevent inferences, the buyer should learn only the number of similar compounds in the seller’s dataset, or, better, only learn whether at least one similar compound exists.

*Peer-to-peer document search.* Privacy-preserving peer-to-peer search engines help users and organizations with strict privacy requirements to collaborate safely. We take the example of investigative journalists who, while unwilling to make their investigations or documents publicly available, want to find collaboration opportunities within their network [56].

To identify those opportunities, a journalist performs a search to learn whether a document owner has documents of interest. A search query consists of keywords relevant to the journalist’s investigation. The document owner compares the query to all documents in his collection. A document is deemed relevant if it contains *all* or *a sufficient number of* queried keywords. Journalists own a collection of a thousand documents (on average), and each document is represented by around a hundred keywords.

The sensitivity of journalists’ investigations demand that both the content of the documents and of the queries remain private [56]. Journalists only need to learn one bit of information – that at least one or a threshold number of documents in the owner’s collection is relevant – to determine whether they should contact the owner.

*Matching in mobile apps.* A common feature in mobile apps is enabling users to find records of interest in the app servers’ databases, e.g., restaurants [57],



routes for running [58], or suitable dating partners [59,70]. Users are typically interested in records that have at least a number of matching characteristics in common with their search criteria or that are a perfect match. Also, users need to be able to retrieve these records.

The user-provided criteria – typically range choices entered via radio buttons or drop-down menus – are compared to the attributes of records. An app database can have millions of records and records can have dozens to hundreds of attributes.

Both search criteria and records are sensitive. Knowing search criteria enables profiling of user interests. These are particularly sensitive for dating applications. Thus, search queries should be kept private. The secrecy of the records in the database is not only at the core of the business value of these apps but also required by law in cases where records contain personal data (e.g., dating apps).

### 3.2.2 PCM requirements

We extract requirements that PCM protocols should fulfill based on the commonalities between the use cases. These come in addition to basic PSI properties such as client privacy.

*RQ.1: Flexible set matching. PCM protocols need to be able to determine matches between sets without revealing other information such as intersections or cardinalities to the client.* In the use cases, clients do not need to know the intersection or its cardinality. They are interested only in whether there is a match. Matches in these examples are a function of the intersection between a client and a server set: a chemical compound is a match when the Tversky or Jaccard similarity with the query exceeds a threshold; a document is a match when it contains some or all query keywords; and a record is a match when it includes a threshold of query attributes. PCM protocols must be able to detect matching sets and compute a *single* one-bit matching status per-set.

*RQ.2: Aggregate many-set responses. PCM protocols need to have the capability to provide an aggregated response for a collection of sets without leaking information about individual sets.* Our use cases highlight that in many applications, a client (buyer, journalist, user) may want to compare their input (compounds under investigation, keywords of interest, search criteria) with a *collection* of sets (compounds in a database, documents in a collection, records in a database). More importantly, we observe that clients wish to know how

interesting the collection is as a whole. For example, a buyer is interested in a chemical dataset if it contains at least one similar compound and a querier journalist may contact a document owner if the owner has a number of relevant documents in their collection. Therefore, to satisfy clients' needs yet protect the server's privacy, PCM protocols should only reveal aggregated per-collection results.

*RQ.3: Extreme imbalance. PCM problems have thin clients and imbalanced input sizes; thus, protocols must not require communication and computation linear to the server's input size from clients.* Drawing from our earlier scenarios, the total input size of the server may be as much as *6 orders of magnitude* larger than the client's input, as shown by the chemical similarity scenario. The server, holding many sets, can safely be assumed to be resourceful, while clients may be constrained in their capabilities, e.g., a client running the PCM protocol from their mobile phone. This can be in terms of computation, e.g., battery has to be preserved in mobile apps; or in terms of bandwidth, e.g., journalists that can be in locations with poor Internet access. Therefore, PCM protocols should not incur a large client-side cost.

### 3.2.3 Formal PCM definition

Let  $X$  be a client set with  $n_c$  elements  $\{x_1, \dots, x_{n_c}\}$  from input domain  $D$  and  $\mathcal{Y}$  be a collection of  $N$  server sets  $\{Y_1, \dots, Y_N\}$  where the  $i$ 'th server set  $Y_i = \{y_{i,1}, \dots, y_{i,n_s^i}\}$  has  $n_s^i$  elements, also from  $D$ , leading to a total server size of  $N_s = \sum_i n_s^i$ . We define two families of functions as follows:

Matching functions  $\lambda_i \leftarrow f_M(X, Y_i)$  take two sets  $X$  and  $Y_i$  as input and compute a binary matching status  $\lambda_i$  determining interest. This family represents our flexible matching criteria RQ.1. For example, in the document search scenario where a server set (document) is of interest when it contains all queried keywords, we define  $f_M(X, Y_i)$  as 1 if  $X \subseteq Y_i$  and 0 otherwise.

Aggregation functions  $A \leftarrow f_A(\lambda_1, \dots, \lambda_N)$  take  $N$  binary matching statuses ( $\lambda_i$ ) and aggregates them into a single response  $A$ . This family represents our aggregation requirement RQ.2. For example, if we want to count the number of relevant documents in a search, we define  $f_A(\lambda_1, \dots, \lambda_N)$  as  $\sum_j \lambda_j$ .

In Table B.2 in Section B.1, we summarize the matching and aggregation functions that we implement.

**Definition 4** (PCM). PCM protocols are two-party computations between a

client and a server with common inputs  $f_M$  and  $f_A$ , where the client learns an aggregated matching status and the server learns nothing. Formally:

$$(A = f_A(f_M(X, Y_1), \dots, f_M(X, Y_N)), \perp) \leftarrow PCM_{f_M, f_A}(X, \mathcal{Y})$$

We use this notation to define formal properties of PCM protocols.

**Definition 5** (Correctness). A PCM protocol is correct if the client output matches the result of  $A = f_A(f_M(X, Y_1), \dots, f_M(X, Y_N))$ .

**Definition 6** (Client privacy). A PCM protocol is client private if the server cannot learn any information about the client’s set beyond the size of the client’s set.

**Definition 7** (Server privacy). A PCM protocol is server private if the client cannot learn any information about the server elements beyond the number of server sets  $N$ , the maximum server set size  $\max n_s^i$ , and the client output  $A = f_A(f_M(X, Y_1), \dots, f_M(X, Y_N))$ .

### 3.3 Related work

While ad-hoc solutions for chemical (Shimizu et al. [55]) and document search (EdalatNejad et al. [56]) exist, most prior work focuses on building private set intersection (PSI) protocols, which are a special case of PCM. We introduce the PSI protocols most relevant to our work. We leave the detailed comparison with ad-hoc solutions to the evaluation (see Section 4.8).

We compare existing work on two critical aspects of PCM problems: privacy and efficiency. We summarize existing schemes and their suitability for the PCM scenario in Table 3.1.

For privacy, we assess whether existing approaches provide flexible matching (RQ.1) and aggregated many-set responses (RQ.2). We note that the majority of prior works do not consider, or support, many sets. When there is no natural extension to support many sets at once, we run a single-set interaction per server set, leading to an  $N \times$  increase in cost. This naive extension does not provide the privacy enhancement of many-set aggregation but enables us to reason about the efficiency of these approaches.

For efficiency, taking into account the extreme imbalance requirement (RQ.3), we focus on the client’s computation and communication cost and require schemes to have a client cost of  $\omega(N_s (\approx N n_s^i))$ .

Table 3.1: Overview of PSI approaches in the PCM setting.

		Privacy		Efficiency
		RQ.1	RQ.2	RQ.3
Comparison	[13, 15, 17, 46, 71, 72]	×	×	×
OPRF	[16, 25, 60, 61, 73]	×	×	✓*
OPE	[45, 74–76]	×	×	✓
Generic SMC	[11, 77, 78]	✓	✓	×
Circuit-PSI	[12, 62, 79–83]	✓	✓	×
Flexible functionality	[47, 55, 84–87]	✓	×	×
<b>Our work</b>		✓	✓	✓

\*Efficient communication requires pre-processing.

**Traditional single-set PSI.** We first study protocols *solely focusing on single-set intersection or cardinality*. We study schemes that offer *enhanced functionality or privacy* below in the Custom PSI Protocols section.

In PSI, clients learn information about the intersection of two sets while (i) not learning anything about the server’s non-intersecting elements, and (ii) not leaking any information about their own set to the server. PSI protocols in the literature focus on providing two possible outputs: the *intersection* (e.g., finding common network intrusions [88], or discovering contacts [89]); and the *cardinality of the intersection* (e.g., privately counting the number of common friends between two social media users [90], or performing genomic tests [91]). Works in this area opt for a variety of trade-offs between the computational capability and the amount of bandwidth required to run the protocol [16, 76, 77, 92]. These works show that PSI can scale to large datasets [62, 93], and support light clients [16].

We classify PSI protocols according to the techniques they use to compute the PSI functionality:

*Comparison-based protocols.* The fastest class of PSI protocols uses bucketing to enable PSI protocols that run optimized comparison protocols between very small client and server buckets [13, 15, 17, 46, 71, 72]. These approaches use hashing to first map elements to small buckets. Then they compare the client and server elements in each bucket using comparison primitives that are efficient when buckets have very few elements, for example built using oblivious transfer (OT). These approaches reveal comparison results, and thus intersections or cardinalities, to the client; therefore, they do not satisfy our single-set privacy requirement (RQ.1). As these approaches reveal individual detailed set

responses, *private* aggregation (RQ.2) is impossible. We discuss approaches that do not reveal the comparison result separately below as ‘Circuit-PSI’. Each client and server element must participate in at least one comparison; consequently, the communication cost is linear in the client size  $n_c$  *and* total server size  $N_s$ . Therefore, comparison-based approaches do not satisfy our efficiency requirement (RQ.3).

In Section B.4.3, we confirm our efficiency assessment by evaluating the cost of SpOT-light [72], one of the fastest comparison-based PSI protocols, in the PCM setting. We show significant improvement in latency (10–65x), client’s computation (1800–24,800x), and transfer cost (1.7–27x).

*OPRFs.* Another technique to construct PSI protocols is to evaluate a pseudo-random function (PRF) over client and server set elements and compare these. To protect privacy, the client obviously evaluates the PRF over its elements together with the server, and the server sends the PRF evaluation of its elements to the client. Oblivious Pseudo-random Functions (OPRFs) can be constructed from asymmetric primitives such as RSA [16, 60, 61], Diffie-Hellman [16, 73], discrete logarithms [25], or by applying SMC on symmetric PRFs [16, 77].<sup>1</sup> What distinguishes these OPRF-based approaches from the previous category is that instead of comparing many *pairs* in small buckets, OPRF-based approaches compute a deterministic tag for each client and server element that can be compared locally. While the communication cost is linear in the size of both client *and* server sets, preprocessing and reusing tags for server elements can make the transfer cost independent of the server size [16]. Similar to comparison-based protocols, OPRF approaches cannot compute flexible set matches without leaking intermediate data nor do they support aggregation.

*Oblivious polynomial evaluation.* Another approach is to use (partial) homomorphic encryption [94] to determine set intersection using oblivious polynomial evaluation (OPE) [45, 74–76]. The client encrypts their elements and sends them to the server. The server constructs a polynomial with its set elements as roots, evaluates the polynomial on encrypted client elements, randomizes the result, and sends them back to the client. The client decrypts the results, a 0 indicates a matching element. We use a similar approach in our schemes. Existing OPE-based approaches do not support flexible set matching or aggregation, but achieve cost independent of the server input size ( $N_s$ ) for the client.

---

<sup>1</sup>We only consider OPRF approaches where the parties can *choose the PRF key*. Primitives where a random key is generated during the execution [71, 72] can only be used for comparing-based approaches.

*Generic SMC.* Some PSI protocols use generic SMC tools [95–97] to construct full circuits such as sort-compare-shuffle [11] to compute the intersection [77,78]. They can be extended to support flexible set matching or many-set aggregation. However, circuits have communication linear in the size of their inputs (wires) which guarantees a transfer cost of  $\mathcal{O}(n_c + N_s)$ . This is a fundamental limit. Thus, circuits cannot satisfy our efficiency requirement (RQ.3). Because circuits can satisfy our privacy requirements, we develop a generic alternative to our framework using an SMC compiler in Section 3.11.2 and show that our system improves latency (2–96x), client’s computation (75–2250x), and transfer cost (93–2800x). Besides for generic protocols, circuits are used to (1) extend OT-based protocols (discussed as ‘Circuit-PSI’ below) or (2) obviously evaluate PRFs [16, 77] such as AES or LowMC [98] (discussed as ‘OPRF’).

**Custom PSI protocols.** Some PSI protocols go beyond cardinality and compute more complex functions over the intersection.

*Circuit-PSI.* A new line of work extends comparison-based PSI protocols to support arbitrary extensions with generic SMCs [12, 62, 79–83]. These works compute the intersection of two sets but instead of revealing the plain result to the client, they secret share the intersection between the two parties. This secret shared output enables parties to privately compute arbitrary functions on top of the intersection. Unfortunately, these work focus on scenarios with equal client and server sizes as their cost is linear in the input of both parties  $\mathcal{O}(n_c + N_s)$ . This linear communication cost is a fundamental limit. As these approaches can satisfy our privacy requirement, we evaluate Chandran et al. [81], a state-of-the-art Circuit-PSI paper, in the PCM setting in Section 3.11.2. We show that our framework significantly improves latency (580x), client’s computation (70,000x), and transfer cost (2360x).

*Flexible functionality.* Several privacy-preserving custom protocols provide functionality beyond computing intersection or cardinality. For instance, computing the sum or statistical functions over associated data [84,86], evaluating a threshold on intersection size [47,85,87], or computing Tversky similarity [55]. These approaches improve privacy by supporting flexible set matching but do not extend well to many-sets scenarios and aggregation. They are optimized for a specific setting and do not achieve cost independent of the server input size.

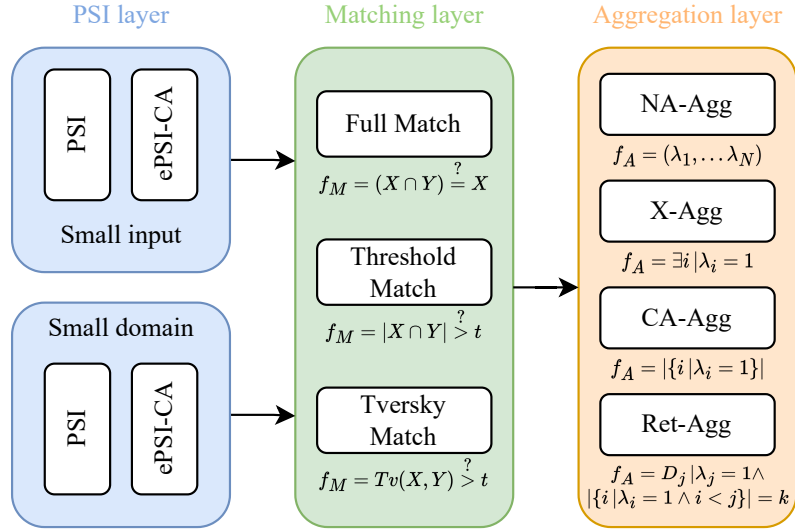


Figure 3.2: An overview of our layers and their composition. Refer to Table B.2 for a summary of protocol definitions.

**Orthogonal works.** We briefly mention two groups of related work that, while of interest, are orthogonal to PCM problems and solve different challenges. *Encrypted databases and ORAM* [50, 99–101] let clients query outsourced data, or, in more recent work [102], subsets of outsourced data via an access control policy. Typically, these approaches do not limit what the client can learn about the entries they have access to. This is in contrast with the PCM setting where the client computes a function over the server’s sets, and clients should not learn more than the function’s outcome. Therefore, traditional applications of encrypted databases and ORAM do not seem to directly enable the construction of PCM solutions. However, we do not rule out the possibility that they can be extended or used in creative ways to solve PCM problems.

In *multi-party PSI* [103–105],  $p$  parties each with a set  $S_i$  compute one intersection  $I = \cap S_i$ . PCM problems, instead, are a two-party protocol where the server holds  $N$  sets.

### 3.4 A framework for PCM schemes

We showed that existing work – except for ad-hoc solutions – cannot solve PCM problems without losing either privacy or efficiency. We now introduce a modular framework that enables the design of PCM solutions with minimal

effort and strong privacy while providing performance close to ad-hoc solutions.

The framework has three layers, shown in Fig. 3.2:

*PSI layer* protocols operate directly on a client’s and a server’s set. These protocols compute single-set PSI functionalities such as intersection or cardinality. Our implementation focuses on two scenarios: small input domain size and small constant-size client sets.

*Matching layer* protocols use PSI layer protocols to compute a binary answer determining whether each of the server sets matches the interest of the client according to a pre-defined matching function  $f_M$  (RQ.1). Computation in this layer is the same regardless of the scenario chosen in the PSI layer.

*Aggregation layer* protocols aggregate  $N$  single-set responses into one collection-wide response according to a pre-defined aggregation function  $f_A$  (RQ.2). This layer achieves constant size responses and ensures efficient client communication (RQ.3).

*Modularity.* While we provide a large number of protocols for the PSI, Matching, and Aggregations layers (see Fig. 3.2), an advantage of our framework is extensibility. Whenever an application requires new matching or aggregation criteria, designers can add (or adapt) a single functionality while taking advantage of the existing optimized layers. As an example, we extend our framework to support the single-set PSI-SUM functionality in Section B.6.

The layers can also be used as standalone protocols. We include blocks such as ‘Naive’ aggregation such that even if an application does not require matching or aggregation the designer can use the framework to enjoy its capability to tackle the many-set scenario.

*Security goals.* The PCM framework should protect the privacy of clients and servers against semi-honest adversaries. Furthermore, it is desirable that the framework provides server and client privacy against malicious adversaries, however, as we discuss in Section 3.9 our framework only guarantees client privacy. Servers are free to choose their input, allowing them to degrade the quality of the protocol’s result without any misbehavior. We accept this inherent weakness of PCM protocols and make the deliberate decision to aim for correctness only in the semi-honest setting, and *not* when the server is malicious.



## 3.5 Technical background

We introduce our notation and define the syntax of the fully homomorphic encryption scheme we use.

*Notation.* Let  $n$  be a security parameter. We write  $x \leftarrow^{\$} X$  to denote that  $x$  is drawn uniformly at random from the set  $X$ . Let  $q$  be a positive integer, then  $\mathbb{Z}_q$  denotes the set of integers  $[0, \dots, q)$ , and  $\mathbb{Z}_q^*$  represent the elements of  $\mathbb{Z}_q$  that are co-prime with  $q$ . We write  $[n]$  to denote the set  $\{1, \dots, n\}$ , and use  $\langle a_i \rangle_m$  to present the list  $[a_1, \dots, a_m]$ . We drop the subscript  $m$  when the list length is clear from the context. We write  $\llbracket x \rrbracket$  to denote the encryption of  $x$ . We write  $\mathbb{1}[E]$  to denote the indicator function that returns ‘1’ if the event  $E$  is true, and ‘0’ otherwise. Table B.1 in Section B.1 summarizes our notation.

### 3.5.1 Homomorphic encryption

Homomorphic encryption (HE) schemes enable arithmetic operations on encrypted values without decryption. We use HE schemes that operate over the ring  $\mathbb{Z}_q$  with prime  $q$ , such as BFV [106].

*Syntax.* HE is defined by the following procedures:

- $\text{params} \leftarrow \text{HE.ParamGen}(q)$ . Generates HE parameters with the plaintext domain  $\mathbb{Z}_q$ .
- $pk, sk \leftarrow \text{HE.KeyGen}(\text{params})$ . Takes the parameters  $\text{params}$  and generates a fresh pair of keys  $(pk, sk)$ . For brevity, we do not explicitly mention evaluation keys  $evk$  and consider them to be incorporated in the public key.
- $\llbracket x \rrbracket \leftarrow \text{HE.Enc}(pk, x)$ . Takes the public key  $pk$  and a message  $x \in \mathbb{Z}_q$  and returns the ciphertext  $\llbracket x \rrbracket$ .
- $x \leftarrow \text{HE.Dec}(sk, \llbracket x \rrbracket)$ . Takes the secret key  $sk$  and a ciphertext  $\llbracket x \rrbracket$  and returns the decrypted message  $x$ .

<sup>g</sup> The correctness property of homomorphic encryption ensures that  $\text{HE.Dec}(sk, \text{HE.Enc}(pk, x)) \equiv x \pmod{q}$ .

*Homomorphic operations.* HE schemes support homomorphic addition (denoted by  $+$ ) and subtraction (denoted by  $-$ ) of ciphertexts:  $\text{HE.Dec}(\llbracket a \rrbracket + \llbracket b \rrbracket) = a + b \pmod{q}$  and  $\text{HE.Dec}(\llbracket a \rrbracket - \llbracket b \rrbracket) = a - b \pmod{q}$ . HE schemes also support multiplication (denoted by  $\cdot$ ) of ciphertexts:  $\text{HE.Dec}(\llbracket a \rrbracket \cdot \llbracket b \rrbracket) = a \cdot b \pmod{q}$ .

---

**Algorithm 1** Check whether  $\llbracket x \rrbracket$  is zero.

---

```

function HE.ISZERO( $pk, \llbracket x \rrbracket$ )
   $\llbracket b \rrbracket = 1 - \llbracket x \rrbracket^{(q-1)}$   $\triangleright b \leftarrow (x = 0)$ 
  return  $\llbracket b \rrbracket$ 

```

---

Besides operating on two ciphertexts, it is possible to perform addition and multiplication with plaintext scalars. In many schemes, such scalar-ciphertext operations are more efficient than first encrypting the scalar and then performing a standard ciphertext-ciphertext operation. We abuse notation and write  $a\llbracket x \rrbracket + b$  to represent  $(\llbracket a \rrbracket \cdot \llbracket x \rrbracket) + \llbracket b \rrbracket = \llbracket ax + b \rrbracket$ .

*Multiplicative depth.* Our framework is designed with fully homomorphic encryption (FHE) in mind and assumes unbounded multiplication depth. For practical purposes, we port the majority, but not all, of our protocols to support execution with somewhat homomorphic encryption (SWHE) and optimize operations in Section 3.10.

### 3.5.2 Core functions

The complex functionality of PCM protocols can be reduced to a sequence of zero detection and inclusion test procedures. These two functions allow us to describe our protocol at a higher abstraction level. Moreover, any improvement to these basic functions automatically enhances our framework.

*Zero detection.* The function  $\llbracket b \rrbracket \leftarrow \text{HE.IsZero}(pk, \llbracket x \rrbracket)$  computes whether the ciphertext  $\llbracket x \rrbracket$  is an encryption of zero. The binary output  $b \in \{0, 1\}$  is defined as  $b = 1$  if  $x \equiv 0 \pmod{q}$  otherwise  $b = 0$ . We use Fermat's Little Theorem for zero detection [107]. We rely on the prime ring structure of  $\mathbb{Z}_q$  as any non-zero variable  $x \in \mathbb{Z}_q^*$  to the power  $q - 1$  is congruent to one modulo the prime  $q$ . We can perform this exponentiation with  $\lg(q)$  multiplications. See Algorithm 1 for the implementation.

The high multiplicative depth of **HE.IsZero** makes it impractical for use with most SWHE schemes. We hope that the research and advances in HE comparison enables efficient instantiations of this function and unlock our framework's full capabilities. When evaluating our framework in Section 4.8, we use ad-hoc techniques to avoid the need for this function.

*Inclusion test.* The function  $\llbracket I \rrbracket \leftarrow \text{HE.IsIn}(pk, \llbracket x \rrbracket, Y)$  checks if  $x$  is included in the set  $Y$  of cardinality  $n$ . We consider two variants. In the first,  $Y$  is a set of ciphertexts  $\llbracket y_i \rrbracket$ , in the second,  $Y$  is a set of plaintexts  $y_i$ . In both

---

**Algorithm 2** Check inclusion of an encrypted variable  $x$  in a plain  $Y = \{y_1, \dots, y_n\}$  or an encrypted  $Y = \{\llbracket y_1 \rrbracket, \dots, \llbracket y_n \rrbracket\}$  set.

---

**function** HE.ISIN( $pk, \llbracket x \rrbracket, Y = \{\llbracket y_1 \rrbracket, \dots, \llbracket y_n \rrbracket\}$ )

$\llbracket I \rrbracket \leftarrow \prod_{i \in [n]} (\llbracket x \rrbracket - \llbracket y_i \rrbracket)$

$r \xleftarrow{\$} \mathbb{Z}_q^*$

**return**  $r \cdot \llbracket I \rrbracket$

**function** HE.ISIN( $pk, \llbracket x \rrbracket, Y = \{y_1, \dots, y_n\}$ )

$[a_0, \dots, a_n] \leftarrow \mathbf{ToCoeffs}(Y)$

$\triangleright$  Such that  $\prod_i (x - y_i) = \sum_i a_i x^i$

$\llbracket I \rrbracket \leftarrow \sum_{i \in [0..n]} a_i \cdot \llbracket x \rrbracket^i$

$r \xleftarrow{\$} \mathbb{Z}_q^*$

**return**  $r \cdot \llbracket I \rrbracket$

---

cases, the output  $I$  equals 0 if and only if a  $y_i$  exists such that  $x \equiv y_i \pmod{q}$ , otherwise  $I$  will be a uniformly random element in  $\mathbb{Z}_q^*$ . See Algorithm 2 for the implementation.

The function **HE.ISIN** relies on oblivious polynomial evaluation (OPE) [45, 74]. We create an (implicit) polynomial  $P$  with roots  $y_i$ , and evaluate  $\llbracket I \rrbracket = \llbracket r \cdot P(x) \rrbracket$ . If  $x$  is in the set, there exists a variable  $y_i$  where  $x \equiv y_i$ , thus  $I$  is zero. Otherwise,  $I$  is the product of  $n$  non-zero factors modulo  $q$ . Since  $q$  is prime, the product of non-zero values is non-zero. The random value  $r$  ensures uniformity in this case. The multiplicative depth of **HE.ISIN** scales with the size of  $Y$ . We use the second form, where  $Y$  is a set of plaintexts, to lower the multiplicative depth when  $\llbracket x^i \rrbracket$  are known, see Section 3.10.3.

## 3.6 PSI layer

The PSI layer of our framework implements basic PSI functionalities: computing intersection or intersection cardinality. These protocols can be used in isolation, but in our framework they serve to form the input to the matching layer (see Fig. 3.2). We build PSI protocols for two scenarios: (1) scenarios where the client set has a small constant size (e.g., document search queries which typically have less than 10 keywords); and (2) scenarios where set elements come from a small input domain (e.g., gender and age in a dating profile). We build basic protocols assuming semi-honest clients, but allow for an extension – see Section 3.6.3 – that ensures queries represent a valid input set even if clients deviate from the protocol.

We structure our single-set protocols following Fig. 3.3. The client generates a HE key pair  $(pk, sk) \leftarrow \mathbf{HE.KeyGen}(\text{params})$  and sends the public key

$pk$  to the server ahead of the protocol. Clients perform `QUERY` and send an encrypted representation of their set to the server. The server runs a protocol-specific processing function `PROCESS` to obtain the result  $M$ . The protocol is either used as the first layer and passes  $M$  into the second layer, or is stand-alone and returns  $M$  to the client. The server optionally runs `QUERY-CHECK` to randomize the result when a client submits a malformed query ( $\llbracket R \rrbracket$  is zero for correctly formed queries). Finally, the client runs the protocol-specific function `REVEAL` to compute the result. We denote algorithms run by the **client** in **red** and by the **server** in **green**. The *optional server-side* checks that ensure well-formedness of queries are denoted in **blue**.

### 3.6.1 Small constant-size client set

We start with scenarios where client sets are small and constant-size, typical for representing a search criteria. Clients use `QUERY` to encrypt their set elements  $x_i \in X$  as a query  $Q$  and send it to the server. Algorithm 3 instantiates small input functions.

*PSI.* The PSI protocol computes  $\text{PSI}(X, Y_k) = X \cap Y_k$ . The server uses the inclusion test `HE.IsIn` (see Section 3.5.1) to compute an inclusion status  $\llbracket s_i \rrbracket$  for each client element  $x_i$  (see `PSI-PROCESS`). An element  $x_i$  is in the intersection if and only if the corresponding inclusion status  $\llbracket s_i \rrbracket$  is zero (recall that the inclusion test produces zero for values *in* the set). When used as a stand-alone protocol, the server returns the list of encrypted inclusion values  $M$ , which the client then decrypts (see `PSI-REVEAL`).

*Cardinality.* The PSI cardinality protocols compute  $\text{PSI-CA}(X, Y_k) = |X \cap Y_k|$ . There exist two variants: the standard PSI-CA variant in which the *client* learns the cardinality  $|X \cap Y_k|$  [25, 108], and the ePSI-CA variant in which the *server* learns an *encrypted* cardinality [47]. We focus on the latter to enable further computation on the intersection cardinality in the next layers.

Our ePSI-CA protocol (see `EPSI-CA-PROCESS`) first computes the inclusion statuses  $\llbracket s_i \rrbracket$  using `PSI-PROCESS` and then uses `HE.IsZero` to compute – in the ciphertext domain – the cardinality, i.e., the number of elements  $\llbracket s_i \rrbracket$  that are zero. When used as a stand-alone protocol, the server returns  $\llbracket \text{ca} \rrbracket$  to the client which decrypts it to obtain the answer (see `EPSI-CA-REVEAL`).

When the cardinality protocol is used as a stand-alone protocol without next layers, it is possible to mimic earlier work [25] and construct a cardinality protocol from the above-mentioned naive PSI protocol by shuffling server

Client		Server
$X = \{x_1, \dots, x_m\} \subseteq D$ $(pk, sk)$		$Y_i = \{y_{i,1}, \dots, y_{i,n_s^i}\} \subseteq D$ $pk$
$Q \leftarrow (\text{SD-})\text{QUERY}(pk, X)$	$\xrightarrow{Q}$	$\llbracket A \rrbracket \leftarrow \text{PROCESS}(pk, Q, Y_k)$ $\llbracket R \rrbracket \leftarrow (\text{SD-})\text{QUERY-CHECK}(pk, Q)$
$R \leftarrow \text{REVEAL}(sk, M)$	$\xleftarrow{M}$	$M \leftarrow \llbracket A \rrbracket + \llbracket R \rrbracket$

Figure 3.3: Single-set protocol structure. SD refers to small domain variants. The optional blue parts ensure queries are well-formed.

---

**Algorithm 3** Single set procedures with *small input size*.

---

```

function QUERY( $pk, X$ )
   $\llbracket x_i \rrbracket \leftarrow \text{HE.Enc}(pk, x_i)$ 
  return  $Q = \langle \llbracket x_i \rrbracket \rangle$ 

function PSI-PROCESS( $pk, Q = \langle \llbracket x_i \rrbracket \rangle, Y_k$ )
   $\llbracket s_i \rrbracket \leftarrow \text{HE.IsIn}(pk, \llbracket x_i \rrbracket, Y_k)$ 
  return  $M \leftarrow \langle \llbracket s_i \rrbracket \rangle$ 

function PSI-REVEAL( $pk, M = \langle \llbracket s_i \rrbracket \rangle$ )
  return  $\{x_i \mid \text{HE.Dec}(sk, \llbracket s_i \rrbracket) = 0\}$ 

function ePSI-CA-PROCESS( $pk, Q = \langle \llbracket x_i \rrbracket \rangle, Y_k$ )
   $\langle \llbracket s_i \rrbracket \rangle \leftarrow \text{PSI-PROCESS}(pk, \langle \llbracket x_i \rrbracket \rangle, Y_k)$ 
   $\llbracket \text{ca} \rrbracket \leftarrow \sum_{i \in [m]} \text{HE.IsZero}(\llbracket s_i \rrbracket)$ 
  return  $M \leftarrow \llbracket \text{ca} \rrbracket$ 

function ePSI-CA-REVEAL( $pk, M = \llbracket \text{ca} \rrbracket$ )
  return  $\text{HE.Dec}(sk, \llbracket \text{ca} \rrbracket)$ 

function QUERY-CHECK( $pk, Q = \langle \llbracket x_i \rrbracket \rangle$ )
   $\llbracket T \rrbracket \leftarrow \prod_{i \in [Q], j \in [i-1]} (\llbracket x_i \rrbracket - \llbracket x_j \rrbracket)$ 
   $\llbracket R \rrbracket \leftarrow r \cdot \text{HE.IsZero}(pk, \llbracket T \rrbracket)$ 
  return  $\llbracket R \rrbracket$ 

```

---

responses  $M$  before returning them.

*Efficiency.* While literature often dismisses OPE-based schemes due to their ‘quadratic’ total computation cost  $\mathcal{O}(|X| \cdot |Y_k|)$ , this approach excels in PCM scenarios with small client input. Our protocols achieve client computation and communication costs of  $\mathcal{O}(n_c)$ , which is independent of the server’s input size. While the ‘extra’ burden for the server is linear in the size of the client set which is a small constant.

---

**Algorithm 4** Single set procedures with *small input domain*.

---

```

function SD-QUERY( $pk, X$ )
   $z_i \leftarrow (d_i \in X)$ 
   $\llbracket z_i \rrbracket \leftarrow \text{HE.Enc}(pk, z_i)$ 
  return  $Q = \langle \llbracket z_i \rrbracket \rangle$ 

function PSI-SD-PROCESS( $pk, Q = \langle \llbracket z_i \rrbracket \rangle, Y_k$ )
   $v_i \leftarrow (d_i \in Y_k)$ 
   $\llbracket s_i \rrbracket \leftarrow \llbracket z_i \rrbracket \cdot v_i$ 
  return  $M \leftarrow \langle \llbracket s_i \rrbracket \rangle$ 

function EPSI-CA-SD-PROCESS( $pk, Q = \langle \llbracket z_i \rrbracket \rangle, Y_k$ )
   $\langle \llbracket s_i \rrbracket \rangle \leftarrow \text{PSI-SD-PROCESS}(pk, \langle \llbracket z_i \rrbracket \rangle, Y_k)$ 
   $\llbracket \text{ca} \rrbracket \leftarrow \sum_{d_i \in D} \llbracket s_i \rrbracket$ 
  return  $M \leftarrow \llbracket \text{ca} \rrbracket$ 

function SD-QUERY-CHECK( $pk, Q = \langle \llbracket z_i \rrbracket \rangle$ )
   $\llbracket t_i \rrbracket \leftarrow \text{HE.IsIn}(pk, \llbracket z_d \rrbracket, \{0, 1\})$ 
   $\llbracket R \rrbracket \leftarrow \sum_{d \in D} \llbracket t_i \rrbracket$ 
  return  $\llbracket R \rrbracket$ 

```

---

### 3.6.2 Small input domain

When the set's input domain  $D$  is small, sets can be efficiently represented and manipulated as bit-vectors [55, 103, 109]. Parties agree on a fixed ordering  $d_1, \dots, d_{|D|}$  of the elements in  $D$ . Then, clients use SD-QUERY to compute a vector of encrypted inclusion statuses  $\langle \llbracket z_i \rrbracket \rangle$ , where  $z_i = 1$  iff  $d_i \in X$  and 0 otherwise, for all  $d_i$ s in  $D$ . We instantiate small domain procedures, except for REVEAL processes that are not impacted by the domain size, in Algorithm 4.

The function PSI-SD-PROCESS creates a bit vector of the intersection. The status  $\llbracket s_i \rrbracket$  is an encryption of 1 if  $d_i$  is present in both sets and 0 otherwise. To do so, the server multiplies the indicator  $\llbracket z_i \rrbracket$  with another binary indicator  $v_i$  determining whether the element  $d_i$  is present in the server set  $Y_k$ . The function EPSI-CA-SD-PROCESS computes the sum of the inclusion statuses  $\llbracket s_i \rrbracket$  for all domain values  $d_i \in D$ . The functions QUERY, QUERY-CHECK, and base layer PROCESS must have the same domain size. The rest of the functions and layers are not impacted by the choice of domain.

*Efficiency.* While the idea of representing sets as bit-vectors is not new [55, 103, 109], existing works dismiss FHE protocols as too costly and focus on additively homomorphic solutions. We use the inherent parallelism of schemes such as BFV [106], that we discuss in Section 3.10, to achieve lower computation and communication costs, especially in the many-set scenario. In Section B.4.1, we show that PSI-CA-SD has a competitive performance to the existing schemes

such as Ruan et al. [109].

### 3.6.3 Ensuring well-formed queries

In this section we consider a specific class of misbehaving clients: those that encrypt a malformed input and submit it as a query. Since the client’s query is encrypted, the server cannot directly verify well-formedness of the query. In this section we show how we can ensure that if a client’s query is malformed, i.e., it does not correspond to a *valid set*  $X$ , the client’s output is random.

Zero-knowledge proofs are not practical in the FHE setting, so we rely on an HE technique to ensure queries are well-formed. The server uses (SD-) QUERY-CHECK to compute a randomizer term  $\llbracket R \rrbracket$  that is random in  $\mathbb{Z}_q$  if the client misbehaves and 0 otherwise. By adding  $\llbracket R \rrbracket$  to the result  $\llbracket A \rrbracket$ , misbehaving clients learn nothing about the real result. With abuse of notation, the server adds a vector of fresh randomizers when the result is a vector such as the output of PSI-SD-PROCESS. The server can amortize the cost of computing  $N$  randomizers  $\llbracket R_i \rrbracket$  for  $N$  variables  $i$ : The server first computes  $\llbracket R \rrbracket$  as before, then picks a fresh randomness  $\delta_i \leftarrow^{\$} \mathbb{Z}_q^*$  and sets  $\llbracket R_i \rrbracket \leftarrow \delta_i \cdot \llbracket R \rrbracket$ .

*Small domain.* Misbehaving clients can submit non-binary ciphertexts  $\llbracket z_i \rrbracket$  to learn more than the cardinality. We compute a term  $\llbracket R \rrbracket$  which is zero when all  $z_i$ s are binary, and is random otherwise. The term  $\llbracket t_i \rrbracket \leftarrow \text{HE.IsIn}(pk, \llbracket z_i \rrbracket, \{0, 1\})$  evaluates to 0 if  $z_i \in \{0, 1\}$  and to a uniformly random element in  $\mathbb{Z}_q^*$  otherwise. Therefore, the distribution of  $\llbracket R \rrbracket \leftarrow \sum_{d \in D} \llbracket t_i \rrbracket$  will be close to uniformly random in  $\mathbb{Z}_q$  as long as at least one non-binary  $z_i$  exists in the client’s query. See Appendix B.2 for the exact distribution.

*Small input.* The client sends a list of encrypted values  $\llbracket x_i \rrbracket$  to the server. This list represents a set as long as all elements are distinct, so the server needs to ensure that no two client elements are equal. First, the server computes  $\llbracket T \rrbracket$ , the product of pairwise differences of the client elements. Since these multiplications are performed in a prime group  $\mathbb{Z}_q^*$ , the product will be zero if and only if two equal elements exist. Second, the server uses a uniformly random element  $r \leftarrow \mathbb{Z}_q$  in combination with the zero detection on  $\llbracket T \rrbracket$  to compute the additive randomizer  $\llbracket R \rrbracket$ .

Since the zero detection function is impractical, we provide a practical alternative protection method which deviates from our structure. The client can deterministically compute  $T$ , allowing us to protect the result in a multiplicative way by returning  $M \leftarrow \llbracket A \rrbracket \cdot \llbracket T \rrbracket$ . As long as  $T$  is not zero, which

signifies a malicious query, the client can reverse  $T$  and recover  $A \leftarrow M \cdot T^{-1}$ .

Without this protection, misbehaving clients are still limited to submitting a list of scalar values, with a limited size, which is equivalent to a *multi-set*. Depending on the flexible matching function, allowing multi-set queries may or may not impact the security. On one hand, if the server reveals PSI-CA, allowing multi-sets may lead to the extraction of the intersection from the cardinality. On the other hand, if the server is computing F-Match, which checks  $X \subseteq Y_k$ , there is no difference between querying a set or a multi-set.

At first glance, QUERY-CHECK's quadratic computation cost  $\mathcal{O}(n_c^2)$  seems expensive. However, we target imbalanced scenarios where  $n_c \ll N_s$ . In Section 3.10.1, we discuss how this optional protection cost will be overshadowed by the cost of our PSI layer  $\mathcal{O}(n_c \cdot N_s)$ ; thus, not having any impact on our final cost.

*Next layers.* We note that the checks on well-formedness in the PSI layer extend to the matching and aggregation layers by instead applying the randomizer  $\llbracket R \rrbracket$  computed on the client's query to the results that the matching and aggregation layers compute.

### 3.7 Matching layer

Given the output of the PSI layer, the matching layer determines whether the server set  $Y_k$  is of interest to the client. The matching layer outputs a matching status  $\llbracket \gamma_k \rrbracket$ . Similar to the inclusion test, **HE.IsIn** in Section 3.5.1,  $\gamma_k$  is zero for sets of interest and a random value in  $\mathbb{Z}_q^*$  otherwise. The value  $\llbracket \gamma_k \rrbracket$  can be revealed as a binary output  $\lambda_k$  or passed-on to the next layer.

These matching operations can use either the small input or the small domain PSI layer protocols. To instantiate a matching protocol, the client and the server proceed as in Fig. 3.3, but plug in the desired PROCESS variant based on  $f_M$ . As MATCH-PROCESS functions have identical outputs, they share the same MATCH-REVEAL method.

We provide three matching functions  $f_M$ : full matching (F-Match), which determines if the client's query set is fully contained in the server set; threshold matching (Th-Match), which determines if the size of intersection exceeds a threshold; and Tversky matching (Tv-Match), which determines if the Tversky similarity between the client's and the server's set exceeds a threshold. The



**Algorithm 5** Processing matching variants.

---

```

function F-MATCH-PROCESS( $pk, Q, Y_k$ ) ▷ Full match
   $\langle [s_i] \rangle \leftarrow \text{PSI-PROCESS}(pk, Q, Y_k)$ 
   $\llbracket \gamma_k \rrbracket \leftarrow \sum_{i \in [n]} [s_i]$ 
  return  $\llbracket \gamma_k \rrbracket$ 

function TH-MATCH-PROCESS( $pk, Q, Y_k, \mathbb{A} = t_{\min}$ ) ▷ Threshold
   $\llbracket \text{ca} \rrbracket \leftarrow \text{EPSI-CA-PROCESS}(pk, Q, Y_k)$ 
   $T \leftarrow \{t \mid t \in \mathbb{Z}_q, t_{\min} \leq t \leq \min(|Q|, |Y_k|)\}$  ▷ Threshold to set
   $\llbracket \gamma_k \rrbracket \leftarrow \text{HE.IsIn}(pk, \llbracket \text{ca} \rrbracket, T)$ 
  return  $\llbracket \gamma_k \rrbracket$ 

function TV-MATCH-PROCESS( $pk, Q, Y_k, \mathbb{A} = (t, \alpha, \beta)$ ) ▷ Tversky
   $\llbracket \text{ca} \rrbracket \leftarrow \text{EPSI-CA-PROCESS}(pk, Q, Y_k)$ 
   $(a, b, c) \leftarrow \text{Tversky-param-process}(\alpha, \beta, t)$ 
   $T = \{t \mid t \in \mathbb{Z}_q, 0 \leq t \leq (a - b - c)|Y_k|\}$ 
   $\llbracket \text{sizeX} \rrbracket \leftarrow \begin{cases} \llbracket |Q| \rrbracket & \text{For small input size variants} \\ \sum_i [z_i] & \text{For small domain var. with } Q = \langle [z_i] \rangle \end{cases}$ 
   $\llbracket Tv \rrbracket \leftarrow a \cdot \llbracket \text{ca} \rrbracket - b \llbracket \text{sizeX} \rrbracket - c|Y_k|$ 
   $\llbracket \gamma_k \rrbracket \leftarrow \text{HE.IsIn}(pk, \llbracket Tv \rrbracket, T)$ 
  return  $\llbracket \gamma_k \rrbracket$ 

function MATCH-REVEAL( $sk, M = \llbracket \gamma_k \rrbracket$ ) ▷ Reveal matching output
   $\gamma_k \leftarrow \text{HE.Dec}(sk, \llbracket \gamma_k \rrbracket)$ 
   $\lambda_k \leftarrow \mathbb{1}[\lambda_k = 0]$ 
  return  $\lambda_k$ 

```

---

associated PROCESS variants are described in Algorithm 5.

*Full matching.* The F-Match variant determines if all the client elements are inside the server's set, i.e.,  $f_M(X, Y_k) = \mathbb{1}[X \subseteq Y_k]$ . The server first computes the inclusion statuses  $[s_i]$  by calling PSI-PROCESS (see F-MATCH-PROCESS). Recall  $[s_i]$  is zero when  $x_i \in Y_k$ . Therefore, when  $X \subseteq Y_k$ , the sum  $\llbracket \gamma_k \rrbracket$  of all  $[s_i]$  is zero. When an element  $x_i$  is not in the server set, its inclusion status  $s_i$  is uniformly random in  $\mathbb{Z}_q^*$ , and therefore the sum  $\lambda_k$  is also random.

The F-Match protocol has a small false-positive probability when more than one  $x_i$  exists such that  $x_i \notin Y_k$ . Adding multiple random  $[s_i] \in \mathbb{Z}_q^*$  PSI responses can, incorrectly, lead to a zero sum  $\llbracket \gamma_k \rrbracket$ . In Appendix B.2, we bound the probability of a false-positive to  $1/(q-1)$ . Moreover, we bound the difference between the distribution of the sum  $R$  and uniformly random over  $\mathbb{Z}_q$  to  $1/(q-1)^2$  at all points when more than one  $x_i$  is missing. The false-positive probability is zero when only one  $x_i$  is missing.

Note that the F-Match protocol computes containment and not equality. Thus, hashing and comparing the client set with server's set does not work as the server would need to hash every combination of  $|X|$  server set elements

resulting in an exponential cost.

*Threshold matching.* The Th-Match variant determines if the two sets have at least  $t_{\min}$  elements in common, i.e.,  $f_M(X, Y_k; t_{\min}) = \mathbb{1}[|X \cap Y_k| \geq t_{\min}]$ . The server first computes the encrypted cardinality  $\llbracket \mathbf{ca} \rrbracket$  using EPSI-CA-PROCESS, then evaluates the inequality  $\mathbf{ca} \geq t_{\min}$  (see TH-MATCH-PROCESS).

Directly computing this one-sided inequality over encrypted values is costly. However, we know that  $|X \cap Y_k| \leq \min(|Q|, |Y_k|) = t_{\max}$ , where we bound the client set size by the size of the query. The server evaluates the inequality  $t_{\min} \leq \mathbf{ca} \leq t_{\max}$  by performing the inclusion test  $\mathbf{ca} \in \{t \mid t_{\min} \leq t \leq t_{\max}\}$  using HE.IsIn.

*Tversky similarity.* The Tv-Match variant determines if the Tversky similarity of the two sets exceeds a threshold  $t$ . Formally, the protocol computes  $f_M(X, Y_k; \alpha, \beta, t) = \mathbb{1}[\text{Tv}_{\alpha, \beta}(X, Y_k) \geq t]$  where

$$\text{Tv}_{\alpha, \beta}(X, Y_k) = \frac{|X \cap Y_k|}{|X \cap Y_k| + \alpha|X - Y_k| + \beta|Y_k - X|}$$

is the Tversky similarity with parameters  $\alpha$  and  $\beta$ . Computing the Tversky similarity in this form is difficult as it requires floating-point operations. We assume  $t, \alpha$ , and  $\beta$  are rational, and known to both the client and the server. We follow the approach of Shimizu et al. [55] and transform the inequality  $\text{Tv}_{\alpha, \beta}(X, Y) \geq t$  to

$$\begin{aligned} (t^{-1} - 1 + \alpha + \beta)|X \cap Y| - \alpha|X| - \beta|Y| &\geq 0 \\ \Rightarrow (a, b, c) \in \mathbb{Z}_q^3, \quad a|X \cap Y| - b|X| - c|Y| &\geq 0 \end{aligned} \quad (1)$$

for appropriate integer values of  $a, b, c$ . The server either knows  $|X|$  ( $|X| = |Q|$ ) or can compute it ( $\llbracket |X| \rrbracket = \sum_i \llbracket z_i \rrbracket$  for small domain protocols). The server also knows  $|Y_k|$  and can compute  $\llbracket \mathbf{ca} \rrbracket = |X \cap Y_k|$  using EPSI-CA-PROCESS. Evaluating the inequality requires two steps (see TV-MATCH-PROCESS):

*Step 1.* Transform coefficients  $(t^{-1} - 1 + \alpha + \beta)$ ,  $\alpha$  and  $\beta$  to equivalent integer coefficients  $a, b, c$ . We describe this in detail in Appendix B.1.

*Step 2.* Evaluate the Tversky similarity inequality (1). We convert this inequality to a two-sided equation. We know  $|X| \geq |X \cap Y_k|$  and  $|Y_k| \geq |X \cap Y_k|$ , thus

$$a|X \cap Y_k| - b|X| - c|Y_k| \leq (a - b - c)|X \cap Y_k| \leq (a - b - c)|Y_k|.$$

Client		Server
$X = \{x_1, \dots, x_m\}$ $(pk, sk)$		$\mathcal{Y} = [Y_1, \dots, Y_N]$ $Y_i = \{y_{i,1}, \dots, y_{i,n_s^i}\}$ Auxiliary MATCH-PROCESS data $\mathbb{A}$ $pk$
$Q \leftarrow (\text{SD-})\text{QUERY}(pk, X)$	$\xrightarrow{Q}$	$[\gamma_j] \leftarrow \text{MATCH-PROCESS}(pk, Q, Y_j, \mathbb{A})$ $[A] \leftarrow \text{AGG-PROCESS}(pk, \langle [\gamma_j] \rangle)$ $[R] \leftarrow (\text{SD-})\text{QUERY-CHECK}(pk, Q)$
$A \leftarrow \text{REVEAL}(sk, [A])$	$\xleftarrow{[A]}$	$[A] \leftarrow [A] + [R]$

Figure 3.4: Aggregation protocol structure.

Therefore, two sets  $X$  and  $Y_k$  satisfy  $\text{Tv}_{\alpha,\beta}(X, Y_k) \geq t$  iff

$$0 \leq a\llbracket \text{ca} \rrbracket - b|X| - c|Y_k| \leq (a - b - c)|Y_k|.$$

The server uses an inclusion test to evaluate this inequality.

### 3.8 Aggregation layer

The aggregation layer combines the outputs of the matching layer, over many sets, into a single collection-wide result. We provide four aggregation functions  $f_A$ : naive aggregation (NA-Agg), which returns outputs as is; existential search (X-Agg), which returns whether at least one server set matched; cardinality search (CA-Agg) which returns the number of matching server sets; and retrieval (Ret-Agg) which returns the index of the  $\kappa$ 'th matching server set.

Figure 3.4 shows the structure of the aggregation protocols. Upon receiving the query  $Q$ , the server runs the desired matching protocol MATCH-PROCESS (e.g., one from Algorithm 5) on each of its sets to compute the matching output  $[\gamma_j]$ . The response  $[\gamma_j]$  is zero if the set  $Y_j$  is interesting for the client and random otherwise. The server next runs an aggregation function AGG-PROCESS that takes  $N$  matching responses  $[\gamma_j]$  as input and computes the final result  $A$ . We show how to instantiate AGG-PROCESS and REVEAL in Algorithm 6. If using naive aggregation (NA-Agg), the client runs NA-REVEAL. Otherwise, the client runs REVEAL to compute the result. Finally, the server can run (SD-)QUERY-CHECK on the query and apply it to the final result  $A$  for protection.

*Naive aggregation (NA-Agg).* The naive variant runs the matching protocol

**Algorithm 6** Processing  $f_M$ .

---

```

function NA-AGG-PROCESS( $pk, \langle \llbracket \gamma_j \rrbracket \rangle$ )
  return  $\langle \llbracket \gamma_j \rrbracket \rangle$ 

function NA-REVEAL( $sk, M = \langle \llbracket \gamma_j \rrbracket \rangle$ ) ▷ Naive output
   $\text{match}_j \leftarrow \text{MATCH-REVEAL}(sk, \llbracket \gamma_j \rrbracket)$ 
  return  $\langle \text{match}_j \rangle$ 

function X-AGG-PROCESS( $pk, \langle \llbracket \gamma_j \rrbracket \rangle$ ) ▷ At least one match
   $\llbracket A \rrbracket \leftarrow \prod_{j \in [N]} \llbracket \gamma_j \rrbracket$ 
  return  $\llbracket A \rrbracket$ 

function CA-AGG-PROCESS( $pk, \langle \llbracket \gamma_j \rrbracket \rangle$ ) ▷ Number of matches
   $\llbracket b_j \rrbracket \leftarrow \text{HE.IsZero}(pk, \llbracket \gamma_j \rrbracket)$ 
   $\llbracket A \rrbracket \leftarrow \sum_{j \in [N]} \llbracket b_j \rrbracket$ 
  return  $\llbracket A \rrbracket$ 

function RET-AGG-PROCESS( $pk, \langle \llbracket \gamma_j \rrbracket \rangle, \mathcal{D}, \kappa$ ) ▷ Retrieve data for the  $\kappa$ 'th match
   $\llbracket b_j \rrbracket \leftarrow \text{HE.IsZero}(pk, \llbracket \gamma_j \rrbracket)$ 
   $\llbracket \text{ctr}_0 \rrbracket \leftarrow \llbracket 0 \rrbracket$ 
   $\llbracket \text{ctr}_j \rrbracket \leftarrow \llbracket \text{ctr}_{j-1} \rrbracket + \llbracket b_j \rrbracket$  ▷ #matches before  $j$ 'th set
   $\llbracket I_j \rrbracket \leftarrow \text{HE.IsZero}(pk, \llbracket \text{ctr}_j \rrbracket \cdot \llbracket b_j \rrbracket - \kappa)$  ▷ Create index
   $\llbracket A \rrbracket \leftarrow \sum_{j \in [N]} \mathcal{D}_j \cdot \llbracket I_j \rrbracket$ 
  return  $\llbracket A \rrbracket$ 

function REVEAL( $sk, M = \llbracket A \rrbracket$ )
  return  $\text{HE.Dec}(sk, \llbracket A \rrbracket)$ 

```

---

on all  $N$  server sets and returns  $N$  results to the client, i.e.,  $f_A(\lambda_1, \dots, \lambda_N) = \lambda_1, \dots, \lambda_N$ . This enables our framework to support many-sets when there is no need for aggregation and reduces the client's cost by computing and sending the query  $Q$  only once.

*Existential search (X-Agg).* The existential search variant determines if at least one server set  $Y_j$  is of interest to the client. Formally, the aggregation computes  $f_A(\lambda_1, \dots, \lambda_N) = \mathbb{1}[\exists i \mid \lambda_i = 1]$ . Recall that interesting sets produce zero matching responses  $\llbracket \gamma_j \rrbracket$ , so a collection will have an interesting set if and only if the product of matching responses is zero (see X-AGG-PROCESS). As responses  $\gamma_j$  are elements of the prime field  $\mathbb{Z}_q$ , their product will never be zero without having a zero response; thus, there will be no false-positives.

*Cardinality search (CA-Agg).* The cardinality search variant counts the number of interesting server sets  $Y_j$ , i.e.,  $f_A(\lambda_1, \dots, \lambda_N) = |\{i \mid \lambda_i = 1\}|$ . This aggregation (see CA-AGG-PROCESS) follows the same process as ePSI-CA and uses HE.IsZero to turn the matching responses  $\llbracket \gamma_j \rrbracket$  into binary values  $b_j$  and computes their sum.

Similar to the single set PSI-CA, we can use shuffling to convert the

naive aggregation into cardinality search with minimal computational overhead. This gain comes at the cost of increased communication as the protocol needs to send the  $N$  shuffled set responses to the client instead of a single encrypted cardinality.

*Retrieval (Ret-Agg).* The retrieval variant returns associated data  $\mathcal{D}_j$  of the  $\kappa$ th matching server set  $Y_j$ . Formally,  $f_A(\lambda_1, \dots, \lambda_N) = \mathcal{D}_j | (\lambda_j = 1) \wedge (|\{i | \lambda_i = 1 \wedge i < j\}| = \kappa)$ . Clients use this variant when they are not concerned about whether a matching set exists, but rather about information related to this matching set – such as an index ( $\mathcal{D}_j = j$ ) for retrieving records. A good example is the matching scenario where apps want to retrieve a lot of data about the matching records. Apps would first run the Ret-Agg protocol to retrieve the index of the matching record and then follow with a PIR request to retrieve the matching set’s associated data.

The Ret-Agg protocol takes an input parameter  $\kappa$  denoting that the client wants to retrieve the associated data of the  $\kappa$ th matching set. The server builds an encrypted index of interesting sets in three steps (see RET-AGG-PROCESS): (1) The server uses **HE.IsZero** to compute  $\llbracket b_j \rrbracket$  indicating if a set is interesting. (2) The server computes a counter  $\llbracket \text{ctr}_j \rrbracket$  to track how many interesting sets exist in the first  $j$  sets. (3) The server combines  $\llbracket b_j \rrbracket$  and  $\llbracket \text{ctr}_j \rrbracket$  to compute an index  $\llbracket I_j \rrbracket$  where  $I_j$  is 1 if  $Y_j$  is the  $\kappa$ th interesting set, and zero otherwise. Adding weighted  $\llbracket I_j \rrbracket$  values produces the result.

### 3.9 Security and privacy

Section 3.2.3 defines correctness (Definition 5), client privacy (Definition 6), and server privacy (Definition 7) of PCM protocols. We now discuss the correctness and privacy properties of our framework in the semi-honest and malicious settings. Our theorems rely on properties of the HE scheme, which are defined formally in Section B.3.1.

*Semi-honest setting.* When both parties are semi-honest, our protocols achieve all three security properties.

**Theorem 5.** Our protocols are correct, client private, and server private against semi-honest adversaries as long as the HE scheme is IND-CPA secure and circuit private.

We simulate our protocols in a real-world/ideal-world setting to prove

Theorem 5 in Section B.3.2.

*Malicious setting.* When the server is malicious, server privacy does not apply. Moreover, our framework does *not* provide any correctness guarantee to clients. Malicious servers can produce arbitrary, fixed (always match/no match) or input-dependent, responses to the clients without detection. However, servers cannot learn information about the clients' private input even if they misbehave.

**Theorem 6.** Our protocols provide client privacy against malicious servers, as long as the HE scheme is IND-CPA.

In Section B.3.3, we prove Theorem 6 by reducing client privacy to the IND-CPA security of the underlying HE scheme. The client's sole interaction is sending encrypted queries. The server cannot extract information from these encrypted queries without knowing the secret key or breaking the security of the HE scheme.

When the client is malicious, client privacy does not apply. The correctness property, which requires that the malicious client's output is correct, does not apply either. In Section B.3.4, we reason about server privacy when the client is malicious in a non-standard model.

*System-wide security.* We proved the security of our framework in isolation, where there is no honest interaction outside our framework. In practice, when integrating our framework into a larger system, e.g., allowing journalists to contact document owners after a document search, user interactions may leak information such as the outcome of the search. Therefore, whenever protocol designers decide to integrate our framework into a larger system, they *must perform a system-wide security analysis*.

## 3.10 From theory to practice

We first analyze the asymptotic cost of our schemes, and explain the optimizations we implement to make our schemes practical.

### 3.10.1 Asymptotic cost

Our framework is modular and supports arbitrary protocol combinations. Hence, we report the cost of layers separately. Table 3.2 summarizes the asymptotic

Table 3.2: Asymptotic cost of our protocols. See Table B.1 for a summary of our parameter definition. SD stands for small domain protocols. The column *Calls* denotes which protocols from earlier layers are called.

	Calls	Add	Mult.	Exp.
PSI	-	$n_c n_s^i$	$n_c n_s^i$	
PSI-SD	-		$ D $	
ePSI-CA	1×PSI	$n_c$		
ePSI-CA-SD	1×PSI-SD	$ D $		
F-Match	1×PSI	$n_c$		
Th-Match	1×ePSI-CA	$ T $	$ T $	
Tv-Match	1×ePSI-CA	$ T $	$ T $	
NA-Agg	$N \times \text{Match}$			
X-Agg	$N \times \text{Match}$		$N$	
CA-Agg	$N \times \text{Match}$	$N$		$N$
Ret-Agg	$N \times \text{Match}$	$2N$	$2N$	$2N$
Query check	-		$n_c^2$	
SD-Query check	-	$ D $	$ D $	

costs of modules in our framework. We report the number of homomorphic additions, multiplications, and exponentiations (counting scalar-ciphertext operations as ciphertext-ciphertext operations; but excluding operations in earlier layers). The PSI and matching layers' costs are reported for a single set (recall  $n_s^i$  is the size of the server's  $i$ 'th set). As the well-formedness check is optional, we report its one-time cost separately.

As an example, we compute the cost of using existential search with full-match interest criteria with no input domain restriction. The client runs a single X-Agg protocol, which per server set calls one F-Match and one small input PSI protocol. This leads to  $N$  multiplications in the aggregation layer,  $\sum_i^N (n_c) = N n_c$  additions in the matching layer, and  $\sum_i^N n_c n_s^i = n_c N_s$  multiplications in the PSI layer. Additionally, the server needs to perform the well-formedness check once, leading to  $n_c^2$  extra multiplications. The total cost will be  $N + N n_c + n_c N_s + n_c^2$ , but knowing that the total number of server elements ( $N_s$ ) is larger than the set number ( $N$ ) or the client size ( $n_c$ ), we simplify the cost to  $\mathcal{O}(n_c N_s)$ .

*Communication.* The client sends a query  $Q$  in the PSI layer and receives a response  $A$  from the aggregation layer. The query's size only depends on the PSI layer variant and is independent from the server's input. Small input queries are an encrypted list of client elements containing  $n_c$  encrypted scalars while

small domain queries are encrypted bit-vectors containing  $|D|$  scalar elements. The response size depends on the aggregation method. Naive aggregation has to produce  $N$  individual responses for  $N$  sets, while other aggregation methods produce a single scalar value. This results in a total cost of  $\mathcal{O}(|Q|)$  for non-naive and  $\mathcal{O}(|Q| + N)$  for naive protocols which is the minimum achievable cost.

*Computation.* The PSI layer dominates the cost of our framework. Each PCM run calls  $N$  instances of PSI leading to a cost of  $\mathcal{O}(n_c N_s)$  for small client input and  $\mathcal{O}(N \cdot |D|)$  for small input domain variants. Our cost is asymptotically higher than alternatives [16, 25, 92] and is often dismissed as “quadratic” and too expensive. In contrast, approaches [60–62, 80, 110] with “linear” cost  $\mathcal{O}(k(n_c + N_s))$ , where  $10 < k < 100$  determines the protocol’s false positive rate, are considered acceptable. The extreme imbalance requirement (RQ.3) leads to scenarios where  $n_c \ll N_s$  or even  $n_c < \log(N_s)$  and impacts how asymptotic costs should be interpreted. We show a how “quadratic” cost  $\mathcal{O}(n_c N_s)$  can outperform the “linear” cost  $\mathcal{O}(k(n_c + N_s))$  in Section 4.8.

### 3.10.2 Implementation

Fully homomorphic encryption schemes assume unbounded multiplication depth and rely on bootstrapping, which is prohibitively expensive. Thus, we use the somewhat homomorphic BFV cryptosystem [106] with a fixed multiplicative depth.

Let  $N_{deg}$  be the degree of the RLWE polynomial,  $m_{pt}$  be the plaintext modulus, and  $m_{ct}$  be the ciphertext modulus. The polynomial degree  $N_{deg}$  and ciphertext modulus  $m_{ct}$  determine the multiplicative depth of the scheme. The plaintext modulus determines the input domain. We define two sets of parameters:  $P_{8k}(N_{deg} = 8192, m_{pt} = 4,079,617)$  and  $P_{32k}(N_{deg} = 32,768, m_{pt} = 786,433)$ , and follow the Homomorphic Encryption Security Standard guidelines [111] to choose  $m_{ct}$ s that provide 128 bits of security. We use relinearization keys to support multiplication, and rotation keys to support some of our optimizations (see next section). Generating and communicating keys is expensive. Therefore, we assume that clients generate these keys once at setup and use them for all subsequent protocols. In Section B.1, we provide more details on our parameters in Table B.3 and a microbenchmark of basic BFV operations and key sizes in Table B.4.

We implement a subset of our protocols using the Go language. Our code



is open-source and 1,620 lines long.<sup>2</sup> We use the Lattigo library [112, 113] for BFV operations. Unfortunately, Lattigo does not support circuit privacy. We discuss the implications of this lack of support and possible countermeasures in Section B.3.1. We run experiments on a machine with an Intel i7-9700 @ 3.00 GHz processor and 16 GiB of RAM. Reported numbers are single-core costs. As the costly operations are inherently parallel, we expect our scheme to scale linearly with the number of cores.

### 3.10.3 Optimizations

We explain how we optimize our implementation and limit the multiplicative depth of (some of) our algorithms to improve efficiency.

*NTT batching.* We use BFV in combination with the number-theoretic transformation (NTT) so that a BFV ciphertext encodes a vector of  $N_{deg}$  elements [114]; BFV additions and multiplications act as element-wise vector operations. This batching enables single instruction multiple data (SIMD) operations on encrypted scalars. Performing operations between scalars in the same ciphertext (such as computing the sum or product of elements) requires modifying their position through rotations. Rotations require rotation keys.

Batching renders **HE.IsZero** infeasible. The exponentiation with  $m_{pt} - 1$  consumes a multiplicative depth of  $\lg(m_{pt})$  in **HE.IsZero**, so  $m_{pt}$  must be small. To use batching, however, the plaintext modulus  $m_{pt}$  must be prime and large enough that  $2N_{deg} \mid m_{pt} - 1$ . Batching does not support small  $m_{pt}$ s and consequently **HE.IsZero**; we prioritize efficiency and only evaluate variants that do not require zero detection. The parameters we use support batching.

*Replication.* The client’s query is small with respect to the capacity of batched ciphertexts, i.e.,  $|Q| \ll N_{deg}$ . We use two forms of replications to make full use of SIMD operations: powers and duplicates. When using a small input PSI variant, the client encodes *powers* of each element  $x$  (e.g.,  $[x, x^2, \dots, x^t]$ ) into the query ciphertext to reduce the multiplicative depth of **HE.IsIn**, see the second variant in Algorithm 2. Next, the client encodes  $k$  *duplicates* of the full query (including powers, when in use) regardless of the PSI variant.

Replication is straightforward when the client is semi-honest, but impacts security when the clients can misbehave. We follow a similar process to QUERY-CHECK to enforce correct replication. The server computes a second

<sup>2</sup>Code available online: <https://github.com/spring-epfl/private-collection-matching>

randomizer  $\llbracket R \rrbracket$  such that  $R$  will be zero when (1) all duplicates are equal and (2) for all consecutive power elements  $x_i$  and  $x_{i+1}$ , the equality  $x_{i+1} = x_1 \cdot x_i$  holds. The server needs to compute this check only once per query. We implemented this check and included its cost in all figures.

## 3.11 PCM in practice

To demonstrate our framework’s capability, we solve the chemical similarity and document search problems. We discuss matching in mobile apps in Appendix B.5. We focus on end-to-end PCM solutions and do not evaluate single-set protocols or scenarios.

### 3.11.1 Chemical similarity

Recall from Section 3.2.1 that chemical similarity of compounds is determined by computing and comparing molecular fingerprints [53, 54, 64–67]. We use the Tversky similarity matching algorithm, Tv-Match, to compute whether a compound in the seller’s database is similar to the query compound. As fingerprints are short, we instantiate Tv-Match with the small-domain ePSI-CA-SD and represent molecular fingerprint as bit-vectors.

We follow a popular configuration [55] where compounds are represented by 166-bit MACCS fingerprints [115] and Tversky parameters are set to  $\alpha = 1, \beta = 1, t = 80\%$ . Processing these raw parameters (see Algorithm 5) leads to the inequality

$$\begin{aligned} a, b, c = 9, 4, 4 &\leftarrow \text{Tversky-param-process}(1, 1, 0.8) \\ 0 \leq 9\llbracket \mathbf{ca} \rrbracket - 4|X| - 4|Y| &\leq |Y|. \end{aligned}$$

We evaluate two aggregation policies. We apply X-Agg aggregation to determine whether at least one compound in the database matches and CA-Agg to count the number of matching compounds. These variants have high multiplicative depth, so we modify them to enable efficient deployment without relying on bootstrapping.

*Existential search.* The X-Agg protocol applied to a collection of  $N$  compounds requires a multiplicative depth of  $\lg(N)$  to compute  $\prod_{j \in [N]} \llbracket \gamma_j \rrbracket$ . This depth is too high for our parameters. Instead, we relax the output requirements and

reduce the output as much as possible: For a fixed depth  $l$ , we return  $\lceil N/2^l \rceil$  encrypted scalars  $\llbracket \lambda_k \rrbracket = \prod_{k2^l \leq j < (k+1)2^l} \llbracket \gamma_j \rrbracket$  to the client. This relaxation reduces the privacy of X-Agg – it is less than full X-Agg but better than CA-Agg – at the gain of efficiency. If this reduced privacy is unacceptable, the client just needs to choose larger HE parameters.

*Cardinality search.* We use the shuffling variant of the CA-Agg protocol due to its lower multiplicative depth. When the server shuffles the  $N$  matching responses, the client only learns the number of interesting compounds. Shuffling batched encrypted values is hard, so the server shuffles the compounds (server sets) as plaintext before processing the query to the same effect.

Our modifications to both aggregators increase the transfer cost from a single aggregated result to linear in the number of server sets. Yet, the X-Agg protocol sends 1 scalar value per  $2^l$  compounds.

*Evaluation.* We evaluate our similarity search on the ChEMBL database [116, 117], which contains more than 2 million compounds and is one of the largest public chemical databases. The database contains compounds in the SMILES format. We use the RDKit library [118] to compute MACCS fingerprints from this format.

Figure 3.5 shows the cost of running our protocols with the BFV parameter set  $P_{32k}$ . We ran 5 times the experiments with databases containing up to 256k compounds, and 3 times the larger experiments. We report the average cost. Standard errors of the mean are small, so error bars are not visible.

The server batches 128 compounds per ciphertext. Performing the PSI layer protocol ePSI-CA only requires 1 multiplication per ciphertext, while computing the binary Tversky score requires a depth of 7. For the X-Agg protocol, we set  $l = 6$  and aggregate up to 64 matching results into each scalar  $\llbracket \gamma_k \rrbracket$ . As each ciphertext holds 32k scalars, the aggregation of up to  $2M$  server sets requires 1 ciphertext. CA-Agg transfers 1 ciphertext per 32k compounds. The cost of the extra X-Agg multiplications is insignificant compared to computing similarity, leading to similar computation costs for both protocols.

The client computation of searching among 2 million compounds is less than a second and the transfer cost is 12MB for X-Agg and 378MB for CA-Agg. These protocols can be run by a thin client. The server, however, requires 3.5 hours of single-core computation.

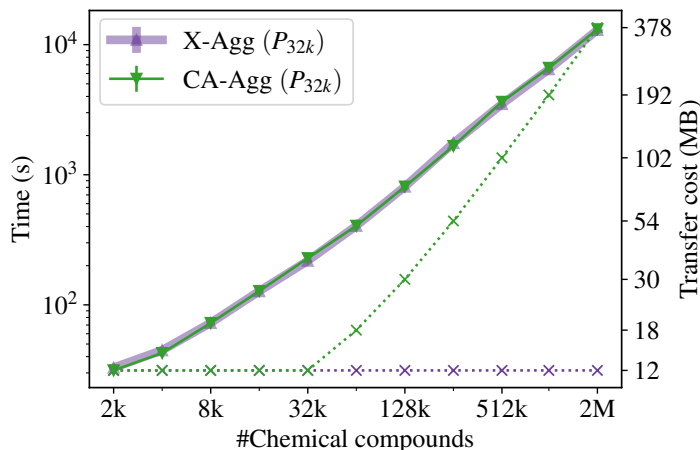


Figure 3.5: Computation time (solid lines) and transfer cost (dotted lines) for computing aggregated chemical similarity.

**Comparison to ad-hoc solution.** Shimizu et al. [55] build a custom chemical search that computes the number of matching compounds. They offer the same functionality/privacy as our CA-Agg protocol, but reveal more information than our X-Agg variant. Shimizu et al. protect privacy against malicious adversaries, but malicious servers can violate correctness. They use the secp192k1 curve which provides less than 100-bit of security, while we offer full 128-bit security. Moreover, their use of differential privacy requires distributional assumptions. Shimizu et al. report the cost of searching *1.3 million compounds* as: 167 seconds of server computation, 172 seconds of client computation, and 265MB of data transfer. Our X-Agg and CA-Agg solutions achieve higher security, lower client computation, and lower ratio of bandwidth consumed per compound, at the cost of higher server computation.

### 3.11.2 Peer-to-Peer document search

To implement peer-to-peer document search, we represent queries and documents by the sets of keywords they contain. A single document, represented by the keyword set  $S$ , is of interest to the querier if it contains all query keywords  $Q$ , i.e.,  $Q \subseteq S$ . This functionality can be implemented with the full matching (F-Match) variant.

The client and the server must agree on how keywords are represented in  $\mathbb{Z}_q$ . We use hash functions to do the conversion. As the search queries typically contain few keywords and the domain for searchable keywords is too large for our small domain variant, we construct F-Match with PSI with a small client

input. There are two sources of false-positive in our setup:

*False positive of mapping words.* The parameter  $m_{pt}$  (recall  $m_{pt} = q$ ) determines the input domain. Since  $m_{pt}$  is small, multiple words could be mapped to the same  $\mathbb{Z}_q$  element. This can lead to F-Match claiming there is a match, even though one of the colliding keywords is absent in the server’s set. Since  $m_{pt}$  impacts the multiplicative depth of our HE schemes, choosing a large value is impractical.

Instead of directly increasing the size of  $m_{pt}$  to reduce the false positive rate, the client hashes the  $n_c$  query keywords with  $t$  hash functions and encodes them into  $t n_c$  scalar values, which reduces the false-positive rate to  $1/q^t$ . When running PSI-PROCESS, the server knows the corresponding hash function for each scalar value and hashes them accordingly. Afterward, the server runs the F-Match protocol on all  $t n_c$  PSI outputs together; a document matches if and only if all hashed keywords are present.

Using multiple hashes to reduce false positives does not impact privacy, as it is straightforward to simulate a query with  $t$  hashes, with  $t$  F-Match queries. This modification increases the computation cost and the number of scalar values in the query by a factor of  $t$ . However, there is no concrete change in the communication cost as the client can still pack  $t n_c$  scalars into one ciphertext.

*False positive of F-Match.* The F-Match protocol itself has a false positive rate of  $\sim 1/q$  (see Section 3.7) caused by internal randomness. An easy way to reduce this FP rate is to run  $r$  instances of F-Match with different randomness and reveal the  $r$  responses. This repetition reduces the FP rate of a single matching response to  $\sim 1/q^r$ , while increasing the server’s computation cost by a factor  $r$ .

*Aggregation.* We explore two aggregation policies: existential (X-Agg) to determine whether at least one document in the collection matches; and cardinality (CA-Agg) to count matching documents. Since the multiplicative depth of F-Match is low, we can fully reduce the X-Agg output to one encrypted scalar. For the CA-Agg variant, we still use the shuffling variant for lower multiplicative depth.

*Evaluation.* We use the parameters  $P_{8k}$  for CA-Agg and  $P_{32k}$  for X-Agg protocols. We base our evaluation on the requirements set out in EdalatNejad et al. [56]. We limit the number of keywords in each query to 8 and generate random documents of 128 keywords. We represent each keyword with two hash

functions leading to false-positive rates of  $2^{-44}$  for CA-Agg and  $2^{-39}$  for X-Agg due to the mapping to  $\mathbb{Z}_q$ . We need to account for false-positive errors as we run a single F-Match per document. This error increases with the number of documents and reaches its peak at 8k documents where the probability of overestimating the cardinality is 0.2% (CA-Agg) and existence is 1% (X-Agg). Our protocols do not have false negatives. We skip QUERY-CHECK from Section 3.6.3 since F-Match is not impacted by repeated keywords in the query, but still perform checks from Section 3.10.3 to enforce honest batching. Our use of power replication leads to a multiplicative depth of 1 in the PSI layer while F-Match only uses addition. We report the cost of our document search in Fig. 3.6 and discuss our performance in Section 3.11.3.

**Comparison to ad-hoc solution.** EdalatNejad et al. [56] build a document search engine that performs PSI-CA in a many-set setting and post-processes the output, in the client, to detect relevant documents. EdalatNejad et al. protect privacy against malicious adversaries, but malicious servers can violate correctness. To enhance performance, they sacrifice privacy and leak more information than individual intersection cardinalities, yet less than vanilla PSI. The protocol of EdalatNejad et al. has a latency of less than 1 second to search 1k documents while our framework requires 8.7s for CA-Agg and 54s for X-Agg. Our CA-Agg search does not reveal information about individual documents and X-Agg *only reveals a single bit about the collection*. As expected, this privacy gain comes at a performance cost.

### 3.11.3 Comparison with generic solutions

After comparing against ad-hoc solutions, we compare our *document search* against generic SMC and circuit PSI that can offer the same privacy and functionality (see Table 3.1). See Section B.4.2 for implementation details. In Section B.4.3, we compare our document search to one of the fastest OT-based solutions, which *cannot* satisfy our privacy requirements.

**Generic SMC.** We use a semi-honest SMC compiler, EMP tool-kit [119], to build two custom search engines. Our circuits take 1 client and  $N$  server sets as input. They compute PSI and F-Match using boolean logic then aggregate the  $N$  matching results following either X-Agg or CA-Agg policy. We call these two approaches ‘SMC-X-Agg’ and ‘SMC-CA-Agg’.

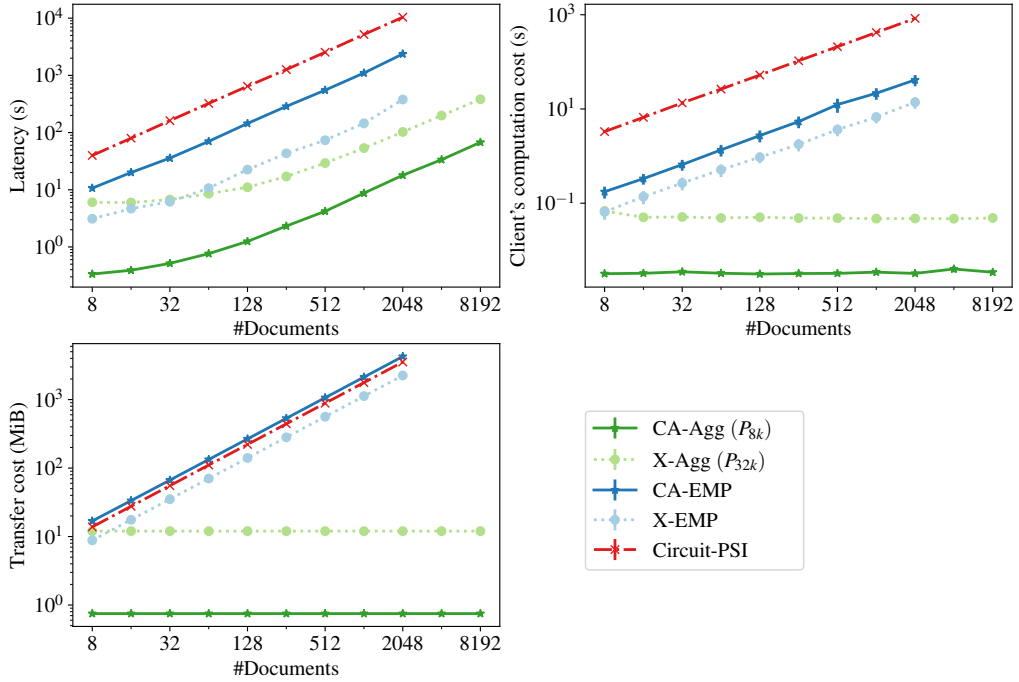


Figure 3.6: The end-to-end search latency (left), client’s computation cost (middle), and communication cost (right) of document search. We limit execution to a single CPU core and enforce a bandwidth of 100 Mbps and an RTT of 100 ms on the connection.

**Circuit-PSI.** We compare against Chandran et al. [81] which is a state-of-the-art single-set circuit-PSI protocol. To support many-set, we run one PSI per document. Chandran et al. support extending the intersection computation with arbitrary circuits allowing us to perform F-Match and then X-Agg or CA-Agg aggregation. However, we consider the cost of computing the intersections as a lower bound for the cost of search and do not extend the circuit.

Figure 3.6 reports the latency, client’s computation, and transfer cost of document search. We repeat each experiment 4 times and report the average cost and the standard error of the mean. Unfortunately, the SMC and circuit-PSI solutions crash on runs with 4k documents and more, hence we only show entries up to 2k documents. The stable trend of our measurements let us think that one could easily extrapolate results for those settings from the measurements that we report. To estimate the latency of our framework, we run it in a single process and add the expected network time (as we have one round, we consider 1 round trip time plus transfer time).

Now we show that our framework supports thin clients by significantly reducing the clients’ computation and communication costs. Moreover, our

framework provides better latency and in Section B.4.2 we show that we have a competitive server cost.

*Latency.* Both SMC and Circuit-PSI require many rounds of communication and are thus heavily impacted by the network’s round-trip time (RTT). We assume a transatlantic RTT of 100 ms. In this setting, our framework consistently outperforms competitors. When performing existential search (X-Agg), our framework cuts the end-to-end latency in half while our improvement factor increases to 96x when doing a cardinality search (CA-Agg) on 1k documents.

*Client’s computation.* Circuit-based approaches have a balanced computation load between clients and servers while we outsource almost all the computations to the server. The client’s computation cost of our framework is independent of the number of the server sets and is only 50 ms for the X-Agg and 5 ms for the CA-Agg search. While slower, the SMC approach is computationally efficient and only requires 3.7s for the existential (X-Agg) and 11.3s for the cardinality (CA-Agg) search for a 1k document collection. SMC’s cost is still acceptable for thin clients, however using our framework leads to a saving factor of 75–2250x on client computation. In contrast, Chandran et al. require a prohibitive cost of 352 seconds, which is 70,000x higher than ours.

*Communication.* The X-Agg protocol has a constant communication cost of 12 MB while the cost of CA-Agg grows linearly with the number of documents. However, this cost is fixed to 768KB for CA-Agg in our evaluation since we can pack up to 8k results in a  $P_{8k}$  ciphertext. Both SMC and Circuit-PSI have costs linear in the inputs size. The SMC search requires 1.1 GiB for X-Agg and 2.1 GiB for CA-Agg queries when searching 1k documents while Chandran et al. require 1.7 GiB; thus, both approaches are prohibitively expensive. Ultimately, our framework reduces communication by a factor of 93–2800x.

### 3.12 Takeaways and future work

In this work, we introduce and formalize private collection matching problems. Using homomorphic encryption, we build a modular framework for solving them. Our framework and its layers-based design simplify the construction of PCM schemes that achieve complex goals while limiting the leakage to what is required by the functionality, nothing more. Relying on homomorphic encryption is extremely advantageous in theory. Our work shows, however, that using it in practice is challenging. We have overcome these challenges by using optimizations from the literature, at the cost of reduced flexibility of our



framework. Ultimately, our framework is competitive with ad-hoc solutions and outperforms generic approaches in all three latency, communication, and computation costs, sometimes by several orders of magnitude. As example, our framework requires 12MB of communication and less than a second of client computation to determine matching of a chemical compound against a database of 2 million compounds; or respectively 768KB and 50ms to determine whether the owner of a thousand documents has content of interest to a querier journalist. We believe further work on homomorphic encryption schemes – and ciphertext-based comparison methods in particular – will allow our framework to operate in a wider range of settings. We hope that our work fosters the evolution of homomorphic encryption in further areas so that the community can build a wider range of privacy-preserving applications.



# CHAPTER 4

## Janus: Safe Biometric Deduplication for Humanitarian Aid Distribution

This chapter is based on the following article:

Kasra EdalatNejad, Wouter Lueks, Justinas Sukaitis, Vincent Graf Narbel, Massimo Marelli, Carmela Troncoso: “Janus: Safe Biometric Deduplication for Humanitarian Aid Distribution”. Under submission 2023.

### 4.1 Introduction

Humanitarian organizations have a long history of providing aid to people in crisis. To maximize impact, they wish to distribute aid among as many recipients as possible given their limited budget. Thus, recipients should receive aid only once for each time that aid is distributed. Wang et al. propose privacy-friendly mechanisms to ensure that recipients enrolled in an aid-distribution system can *receive* aid at most once per round [120]. Their work relies on the assumption that recipients cannot register more than once.

In this chapter, we tackle the problem of *preventing double registration*. To understand the requirements behind double registration prevention in the humanitarian sector, we collaborate with the International Committee of the Red Cross (ICRC). We learn that common ways to tackle this issue are to rely on the use of official government-issued identity documents or on the input of local trusted sources of information (e.g., community representatives).

Both methods have shortcomings. Government-issued IDs are not universal. Aid recipients may not have a reliable government identity document, or may not have these documents in their possession at registration time (e.g., after evacuating their homes under pressure in conflict zones). This means

that people in need may be refused by the system, opposite to the ‘humanity’ and ‘impartiality’ humanitarian principles stating that aid and protection must be given to everyone that needs it [121]. Reliance on community representatives may not be available in every scenario, e.g., in dangerous situations. Moreover, it suffers from efficiency issues. Verification by local actors is slow and thus typically can only be done after registration when prevention may not be as useful. Additionally, both of these methods may increase the risks for the participants in the aid program as they create a single point of failure (e.g., a paper list, a community representative) which, if compromised, reveals the identity of all participants. This membership may be associated with, for example, political or religious beliefs which can be pursued under some governmental regimes [122].

A natural path to address the issues above is digitalization. Our conversations with the ICRC reveal that, besides addressing the shortcomings of current proposals, digital solutions need to also minimize the amount of (personal) data collected. This is not only to uphold the fundamental right of aid recipients to personal data protection and privacy, but to also help the organization avoid sharing data with third parties (e.g., financial service providers) as such sharing may put recipients in danger especially when those institutions are closely linked with local government or cross-national financial institutions.

A digital means that humanitarian organizations have started to use is biometrics [123]. Biometrics are available in many more situations than identification documents, and the digital nature of biometrics systems helps speed up the registration process. However, existing biometric-based solutions come at a high risk for recipients. They require the collection of biometric samples into databases which are typically stored in the clear, becoming a tempting target for entities seeking information about program participants [122]. Moreover, due to the nature of biometric data, such databases can reveal sensitive information about health or ethnicity [124, 125], and because biometrics are unique they can be used to link entries across databases. These risks are exacerbated by the long lifespan of biometric data.

Secure and privacy-preserving biometrics solutions [126–132], which would enable the use of encrypted databases to store the biometric templates, cannot be directly applied to the humanitarian setting. They often do not provide the low error rates required to ensure that legitimate recipients are not erroneously refused from the aid-distribution program. This is because typical error-oriented measures, such as allowing multiple biometric match attempts for authentication, are not a suitable solution for the humanitarian context as

in which participants in the matching might want to evade detection.

In this chapter, we introduce Janus, a biometrics-based deduplication system tailored to the humanitarian sector needs. Janus only reveals a single *bit* of information determining whether a registration request is valid. Instead of creating a plaintext biometric database, Janus either secret shares biometric data between the registration station and a humanitarian organization-operated biometric provider, encrypt data using a quantum secure HE scheme, or seals the data in a secure enclave. This approach provides strong privacy protections by ensuring that not even the humanitarian organization has access to plain biometric data; provides long-term protection of sensitive biometric data in the case of data breaches even in the presence of quantum computers; and avoids creating a high-value target for hackers. Finally, Janus enables humanitarian organizations to remotely disable query access, even when the registration station is physically compromised during armed conflicts.

Our work makes the following contributions:

- ✓ We elicit the functionality, safety, and deployment requirements of protecting against double registration in the context of humanitarian aid distribution.

- ✓ We propose Janus, which addresses these requirements. We build three instantiations of Janus based on secure multiparty computation, somewhat homomorphic encryption, and trusted execution environments. We show that Janus can support multiple biometric sources including fingerprints, irises, and facial recognition; as well as biometrics alignment and fusion.

- ✓ We implement and evaluate Janus’s instantiations to demonstrate that they can satisfy the requirements of humanitarian organizations.

*Ethical considerations.* We do not collect or use biometric data in our evaluation. Therefore, there are no ethical concerns with our experiments.

## 4.2 Deduplication for aid distribution

We take as starting point for our work, the aid distribution scenario identified by Wang et al. [120], depicted in Fig. 4.1. This scenario is largely based on the aid distribution process of the International Committee of the Red Cross (ICRC). In the registration phase, potential *aid recipients* (or recipients, for short) visit a *registration station* (RS) and requests aid. If this person is eligible for aid, the registration station enrolls them into the system and allocates

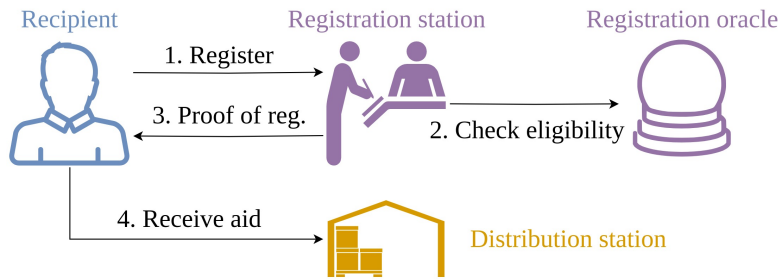


Figure 4.1: Humanitarian aid distribution workflow [120].

them an aid budget. Registration stations are often directly controlled by the humanitarian organization or other trusted local parties. In the case of the ICRC, these stations are protected by the ICRC’s special privileges and immunity [133]. To ensure the legitimacy of aid recipients, we assume, like Wang et al. [120], the existence of a registration oracle that determines eligibility during registration. This role may be fulfilled by verifying recipients’ ID documents, checking against local government lists, or asking community representatives. (See Section 4.3 for why these do not suffice for deduplication.)

At the distribution phase, registered users visit aid distribution centers, show proof of their registration, and receive the aid they are entitled to. As security with respect to the distribution station has been tackled by Wang et al., we leave it out of the scope of this work together with the distribution auditing step included in [120].

#### 4.2.1 Deduplication requirements

We collaborate with the International Committee of the Red Cross to understand the requirements associated with preventing double registration, i.e., ensuring that recipients can only register *once* at the registration stations. We gather the requirements for deduplication through conversations with the ICRC’s Data Protection Office and from ICRC documents on data protection and biometric policy [134, 135].

**Functional requirements.** The deduplication mechanism must provide the following functionality:

*RQ.F1: Identify duplicate aid requests.* The registration station needs to determine if a potential recipient is already registered to proceed with a reg-

istration request. For this, the deduplication system must provide a *single bit* indicating the potential recipient's presence or absence in the database of registered users.

*RQ.F2: Dynamic addition.* If a registration is successful, the newly accepted individual must be added to the registered users database. Newly accepted users must be added individually so that attempts to re-register can be immediately detected. As many recipients may need to be added in a short amount of time, adding individual users to the database should be efficient and not require an operation on all elements in the database.

*RQ.F3: Low failure rate.* Recipients that are eligible should not be erroneously denied registration. Humanitarian organizations should be able to set an arbitrarily low failure rate for the deduplication system on falsely detecting a new recipient as a duplicate.

*Non-goals.* The deduplication system only provides information for the registration station to make decisions regarding the legitimacy of the registration. The deduplication system does not aim to prevent the registration station from proceeding with registration in case of a detected duplicate. For such cases, the humanitarian organization must rely on other processes that ensure that the staff at the registration station use the deduplication information correctly.

**Safety requirements.** Augmenting aid distribution systems with technology may bring new risks to recipients. The introduction of deduplication detection must minimize the introduction of such risks.

*RQ.S1: Single functionality.* The output of the deduplication system should not reveal more information about recipients than the binary answer necessary to detect duplicates. Aid recipients often belong to sensitive groups such as refugees or survivors of violent incidents where their data is of interest to various adversaries such as governments. While some humanitarian organizations (such as the ICRC) enjoy legal protection that prevents subpoenas, creating a valuable database increases the risk for recipients if the data is lost or stolen via hacking attempts [136].

Therefore, to ensure single functionality, deduplication systems should limit the storage of recipients' personal data, and ensure that any personal data stored cannot be used to recover the identity or any other traits of the recipients. This should also hold for adversaries that have access to other data sources and try to link recipient data to these sources.

*RQ.S2: Protection against passive compromise.* Traditional mechanisms for deduplication typically create a single point of failure (e.g., a paper list, a database community representative) which, if compromised, reveals the identity of all participants and their data. This membership may be associated with, for example, political or religious beliefs which can be pursued under some governmental regimes. To increase safety, a deduplication system should ensure that as long as an adversary compromises at most one actor in the aid distribution system *at the same time*, this adversary cannot learn any information about recipients' data.

*RQ.S3: Protection against active compromise.* Humanitarian organizations operate in dangerous and volatile situations, where the registration station is at risk of being physically compromised during armed conflicts. In such scenarios, even the single-bit answer resulting from a membership query for a humanitarian program may put recipients at risk. For example, when the Taliban took control of Afghanistan, they got access to biometric devices left behind by the US Army. These devices contained data about Afghan civilians, and the Taliban used them to determine who had a relation to the US Army [122]. Deduplication systems should prevent that third parties can query the system without involving the humanitarian organization.

**Deployment requirements.** Following humanitarian principles, aid programs have to enable humanitarian organizations to serve a large number of users in very diverse environments. We derive the following requirements.

*RQ.D1: Universality.* Humanitarian organizations aim to bring assistance without discrimination. Thus, deduplication systems should only rely on information available in all kinds of contexts, including conflict zones.

*RQ.D2: Medium scalability.* Humanitarian organizations help millions of humans around the world. However, the process of providing aid to recipients is usually localized in geographically-diverse regions where a conflict or disaster happens. Deduplication is only needed within a given aid program in one of these regions. A typical ICRC program supports 1000–10,000 recipients. Occasionally, there are larger projects that support up to a 100,000 recipients. We aim to efficiently operate at the medium scale (around 10,000) so that humanitarian organizations can rely on commodity devices and limited network connections, which are typically the only infrastructure available in many of the settings where these organizations operate. To scale to larger projects, humanitarian organizations would require higher computational and commu-



nication capabilities, which may limit the scope of application of a solution.

### 4.3 Towards a safe deduplication system

We now discuss methods that humanitarian organizations use, or could use, to prevent duplicate registrations.

*Government-issued ID.* Government IDs are a simple form of identification. However, they cannot be assumed to be available in humanitarian contexts. Humanitarian organizations often operate in areas there may be no government, or the government refuses to issue IDs to certain groups. Additionally, people fleeing conflict zones often do not prioritize protecting their IDs, and losing them is not uncommon.

*Other unique identifiers.* An alternative to government IDs is to use government-based (e.g., social security numbers [137]) or commercial-based (e.g. phone numbers or social network aliases) identifiers. These identifiers, however, may not be available everywhere, or it may be easy to obtain several, and given their uniqueness act as (pseudo-)identifiers that can be used to link databases.

*Trusted local actors.* Humanitarian organizations may rely on trusted local actors, such as local committees, focus rooms, or local community leaders, to identify and verify potential aid recipients. However, this approach is limited by the availability of these trusted actors. Moreover, while local actors are suitable for determining eligibility, humans are not good at remembering who has registered over a long span of time to prevent double registration.

*Biographic information.* The above measures are sometimes complemented by the use of biographic information to better separate records [138]. This can be done using statistical patterns or fuzzy matching, reducing the number of false positives. This method, however, requires the availability of such biographic information, as well as storing it with the corresponding risk.

*Biometrics.* Humanitarian organizations [123] have started to use human biometrics – under the assumption that they do not change, and are almost always present – to implement deduplication while avoiding the weaknesses of the previous methods. For example, the UNHCR introduced the Biometric Identity Management System [139] and the World Food Program (WFP) uses SCOPE [140]. Yet, using biometric data brings serious security and privacy concerns. Biometric data are personal, sensitive, and can be used for purposes

beyond deduplication. This creates risks for humanitarian organizations in forms of surveillance, hacking attempts, or pressure to share data and risk for aid recipients in forms of losing asylum or extradition [123, 141–143].

Our discussion with the ICRC’s Data Protection Office revealed that while government IDs, identifiers, biographic information, and trusted local actors may be suitable for specific aid programs, their lack of availability and accuracy prevents them from fully meeting the functional needs of the ICRC. Therefore, in this work, we focus on biometrics-based solutions combining them with privacy-enhancing technologies to mitigate the risks arising from their use.

**Biometric-based additional requirements.** Due to their sensitive nature, the use of biometric data requires stricter protection measures to prevent harm. For instance, the ICRC’s biometrics policy [135] sets strict conditions under which biometric data may be used. This policy also requires that, in general, biometric materials are stored on users’ devices. However, this requirement is not compatible with preventing double registrations: we cannot trust malicious recipients to provide biometric data from their previous registrations. Therefore, an effective biometric-based deduplication system must store biometric data on devices operated by the humanitarian organization.

To ensure that the centralized biometric storage does not increase risk (and complies with the biometrics policy) when biometrics are involved, the *single functionality* requirement (RQ.S1) implies that the mechanism must prevent the extraction of the biometric templates (or the recovery of the biometric samples), and any inference on the similarity between two biometric samples. Crucially, the information that the deduplication mechanism requires should not enable the authentication of users.

Finally, given the long lifespan of biometric data and its potential for impersonation and inferring information even decades after collection, we introduce a new requirement:

*RQ.S4: Long-term safety.* The deduplication system should be designed to maintain the protection of the information it stores for at least 20-30 years.

## 4.4 Janus' architecture

In this section, we introduce the architecture of Janus, a biometric-based deduplication system that fulfills the requirements in Sections 4.2 and Section 4.3.

*Non goals.* Janus focuses on preventing double registrations. Our design does *not* aim to: (1) *improve plaintext biometrics* – we use existing biometric approaches as black boxes; or (2) *secure biometric readings* – the registration station has physical access to plaintext biometric samples during registration (via the registration devices). It is trusted to delete them at the end of the registration. Using biometric sensors with hardware protection would eliminate this need.

### 4.4.1 Janus-enabled registration workflow

In order to fulfill the requirements for protection against passive and active compromise (RQ.S2 and RQ.S3), the system must include an extra actor besides the registration station and the recipients (which are malicious in our setting). To this end, we introduce a *biometric provider (BP)*, an honest-but-curious actor under the control of the humanitarian organization. The biometric provider should not learn any information about aid recipients' biometric data or the success of their registration attempt. We also consider the registration station to be honest-but-curious, and that there is no collusion between the registration station and biometric provider.

In Section 4.6, we propose three different instantiations of Janus' design with different trade-offs. Each of these designs, however, implement the same set of four protocols. A **Setup** procedure initializes the two parties and sets up key material, and initialize a distributed database of registered recipients **db**. To test for membership of a new recipient in the database **db**, the two parties jointly run the **Membership** protocol. To add a new recipient to **db**, the two parties run the **AddMember** protocol. Finally, to provide proactive security, the parties can run the **Ratchet** protocol to recover from earlier compromises of either party.

When including Janus to prevent double registration, the registration workflow is as follows (see also Figure 4.2):

- (i) A potential aid recipient visits the registration station (RS) to register (step 1). The RS determines the eligibility and the aid entitlement of the

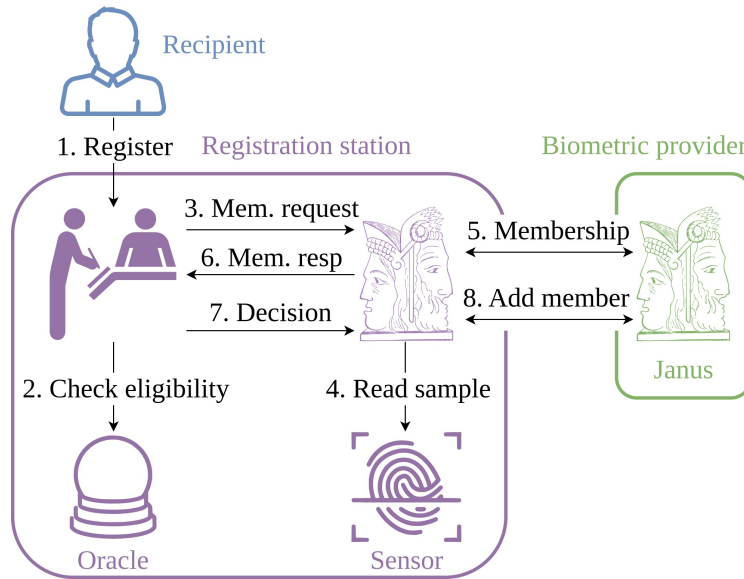


Figure 4.2: An overview of registration workflow.

recipient (step 2).

- (ii) The RS sends a membership request to the RS component of Janus (step 3). The RS then obtains a biometric sample from the recipient (step 4) and runs the **Membership** protocol with the biometric provider component of Janus (step 5). Janus returns the membership response (step 6).
- (iii) Upon receiving the membership response from Janus, the RS determines if the registration should proceed or be aborted. If successful (step 7), the Janus components in RS and BP run the **AddMember** protocol to add the new recipient to the user database (step 8).
- (iv) Regardless of the outcome, the registration station *deletes* all the plain-text biometric data, and it informs the recipient of the registration's outcome.

From now on, we simply write registration station (RS) to refer to the RS component of Janus and biometric provider (BP) to refer to the BP component of Janus.

#### 4.4.2 Requirements achieved by design

Our design choices for Janus help ensure we meet the requirements in Section 4.2. By design, the interface provided by the **Membership** and **AddMember** protocols satisfy the functional requirements to be able to query for mem-

bership (RQ.F1) and to add new recipients (RQ.F2). Because Janus uses biometrics, it ensures universality (RQ.D1).

In all instantiations, the separation of trust between the RS and BP ensures that Janus protects against passive compromises of any single party (RQ.S2). Moreover, because the instantiations support ratcheting, the system can recover from passive compromises of any single party as long as `Ratchet` is run between compromises (RQ.S2).

Our three instantiations of Janus differ in the cryptographic building blocks for distributing trust between the RS and the BP; which reflects on how they guarantee that each query reveals only a single bit (RQ.S1) and how they achieve scalable designs (RQ.D2).

*Achieving non-collusion.* The non-collusion assumption between the RS and the BP is critical to satisfying a number of requirements. In the context of the ICRC, this assumption can easily be realized. The ICRC has a hierarchical structure. The headquarter (HQ) of the ICRC is located in Switzerland, and it has delegations in over 50 countries. The ICRC also has local branches in the 190 countries in which it operates.

The HQ and majority of delegations are located in physically secure and stable locations and have their own IT infrastructure. Delegations and the HQ are autonomous and suitable for the role of the biometric provider. Registration stations need to be closer to potential aid recipients who need to visit the station in person and provide biometric samples. This makes local branches a suitable choice for the role of the registration station. The local branches have the IT infrastructure necessary to support gathering and processing biometric samples which enables them to actively participate as a computing party in the deduplication. This internal structure promotes a significant level of independence between headquarters, delegations, and local branches. With the roles above, the RS and BP are typically situated in distinct geographical locations and have separate IT systems; and both benefit from a special privilege of non-disclosure of confidential information [133], which gives them judicial immunity against subpoenas and requests for data.

## 4.5 Biometrics

We introduce a simplified and generic abstraction for biometric operations. We first define biometric templates then show how to match two biometric

templates to determine if they belong to the same user. We demonstrate that our abstraction supports matching different biometric sources such as fingerprint, iris, and face in Section 4.7.

**Notation.** We write  $x \leftarrow^{\$} X$  to denote that  $x$  is drawn uniformly at random from the set  $X$ . Let  $q$  be a positive integer, then  $\mathbb{Z}_q$  denotes the set of integers  $[0, \dots, q)$  and  $\mathbb{Z}_q^*$  represents the elements of  $\mathbb{Z}_q$  that are co-prime with  $q$ . We write  $\langle a_i \rangle_m$  to represent a list of  $m$  elements  $[a_0, \dots, a_{m-1}]$ .

**Biometric template.** Biometric sensors produce raw images. These images are processed into biometric templates to facilitate matching. We use templates modeled as a *fixed-sized* array of **TS** values taken from an integer domain  $\mathbb{Z}_d$ . We denote the process of reading and processing a biometric sample from a sensor as:

$$\begin{aligned} S &\leftarrow \text{Sensor}(\text{TS}, d) \\ B \in \mathbb{Z}_d^{\text{TS}} &\leftarrow S.\text{ReadBio}(\text{recipient}). \end{aligned}$$

We use  $B[i]$  to denote the  $i$ 'th value in a biometric template. The process of reading biometrics is probabilistic and multiple readings of the same biometric may result in slightly different templates.

**Biometric matching.** Matching takes two biometric templates  $X$  and  $Y$  and determines if they belong to the same person. We formalize the matching process as:

$$0/1 \leftarrow \text{Match}(X, Y, D, t) = (D(X, Y) < t).$$

where  $D$  is a distance measure that determines the similarity between two templates, and  $t$  is a threshold that determines whether two templates correspond to the same person. The choice of the threshold  $t$  impacts the error rate of the system and creates a trade-off between false acceptance and rejection (see Section 4.7). This parameter is non-sensitive and can be hardcoded into the system. When the distance and threshold are clear from the context, we drop them from the notation.

In this work we use three distance measures: *Euclidean distance* ( $D.\text{Euclidean}(X, Y) = \sum_i (X[i] - Y[i])^2$ ), *Hamming distance* ( $D.\text{Hamming}(X, Y) = \sum_i (X[i] \oplus Y[i])$ ), and *normalized Hamming distance* ( $D.\text{NormHamming}((X, M_X), (Y, M_Y))$ ),

where  $M_\alpha$  denotes a mask of a template used to exclude certain values during comparison). We include Euclidean and Hamming distances in the body of the thesis and address the normalized Hamming distance in Section C.2.

**Alignment.** Biometric samples may need alignment before matching. We represent the aligned matching of templates  $X$  and  $Y$  as producing  $a$  aligned templates  $\langle X_i \rangle_a \leftarrow \text{Align}(X, a)$  from  $X$  then computing pairwise matching between  $Y$  and the aligned  $X_i$  templates as follows:

$$M \in \{0/1\} \leftarrow \text{AlignedMatch}(X, Y, D, t, a) = \begin{cases} \langle X_i \rangle_a \leftarrow \text{Align}(X, a) \\ d_i \leftarrow D(X_i, Y) \\ M \leftarrow \bigvee_{i=1}^a (d_i < t). \end{cases}$$

Janus only supports alignment approaches that do not need to have plaintext access to both  $X$  and  $Y$  at the same time. We discuss how to instantiate **Align** for different biometric sources in Section 4.7.

## 4.6 Instantiating Janus

We design three Janus instantiations that use secure multiparty computation, homomorphic encryption, and trusted hardware to offer different trade-offs. These instantiations are inspired by existing biometric systems that we modify to satisfy our specific requirements.

### 4.6.1 SMC-Janus

Our first instantiation uses secure multiparty computation (SMC) between the RS and the BP. We are not the first to propose the use of SMC in the biometrics context. Existing privacy-preserving biometric identification approaches use SMC to compute the distances between a client sample and a database of  $N$  samples held by a server [131, 144, 145]. These works prevent the server from learning the client’s sample and prevent the client from learning the server’s database. However, in these approaches the server holds plaintext biometric templates, violating the single functionality requirement (RQ.S1). In our design, we instead store the  $N$  samples of registered recipients in secret-shared form between the RS and BP. We redesign the SMC-based biometric approach

to reconstruct the secret shared templates before computing distances.

**Secure two-party computation.** Secure two-party computation is an interactive protocol between Alice with private data  $X$  and Bob with private data  $Y$  where they want to compute a known function  $(A, B) \leftarrow F(X, Y)$  without revealing any information about their private data. The security of S2PC guarantees that the view of each party can be simulated with its input and output. We use Yao’s garbled circuit (GC) [96] for our SMC instantiation.

**Secret sharing.** A secret-sharing scheme shares a secret value  $x$  between  $n$  party while providing an information-theoretic guarantee that no combination of  $t < n$  shares will reveal any information about the secret  $x$ . We focus on the two-party scenario, i.e.,  $n = 2$ . A secret-sharing scheme is given by the methods **SS.Share** and **SS.Reconstruct**. We define two types of secret sharing, binary and additive:

$(x^{(0)}, x^{(1)}) \leftarrow \mathbf{SS.Share}(x)$ . shares a secret value  $x$  between two parties. For binary sharing,  $x \in \{0, 1\}^n$ , for additive sharing,  $x \in \mathbb{Z}_\alpha$ . The function **SS.Share** picks the first share  $x^{(0)} \leftarrow \{0, 1\}^n$  (resp.  $x^{(0)} \leftarrow \mathbb{Z}_\alpha$ ) uniformly at random in binary (resp. additive) sharing and sets the second share  $x^{(1)}$  such that  $x^{(0)} \oplus x^{(1)}$  (resp.  $x^{(0)} + x^{(1)} \pmod{\alpha}$ ) equals  $x$ . We write  $\mathbf{SS}_b$  to denote binary secret sharing and  $\mathbf{SS}_\alpha$  to denote arithmetic sharing modulus  $\alpha$ .

$x \leftarrow \mathbf{SS.Reconstruct}(x^{(0)}, x^{(1)})$ . takes two shares and outputs the original secret  $x = x^{(0)} \oplus x^{(1)}$  in the binary and  $x = (x^{(1)} + x^{(0)}) \pmod{\alpha}$  in the additive setting.

**SMC-Janus.** We now instantiate the SMC-Janus functions.

**Setup.** We initialize the system as follows:

1. The RS initializes a new biometric sensor  $S \leftarrow \mathbf{Sensor}(\mathbf{TS}, d)$ .
2. The RS creates an empty database **RS.db** for storing secret shared biometric templates.
3. The BP creates an empty database **BP.db** for storing secret shared biometric templates.

**Membership.** The RS reads a biometric sample  $B$  from the new recipient, then interacts with the BP to compute whether it matches any of the  $N$  registered recipients in the database using a garbled circuit:



1. The RS reads a new biometric sample  $B \leftarrow S.\text{ReadBio}(\text{recipient})$  from the potential recipient.
2. The RS computes  $a$  alignments of  $B$  using  $\langle B_j \rangle_a \leftarrow \text{Align}(B, a)$ .
3. The RS and BP use a garbled circuit to compute the **SMC-Matching** function from Algorithm 7 as follows:
  - (a) The RS retrieves the registered users' template shares from its database  $\langle X_i^{(0)} \rangle_N \leftarrow \text{RS.db}$  and uses  $(\langle X_i^{(0)} \rangle_N, \langle B_j \rangle_a)$  as its private input.
  - (b) The BP retrieves the secret shares of registered users' templates  $\langle X_i^{(1)} \rangle_N \leftarrow \text{BP.db}$  as its private input.
  - (c) The RS and BP jointly compute **SMC-Matching** and RS learns the binary value **member**.
4. The RS outputs the binary value **member**.

When computing the **SMC-Matching** function, we first reconstruct biometric templates for registered users in line 3. Next, we compute the Hamming distance between the  $i$ 'th user and the  $j$ 'th alignment of  $B$  in line 5. We check the distance against the public threshold  $t$  to decide if a pair of templates are a match in line 6. The user  $i$  is a match for the new recipient if the template  $X_i$  matches any of the  $a$  alignment  $B_j$  as seen in line 7. Finally, recipients are duplicates if they match any of the existing users (line 8). Changing the Hamming distance to Euclidean distance only requires replacing line 5 with  $d_{i,j} \leftarrow \sum_{k=0}^{\text{TS}-1} (B_j[k] - X_i[k])^2$ .

**AddMember.** To add a new recipient, the registration station loads the biometric from the pending registration request and secret shares the template  $B$  with the biometric provider (without including  $a$  alignments) as follows:

1. The RS loads the plaintext biometric template  $B$  from the last membership request and secret shares  $B$  as  $(B^{(0)}, B^{(1)}) \leftarrow \text{SS}_b.\text{Share}(B)$ .
2. The RS runs  $\text{RS.db.Insert}(B^{(0)})$  to insert  $B^{(0)}$  into its database and sends  $B^{(1)}$  to BP. The BP inserts  $B^{(1)}$  into its the user database by running  $\text{BP.db.Insert}(B^{(1)})$ .
3. The RS deletes plaintext biometrics templates  $B$ , alignments  $\langle B_j \rangle_a$ , and BP's share  $B^{(1)}$ .

**Ratchet.** Ratcheting refreshes the secret shares of the registration station and biometric provider (see RQ.S2). To minimize communication costs, instead of sending new shares for all values, we send a single randomness seed and use a keyed PRF function to refresh shares locally as follows:

1. The RS chooses a randomness seed  $r$  and sends it to the biometric

---

**Algorithm 7** The matching function computed using a garbled circuit inside SMC-Janus (for hamming distances).

---

```
1: function SMC-MATCHING( $(\langle X_i^{(0)} \rangle_N, \langle B_j \rangle_a), \langle X_i^{(1)} \rangle_N$ )
2:   for  $i \leftarrow \mathbb{Z}_N$  do
3:      $X_i \leftarrow X_i^{(0)} \oplus X_i^{(1)}$ 
4:     for  $j \leftarrow \mathbb{Z}_a$  do
5:        $d_{i,j} \leftarrow \sum_{k=0}^{TS-1} (B_j[k] \oplus X_i[k])$ 
6:        $m_{i,j} \leftarrow d_{i,j} < t$ 
7:        $match_i \leftarrow \bigvee_{j=0}^{a-1} m_{i,j}$ 
8:    $member \leftarrow \bigvee_{i=0}^{n-1} match_i$ 
9:   return member
```

---

provider.

2. The BP receives the seed  $r$  and randomizes template shares  $\langle X_i^{(1)} \rangle_N \leftarrow \mathbf{BP.db}$  as  $X_i^{\prime(1)} \leftarrow X_i^{(1)} \oplus \mathbf{PRF}_r(i)$ . The BP deletes its database  $\mathbf{BP.db}$  and the seed  $r$  then create a new user database with new shares using  $\mathbf{BP.db.Insert}(\langle X_i^{\prime(1)} \rangle_N)$ .
3. The RS refreshes its shares using the seed  $r$  following the same process as BP. Afterward, RS deletes the seed  $r$  and all old secret shares.

**Safety.** The SMC-based design satisfies the safety requirements. First, we argue that SMC-Janus achieves single functionality (RQ.S1). By inspection, the only protocol that might reveal data related to recipients is **Membership** (at **Setup** there is no data to leak, and **AddMember** and **Ratchet** have no output). However, in **Membership** the private inputs of either party are fed into a two-party SMC protocol that guarantees that as long as the parties are honest-but-curious (which they are by assumption), only the RS learns the single output bit, and nothing more. Thus RQ.S1 is satisfied.

Second, at rest, recipient-related data is secret-shared between the RS and the BP, ensuring information-theoretic protection against single compromises. As a result, a compromise of either party does not reveal any sensitive data (satisfying RQ.S2) and long-term safety is guaranteed (RQ.S4). Any leakage of information resulting from queries can only be the result of a call to **Membership** by an attacker compromising the RS. However, the SMC protocol can only be run with the cooperation of the BP (satisfying RQ.S3).

### 4.6.2 SHE-Janus

Next, we instantiate Janus using homomorphic encryption. Biometric matching and identification can be computed using either additively homomorphic encryption [146, 147] or fully homomorphic encryption [129, 148]. Computing the Euclidean distance is straightforward with homomorphic encryption. However, many HE schemes are not suitable for performing the comparison needed for applying the decision threshold. While many of the prior works [149, 150] reveal this distance and perform the thresholding in the clear, some papers like Huang et al. [146] extend the HE distance computation with an SMC comparison to keep the whole computation in an encrypted domain. We use the BFV [106] somewhat homomorphic encryption to compute Euclidean distance and use a garbled circuit to perform the comparison. We compare our performance against a scheme that reveals the distance [150] and Huang et al. [146] that protects the distance in Section 4.8.2.

**Somewhat homomorphic encryption.** Somewhat homomorphic encryption schemes allow arithmetic operations like additions and limited multiplications without decryption. This chapter relies on the BFV scheme over the prime ring  $\mathbb{Z}_q$ . There are 4 major methods in SHE schemes:

- $\text{params} \leftarrow \text{HE.ParamGen}(q)$ . Takes a plaintext domain  $\mathbb{Z}_q$  and generates an HE parameter set.
- $pk, sk \leftarrow \text{HE.KeyGen}(\text{params})$ . Generates a new key pair  $(pk, sk)$  based on the parameter set **params**. Evaluation keys are assumed to be part of the public key.
- $\llbracket x \rrbracket \leftarrow \text{HE.Enc}(pk, x)$ . Takes the public key  $pk$  and a message  $x \in \mathbb{Z}_q$  and returns the ciphertext  $\llbracket x \rrbracket$ .
- $x \leftarrow \text{HE.Dec}(sk, \llbracket x \rrbracket)$ . Takes the secret key  $sk$  and a ciphertext  $\llbracket x \rrbracket$  and returns the decrypted message  $x$ .

The correctness property of the encryption ensures  $\text{HE.Dec}(sk, \text{HE.Enc}(pk, x)) \equiv x \pmod{q}$ .

*Homomorphic operations.* SHE schemes support homomorphic addition (denoted by  $+$ ), subtraction (denoted by  $-$ ), and a limited number of multiplication (denoted by  $\cdot$ ) of ciphertexts:  $\text{HE.Dec}(sk, \llbracket a \rrbracket \cdot \llbracket x \rrbracket + \llbracket b \rrbracket) = ax + b \pmod{q}$ . It is also possible to perform addition and multiplication with scalar values in addition to operating on two ciphertexts.

*SIMD batching.* We can combine number theoretic transformation (NTT) with

---

**Algorithm 8** The matching functioned computed using a garbled circuit inside SHE-Janus.

---

```

1: function SHE-MATCHING( $\langle d_{i,j}^{(0)} \rangle_{aN}, \langle d_{i,j}^{(1)} \rangle_{aN}$ )
2:   for  $i \leftarrow \mathbb{Z}_N$  do
3:     for  $j \leftarrow \mathbb{Z}_a$  do
4:        $d_{i,j} \leftarrow d_{i,j}^{(0)} + d_{i,j}^{(1)}$ 
5:       if  $d_{i,j} \geq q$  then
6:          $d_{i,j} \leftarrow d_{i,j} - q$ 
7:          $m_{i,j} \leftarrow d_{i,j} < t$ 
8:        $match_i \leftarrow \bigvee_{j=0}^{a-1} m_{i,j}$ 
9:    $member \leftarrow \bigvee_{i=0}^{n-1} match_i$ 
10:  return member

```

---

the BFV scheme to allow batching  $N$  scalar values into each ciphertext [114]. This transforms each arithmetic operation to a SIMD alternative that performs pairwise operations between two  $N$ -ary vectors.

**SHE-Janus.** We now instantiate the SHE-Janus functions.

**Setup.** We initialize the system as follows:

1. The RS initializes a new biometric sensor following  $S \leftarrow \mathbf{Sensor}(\mathbf{TS}, d)$ .
2. The BP creates a new SHE key pair  $pk, sk \leftarrow \mathbf{HE.KeyGen}(\mathbf{params})$  and sends the public key to the registration station. The RS saves  $pk$ .
3. The RS creates an empty database  $\mathbf{RS.db}$  for encrypted biometric templates.

**Membership.** To determine if a recipient has registered before, the registration station reads a biometric sample  $B$  and performs the membership matching in three phases: computing Euclidean distance using SHE, secret sharing the encrypted distance using an additive sharing modulus  $q$ , and using a GC to perform comparison over secret shared values.

1. The RS reads a new biometric sample  $B \leftarrow S.\mathbf{ReadBio}(\mathbf{recipient})$  from the recipient and computes  $a$  alignments of  $B$  using  $\langle B_j \rangle_a \leftarrow \mathbf{Align}(B, a)$ .
2. The RS retrieves the encrypted templates of registered recipients  $\langle \llbracket X_i \rrbracket \rangle_N \leftarrow \mathbf{RS.db}$  from its database.
3. The RS computes the Euclidean distance of the alignment  $B_j$  and user template  $\llbracket X_i \rrbracket$  as  $\llbracket d_{i,j} \rrbracket = \sum_{k=1}^{\mathbf{TS}} (\llbracket X_i[k] \rrbracket - B_j[k])^2$ .
4. The RS secret shares the encrypted Euclidean distances using  $d_{i,j}^{(0)} \leftarrow_{\$} \mathbb{Z}_q$ ,  $\llbracket d_{i,j}^{(1)} \rrbracket \leftarrow \llbracket d_{i,j} \rrbracket - d_{i,j}^{(0)}$  and sends  $\langle \llbracket d_{i,j}^{(1)} \rrbracket \rangle_{aN}$  to the biometric provider.
5. The BP receives and decrypts  $\langle \llbracket d_{i,j}^{(1)} \rrbracket \rangle_{aN}$ .
6. The RS and BP interact together to compute the **SHE-Matching** function

in Algorithm 8 using a garbled circuit as follows:

- (a) The RS uses  $(\langle d_{i,j}^{(0)} \rangle_{aN})$  as its private input.
- (b) The BP uses  $(\langle d_{i,j}^{(1)} \rangle_{aN})$  as its private input.
- (c) The RS and BP jointly compute the garbled circuit and RS learns the value **member** as output.

7. The RS outputs the binary value **member**.

Computing the Euclidean distance  $\llbracket d_{i,j} \rrbracket$  between the  $j$ 'th alignment  $B_j$  and  $i$ 'th user template  $\llbracket X_i \rrbracket$  is straightforward and follows the definition of Euclidean distance. The Hamming distance is equivalent to Euclidean distance in binary arrays; we discuss an optimized version of Hamming distance for our HE approach in Section C.2. We use additive secret sharing modulo  $q$  to secret share the distance. When performing the secret sharing over the ciphertexts, the BFV scheme automatically performs the modular reduction. However, we need to manually perform the modular reduction needed for reconstruction in the GC. Instead of running a costly modulus operation in GC, we take advantage of the fact that both shares  $d_{i,j}^{(0)}, d_{i,j}^{(1)}$  are less than  $q$  and consequently their addition is less than  $2q$ . This allows us to simplify computing modulus in line 6 to subtract  $q$  if  $d_{i,j}$  is greater than  $q$ . After computing  $d_{i,j}$  in our garbled circuit protocol, comparing against the decision threshold is straightforward and similar to our SMC approach.

**AddMember.** To add a new recipient, the registration station encrypts then stores the the resulting ciphertext template in **RS.db** as follows:

1. The RS loads the plaintext templates  $B$  from the last membership request, encrypts  $\llbracket B \rrbracket \leftarrow \text{HE.Enc}(sk, B)$ , and inserts it to the user database  $\text{RS.db.Insert}(\llbracket B \rrbracket)$ .
2. The RS deletes all plaintext and alignment templates  $B, \langle B_j \rangle_a$ .

**Ratchet.** The biometric provider generates a new key pair then interacts with the RS to transition the registration station's old ciphertext to the new key as follows:

1. The BP generates a new key pair  $pk', sk' \leftarrow \text{HE.KeyGen}(\text{params})$  and sends the public key  $pk'$  to the registration station.
2. The RS receives and saves the public key  $pk'$ .
3. In a similar manner to membership, the RS secret shares the templates  $X_i^{(0)} \leftarrow^{\$} \mathbb{Z}_q, \llbracket X_i^{(1)} \rrbracket \leftarrow \llbracket X_i \rrbracket - X_i^{(0)}$  and sends  $\langle \llbracket X_i^{(1)} \rrbracket \rangle_N$  to the biometric provider.
4. The BP receives  $\langle \llbracket X_i^{(1)} \rrbracket \rangle_N$ , decrypts then re-encrypt template shares with

the new key  $pk'$  as  $\llbracket X_i'^{(1)} \rrbracket \leftarrow \text{HE.Enc}(pk', \text{HE.Dec}(sk, \llbracket X_i^{(1)} \rrbracket))$ .

5. The BP sends  $\langle \llbracket X_i'^{(1)} \rrbracket \rangle$  to the registration station.
6. The RS deletes the old ciphertext database and then reconstructs and inserts new ciphertext to the database as  $\text{RS.db.Insert}(\langle \llbracket X_i'^{(1)} \rrbracket + X_i^{(0)} \rangle_N)$ .
7. The BP deletes the old key pair  $(pk, sk)$ .

**Safety.** Our SHE-based version satisfies the safety requirements. Similar to SMC-Janus, the only protocol that reveals an output about recipients is **Membership**. The **Membership** protocol has three stages. First, the registration station computes the distance in the encrypted domain, thus cannot gain any information. Second, RS secret shares the distances and sends shares  $d_{i,j}^{(1)}$  to the BP, thus the security of **SS** prevents any leakage. Third, both parties feed their secret shares into a two-party SMC protocol that guarantees that as long as the parties are honest-but-curious (which they are by assumption), only the RS learns the single output bit. Therefore, RQ.S1 is satisfied.

At rest, the biometric provider does not have any information about recipients, and the registration station stores a database encrypted with a quantum secure encryption scheme without knowing the decryption key. As long as only the data of a single party gets compromised during each ratchet epoch, SHE-Janus satisfies both RQ.S2 and RQ.S4. Similar to SMC-Janus, the leakage from queries is limited by the need for cooperation from the BP to run calls to **Membership** (satisfying RQ.S3).

### 4.6.3 TEE-Janus

An alternative for using advanced cryptographic tools such as SMC and SHE is relying on trusted hardware. In this section, we instantiate Janus based on a trusted execution environment (TEE). While TEEs allows us to ensure the security of biometric templates with only one party, we still keep our two-party setting to protect against active compromises (RQ.S3). In TEE-Janus, we use the biometric provider as an authentication server that blindly signs membership requests. If an adversary takes control of the registration station, the humanitarian organization can disable the biometric provider to prevent adversaries from performing queries and learning information from RS.

Recent attacks on trusted hardware [151–153] demonstrate that extracting secrets from a TEE is not impossible. While we acknowledge the vulnerability of TEE solutions to hardware attacks as a limitation of TEE-Janus, our design

safeguards against side channels like memory access and timing patterns. We deploy TEEs on humanitarian organization-controlled devices and not on a cloud service, therefore, there is no need for TEE attestation.

**Trusted execution environment.** TEEs are hardware security modules that ensure the confidentiality and integrity of the data and code during execution. TEE programs have two components: an enclave that runs the sensitive portion of the code and benefits from the hardware security guarantee, and a host component that runs outside of the secure module and is in charge of managing the interactions of the enclave with non-secure components (e.g., the network).

**Blind signature.** A blind signature scheme allows a client to interact with a server to compute a signature on private message  $m$  with the server's secret key without revealing  $m$  to the server. There are 4 major methods in blind signatures:

- $vk, sk \leftarrow \text{BS.KeyGen}()$ . Generates a verification  $vk$  and a signing key  $sk$ .
- $\sigma \leftarrow \text{BS.Sign}(sk, m)$ . The server signs a known message  $m$  with its signing key  $sk$ .
- $\sigma \leftarrow \text{BS.BlindSign}(sk, m)$ . A client with a secret message  $m$  interacts with a server holding the signing key  $sk$  to compute a signature  $\sigma$  on  $m$ . Blindness guarantees that the server does not learn any information about the message.
- $\{0, 1\} \leftarrow \text{BS.Verify}(vk, m, \sigma)$ . Takes a message  $m$ , a signature  $\sigma$ , and a verification key  $vk$  and determines if  $\sigma$  is a valid signature for  $m$ .

**TEE-Janus.** We now instantiate the TEE-Janus functions.

**Setup.** We initialize the system as follows:

1. The BP creates a new blind signature key  $vk, sk \leftarrow \text{BS.KeyGen}()$  and sends the verification key  $vk$  to the registration station.
2. The RS creates a secure enclave and stores  $vk$  as the authority key inside. The enclave securely creates an empty database **RS.db** for biometric templates.
3. The RS initializes a new biometric sensor through the host following  $S \leftarrow \text{Sensor}(\text{TS}, d)$ . Remember that running the sensor inside an enclave is a non-goal for this work (see Section 4.2.1).

**Membership.** To check if a new user is already registered in the database, the registration station reads and processes a biometric sample  $B$  on the host, requests a blind signature from the biometric provider on this template, and passes both the template and signature to the enclave which can directly perform the plaintext biometric matching after verifying the signature.

1. The RS reads a new biometric sample  $B \leftarrow S.\text{ReadBio}(\text{recipient})$  from the potential recipient.
2. The RS host interacts with the BP to compute a blind signature on the template  $\sigma \leftarrow \text{BS.BlindSign}(sk, B)$ , and passes  $(B, \sigma)$  to the enclave.
3. The RS enclave verifies the signature with the stored authority key  $vk$  by running  $\text{BS.Verify}(vk, B, \sigma)$ . If the verification fails, the enclave aborts.
4. The RS enclave runs a plaintext biometric membership check by iterating over *all* registered templates and comparing them with the plain template  $B$  to compute the binary result **member**. Comparing against all templates prevent memory access pattern and timing attacks by ensuring that the operations performed are independent of the data.
5. The RS outputs the membership result **member**.

**AddMember.** To add a new recipient, the RS inserts the biometric sample from the last membership query to the sealed registered user database inside the enclave as follows:

1. The RS enclave loads the plaintext biometric templates  $B$  from the last membership request and inserts the template to its user database  $\text{BP.db.Insert}(\llbracket B \rrbracket)$ .
2. The RS host deletes the plaintext templates  $B$ .

The registration station is adding the biometric template from the last membership request to the database without reading a new sample. As RS has already received and verified a blind signature on this template, there is no need for repeating the blind signing.

**Ratchet.** We assume that the biometric data stored in the enclave are secure and the adversary cannot extract secrets from a TEE. Therefore, our ratchet protocol only supports rotating the key of the biometric provider after verifying the signature of the ratchet request as follows:

1. The biometric provider generates a new key pair  $vk', sk' \leftarrow \text{BS.KeyGen}()$ , signs the verification key with the old secret key  $\sigma \leftarrow \text{BS.Sign}(sk, vk')$ , and sends the verification key  $vk'$  and the signature  $\sigma$  to the registration



station. The RS host receives the new key and signature  $(vk', \sigma)$  and passes them to the enclave.

2. The RS enclave loads the old verification key  $vk$  from the sealed memory and runs **BS.Verify** $(vk, vk', \sigma)$ . If the verification passes, the RS enclave replaces the old verification key  $vk$  with the new one  $vk'$ .

**Safety.** TEE-Janus satisfies the safety requirements. First, we argue that the design achieves single functionality (RQ.S1). All recipient data is stored inside the TEE enclave. The only time that the enclave reveals any information about these data is when running **Membership**. By design, the enclave code will compute and output exactly the allowed 1-bit membership response, thus satisfying RQ.S1.

Second, the sealing properties of the TEE guarantee security against passive compromises of either party, thus satisfying RQ.S2. Third, the TEE only replies with a 1-bit answer when it receives a (blindly) signed request from the BP. Thus, TEE-Janus also protects against active compromises, satisfying RQ.S3. Finally, provided that the security guarantees of TEEs hold, TEE-Janus provides long-term safety (RQ.S4). However, given the recent attacks on TEEs [151–153] we do not want to claim that TEE-Janus provides long-term safety in practice.

## 4.7 Biometrics in practice

Biometric pipelines start with a raw image such as an eye image or a fingerprint scan. Directly comparing images is complex and costly. Thus, typically images are transformed into lower-dimension representations called ‘templates’. This transformation can rely on classic algorithms [154–156] or on neural networks [157–159]. As we assume the registration station processes biometrics in plaintext, Janus can use prior biometric works as a black-box inside the **ReadBio** function, as long as the resulting template complies with the abstraction in Section 4.5.

**Template creation.** We show how to generate templates and handle alignment for major biometric sources. We note that when choosing biometrics, humanitarian organizations must consider factors such as culture and religion which may vary between regions where aid needs to be distributed.

*Fingerprint.* The traditional approach for fingerprint matching is minutiae matching. Unfortunately, it does not satisfy our abstraction as the number of extracted minutiae varies between readings and minutiae do not have a fixed ordering. Thus, we use a different matching technique relying on *finger codes* [154, 160–162], that transform fingerprints into fixed-size arrays of integers where similarity is computed based on Euclidean distance. Different fingerprint scans can have different angles. Some finger code approaches [154, 161] create codes that are not rotation-invariant. Instead, they use an alignment process that rotates the raw image  $a$  times (commonly by  $11.5^\circ$  degree), extract  $a$  finger codes, and check if any of these  $a$  templates match the target biometric to improve accuracy.

*Iris.* Irises are a highly accurate source of biometric data. A common matching approach is *iris code* [155, 163, 164], where an eye image is converted to a fixed-size binary array. Not all iris bits are equal. Parts of the eye may be obstructed (e.g., by glares or eyelashes) and lead to unreliable bits [164]. To address this issue, iris codes are accompanied by mask vectors to exclude unreliable bits from the comparison, and similarity is computed using normalized Hamming distance. Iris codes have an alignment procedure that applies  $a$  circular shifts (by steps of 1 or 2) on the bit vector and repeats the similarity comparison.

*Face.* Prior works such as ArcFace [157] allow us to represent faces with a fixed-size array of real values where the similarity of two faces is computed using Euclidean distance. We can discretize these real values into integers fitting into an arbitrary domain  $\mathbb{Z}_d$ . This discretization can have a minor impact on accuracy that depends on the choice of  $d$ . The templates produced in ArcFace do not require alignment.

#### 4.7.1 Membership with a single sample

We study membership performance with respect to two types of errors, which we name following the same convention as in biometric identification and matching: *false accept rate (FAR)*, the probability of incorrectly identifying a new person as an existing user in a biometric membership; and *false reject rate (FRR)*, the probability of failing to detect an existing user in a biometric membership test.

In the humanitarian setting, the cost of false acceptance is considerably higher than false rejection. False acceptances prevent legitimate recipients from receiving aid. False rejection enables recipients to register more than

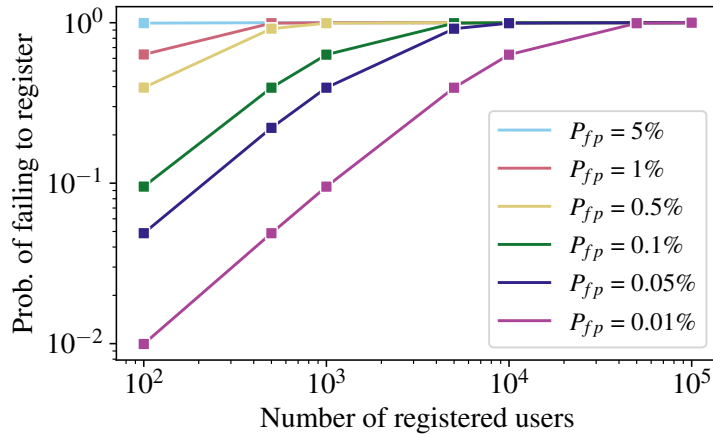


Figure 4.3: Impact of biometric matching  $P_{fp}$  on registration.

once. While allowing cheating is undesirable, humanitarian principles imply that we must favor lower FAR over FRR (see RQ.F3).

The membership operation can be broken down into individual matching operations between the queried biometric sample  $B$  and the  $N$  templates of recipients already registered in the system. The accuracy of each matching operation is limited by the error rate of the plaintext matching operation **Match**. Assuming that the underlying plaintext matching has a false positive rate of  $P_{fp}$  and a false negative rate of  $P_{fn}$ , and these probabilities are independent for different match operations, we can compute FAR as  $1 - (1 - P_{fp})^N$  and FRR as  $P_{fn}(1 - P_{fp})^{N-1}$ .

The error probabilities of plaintext matching algorithms for a single biometric sample are typically in the range of 0.01 – 5%. These error rates often satisfy the accuracy requirements of authentication systems and choosing the decision threshold value  $t$  allows configuring a trade-off between  $P_{fp}$  and  $P_{fn}$ . These rates however are not sufficient in the context of humanitarian aid distribution. Figure 4.3 shows the probability of *falsely* rejecting a legitimate new recipient depending on the number of registered users and biometric failure probability  $P_{fp}$ . Looking at this graph, deduplicating a new recipient with a single biometric sample against a database of 10,000 fails with the probability of 63% for  $P_{fp} = 0.01\%$  and almost always for higher error rates. These failure rates do not satisfy RQ.F3.

### 4.7.2 Membership at scale

We explore the potential of biometric fusion to lower the error rate, inspired by existing biometric identification systems such as Aadhar [165] or US-VISIT [166] that use multiple samples (10 fingerprint scans, 2 iris scans, and 1 facial image; and 10 fingerprints and 1 facial image, respectively) to lower their error rates when operating on a billion, respectively tens of millions, people.

The fusion procedure takes  $f$  samples from different sources to create a  $f$ -ary template  $\langle X_i \rangle_f$ . As our primary goal is reducing FAR, our fusion requires all  $f$  sample pairs of two templates to match as follows:

$$M \in \{0/1\} \leftarrow \text{FusedMatch}(\langle X_i, Y_i, D_i, t_i, a_i \rangle_f) = \begin{cases} m_i \leftarrow \text{AlignedMatch}(X_i, Y_i, D_i, t_i, a_i) \\ M \leftarrow \bigwedge_{i=1}^f m_i. \end{cases}$$

A fused false positive only happens when all  $f$  underlying samples produce false positives at the same time. Therefore, the fused membership will have an FPR of  $\prod_{i=1}^f P_{fp_i}$  where  $P_{fp_i}$  is the FPR of the  $i$ 'th sample. At the same time, the fused false negative rate is bound by  $\sum_{i=1}^f P_{fn_i}$ . Providing support for fusion is very similar to our alignment process (**Align**). The only difference is that when performing pairwise matching instead of performing the logical or ( $\vee$ ) as in alignment, we use logical and ( $\wedge$ ) in fusion. We support biometric fusion in all three instantiations of Janus. This enables Janus to scale its FAR rate to support very large userbases *as long as Janus receives the necessary number of biometric samples*.

We evaluate the FAR of membership when fusing  $f$  samples with false positive rates of  $P_{fp}$  for various numbers of registered users in Fig. 4.4. To satisfy RQ.F3, we aim for a false accept rate of  $\text{FAR} = 10^{-4}$  on deduplicating against a database of 10,000 registered users. Two ways for achieving this error rate are: (1) taking 4 samples from biometric modalities with  $P_{fp} = 1\%$  (e.g., 4 fingers); or (2) with 2 samples with  $P_{fp} = 0.01\%$  (e.g., 2 irises). Since there exist biometric devices that scan 4 fingers or 2 irises at the same time, Janus could be efficiently deployed even when requiring more than one sample. If lower error rates are desired, the ICRC can combine modalities (4 finger and 2 irises) or collect more finger samples from recipients.

**Template configuration.** While we use existing biometric matchings as black boxes, they offer various configurations that provide accuracy/performance

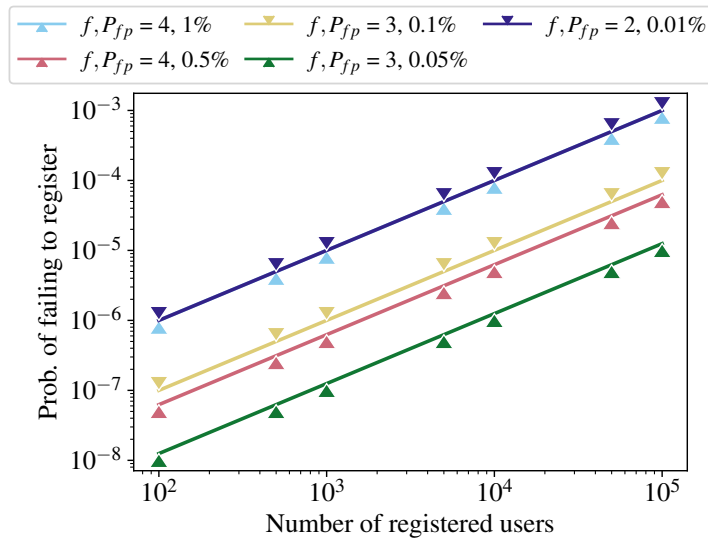


Figure 4.4: Impact of fusing  $f$  samples with FPR of  $P_{fp}$ .

trade-offs that are of interest in Janus. For example, having a larger template size (e.g.,  $\mathbf{TS} = 10k$  over  $\mathbf{TS} = 2k$ ) or more alignments (e.g.,  $a = 8$  over  $a = 1$ ) may improve both the false positive and negative error rates (up to a limit) at a performance cost.

Biometric fusion introduces the number of samples as a new trade-off dimension, where two short templates (e.g.  $f = 2, \mathbf{TS} = 2k$ ) might outperform and offer better error rates than a single large template (e.g.  $f = 1, \mathbf{TS} = 10k$ ).

**Failure rate of Janus.** The error rates of Janus are mainly limited by the plaintext biometric matching. Our instantiations follow the plaintext computation with two minor differences: *Discretization*: SMC-Janus and SHE-Janus expect biometric templates to have integer values from the domain  $\mathbb{Z}_d$ . While iris and finger codes follow this restriction, face templates are often real-valued and need discretization. *Threshold*: The public threshold  $t$  has limited precision. As the encrypted computation is nearly identical to its plaintext counterpart, and measuring the accuracy of the underlying plaintext biometric approaches is out of the scope of this work, we do not measure the error rates of Janus.

## 4.8 Evaluation

In this section, we evaluate the performance of Janus.

**Biometrics setup.** As we do not need to measure accuracy (see Section 4.7.2), we do not collect or process biometric data from real humans, and instead, take existing biometric template generation schemes and replace their output with random data of the same dimension as follows:

*Iris.* We base our iris code setting on templates produced by the University of Salzburg Iris Toolkit (USIT) v3 [167, 168]. The default parameters create  $\text{TS} = 10240$  bits template, but smaller 2048 bits templates are used in prior privacy-preserving works [131]. Following our notation in Section 4.5, we create iris sensors as  $\text{Iris}_{10240} \leftarrow \text{Sensor}(10240, 2)$  and  $\text{Iris}_{2048} \leftarrow \text{Sensor}(2048, 2)$ .

*Fingerprint.* We use two instantiations of finger codes. The first sensor  $\text{Finger}_{640} \leftarrow \text{Sensor}(640, 256)$  follows Jain et al. [160] and uses Gabor filters to create a byte array of  $\text{TS} = 640$  elements as the template. The second sensor  $\text{Finger}_{64} \leftarrow \text{Sensor}(64, 256)$  follows more recent CNN-based approaches [150, 162] and creates  $\text{TS} = 64$  byte templates.

*Face.* We base our face sensor  $\text{Face} \leftarrow \text{Sensor}(512, 256)$  on ArcFace [157]. ArcFace generates 512-dimensional real-values templates, but we can discretize template elements into bytes to fit Janus’s requirements. Since face and fingerprint templates are similar and their matching follows the same process, we only evaluate fingerprints.

**Instantiations setup.** We run our experiments on a machine with an Intel i7-8650U CPU and 16 GiB of RAM, using the following libraries and parameters for our instantiations.

*SMC-Janus.* We rely on a high-level SMC compiler called EMP-Toolkit [119] to generate and execute garbled circuits. More specifically, we use ‘EMP-sh2pc’ [169] which offers semi-honest security in a two-party setting. We implemented SMC-Janus fully in C++ (320 lines).

*SHE-Janus.* We use the BFV [106] somewhat homomorphic encryption scheme implemented in the Lattigo library [170] for our homomorphic operations. Let the degree of the RLWE polynomial be  $N_{deg}$ , the plaintext modulus be  $m_{pt}$ , and the ciphertext modulus be  $m_{ct}$ . We generate BFV keys with a set of parameters ( $N_{deg} = 4096$ ,  $\lg(m_{ct}) = 109$ ,  $m_{pt} = 65\,929\,217$ ). Our parameter set follows the homomorphic encryption security standard [111] to provide 128 bits of security. Moreover, our parameter supports batching  $N_{deg}$  scalar values into each ciphertext and performing SIMD operations. We implement SHE-Janus partially in Go (650 lines) and partially in C++ (150 lines).

*TEE-Janus.* We use Intel SGX for our TEE-based solution and rely on the Fortanix enclave development platform [171] to run Rust code in an SGX enclave. We implement TEE-Janus fully in Rust (300 lines).

We will make all our implementations open source.<sup>1</sup>

#### 4.8.1 Performance of Janus

We evaluate the performance of our three Janus instantiations. We measure their (single-core) computation and communication cost when using various database sizes (generated as described above). We assess both the single- and multi-sample settings with a single alignment. We use 4 fingers and 2 irises for the latter to ensure the instantiations satisfy RQ.F3 (see Section 4.7.2).

*TEE-Janus.* Our TEE-based solution significantly outperforms SMC-Janus and SHE-Janus, providing near plaintext biometric efficiency. Deduplication over an 8k user database takes under 50 ms and uses less than 1 KiB of communication, meeting the scalability requirement (RQ.D2) easily.

*SHE-Janus.* Fig. 4.5 displays the computation and communication costs for single and multi-sample membership across different database sizes using **Finger**<sub>64</sub> and **Iris**<sub>2048</sub> sensors. Deduplication 8k recipients with 2 irises takes 40 s of computation on RS and 56 MiB, while deduplication with 4 fingerprints requires 4.4 s and 155 MiB. This duration, under a minute, is suitable for in-person registration, and SHE-Janus meets RQ.D2. When templates are larger, computation increases linearly, but not the communication cost (see Section C.1 for more details).

*SMC-Janus.* SMC-Janus offers information theoretic data security. With this instantiation, deduplicating 10,000 users is prohibitively costly (see Section C.1); it fails to meet RQ.D2 and is better suited for smaller aid projects. Deduplicating 1k recipients with 2 irises requires 0.93 GiB and 39 s of computation on both RS and BP, while deduplication with 4 fingerprints requires 52 s and 2.36 GiB.

**Comparison.** The TEE-Janus version is the most efficient but it requires SGX-supported devices at the registration station, as well as trust in the manufacturers of these devices. Also, the security of the scheme is limited by the existence of hardware vulnerabilities that might expose biometric data from

<sup>1</sup>Code available online: <https://github.com/spring-epfl/Janus>

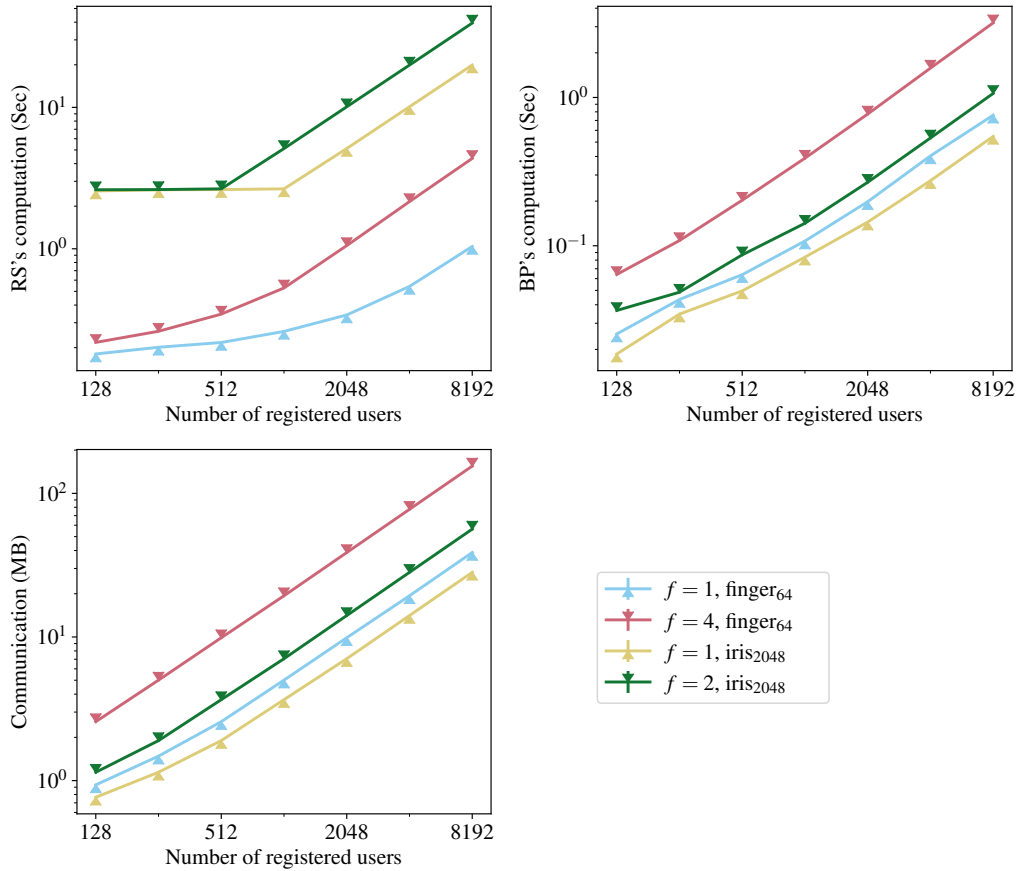


Figure 4.5: Evaluating the single-core membership performance of SHE-Janus with  $f$  iris or fingerprint samples.

the registration station. While SMC-Janus offers the strongest data protection, its cost makes it unsuitable for large aid distribution projects. SHE-Janus ensures biometric data safety using a quantum-secure encryption scheme, and registration completes in under a minute. We would recommend SHE-Janus for large aid projects given its balance between security and performance; and SMC-Janus as long as the size of the group receiving aid is such that performance is acceptable.

#### 4.8.2 Comparison with Closely Related Work

In this section, we compare the cost of SHE-Janus to two of the most relevant prior works:



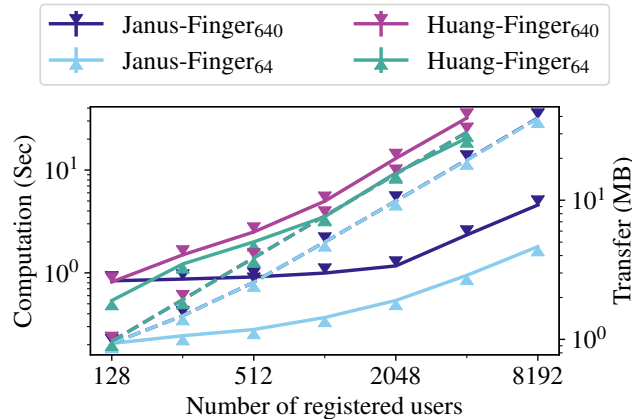


Figure 4.6: Single sample computation (solid lines) and communication (dashed lines) cost of Huang et al. [146] (80-bit security) vs SHE-Janus (128-bit security).

**Same protection.** The closest paper to Janus is Huang et al. [146], which supports identification with a single finger code sample and has a structure similar to SHE-Janus. This work can be modified in a straightforward manner to support membership, but due to being limited to a single biometric sample and relying on the RSA assumption for security (which is not quantum secure), they do not satisfy RQ.F3 or RQ.S4.

Huang et al. use Pailler homomorphic encryption to compute the Euclidean distance of templates, then secret shares the distance between two parties, and finally uses a garbled circuit to determine the template with the highest similarity. Huang et al. have an expensive offline phase whose cost cannot be fully aggregated in our setting due to frequent user additions (see RQ.F2) and provides a protocol called ‘backtracking’ to retrieve the identity of the matching template. We only consider the online cost of Huang et al. in the comparison without performing backtracking. We run the code of Huang et al. on the machine described above using the original parameter of the paper that provides 80-bit (classic) security. Fig. 4.6 shows the cost of deduplicating over databases of various sizes with both  $\text{Finger}_{64}$  and  $\text{Finger}_{640}$  sensors. SMC-Janus has a similar communication cost to Huang et al., but despite having a higher security guarantee of 128-bit, Janus reduces the cost of deduplicating 4k users by a factor of 20x.

**Less protection.** HERS [150] provide privacy-preserving single sample fingerprint identification. HERS introduces a novel short finger code representation (the basis for our  $\text{Finger}_{64}$  sensor) and computes and *reveals* the distance of templates using the BFV [106] SHE scheme. HERS focuses on finding the

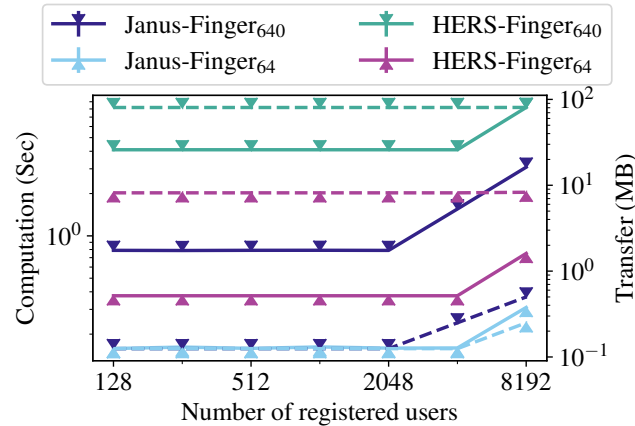


Figure 4.7: Single sample computation (solid lines) and communication (dashed lines) cost of computing template distance of HERS [150] and SHE-Janus (without thresholding).

most similar templates as part of identification and do not support or evaluate the accuracy of membership or deduplication. HERS is limited to a single biometric sample which prevents achieving a low membership error rate (RQ.F3), reveals the distance against every template in the server’s database that violates RQ.S1, and does not offer a key rotation mechanism to protect against RQ.S2 when actors are compromised at different times. Fig. 4.7 shows the cost of distance computation in SHE-Janus (without thresholding) and HERS. Both approaches have similar computation costs but SHE-Janus significantly improves the communication in our setting.

## 4.9 Related work

In Section 4.6, we discussed prior works related to the techniques we use to instantiate Janus: secure multiparty computation, homomorphic encryption, and trusted execution environments; and discuss how they compare to our solution. We now discuss other techniques to build privacy-preserving biometrics systems.

*Fuzzy matching.* Fuzzy matching approaches model the probabilistic biometric sampling process by assuming that a unique biometric ground truth exists, and taking each biometric sample read as a noisy version of this truth. Then, they decode each biometric reading using error correction codes. When biometrics samples are taken from the same user, the difference between them should be small, and decoding them should lead to the same base code. There are three

common fuzzy matching approaches: fuzzy commitments [127, 128, 172], that target biometric sources represented by fixed-size arrays; fuzzy vaults [126, 173] which can support comparison of order-invariant arrays such as fingerprint minutiae; and fuzzy extractors [174, 175], that generate pseudorandom keys based on biometric samples.

The use of error correction codes negatively impacts accuracy, and thus fuzzy approaches cannot fulfill the low failure rate requirement (RQ.F3). Also, many fuzzy matching protocols are vulnerable to statistical attacks [176, 177], violating the single functionality requirement (RQ.S1).

*Convolutional neural network.* Besides using a CNN to learn a representation of biometric samples [162], it is possible to train a model to classify biometric images in a closed world [178]. These approaches require retraining when adding a new recipient to the system, thus they do not satisfy the dynamic addition requirement RQ.F2.

*Biometric cryptosystems.* Biometric cryptosystems [179] use biometric data to derive a secret key per user, either by deriving it from the biometric sample [174] or by hiding the key in helper data which can be opened using a biometric sample from the user [127, 173]. The secret key can be used to encrypt data or in authentication protocols. These approaches can be combined with so-called BioHashing [180, 181] that enhances the key derivation process by adding a user secret (password) to support 2-factor authentication and reduce the false accept rate. Deduplication cannot provide a secret key and thus biometric cryptosystems are not a suitable solution in our setting.

## 4.10 Conclusion

To ensure efficient distribution of aid, humanitarian organizations need to prevent recipients from registering more than once in their programs. Current approaches to achieving this are based on slow and not widely-available means. The humanitarian sector is turning to biometrics-based solutions to address these problems, but not carefully done these solutions can have a strong impact on the safety of aid recipients.

In this chapter, we introduced Janus, a biometric-based deduplication system designed to provide safety-by-design. Janus can be combined with major biometric sources, and supports the fusion of multiple encrypted biometric samples to improve accuracy. We show that, while providing better protec-

tion, Janus improves by orders of magnitude the performance of prior systems and, to the best of our knowledge, is the first to achieve the low error rate needed for deduplication in humanitarian aid.

After Wang et al.'s system [120], our work is a step forward toward enabling humanitarian organizations to harness digitalization to increase their efficiency while protecting the safety and dignity of those that are most in need. Further research is essential for safe digitalization in other stages, like pre-program recipient identification and post-aid monitoring and evaluation [182].

# CHAPTER 5

## **Brutus: A Decision Support System to Prevent the Use of Insecure Communication in Aircraft**

This chapter is based on the following article:

Kasra EdalatNejad\*, Theresa Stadler\*, Martin Strohmeier, Vincent Lenders, Wouter Lueks, Carmela Troncoso: “Brutus: a decision support system to prevent the use of insecure communication in aircraft”. Under submission 2023.

This article is a collaborative effort. Kasra EdalatNejad designed the detection method based on text compression, while methods utilizing convolutional neural networks and random forest are designed by Theresa Stadler. Remaining contributions were made jointly by all authors.

\* denotes equal contribution.

### **5.1 Introduction**

Aviation protocols, including the Aircraft Communications Addressing and Reporting System (ACARS) [183], were designed decades ago. At that time, eavesdropping on aircraft communication required specialized tools. The confidentiality provided by these protocols relied solely on such tools being unavailable to adversaries. With the advent of software-defined radios, however, eavesdropping on aircraft communications became possible at a low cost (e.g., with equipment worth less than 100\$) and without requiring any technical background.

In response, secure protocol standards such as ACARS Message Security (AMS) [184] were introduced. Yet, these protocols are rarely, if ever, used in

the wild. Instead, some airlines and aircraft manufacturers developed home-made protocols to enable message confidentiality. However, previous works that manually inspected ACARS traffic discovered that these proprietary protocols rely on insecure ciphers as encryption method [185, 186].

The use of insecure ciphers, such as substitution ciphers, poses a major risk to the users of the ACARS network as they give the communicating parties an illusion of confidentiality. This might encourage them to send sensitive content, such as credit card details, passenger information, or location data [186].

To improve data privacy, aviation authorities thus need to be able to easily and reliably identify air traffic operators and manufacturers that use proprietary protocols based on insecure ciphers. This is a difficult task that requires expertly trained analysts to manually inspect large volumes of messages and identify suspicious patterns among a trove of machine-generated data. While such manual inspection has shown to be effective [185], it does not scale.

*Goal.* We develop methods to identify and flag encrypted ACARS messages at scale with a focus on weak ciphers. Our methods do not distinguish secure and insecure ciphertexts. The goal is to provide authorities with a tool that filters out messages that are likely to be encrypted and should be further inspected by a human analyst. Our tool aims to reduce the manual labor of identifying entities that use insecure ciphers. Distinguishing plaintext messages from encrypted messages is a challenging task due to several reasons.

First, ACARS messages are highly heterogenous and range from machine-generated sensor data to manually typed messages written in multiple languages. A manual inspection of ACARS messages shows that the message structure differs significantly from natural language. Many sensor messages are a sequence of characters drawn from the full set of ASCII characters instead of words. Moreover, user-generated texts contain words from a mix of natural languages and include jargon and abbreviations.

Second, ACARS messages that contain sensor data look similar to the (somewhat) random stream of characters produced by (insecure) ciphers. This makes it challenging to robustly differentiate between these two classes.

Third, the data available through the collection of real-world ACARS messages is not naturally labeled and contains a mix of plain- and ciphertexts. This requires careful consideration when pre-processing and cleaning the raw data for classification. Moreover, the lack of ground truth labels complicates the evaluation of detection methods.

Our work makes the following contributions:

- ✓ We design three methods to automatically detect ciphertexts; one based on text compression techniques and two based on supervised learning paradigms.
- ✓ Our detectors are the first methods that support flagging insecure classic ciphers in a scenario where the plaintext distribution is unknown and may include multiple natural languages in addition to machine-generated messages.
- ✓ We introduce a methodology to obtain labeled data for training and evaluation at scale when ground truth is unavailable and labeling requires specialized knowledge.
- ✓ We design and build a decision support system using our detector. To demonstrate the capability of our system, we performed a large-scale analysis of real-world data; we found 9 new unknown (potentially insecure) encrypted protocols.

## 5.2 Aircraft communications addressing and reporting system

In this section, we provide technical background on the ACARS protocol. We discuss the system model, message structure, and typical usage by stakeholders in aviation.

### 5.2.1 ACARS system model

ACARS was developed and standardized by Aeronautical Radio Inc. (ARINC) in the 1970s. ACARS is a major datalink in commercial aviation. Almost all commercial airliners use it to communicate with local air traffic control as well as with their own business and flight operations centers.

ACARS uses different physical channels for transmission: (Very) High Frequency (HF/VHF) air-to-ground communication and via satellite (UHF/SHF). Messages are transmitted between an aircraft and ground stations managed by two service providers, who handle the infrastructure between aircraft and endpoints. Fig. 5.1 illustrates this system model.

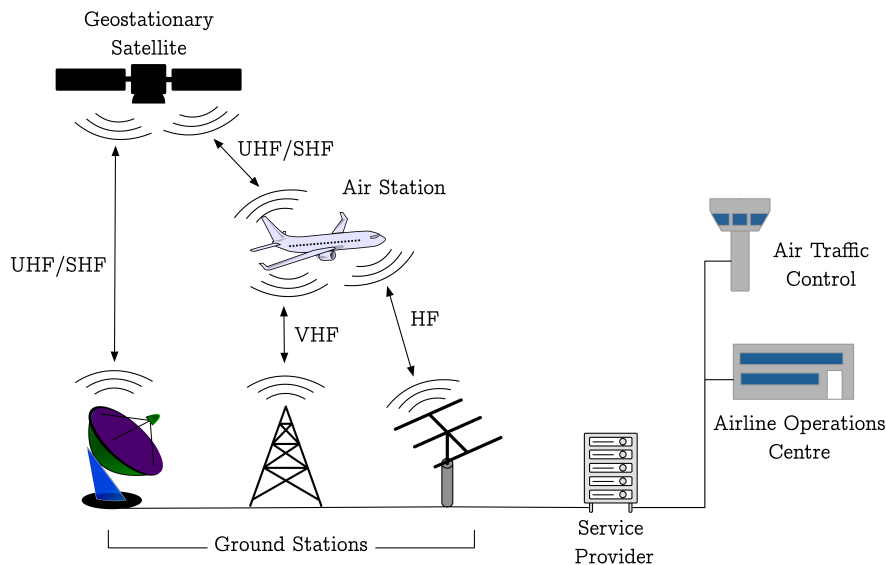


Figure 5.1: Overview of the ACARS system taken from Smith et al. [185].

### 5.2.2 ACARS messages

ACARS is a text-based communication channel supporting 7-bit ASCII characters. Yet, in practice, the Baudot character set<sup>1</sup> is used to ensure compatibility across all parts of the network and applications. We focus on three fields of ACARS message structure that are relevant for our work:

*Aircraft registration.* With the help of open-source aircraft databases, the 7-character aircraft registration field enables the retrieval of metadata about the sender, including aircraft type, owner, operator, and manufacturer.

*ACARS label.* The 2-character ACARS label field is crucial in routing a message to the intended endpoint in the ACARS network. Labels are principally defined in the ARINC 620 standard. However, there is space for user-defined labels [187], which are used extensively.

*Free text.* The core part of the message is formed by the 210-character text field, which allows the transmission of arbitrary content. For our work, it is crucial to distinguish between several types of messages by user type and intent:

*Arbitrary free text entered by humans.* ACARS is often described as “SMS for aircraft”, whereby pilots and cabin crew communicate with the ground. This

<sup>1</sup>Consisting of A-Z, 0-9, , - . / ?, and some control characters.



can take any form and content, from concerns about unruly or sick passengers to small talk and exchange of connecting flight information. Similar to SMS messages, the character limit often leads to use of word contractions (“THX”, “PLS”). There have been reports of even basic emails transmitted via ACARS [186].

*Automated human-readable content.* Many messages are sent to/from the aircraft by automated systems but intended for human consumption. Examples include passenger and crew lists, weather data, positional information, and flight plans. In principle, ACARS can be used as an all-purpose datalink with larger messages spread over several consecutive messages.

*Automated machine-readable content.* Mostly employed by commercial airlines, this type of message typically includes sensor and performance data regarding engines and other crucial aircraft parts that require monitoring. The input for these messages comes mostly from the Aircraft Condition Monitoring System (ACMS), which provides vibration, temperature, and pressure monitors as well as speed measurements. The real-time nature of these applications creates a significant data throughput. Besides the ACMS, there can be further proprietary, user-defined applications with similar content.

*Dataset.* We use the Smith et al. dataset [185] throughout this chapter. This dataset includes 1.18 million ACARS messages from satellite, VHF, and VDL2 transmission channels. The reception coverage of messages is focused on Switzerland.

### 5.2.3 Usage

In modern aviation, ACARS is used as a general-purpose datalink, outgrowing its original purpose. Aircraft transmit and receive safety-critical data such as aircraft weight, fuel, engine data, or weather reports; privacy-related information about passengers and crew; and critical business operation data such as gate assignments, crew schedules, or flight plan updates.

ACARS is generally used by larger aircraft with expensive avionics, i.e., not typically by individual citizens in small general aviation aircraft such as a Cessna 172. Besides commercial aviation, its users include governments, corporations, and the military [185]. Using ACARS is costly, both providers charge a significant bandwidth usage fee.

### 5.2.4 Security

ARINC made no considerations regarding information security in ACARS' original design, similar to most legacy aviation technologies [188]. Consequently, eavesdropping on ACARS reception has been a known problem for over two decades. This was worsened significantly by the ease of access provided by modern software-defined radios and large-scale website-based collectors. In response to this issue, several security add-ons were developed in the 2000s including the AMS standard [184], which has been independently validated by academic researchers [189].

Unfortunately, AMS incurs a financial cost, which seems to strongly impede its deployment – previous research has found little to no evidence of its use in the wild [185, 186]. Potentially in response to privacy concerns and the cost of AMS, proprietary solutions by individual stakeholders have been developed such as the insecure protocol analyzed in Smith et. al [185]. The prevalence of such proprietary ‘hacks’ in the aviation datalink ecosystem motivates our present work.

### 5.2.5 Ethics

Our experiments use publicly broadcasted data. The existence of ciphers shows that some aviation actors care about the secrecy of these messages, yet they fail to achieve it due to using insecure classic ciphers. We disclosed our findings to impacted aircraft manufacturers without decrypting insecure ciphertexts. Our aim is to enable regulators to notify impacted parties and ensure that actors do not have wrong impressions of security. A false sense of secrecy is more dangerous and harmful than knowingly using public channels. As our research involves real data, we reached out to our organization's ethics committee and received approval.

## 5.3 Related work

*Classic cryptanalysis.* Classical methods such as the Kasiski test [190], index of coincidence [191], and frequency analysis [192] help detect and decrypt monoalphabetic and polyalphabetic substitution ciphers. A more general approach is to use entropy [193] to distinguish plaintexts from ciphertexts: plaintexts are expected to have a lower entropy. Unfortunately, entropy-based

metrics do not work well for ACARS messages. These messages are more complex than natural language texts, have lower repetition, and thus have higher entropy. Moreover, classic ciphers do not manage to hide the underlying characteristics of ciphertexts well leading to lower entropy than their secure counterparts. Combining these two facts results in entropy being an ineffective detection measure.

*Automatic classic cipher decryption.* The basic idea behind automatic decryption is to search for keys that decrypt the ciphertext to promising plaintext candidates and then ranking these candidates based on their likelihood of being a proper human written message [194–198]. These methods work well for long natural language texts, but fail for ACARS messages that are short and heterogeneous. We adapt and extend one of the compression-based ranking mechanisms [199] to distinguish plain- from ciphertexts.

*Traffic classification.* A related topic to our goal is network traffic classification as it aims to detect encrypted network traffic. Recent works use statistical and machine learning methods to detect encrypted traffic [200–202]. However, none of these support insecure classic ciphers as they exclusively focus on modern (secure) ciphers. Moreover, most methods heavily rely on traffic flow and network features beyond text in classification which are unavailable in our scenario [203].

*Data leakage.* A related yet orthogonal problem is automatically detecting *data leakage* in network traffic. For example, Reardon et al. [204] study Android applications at scale to detect patterns leading to leakage of personal data.

*Avionics.* The unique legacy communication protocols found in aviation have been subject of recent research. Much of the literature focuses on active attacks, see Strohmeier et al. for an overview [188], but the (lack of) confidentiality of communications [185, 186, 205] and the privacy of aviation stakeholders [206] has recently received increased attention, too.

## 5.4 Detection methods

We develop three automated methods to distinguish encrypted ACARS messages from those sent in clear at scale. Despite flagging all encrypted communication, our goal is to identify weak ciphers. Thus, we focus our design and evaluation on detecting insecure ciphers.

### 5.4.1 Cipher detection based on text compression techniques

Plaintext ACARS messages can be modeled as a (mix of) language(s). A message is a ciphertext if it is unlikely to be generated from the plaintext language. We first build a Hidden Markov Model (HMM) of plaintext ACARS messages and subsequently use it to determine whether a message is a plaintext that originates from the same language.

*Language definition.* Languages are well studied in computer science. Despite the popularity of simpler languages (e.g., regular or context-free) in designing protocols, many ACARS protocols are semi-decidable: they support free-text messages [207]. Furthermore, many of the protocols used on top of ACARS are proprietary and their specifications are inaccessible. Therefore, classic language theory tools that rely on the definition of a language, e.g., grammars, are infeasible in the ACARS setting.

Instead, we use an empirical approach to build a statistical language model of the ACARS plaintext language. We assume that plaintext ACARS messages, which are strings of characters sent in the clear, form a language  $\mathcal{L}$ . This language is more extensive than the traditional ‘comprehensible human-written text’ language definition because ACARS includes sensor-generated machine-readable messages.

Statistical language models have been used to support the automatic decryption of English messages encrypted with classical ciphers [196–198]. Al-Kazaz [199] uses the prediction by partial matching (PPM) compression algorithm, which uses an internal language model trained on a corpus of English texts, to determine whether a message is a proper English text. The algorithm achieves a high compression ratio for English texts but this ratio declines otherwise. We follow a similar method to distinguish plain- from ciphertexts.

**Prediction by partial matching.** The PPM compression algorithm processes text in a streaming fashion with two main components: a statistical model to determine the probability of each character’s occurrence, and an encoder to output characters based on their probability.

*Statistical Model.* The statistical model underlying PPM compression is an HMM of order  $k$ . This model determines the probability distribution over each character  $c_i$  in a message  $m = (c_1, \dots, c_l)$  based on the length- $k$  prefix  $s_i = (c_{i-(k+1)}, \dots, c_{i-1})$  [208]. The model is trained on a training set  $M = \{m_1, \dots, m_n\}$  with messages  $m_i$  from the target language  $\mathcal{L}$ . The trained

model is a table that represents the probability of a character  $c_i$  following a prefix  $s_i$  of length  $k$ .

Combinations of prefix  $s$  and characters  $c$  that do not appear in the training set are assigned a probability of zero, resulting in a failure to encode the character. Therefore, PPM compression uses  $k$  Markov models, of order 1 to  $k$ . Whenever the  $i$ 'th model fails to represent a character sequence, it produces a special character that tells the encoder to use the  $i - 1$ 'th model instead. This “skip model” character is assigned a probability  $p_s$  in each Markov model. There are various approaches to computing  $p_s$ ; based on prior work [198] we use the PPMD approach and set order  $k = 5$ .

*Encoder.* PPM compression encodes messages with an arithmetic encoder using the trained statistical model [208, 209]. We compute the compression ratio of a message  $m$  as  $r(m) = |\text{Encode}(\text{Model}(\mathcal{L}), m)| / |m|$ .

**Ciphertext detection.** The compression ratio  $r(m)$  captures the likelihood that the message  $m$  is generated by the language model underlying the PPM compression: the lower the ratio, the higher the probability that the message originates from the target language, i.e.,  $m \in \mathcal{L}$ . We build two PPM-based detectors to classify messages as `plain` (comes from the language) or `cipher` (does not come from the language).

*Gaussian detector (G-PPM).* The G-PPM method assumes that the compression ratios of messages generated from the same language follow a normal distribution. We take a second set of plaintext messages  $M'$ , disjoint from the set  $M$  used to train our compression model, compress each message in  $M'$  using the trained PPM model, and fit the parameters of a Gaussian distribution to the resulting compression ratios. We then label a message as `plain` if its compression ratio  $r(m)$  is within  $d = 4$  standard deviations of the mean of the fitted Gaussian. Otherwise, we label the message as `cipher`.

*Two-class detector (2C-PPM).* The 2C-PPM method takes an empirical, non-parametric approach to determine the compression ratio that separates plaintext messages, coming from the target language modeled by the trained HMM, from ciphertexts. It only relies on the assumption that the compression ratio of a message  $r(m)$  is directly proportional to the likelihood that  $m \in \mathcal{L}$  and should be labeled as `plain`. To find the best compression threshold that separates the two classes, we use a *labeled* dataset  $M'$  where half of the messages are labeled as `plain`, i.e.,  $m \in \mathcal{L}$ , and the other half is labeled `cipher`. We compress each message in  $M'$  and perform a grid search to determine the

threshold  $r'$  that maximizes the F1 score of separating messages based on their class. We then label a message as **plain** if  $r(m) \leq r'$  and **cipher**, otherwise.

**Implementation.** We modify an existing python implementation of the PPM algorithm [210] for our use case.<sup>2</sup> We implement three core functionalities: train a compression model, train a G-PPM or 2C-PPM detector, and classify messages.

### 5.4.2 Supervised text classification with a CNN model

The problem of detecting encrypted messages can also be cast as a supervised learning task. Similar to existing text classification tasks, in which a free text document is assigned to one out of a fixed set of predetermined categories [211], we treat the content of each message  $m$  as a sample from the class distribution indicated by its label. We design a CNN detection method using a Convolutional Neural Network (CNN) model that learns a representation of messages to separate the two distributions underlying the **plain** and **cipher** messages.

A prerequisite for the supervised learning approach is access to a sufficiently large set of labeled messages that serves as a training set. It is crucial that the messages in the model's training set are *representative* of the messages encountered at test time, or during deployment. Otherwise, the model will have a *large generalization gap* and fail to detect ciphers during deployment under real-world settings.

*Message encoding.* While almost all text classification models use the natural construct of words to quantize raw text data into numerical features [211, 212], we follow the approach presented by Zhang et al. [213] and treat each message as a sequence of single characters. We model a message  $m$  as a sequence of characters  $(c_1, c_2, \dots, c_l)$  from a pre-defined character set, the alphabet  $\mathcal{C}$ . This approach addresses many of the challenges described in Section 5.1: it makes the classification agnostic to the language in which a message was written, it supports abnormal character sequences, such as sensor messages, and enables the use of CNNs which have been found useful in extracting information from sequential data.

We encode each message  $m$  as a fixed length sequence of one-hot vectors. Given an alphabet  $\mathcal{C}$  of size  $k$ , each character  $c_i$  is translated into a vector of

---

<sup>2</sup>We will open-source our code upon publication without releasing data.

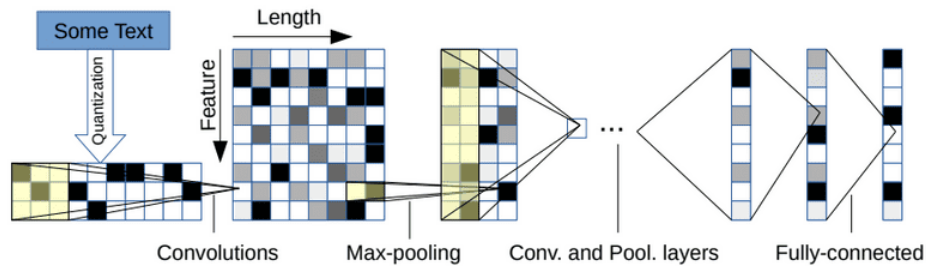


Figure 5.2: Character-level CNN by Zhang et al. [213]

size  $k \times 1$  which is all zeros except at  $k_i$ , the position of character  $c_i$  in the alphabet. Starting from the last position, we transform a message  $m$  into a binary array of size  $k \times l$ , where  $l$  is the number of characters in the message. We fix the maximum sequence length of any message to  $l_{max}$  and ignore any characters that exceed this limit. Messages with  $l < l_{max}$  are padded to the maximum sequence length with zeroes.

*Model architecture.* The model described by Zhang et al. [213] is a 9-layers deep CNN with 6 convolutional and 3 fully-connected layers. We implement the small version of the model with 256 1-dimensional kernels in each convolutional layer and varying kernel sizes (see original paper and implementation [213]). The first two and the last convolutional layers are followed by temporal max-pooling which enables training deeper models. The first two fully-connected layers present a dropout layer with dropout probability  $p = 0.5$ . The number of input features to the first convolutional layer and the number of output features of the last fully-connected layer are determined by the problem. The alphabet size  $k$  is determined by the chosen plaintext alphabet (see Section 5.5.6) and the number of output classes is equal to 2 (`plain` and `cipher`). We use the same non-linearity (ReLU) and training procedure (mini-batch SGD) as in the original paper.

*Implementation.* We implement the model using the `PyTorch` framework [214]. The implementation provides functions for training and evaluating a trained model, and can be adapted to run either the small or large version of the CNN presented by Zhang et al. [213].

### 5.4.3 Supervised classification with a Random Forests model

So far, we considered purely text-based approaches only. Now, we augment our classification with metadata. We design the `MetaRF` detection method using a

Random Forests (RF) model that combines metadata with features extracted from the messages' text to label them as either `plain` or `cipher`.

*Message metadata.* Each message sent over the ACARS network includes an aircraft registration field (see Section 5.2.2). We use the registration to identify the aircraft in open-source avionics datasets such as OpenSky's aircraft registration [215] and gather additional categorical information about the aircraft such as manufacturer, model, owner, and operating airline. Previous studies showed that the use of certain weak ciphers is most prevalent among a small number of manufacturers and aircraft models [185]. This suggests a high correlation between a message's metadata and its label. We leverage this relationship for training our Random Forests classifier.

*Text features.* A model trained on message metadata only would not be able to correctly label messages from unknown categories, such as any manufacturer not included in the model's training set. We hence combine the metadata with a set of text features extracted from the actual message text. The combination of these two types of features, message text and categorical metadata features, ensures that the RF classifier is able to both flag *new types of ciphertexts* and the use of known ciphers by *new actors* (e.g., manufacturer or airline). We leave the detailed description of features used to train the model to Table D.1 in Section D.1.

*Limitations on application.* The `MetaRF` detector combines domain knowledge through metadata with the power of natural language processing. A major drawback, however, is its reliance on a fixed set of metadata fields to classify messages. This *restricts its usage to use cases where such metadata is readily available and correlated with the target label*. While our primary task, flagging encrypted ACARS messages, fulfills this requirement, the purely text-based detectors (i.e., `G-PPM`, `2C-PPM`, and `CNN`) are more widely applicable to similar use cases that do not meet this criteria. We hence focus our evaluation in Section 5.5 and 5.7 to these text-based detectors.

*Implementation.* We use the `sklearn` library [216] to implement an RF model with 10 estimators. The implementation provides model training and classification functionalities.



## 5.5 Cipher detection on generated datasets

Our ultimate goal is to apply the detection methods introduced in Section 5.4 on real ACARS messages, flag encrypted ones, and then ask an analyst to identify weak ciphers. We have to solve three major challenges to achieve this goal.

First, ACARS messages available through data collection of real-world communication are not labeled and contain a mix of plain- and ciphertexts. Without labeled data, we can neither create a clean plaintext corpus to feed to the PPM detectors nor train any of the supervised learning-based detectors. Labelling ACARS messages, however, requires specialist knowledge about both aircraft communication protocols and (weak) encryption schemes. Thus, typical approaches for gathering a labeled dataset such as crowdsourcing that rely on untrained volunteers are not applicable. We further discuss the challenges of obtaining labeled data and design a methodology to overcome them in Section 5.6.

Second, the content of ACARS messages is highly heterogeneous as seen in Section 5.2.2. The PPM detectors assume that plaintext messages form a single well-defined language. This heterogeneity increases the complexity of building a unified language model and finding a threshold that optimally separates the compression ratio of plain- and ciphertext messages. Similarly, a highly heterogeneous class of plaintext messages makes it difficult for our supervised classification methods, *CNN* and *MetaRF*, to find a good representative of the plaintext class that cleanly separates the two classes.

Third, we do not know what ciphers generate encrypted ACARS messages. While there is evidence for the use of classical ciphers, such as mono-alphabetic substitution ciphers [185], other types such as transposition or more modern ciphers may be in use. Thus, our detectors must be able to detect new ciphers that were not present in their training set.

We explore how each of these factors might affect our detectors in a simplified setting before addressing them on real-world ACARS data in Section 5.7. For our evaluation, we use synthetic data generated from pre-processed clean (plain-)text corpora. The evaluation with synthetic datasets brings two main benefits. First, we know the ground truth label for every message. Second, it enables us to study how heterogeneous text sources and domain shifts between train and validation sets affect our detectors.

Table 5.1: Summary of plaintext sources.

Name	#Msgs	Mean(len)	Max(len)	Examples
Bible	42k	94	150	for his mercies are great.. And he said unto me, Go..
SMS	56k	52	500	Okay set! Put on Facebook? Love u back
Sensor	10k	64	438	PIKCPYA.ADS.N407KZ08081438DA PIKCPYA.AT1.G-CIVL212E8168.. NIMCAYA.CR1.G-CIVO205837A8..

### 5.5.1 Synthetic data generation

*Text sources.* We use three plaintext sources for our experiments. (1) *Bible*: an English Bible taken from the Canterbury corpus [217]. We split the text into sentences with at most 150 characters to simulate the challenge of short messages. Books provide a simple structured language and are the traditional choice used in prior work. (2) *SMS*: a corpus of English SMS messages from students of the National University of Singapore [218]. The informal conversations in the SMS dataset are more similar to the pilot written messages that we encounter in the ACARS dataset. (3) *Sensor*: machine-generated ACARS messages from a single sensor selected from Smith et. al’s dataset [185]. These selected sensor messages are plaintexts with high probability and represent the plaintext sensor data which we face in ACARS. Table 5.1 shows a summary.

*Ciphertext generation.* From each of these sources, we extract plaintext messages and encrypt them with classical ciphers to generate ciphertext samples.

Classic ciphers were originally designed to encrypt messages written in the standard Latin alphabet (A-Z). To be able to encrypt messages that contain characters outside this set, such as numbers or the special characters found in ACARS messages (see Section 5.2), we need to extend the set of supported plaintext characters which we call the *plaintext alphabet*. We refer to the set of all possible characters appearing in ciphertexts as the *cipher alphabet* and the set of all possible keys as the *keyspace*.

We use four types of ciphers in our synthetic experiments:

*Mono-alphabetic substitution.* A mono-alphabetic substitution cipher creates a one-to-one mapping between the plaintext and cipher alphabets as the encryption key. Both encryption and decryption use this key to translate characters from plaintext to cipher alphabet, and vice versa. We represent this category

with Caesar and simple substitution ciphers.

*Poly-alphabetic substitution.* Poly-alphabetic substitution ciphers are an extension of mono-alphabetic ciphers that use *multiple* plaintext to cipher alphabet mappings for encryption. We represent this category with the Vigenere cipher.

*Transposition.* Transposition ciphers permute the characters in a plaintext message. We represent this category with the columnar cipher.

*Modern.* We use this category to represent modern ciphers that produce a pseudo-random stream of characters. As we are not assessing the security of modern ciphers in this chapter, we assume these ciphers to be secure. We represent this category with an ideal primitive (PRF) that replaces plaintexts with a uniformly random binary string of the same length. As the ACARS protocol does not support binary transmission, we randomly use one of hex, base-32, base-64, or base-85 encodings to convert the binary ciphertext to an ASCII string.

### 5.5.2 Experiment setup

The basic experiment setup in this section is built on a set of plaintext messages  $M$  extracted from one of the three text sources (Bible, SMS, and Sensor) and a set of  $n_c$  ciphers  $\mathcal{E}$ . Unless explicitly defined, the default set  $\mathcal{E}$  contains the Caesar, Substitution, Vigenere, Columnar, and Modern ciphers.

For each experiment, we run a five-fold cross-validation and split  $M$  into 5 train and validation pairs,  $M_{train}$  and  $M_{val}$ . In each fold, we split both training and validation sets into two halves. One half we leave unencrypted and label as `plain`. The other half, we split into  $n_c$  equal-sized groups, encrypt each group with a different cipher from our cipher suite  $\mathcal{E}$ , and label them as `cipher` messages. For each cipher in  $\mathcal{E}$ , we use 10 random keys for encrypting messages in  $M_{train}$  and 10 *fresh* random keys for encrypting messages in  $M_{val}$ . We set all characters that occur in the plaintext source  $M$  as our plaintext alphabet.

We use the entire training set  $M_{train}$  of each fold to train the CNN and MetaRF detectors. For the PPM detectors, G-PPM and 2C-PPM, we split  $M_{train}$  into two unequal parts: 80% of the data is used for building the compression model, and the remaining 20% is used for finding the decision threshold. The G-PPM detector does not use ciphertext samples at all while the 2C-PPM detec-

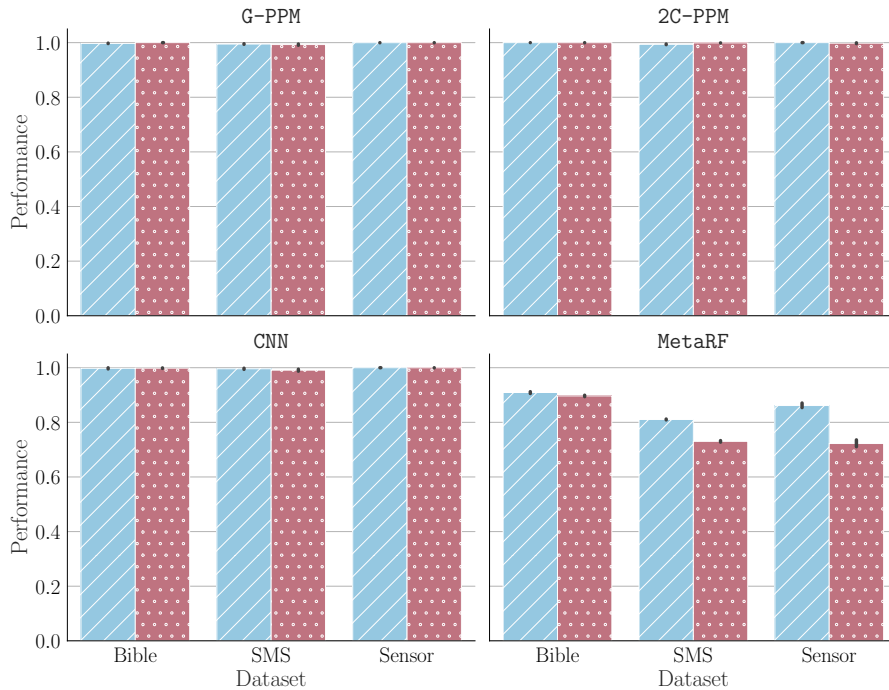


Figure 5.3: Precision  and recall  of ciphertext detection.

tor only uses ciphertexts in the 20% of  $M_{train}$  that is dedicated to finding a suitable threshold. In plots, we report the mean precision or recall of ciphertext detection for each detector on the five validation sets, and error bars show the 95% confidence interval.

### 5.5.3 Baseline

We first demonstrate the effectiveness of our detectors if their respective requirements (see Section 5.4) are met. We use the experiment setup from Section 5.5.2 for all sources.

The text-based detectors (G-PPM, 2C-PPM, CNN) achieve near-perfect accuracy in distinguishing ciphertexts from plaintexts with an average precision and recall of 99% across all three datasets if clean, labeled data is available for training (see Fig. 5.3). The MetaRF detector only reaches an average precision and recall of 84% and 76% across datasets, respectively. However, this is because in these experiments there is no metadata available for classification. The model hence needs to rely purely on the features extracted from message texts which substantially reduces its predictive power. In the following, we focus our evaluation on the three purely text-based detectors.

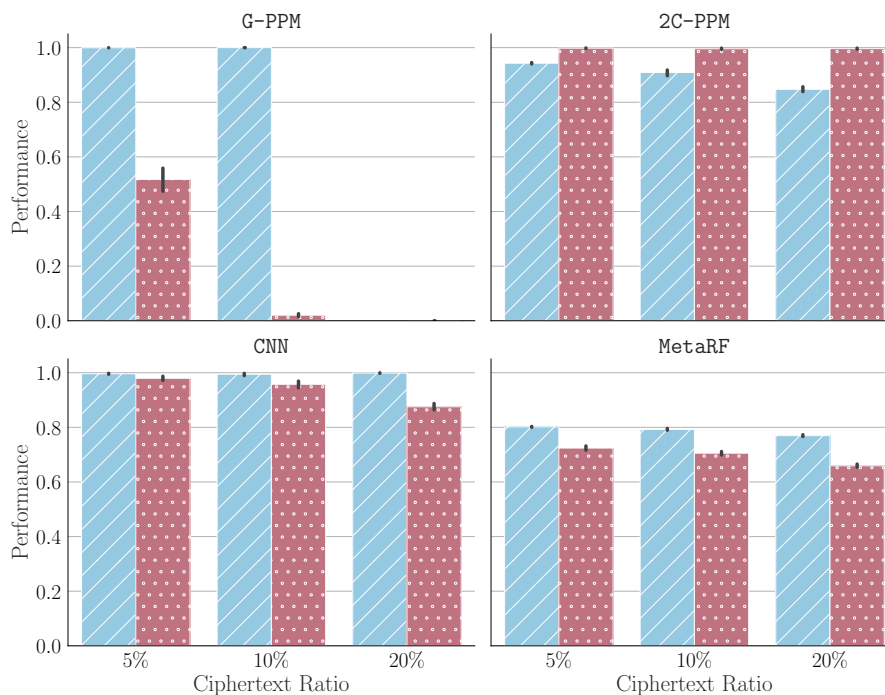


Figure 5.4: Precision ▨ and recall ▨ of ciphertext detection when models are trained on the SMS corpus with a varying ratio of wrongly labeled ciphertexts.

#### 5.5.4 Noisy data labels

Data collected from real-world ACARS traffic is unlabeled, and manual labeling by experts is costly and prone to errors. Given the class imbalance in the ACARS data – only a small fraction of messages is suspected to be ciphertexts – we expect that a random sample of messages contains a large majority of plaintexts. Thus, one could use such a random sample directly as a (noisy) plaintext training set.

To study the impact of noisy labels during model training, i.e., ciphertexts wrongly labeled as plaintext messages, we intentionally add noise to our training data and vary the ratio of ciphertexts that are wrongly labeled as `plain`. We use the same basic experiment setup as before. We present experiments for the SMS dataset but observe similar results for all other text sources. In each experiment, we replace  $e\% \in [5, 10, 20]$  of `plain` messages in the training set with `cipher` messages and randomly drop  $e\%$  of `plain` messages to ensure the class balance during training. Fig. 5.4 shows the performance of our detectors when the training plaintext data is noisy.

The CNN detector is most robust to noise in its training labels. Recall drops

from 99.6% to 87.6% if 20% of plaintext training samples are wrongly labeled. Precision remains unaffected. In contrast, the **G-PPM** detector performs very poorly when trained on noisy inputs. Even just 5% of wrongly labeled training samples leads to precision as low as 50%. Increasing the noise to 20%, leads to **G-PPM** classifying all validation messages as plaintext. Hence, there is no ciphertext precision bar in the graph. The **2C-PPM** detector is considerably more robust: precision drops slightly to a minimum of 84.7% if 20% of samples are wrongly labeled while recall stays perfect.

Our experiment (intentionally) uses the same set of keys for encrypting wrongly labeled plaintexts and correctly labeled ciphertexts. Unlike modern ciphers, messages encrypted with the same key using classic ciphers produce highly correlated ciphertexts. This correlation amplifies the performance drop but is necessary as there is no guarantee that ciphertexts labeled as **plain** have different keys from ciphertexts that we want to detect when using a noisy training set.

*Conclusions.* Given that we cannot guarantee to obtain clean, labeled training data in the ACARS setting, we exclude the **G-PPM** detector from further analysis. Both the **CNN** and **2C-PPM** detectors tolerate noisy training data, but with different trade-offs. While the **CNN** offers higher precision the **2C-PPM** provides close to perfect recall.

### 5.5.5 Heterogeneous plaintext sources

ACARS messages are highly heterogeneous and vary in purpose (conversations, requests, etc.), language, and authorship (pilot, sensor). Besides, many proprietary protocols use ACARS as a transmission channel. These protocols are often private and there is no public list of existing protocols. Even aviation communication experts struggle to decide if a given message is from known standard protocols or from a new proprietary one due to the high entropy of acars messages. Given these factors, manually detecting and separating languages in the ACARS dataset is infeasible, and the ‘ACARS language’ is best modeled as a mixture of smaller languages as opposed to a single well-structured language.

In this section, we hence study the behavior of the **CNN** and **2C-PPM** detectors in the presence of plaintext heterogeneity.

*Mix of languages.* To assess how training our detectors on messages from a mixture of languages affects performance, we use the experiment setup in

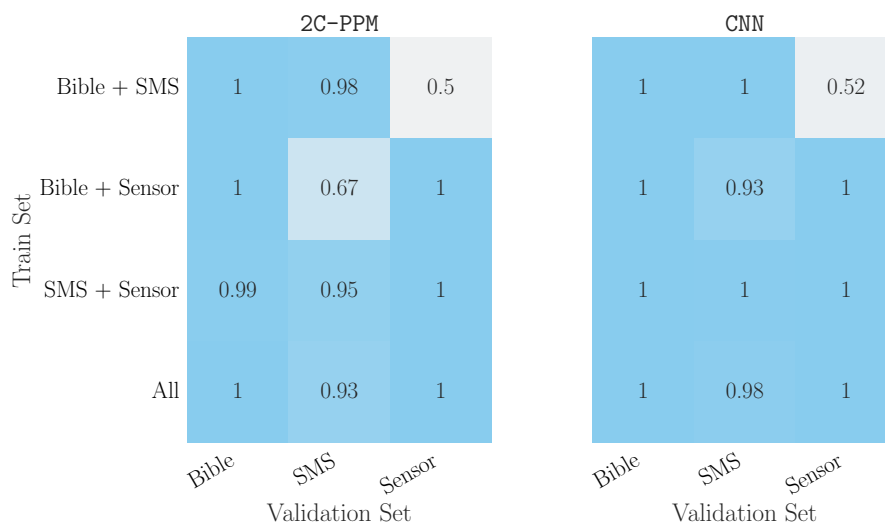


Figure 5.5: Precision across validation sets from different text sources (columns) for models trained on a mix of languages (rows).

Section 5.5.2 but create datasets that contain a mixture of messages from different plaintext sources. We randomly sample 8000 messages from each plaintext dataset and combine them into 4 mixed-language training datasets that contain a mix of 2 or 3 different languages. For validation, we use the original single-source validation sets. Fig. 5.5 shows the precision of detectors trained on mixed-language plaintext sources. The detectors’ recall remains unaffected under this experiment (see Fig. D.1 in Appendix D.2).

Training on a mix of languages affects the detectors’ performance only in some cases. The average precision of the 2C-PPM detector on validation sets from the SMS corpus drops from 98.2% to 93.1% when training on a dataset that contains messages from all three sources instead of SMS texts only. The CNN’s performance remains unaffected. For both detectors, however, we observe a clear decrease in precision when their training sets did not include any plaintexts from the validation source. Next, we study this effect in more detail.

*Plaintext distribution shifts.* ACARS messages come from a variety of sources and there is no guarantee that messages observed during training are representative of those that will be classified later. In other words, there is a high likelihood that there will be a *shift* between the distribution of messages in the models’ train and test sets.

To assess how well models trained on messages from a specific language perform on test messages from a different language, we use the experiment



Figure 5.6: Precision across validation sets from different text sources (columns) for models trained on different texts (rows).

setup in Section 5.5.2 but pair training and validation sets of different text sources. We show the average precision of the 2C-PPM and CNN detectors under this setting in Fig. 5.6. The diagonal displays the results for training and validation sets from the same text source.

The precision of both detectors is significantly affected by certain shifts in the plaintext distribution between their train and validation sets. While both detectors retain a high average recall across all text combinations ( $\geq 99.0\%$ , see Fig. D.2 in Appendix D.2), there is a high likelihood to misclassify plaintext messages from unseen sources as `cipher`.

The CNN detector is more resilient to plaintext domain shifts than 2C-PPM. For instance, only when trained on SMS and tested on Sensor messages, CNN’s average precision drops significantly to 69.4%. This robustness is most likely due to CNN’s ability to find a coherent representation of *ciphertxts*. As long as ciphertxts characteristics remain unchanged between training and validation sets, a CNN detector may accurately distinguish between the two classes. Our experiments in Section 5.5.6 further strengthen this hypothesis.

The 2C-PPM detector generalizes well across human-written text sources (SMS and Bible). However, moving between human-written texts and machine-generated sensor messages leads to a large drop in precision. A detector trained on the SMS or Bible corpus labels nearly all Sensor messages as `cipher` because its underlying language model confidently detects that sensor plaintexts are not generated from the language it was trained on; therefore marks them



as encrypted.

*Unbalanced mix of languages.* In the previous experiments, we observed that training on a balanced mix of languages with *an equal number of messages* from each source affects our detectors’ performance only marginally but that they tend to perform poorly on messages from languages never seen before. An analysis of real-world ACARS communications shows that the situation encountered in practice lies somewhere in-between: While the majority of messages stem from a small number of common protocols and natural languages, there is a large number of proprietary protocols that contribute a small number of messages each. It is thus highly likely that a random sample of ACARS messages used for training our detectors contains an unbalanced mix of plaintext types.

To assess the performance of our detectors trained on an unbalanced language mix, we use the experiment setup in Section 5.5.2 and generate training sets of fixed size in the following manner: From each data source, Bible, SMS, and Sensor, we select 6400 (80%), 7200 (90%), and 7840 (98%) messages and complement this set with an equal number of messages from the other two sources to obtain a total of 8,000 messages in each training set. Fig. 5.7 shows the average precision of the 2C-PPM and CNN on a training set where 98% of messages come from the majority source. In this case, the two minority classes contribute only 40 plain and 40 cipher messages each to the training set. The detectors’ recall remains unaffected under this experiment (see Fig. D.3 in Appendix D.2).

Even a very small number of training samples from a specific source is sufficient for the CNN detector to achieve high performance on messages from that source. Mixing languages does not degrade the performance of either detector on the majority source. However, 2C-PPM does not achieve a high precision on SMS messages when this source contributes only 1% of samples in its training set. Fig. D.4 in Section D.2 shows that improving 2C-PPM’s performance on minority languages requires at least 5% representation in the training data.

*Conclusions.* To summarize, the CNN detector shows a high tolerance towards shifts in plaintext distributions as long as its train set contains at least a small number of samples from the target distribution. The 2C-PPM detector requires at least 5% of training messages to originate from the test language or high similarity between train and test languages.

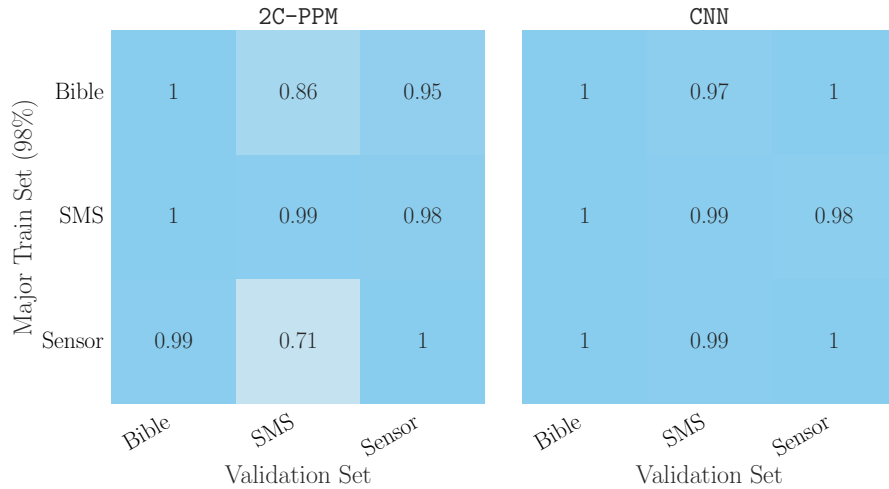


Figure 5.7: Precision across validation sets from different text sources (columns) for models trained on train sets with an unbalanced mix of languages (rows). Row labels indicate the majority source of training samples (98%).

### 5.5.6 Heterogeneous cipher suites

As we have little knowledge about the types of ciphers and their implementations used by aircraft providers [185], our detectors need to generalize well across ciphertext distributions: i.e., they should correctly identify ciphertexts generated by ciphers not observed at training time. We intentionally create domain shifts in our synthetic ciphertexts to study the impact of these shifts. We report results for the SMS dataset but observe equivalent results on all other text sources.

*Generalization across cipher types.* As it is close to impossible to create training sets that include all possible cipher types and implementations, we study whether our detectors are able to detect cipher types not seen during training.

From the SMS source, we generate four different datasets, each using a different cipher suite  $\mathcal{E}$  to generate encrypted messages (see Section 5.5.2 for our general experiment setup). We generate one dataset per cipher type described in Section 5.5.1: mono-alphabetic ( $\mathcal{E} = \{\text{Caesar, Substitution}\}$ ), poly-alphabetic ( $\mathcal{E} = \{\text{Vigenere}\}$ ), transposition ( $\mathcal{E} = \{\text{Columnar}\}$ ), and modern ( $\mathcal{E} = \{\text{PRF}\}$ ) ciphers. During cross-validation, we train a detector on each cipher’s training set then evaluate its performance on all four validation sets. This way we obtain the detectors’ average recall in detecting ciphertexts from the same family as its training examples (diagonal entries) and across cipher types (off-diagonal entries) shown in Fig. 5.8. The detectors’ precision remains

		2C-PPM				CNN			
Train Set	Monoalphabetic	0.99	1	0.99	1	0.98	0.99	0.23	0.99
	Polyalphabetic	0.99	1	0.96	1	0.97	0.99	0.097	0.96
	Transposition	0.99	1	0.99	1	0.97	0.98	0.98	0.98
	Modern	0.97	0.99	0.68	1	0.061	0.061	0.018	1
		Validation Set				Validation Set			
		Monoalphabetic	Polyalphabetic	Transposition	Modern	Monoalphabetic	Polyalphabetic	Transposition	Modern

Figure 5.8: Recall across validation sets from the SMS data when the training cipher suite (rows) differs from the validation cipher suit (columns).

unaffected under this experiment (see Fig. D.5 in Appendix D.2).

The 2C-PPM detector’s performance is rarely impacted by cipher domain shifts. Only a detector trained on Modern ciphers fails to classify ciphertexts generated by a transposition cipher. This robustness is due to learning a *coherent representation of plaintext messages*. Only when the chosen threshold  $r'$  to separate plain- from ciphertexts is too high (see the last row, Fig. 5.8 left), the 2C-PPM detector misclassifies ciphertexts with a low compression ratio. In contrast, the CNN detector consistently fails to detect messages encrypted with ciphers less secure than those seen during training. For instance, when trained on Modern ciphers (last row, Fig. 5.8 right), it does not detect classic ciphers. Similarly, if the CNN detector is trained on substitution ciphers that impact the character frequency of messages, the detector labels transposition ciphertexts, which do not hide the character frequency, as `plain`. We suspect that the CNN detector uses the characteristic that the train ciphers hide to distinguish classes. For example, modern ciphers hide the character frequency so the detector uses non-uniform character frequency to detect plaintexts; and misclassifies insecure ciphers that retain these characteristics.

*Generalization across keyspace.* The ciphertext distribution may be impacted by the choice of keyspace, especially in substitution ciphers, where the key determines how plaintext characters map to ciphertext characters. The ACARS transmission channel supports sending 7-bit ASCII characters, but many proprietary protocols *use a reduced character set*, e.g., only uppercase alphanumeric characters plus a small set of special characters. It is likely that the

Table 5.2: Character sets used as keyspace.

Name	Size	Character Set
$\mathcal{C}_{\text{Latin}}$	33	[a-z, -./?:*]
$\mathcal{C}_{\text{AlphanumericLower}}$	43	[a-z0-9, -./?:*]
$\mathcal{C}_{\text{Alphanumeric}}$	69	[a-zA-Z0-9, -./?:*]
$\mathcal{C}_{\text{SMS}}$	97	[a-zA-Z0-9, .;:!'?"\t\n#%&() [] *+~/<=>\@]

custom encryption methods, designed to encrypt ACARS messages from specific protocols, directly use the protocol’s reduced plaintext alphabet as the ciphertext space. This poses a problem when generating training sets: As we do not know the specific plaintext alphabets of ACARS protocols that contain weak ciphertexts nor how the ciphertext space of their proprietary encryption is chosen, our detectors are likely to be trained on ciphertexts with a much larger cipher alphabet than those encountered at test time. We measure how differences in the ciphertext space between training and validation sets impact our detectors’ performance on substitution ciphers.

To mimic such shifts, we choose 4 character sets that are subsets of each other (see Table 5.2). We run our basic experiment setup from Section 5.5.2 but use the simple substitution as the only cipher. We train one detector using the full SMS dataset and set  $\mathcal{C}_{\text{SMS}}$  as the keyspace to generate synthetic ciphertexts. We then generate four different versions of our validation set by choosing one of the four character sets listed in Table 5.2 as our plaintext/ciphertext space and discarding any messages from the validation set that contain characters outside this set. Fig. 5.9 shows the performance of our detectors trained on datasets using the full SMS character set as ciphertext space validated over datasets containing plain- and ciphertext messages from a (reduced) character set.

The 2C-PPM detector’s performance is unaffected by shifts in ciphertext space. In contrast, we observe a significant drop in the CNN’s recall when the character set of validation messages is substantially smaller than that of training messages.

### 5.5.7 Conclusions

Our extensive evaluation of the four detectors introduced in Section 5.4 uncovers their varying trade-offs. When trained on a labeled dataset that is representative of the messages encountered at test time, the three purely text-based

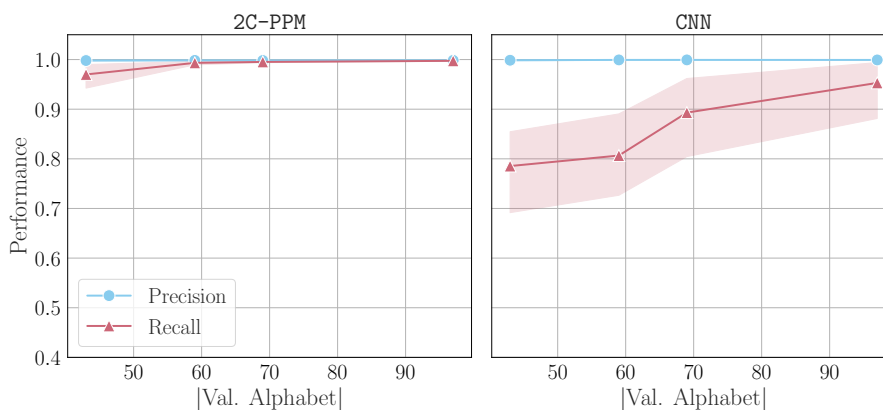


Figure 5.9: Precision and recall when varying ciphertext spaces in validation set.

detectors, **G-PPM**, **2C-PPM**, and **CNN**, clearly outperform the **MetaRF** detector if no metadata is available. The **G-PPM** method, however, does not tolerate noise in training labels and is not a viable option for classifying ACARS messages.

In contrast, the **2C-PPM** detector still achieves close to perfect recall even when the training data includes a small ratio of wrongly labeled ciphertexts and only suffers a slight reduction in its precision. It is furthermore robust to cipher domain shifts such as mismatching cipher type or keyspace. The drawback of the **2C-PPM** detector is sensitivity to plaintext domain shifts. Large shifts, such as those between human-written texts and machine-generated sensor data, between the detector’s train and validation sets prompt the **2C-PPM** to misclassify plaintexts from the target language as ciphertexts.

The **CNN** detector shows resilience to noisy training labels and plaintext domain shift. However, it often fails to recognize ciphertexts generated from a new cipher type not included in its training set. Our experiments suggest that the **CNN**’s tendency to misclassify ciphertexts as plaintexts is largest when the test cipher is less secure than the ciphers the model has been trained on and retains certain plaintext characteristics, such as non-uniform character frequencies.

## 5.6 A pipeline for labeling ACARS messages

We investigated the behavior of our detectors on synthetic data in a setting where we control any experimental factors that might influence their performance. Now, we draw on the insights from these experiments to design a data pipeline that starts with a set of unlabeled ACARS messages collected from

real-world aircraft communications and results in a labeled dataset that can be used to train any of the detection methods introduced in Section 5.4. We build our pipeline in a way that minimizes the manual labor needed and maximizes the number of flagged messages that are deemed relevant by an expert analyst. We describe three alternative approaches to obtaining labeled datasets and discuss their varying trade-offs in terms of scalability and accuracy of the resulting predictions.

*Data requirements.* As discussed before, ACARS data available through data collection of real-world aircraft communications is not naturally labeled. However, training our detectors requires a labeled dataset. We have two main dataset requirements. First, naturally, the labeled set must include a sufficient number of encrypted messages. Prior analysis of ACARS messages has shown that only a small fraction of sent messages are encrypted [185]. Consequently, satisfying this requirement can be challenging in the ACARS setting as a small random sample of messages might not meet this requirement. Second, the labeled set must cover a wide variety of message types (e.g., originating from various protocols). As we demonstrate in Section 5.5.5, if the detectors’ training set does not include any examples of a specific message type, there is a high risk that messages of this type are misclassified.

*Labeling approaches.* The standard approach to label data at scale, crowd-sourcing, is unfortunately unsuitable for our setting. Labeling ACARS messages requires specialist knowledge about both ACARS communication protocols and (weak) encryption schemes. Thus, we cannot outsource this task to untrained volunteers. The next option is expert labeling. This approach has two problems: (1) labeling messages *correctly* is hard and time-consuming even for experts and (2) it does not scale beyond a few thousand messages. We performed a small study in which we asked a group of ACARS specialists to label a small set of messages. They perceived the task as cumbersome and struggled to produce high-confidence labels. Now, we study approaches to overcome these limitations,

### 5.6.1 Synthetic data generation based on noisy samples

The fraction of ciphertexts in ACARS messages is low [185]. We leverage this observation and combine it with the synthetic ciphertext generation (see Section 5.5.1) to design a scalable approach to obtain a labeled dataset. We first randomly select a large sample of ACARS messages and label all messages as `plain`. Due to the class imbalance, the majority of labels are assumed to

be accurate. The random sample, however, only provides us with (noisily) labeled plaintext messages. We use our experimental setup (see Section 5.5.2) with the default cipher suite to generate `cipher` messages from our plaintexts. This provides us with a dataset similar to the noisy labels which we study in Section 5.5.4.

*Discussion.* This approach scales easily but produces low-quality and noisy labels. Moreover, it relies on synthetic ciphertext generation to obtain `cipher` samples. Hence, the performance of our detectors on flagging real-world encrypted ACARS messages depends on the similarity between the generated ciphertext examples and the ciphers found in ACARS.

### 5.6.2 Manual labeling

We show how to address the lack of ciphertext samples when manually labeling a small set of messages. Next, we show how to bootstrap this small set of high-confidence manual labels to produce high-quality labels at scale.

*Message selection.* The main constraint on manual labeling by experts is its scalability: To collect high confidence labels on ACARS messages requires a substantial effort even by trained experts. The problem with letting experts label a small *random* selection of ACARS messages is that it does not meet our two main requirements: The random sample is unlikely to contain enough examples of the `cipher` class and a variety of message types from different ACARS protocols.

We want to filter out a small set of messages that is likely to contain a sufficient number of ciphertexts. Detectors trained on noisy data may suffer from performance drop, but they suffice to guide out filtering criteria and make a biased message selection. We first collect a large random sample of ACARS messages. Similar to our experiment setup (see Section 5.5.2), we split the data into 5 disjoint subsets then process each subset into two `plain` and `cipher` halves. This makes each subset a noisy training set where some ciphertexts are incorrectly labeled as `plain`. On each of these subsets, we train a `CNN` detector and use it to classify all messages in the remaining 4 subsets. At the end of this process, we have 4 labels on each message. We select messages that are flagged as `cipher` by at least one detector for manual labeling. Using `2C-PPM` detectors instead of `CNN` detectors is possible and may improve the ciphertext recall rate at the cost of having more messages for manual labeling due to the lower precision.

*Manual expert labeling.* We randomly select 50,000 ACARS messages and use cross-validation with our CNN detector to filter out a subset of messages for manual labeling. Following this process, we obtain 1,811 messages marked as **cipher**. One of the authors, who is familiar with aviation protocols, labeled these messages, together with an additional control sample of 100 messages not flagged by any of the 4 detectors and marked as **plain** (1,911 messages in total).

In addition to the free text, our labeling tool provides the labeler with message metadata such as aircraft manufacturer, operator, and ACARS label. This allows experts to use their domain knowledge to make more informed decisions. We asked labelers to indicate their confidence in each label as low, medium, or high. Our expert labels 422 out of 1,911 messages as ciphertexts, 276 of those with high or medium confidence. A total of 738 messages only receive a low confidence label.

The high number of messages that only receive a low confidence label can partly be explained by our biased selection. Our expert mainly observes challenging plaintexts that look similar to ciphertexts. This makes it harder to distinguish these messages from their encrypted counterparts. This is also shown by the fact that all 100 messages not flagged by our CNN included as a control sample are labeled as **plain**, 67% of which with high confidence.

### 5.6.3 Bootstrapping manual labels

The result of the previous step is a set of 1,911 manually labeled messages that contains both plain- and ciphertext examples. This dataset is still too small to fully leverage the capabilities of our CNN and 2C-PPM detectors and the biased message selection is not representative of the entire ACARS corpus. This leads to a high risk of purely text-based detectors producing random labels on any type of plaintext message that is not included in the labeled set

We leverage the ACARS metadata and use our **MetaRF** detector to overcome this limitation. As discussed in Section 5.4.3, the metadata attached to each ACARS message is highly correlated with its likelihood to be encrypted. The **MetaRF** detector uses this relationship to learn which actors are likely to encrypt their communications and which ACARS subprotocols contain a high ratio of ciphertexts. Combined with the text features listed in Table D.1, the **MetaRF** detector, which can be trained on small datasets, provides us with a powerful tool to expand our manual labels.



Table 5.3: An overview of labeling approaches and their suitability for our detectors. HC stands for high-confidence.

Labeling	Scalability	No manual labor	HC plain labels	HC cipher labels	Real ciphertexts	CNN	2C-PPM	MetaRF
Noisy	✓	✓	✗	✓	✗	~	~	✗
Manual expert	✗	✗	✓	✓	✓	✗	✗	✓
Bootstrapped	✓	✗	~	~	✓	✓	✓	✗

We train a **MetaRF** detector on our manually labeled dataset and use it to label a random sample of 150,000 ACARS messages, disjoint from the dataset used to create our manual labels. We obtain 9,246 messages labeled as **cipher** and 140,754 messages as **plain**.

*Discussion.* As shown by our evaluation in the next section, our approach to scaling manual labels by experts to a larger dataset produces high-quality training sets for our detectors. However, compared to synthetic data generation on noisy labels (see Section 5.6.1), it comes at a significant cost in manual labor and might not be applicable to other data use cases outside ACARS. The metadata needed for bootstrapping is not always available or correlated with the target label. Furthermore, the generalization capabilities of our **MetaRF** detector are limited. While **MetaRF** is capable of detecting new cipher types sent by aircraft models and operators known to use proprietary encryption methods, it might fail to flag ciphertexts that originate from new aircraft types or have entirely different text features from training ciphertext.

#### 5.6.4 Conclusion

We present three approaches to gathering a labeled dataset of ACARS messages that provide different trade-offs on scalability, manual labor, and label confidence. Table 5.3 summarizes the properties of our labeling methods and their compatibility with our detectors.

## 5.7 Insecure ciphers in aerospace

Our goal is to detect ciphertexts in ACARS communications and identify actors who use insecure encryption schemes. We assess our detectors' performance in

a real scenario and observe how they compare. We cannot compute traditional performance measures such as accuracy or F1 score since there is no ground truth for real messages. Instead, we design an evaluation criteria based on how experts use detectors to catch bad actors in aerospace. We require all our measures to be *non-intrusive*, i.e., measurements should not cause a high burden on experts when performing the analysis.

We take the manual analysis of ACARS messages by Smith et al. [185] as the baseline and assess how our detectors impact the analysis. We design and evaluate four criteria:

- Does the detector flag messages encrypted under known insecure ciphers (Section 5.7.2)?
- Does the detector enable our human analyst to identify previously unknown insecure ciphers (Section 5.7.3)?
- How much overhead in manual labor does the detector cause (Section 5.7.4)?
- How much effort is needed to effectively use our detectors in a new region or over time (Section 5.7.5)?

### 5.7.1 Setting-up detectors

Building detectors requires 2 parts: a detection method and a training label source. We designed 3 detection methods (PPM, CNN, RF) in Section 5.4 and identified 3 approaches for obtaining labeled training data (Noisy, Manual, Bootstrapped) in Section 5.6. We use Table 5.3 to select 5 detectors:

**SynCNN** uses noisy data labeling with CNN detection.

**SynPPM** uses noisy data labeling with 2C-PPM detection.

**MetaRF** uses manually labeled data with MetaRF detection.

**CNN** uses bootstrapped data, that includes real ciphertexts, with CNN detection.

**2C-PPM** uses bootstrapped data, where real ciphertexts are replaced with synthetic ones, with 2C-PPM detection.

Our detector selection assesses our three detection methods with low and high labeling effort. Since the 2C-PPM detector is sensitive to noisy labels

Table 5.4: Number of insecure ciphers flagged out of 52 ciphertexts manually detected in Smith et al.

	SynCNN	SynPPM	MetaRF	CNN	2C-PPM
# cipher	1	16	51	52	52

but robust to cipher domain shifts, we replaced bootstrapped cipher messages (which may contain false positives) with synthetic cipher messages (which are guaranteed to have the correct label).

Our goal is to confirm the impact of our detectors rather than doing a full-scale study. Hence, we limit the number of studied messages to reduce our analyst’s workload. Of course, looking at more data may lead to finding more insecure ciphers. We use the 150k randomly selected messages from Section 5.6.3 for training and randomly select 50k test messages for analysis. We label these 150k training messages according to our noisy and bootstrapped approaches in Section 5.6 and train our detectors. Note that the **MetaRF** detector is trained on the set of 1,911 manually labeled messages in Section 5.6.2. Finally, we classify the 50k test messages with our 5 detectors.

### 5.7.2 Automated detection of known ciphers

First, we assess whether our automated tools can detect manually identified insecure ciphers. Smith et al. [185] manually crafted a regular expression for detecting a specific class of classic ciphers. We apply this expression to our 50k test messages, which flags 52 messages. Table 5.4 reports the number of known ciphertexts flagged by each detector.

Detectors trained on manual and bootstrapped labeled data have near-perfect recall; **CNN** and **2C-PPM** detectors flag all weak ciphers while **MetaRF** only misses one (98%). In contrast, detectors trained on noisy data only mark 31% (**SynPPM**) and 2% (**SynCNN**) of known ciphertexts.

### 5.7.3 Exploring unknown insecure ciphers in ACARS

To answer “Does our detectors help analysts to detect unknown insecure ciphers?”, we asked an analyst – who is one of the authors and has extensive knowledge of ACARS – to repeat the original experiment (Smith et al. [185]) on a smaller scale with our detectors.

*A decision support system.* We observed the expert’s analysis process and interactively designed a dashboard to support them. The dashboard enables viewing and searching messages. Moreover, the dashboard supports grouping messages by their metadata since experts study messages in groups based on the ACARS label as it approximately separates messages by their functionality and underlying protocol (see Section 5.2). The dashboard further displays statistics on the flagged messages per aircraft manufacturer, model, owner, and operating airline. We further describe our dashboard in Section D.3.

We loaded our dashboard with the 50k test messages and classification labels from the **MetaRF**, **CNN**, and **2C-PPM** detectors. Then, we asked our expert to analyze this data and report insecure ciphers. We only use high-performance detectors in our dashboard to reduce the load on our analyst. We compared labels from **SynCNN** and **SynPPM** to our analyst’s findings to assess their effectiveness at the end.

Whenever our analyst suspects a group of messages to be encrypted under an (insecure) cipher they first identify which actors are most likely responsible for using this cipher and then contact the responsible actor, either directly or through regulating bodies. The number of contacts initiated based on the analysis is a good measure of the effectiveness of our tools. It demonstrates that our detectors flag messages that our specialists deem suspect and worth a follow-up request.

Our analyst identified 9 new potentially insecure ciphers, see Table 5.5. The majority of suspected ciphers have a fixed acars label and are used by a specific manufacturer. Only ciphers #4, #5, and #9 are found across multiple aircraft types. Ciphers #4 and #8 are found across multiple ACARS labels. We *disclosed our findings to aircraft manufacturers* and are following up to gather more information.

Our analyst believes one of the detected cipher groups originates from the same encryption scheme as the original manual study but eluded inspection. A metadata analysis confirms that these messages originate from aircraft using HoneyWell devices, the device type identified as the source of the cipher in the manual study. Thus, beyond detecting new ciphers, our detectors enabled our analyst to identify and alert a new manufacturer who uses encryption known to be insecure.

*Detector performance.* The last five columns of Table 5.5 indicate whether the cipher type would have been identified by our analyst based on the detector’s predictions.

Table 5.5: Suspected (insecure) ciphers in ACARS, and detectors’ performance on flagging all (✓), some (∼), or none (✗) of the cipher’s message groups.

	Labels	# Actors	SynCNN	SynPPM	MetaRF	CNN	2C-PPM
#1	MA	1	✓	✓	✓	✓	✓
#2	H1	1	✗	✓	✗	✗	✓
#3	H1	1	✗	✓	✓	✓	✓
#4	1L, 3L, H1, H2	4	∼	∼	∼	✓	✓
#5	41	2	✗	∼	✓	✓	✓
#6	26	1	✗	✓	✗	✓	✓
#7	H1	1	✗	✓	✓	✓	✓
#8	4D, 5Z	1	✗	✓	✓	✓	✓
#9	42	2	∼	✓	✓	✓	✓

Our purely text-based detectors trained on a bootstrapped dataset outperform the **MetaRF** detector that is using both text- and metadata-based features. The **MetaRF** detector fails to flag 2 out of the 9 ciphers and only partially flags cipher #4. In comparison, the **2C-PPM** detector provides a complete list of findings while the **CNN** detector only misses one interesting group. These results demonstrate the success of text-based detectors in ACARS when supplied with a training set that is representative of the target distribution.

Out of our two detectors trained on noisy data, **SynCNN** misses many of the analyst’s crucial findings. We expect that the low recall of **SynCNN** is due to cipher domain shift (see Section 5.5.6) as CNN detectors generalize poorly across unseen cipher types. Surprisingly, **SynPPM** far exceeds our expectations and even outperforms our **MetaRF** detector.

#### 5.7.4 Analysis overhead

While the number of contacts is indicative of whether a detector is likely to miss crucial findings, it does not take into account how precise the labels are. While some detectors have high coverage, they might produce a significant overhead for the analyst if the total number of flagged messages is high but only a low fraction is actually relevant.

There are two common (direct) approaches for measuring the overhead of detectors: measure the necessary time for an expert to perform the analysis with a detector or ask experts to label messages as interesting/non-interesting to measure the ratio of interesting messages. However, both approaches are intrusive. Measuring the analysis time for each detector adds a huge burden on

Table 5.6: Number of messages labeled as `cipher` from 50k test ACARS messages.

	SynCNN	SynPPM	MetaRF	CNN	2C-PPM
# <code>cipher</code>	300	2278	2909	3193	3015

analysts while reducing the available information to only 1 classification label. Similarly, explicitly labeling all messages is very time-consuming. Therefore, we come up with two indirect measures to estimate the overhead:

*Number of flagged messages.* Analysts check a large portion of flagged messages. Thus, the number of messages labeled as `cipher` is a coarse estimate of the effort needed. There are two drawbacks to this measure: it estimates total work and not overhead, and the time spent on messages varies depending on the underlying protocol and the number of similar messages.

Table 5.6 reports the number of messages labeled as `cipher` by each detector. Except for the `SynCNN` detector, the number of flagged messages does not vary much; `MetaRF`, `CNN`, and `PPM` detectors flag 5.8 – 6.4% of messages while `SynPPM` only flags 4.5%.

*Number of flagged groups.* Experts study flagged messages in groups (formed by metadata) as having multiple messages of the same type/purpose gives a better view of the protocol. We follow the same process to group all flagged messages. We mark a group as interesting if it is marked for further investigation in Section 5.7.3. The number of groups not marked as interesting is a measure of overhead. Fig. 5.10 reports the number of flagged groups for each detector and how many groups were marked as interesting.

The detection method has a higher impact on the overhead than the source of training labels. The `PPM` detectors, `SynPPM` and `2C-PPM`, tend to have the highest number of non-interesting flagged message groups (85 to 87). The `CNN` detectors, `SynCNN` and `CNN`, flag 55-61 non-interesting groups, while the `MetaRF` detector achieves the lowest overhead by only flagging 40 non-interesting groups.

Our measures are a rough estimate of the actual overhead. They give a first impression of the trade-offs our detectors pose. Our experiment suggests that the higher recall rate of `PPM` detectors comes at the price of doubling the overhead.

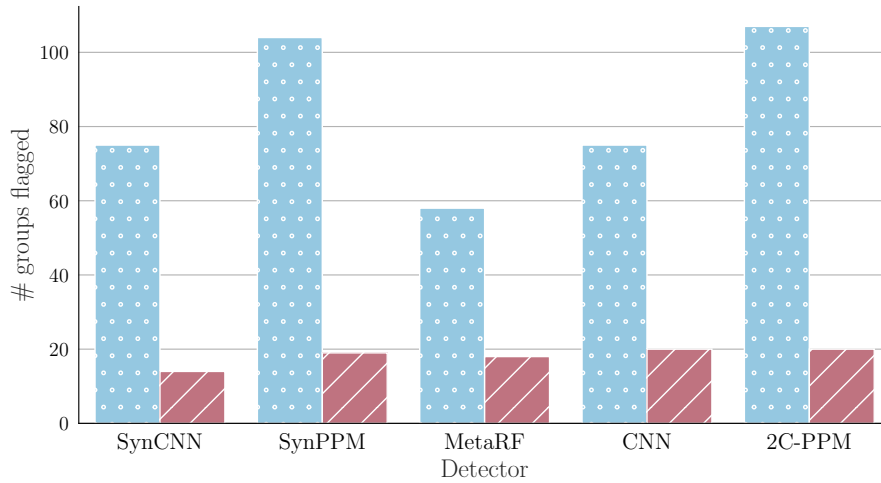


Figure 5.10: Message groups flagged by our five detectors. ■ shows the total number of groups flagged and ▨ the groups marked by analysts as suspicious.

### 5.7.5 Generalization over time and geographic regions

Ensuring the security of aerospace is a continuous task. Regulators keep track of known ciphers to push manufacturers towards secure options. Moreover, aviation is an international industry and different geographical regions may have distinct airlines, aircraft, and sometimes even different protocols. This may lead to plain- and ciphertext distribution shifts. Thus, we assess our detectors’ robustness when faced with new data.

*New dataset.* We used an existing infrastructure [219] to obtain over 2 million ACARS messages from satellite, VHF, and VDL2 sources during one week starting April 6, 2022. We mainly cover North America.

We use our five detectors, without re-training, to classify a random selection of 100k messages from the new dataset. Based on our experiment in Section 5.7.4, we expect roughly 5% of messages to be flagged if there is no change in the situation. **SynCNN** and **MetaRF** only label 303 and 512 messages as **cipher** which is a sign of having a low recall. The PPM detectors label 37% of messages as **cipher** which shows low precision and high overhead. In contrast, **CNN** is robust and labels 5.5% of messages as **cipher** as expected. Inspecting the compression ratio of PPM detectors suggests a plaintext distribution shift. Re-training the **SynPPM** detector brings the **cipher** ratio to the expected range at a low cost.

Finally, we used the **CNN** result to check the persistence of ciphers detected

in Section 5.7.3. We observed that not only weak ciphers from 2017 are still in use in 2022 but that they also have spread to more manufacturers and aircraft.

### 5.7.6 Conclusion

We recommend using both **CNN** and **SynPPM** detectors simultaneously for continuous deployment. The **CNN** detector provides great recall and the highest robustness; it can be used on new datasets without retraining. **2C-PPM** provides the highest recall but is sensitive to plaintext domain shift. Changing the geographical region may require re-training **PPM** detectors. As the **SynPPM** detector has a recall and overhead rate close to the **2C-PPM** detector, the lower labeling effort makes **SynPPM** a better choice for deploying to new datasets. Training the first **MetaRF** detector was necessary to enable bootstrapped labeling and build robust **CNN** detectors. However, the performance/overhead trade-off of **MetaRF** does not justify repeating the manual labeling and retraining a **MetaRF** detector for new datasets.

## 5.8 Discussion and conclusion

We presented new methods for detecting the use of weak ciphers in the ACARS protocol. However, our design and methodology applies to any scenario in which weak ciphers may still be in use. Unfortunately, such scenarios might be more common than security researchers might like. Nautical and satellite communications are also notorious for having been designed before security was common place. Moreover, IoT devices have low communication and computation power and they are infamous for the existence of numerous private proprietary protocols. IoT designers may cut corners and use obsolete ciphers. Similar conditions apply to embedded systems. Our tools can easily be adjusted to detect ciphers in various communication channels.



# CHAPTER 6

## Conclusion

When designing systems, some argue that enabling privacy will be prohibitively expensive and use this excuse to ignore the possibility of building privacy-preserving alternatives. However, In this thesis, we demonstrate that privacy-enhancing technologies can be practical in many real-world scenarios. While there is no silver bullet that can achieve *efficient privacy out of the box*, tailoring cryptographic tools for a specific challenge can drastically improve performance; allowing the system to scale to practical scenarios.

We showed that identifying and bridging the gaps between theoretical cryptographic solutions and real-world problems can improve the practicality of privacy-preserving systems. Although we have not reached a stage where existing systems can seamlessly integrate privacy via a ‘privacy library’, privacy is no longer an out-of-reach property.

In collaboration with the International Consortium of Investigative Journalists (ICIJ), we designed DatashareNetwork a privacy-preserving peer-to-peer document search engine tailored for investigative journalists. We demonstrated that DatashareNetwork can scale up to thousands of journalists and millions of documents; satisfying ICIJ’s current needs and supporting future expansions. While a practical academic design was a crucial step in conceiving DatashareNetwork, the academic proof of concept and publication was merely the beginning. Real-world applicability requires more. We continued our collaboration with ICIJ to help them develop and deploy the system. DatashareNetwork not only brings novel search features to journalists but also fosters collaboration and opens up new avenues At the moment, DatashareNetwork is an open-source software in the beta test stage and soon will enable journalists to tackle upcoming challenges.

The efforts made to ensure that DatashareNetwork satisfies a real-world need and can be deployed as a useful tool for investigative journalists were rec-

ognized by receiving awards such as runner-up for CNIL-Inria data protection award [220] and runner-up for Caspar Bowden award for outstanding research in privacy enhancing technologies [221]. Beyond the original paper, the lessons learned during the DatashareNetwork project were presented at the real-world crypto symposium (RWC 2023) [222]

We introduced a new class of problems called PCM and showed their use in various real-world scenarios such as privately searching chemical compound datasets or collections of documents. Our approach to PCM problems takes the principle of ‘data minimization’ to the maximum and states that ‘if a user’s need can be satisfied with a single bit of information, there is no need for revealing more’. We designed a layered framework that separates concerns such as basic set comparison, matching criteria, and aggregation from each other and allows simple customization of the protocol for various use cases. We developed our framework and showed that it improves latency, client computation, and communication cost with respect to the prior generic systems that offer the same privacy guarantee.

We designed Janus, a privacy-preserving biometric deduplication system to protect against double registration in humanitarian aid distribution. We demonstrate that Janus is compatible with fingerprints, irises, and face recognition; Janus is also capable of fusing multiple biometric samples together enabling it to support the necessary error rate for large-scale deduplication. We develop three instantiations of Janus based on secure multiparty computation, homomorphic encryption, and trusted hardware and demonstrate that they can scale to practical scenarios. Our design may impact the future decisions of the International Committee of Red Cross when integrating protection against double registration in their digital system.

We designed a decision support system called Brutus that aids human analysts to detect the use of insecure ciphers in aircraft communication. We design three methods to automatically detect ciphertexts in the ACARS domain and a new methodology to gather labeled data at scale when the labeling process requires high expertise. Our decision support system led to the detection of 9 (potentially insecure) ciphers used in aviation which we responsibly disclosed to the stakeholders involved. Brutus may be used in the research branch of Armasuisse as a tool to help monitor and improve the security of aircraft communication.

## 6.1 Future work

This thesis introduces new practical scenarios with real-world impacts, such as constraints that investigative journalists and humanitarian organizations face. We hope to see more papers and systems that are compatible with these requirements in the future. Beyond introducing new problems and scenarios, this thesis identifies the following areas to address:

**A methodology for designing private systems.** We observe a common pattern emerging when looking at the process of building DatashareNetwork, PCM framework, and Janus together. Each project includes steps to: (1) Identify a real-world privacy problem. (2) Study users to understand their functional needs and the risks that face. (3) Systematize users' needs and risks into a set of technical requirements and a threat model. (4) Design a solution that satisfies all requirements, (5) Perform a system-wide security and privacy analysis. (6) Develop a prototype and evaluate the performance of the system. Creating a methodology on how to study privacy problems and design privacy-preserving solutions can facilitate the process of building new systems.

**Evaluator privacy in homomorphic encryption.** Homomorphic encryption is a valuable tool that enabled the construction of many new cryptographic primitives including the ones introduced in our PCM paper. HE offers strong formal privacy guarantees for the person holding the private key, but the protection provided to the evaluators is lacking. The notion of circuit privacy was introduced to help with the challenge of keeping the evaluators' data private but it still has shortcomings: circuit privacy requires (1) honest generation of public and evaluation keys, (2) can have a high impact on noise, and (3) only supports the semi-honest setting and fails to protect against malicious adversaries.

**Private tools for journalist.** We studied the constraints that investigative journalists face which are quite different from common assumptions that appear in cryptographic papers. While DatashareNetwork satisfies journalists' need for search, there is still a lot of room for improvement. The performance and scalability of DatashareNetwork come at the price of privacy compromises. It is possible to design and build more private solutions. Our PCM framework shows that it is possible to achieve the optimal privacy of only revealing 1 bit per search. However, the extra privacy of our PCM solution comes with higher

computation and communication costs. We hope that future work can reduce the performance and privacy gap between PCM solutions and DatashareNetwork.

# APPENDIX A

## Appendix for DATASHARENETWORK

### A.1 Security of MS-PSI

In this section, we prove that MS-PSI is correct and private. Proving privacy requires showing that neither a malicious client nor a malicious server can learn anything beyond the intended output of the protocol. The client's interaction with the server is identical to the PSI [25] and C-PSI [16] protocols. Hence, they have the same client privacy against a malicious server. To prove server privacy, we use the ideal/real world paradigm in the random oracle model and show that a malicious MS-PSI client does not learn anything beyond the intended output of the protocol as long as the One-more-Gap-DH assumption holds. We first prove correctness.

**Theorem 7.** The MS-PSI protocol is correct.

*Proof.* We show that the intersection  $I_d$  of the  $d$ 'th set represented as  $Y_d = \{y_{d,1}, y_{d,2}, \dots, y_{d,n_d}\}$  and the client's set  $X = \{x_1, x_2, \dots, x_m\}$  is equal to  $I_d = X \cap Y_d$ .

Recall that the client computes the intersection with set  $Y_d$  as  $I_d = \{x_i \mid \mathsf{T}_i^{(d)} \in \mathsf{TC}\}$ . We prove that  $\mathsf{T}_i^{(d)} \in \mathsf{TC}$  iff  $x_i \in X \cap Y_d$ . For each client keyword  $x_i$ , the client computes the pretag  $\tau_i = \hat{x}_i^{c^{-1}} = \tilde{x}_i^{sc^{-1}} = \hat{H}(x_i)^{csc^{-1}} = \hat{H}(x_i)^s$ . On the other hand, the server computes its pretags for set  $Y_i$  as  $\tau^{(i)} = \{\hat{H}(y)^s \mid y \in Y_i\}$  and computes its tag collection as  $\mathsf{TC} = \{\mathsf{H}(i \parallel t) \mid i \in [N] \wedge t \in \tau^{(i)}\} = \{\mathsf{H}(i \parallel \hat{H}(y)^s) \mid i \in [N] \wedge y \in Y_i\}$ . Hence, the intersection will be computed as

$$\begin{aligned} I_d &= \{x_i \mid \mathsf{T}_i^{(d)} \in \mathsf{TC}\} \\ &= \left\{ x_i \mid \mathsf{H}(d \parallel \tau_i) \in \left\{ \mathsf{H}(i \parallel \hat{H}(y)^s) \mid i \in [N] \wedge y \in Y_i \right\} \right\}. \end{aligned}$$

The hash functions  $\hat{H}$  and  $H$  are cryptographically secure, and the probability of collision is negligible. Hence, two hash values will only be equal when their inputs are equal. Since  $d$  is an input to  $H$ , only the keywords from the  $d$ 'th set in the server's tag collection can be in the intersection. Therefore:

$$I_d = \{x_i \mid \hat{H}(x_i)^s \in \{\hat{H}(y)^s \mid y \in Y_d\}\}.$$

Similarly,  $x_i$  is an input to  $\hat{H}$  and it will be in the intersection  $I_d$  if  $x_i$  is present in both  $X$  and  $Y_d$  sets. Consequently:

$$I_d = \{x_i \mid x_i \in \{y \mid y \in Y_d\}\} = \{x_i \mid x_i \in Y_d\} = X \cap Y_d. \quad \square$$

The MS-PSI protocol is interactive: the client asks keywords in multiple queries and receives the response of the  $i$ 'th query before making the  $i + 1$ 'th one. To measure the client's interaction with the server, we define  $q$  as the number of queried keywords. We chose the number of queried keywords over the number of queries since the server reveals the same information about these  $q$  keywords regardless of how many queries they were asked in. Without loss of generality, we assume an adaptive adversary in which the adversary asks her keywords one by one and receives responses immediately. The non-adaptive versions or versions where the client queries multiple keywords simultaneously only delay when the adversary receives the response. Hence, they have the same security guarantee as the adaptive version.

**An adaptive PSI functionality.** Let  $\lambda$  be an empty string,  $w$  be the client's input keyword, and  $\mathcal{Y} = [Y_1, \dots, Y_n]$  be a list of  $n$  server sets  $Y_i = \{y_{i,1}, \dots, y_{i,n_i}\}$ . We define the adaptive PSI functionality  $\mathbf{PSI}_{\text{adt}}$  as a two party function in which the client learns the sets which contain the keyword  $w$ , and the server learns nothing:

$$\mathbf{PSI}_{\text{adt}}(w, \mathcal{Y}) = (\{i \mid i \in [n] \wedge w \in Y_i\}, \lambda).$$

We define  $\mathbf{Ideal}_q$  as an ideal instantiation of  $\mathbf{PSI}_{\text{adt}}$  in which a trusted third party receives the server's input and responds to the client's  $\mathbf{PSI}_{\text{adt}}$  queries at most  $q$  times.  $\mathbf{Ideal}_q$  provides an oracle  $\mathcal{O}_{\text{ideal}}(\mathcal{Y}, w) \rightarrow \{i \mid i \in [n] \wedge w \in Y_i\}$  which responds to ideal queries. The ideal oracle can answer non-adaptive queries with  $t$  keywords by calling the  $\mathbf{PSI}_{\text{adt}}$  process  $t$  times and concatenating their responses. Note that this operation costs  $t$  adaptive queries.

We define  $\mathbf{Real}_q$  as the real world instantiation of  $\mathbf{PSI}_{\text{adt}}$  which runs the

MS-PSI protocol and allows the client to ask up to  $q$  keywords. The MS-PSI protocol consists of 2 parts: 1) publish, which corresponds to the pre-process phase and 2) exponentiation, which corresponds to the online interaction. The simulation implicitly assumes a known fixed size for the parties' inputs as an adversary can distinguish different input sizes. To make this explicit, we reveal the number of server sets  $N$  and the size of the tag collection  $\mathcal{N} = |\mathbf{TC}| = \sum_{i=1}^N n_i$  to the simulator and the adversary. Bear in mind that the MS-PSI protocol reveals an upper bound on  $N$  and  $\mathcal{N}$ . MS-PSI uses two hash-functions  $\mathbf{H} : \{0, 1\}^* \rightarrow \{0, 1\}^l$  and  $\hat{H} : \{0, 1\}^* \rightarrow \mathbb{G}$ , which are modeled in the random oracle model (ROM) as oracles  $\mathcal{O}_{H_{kw}}$  and  $\mathcal{O}_{H_G}$  respectively. We define the following oracles to represent  $\mathbf{Real}_q$ :

$x \leftarrow \mathcal{O}_{H_G}^{\mathbf{Real}}(w)$  hashes the keyword  $w \in \{0, 1\}^*$  into a uniformly random group element  $x \in_R \mathbb{G}$ .

$TC \leftarrow \mathcal{O}_{Pub}^{\mathbf{Real}}()$  pre-processes the server's input, i.e., chooses the server's secret key  $\alpha$ , and publishes the server's tag collection  $\mathbf{TC} = \{\mathbf{H}(i \parallel \hat{H}(y)^\alpha) \mid i \in [N] \wedge y \in Y_i\}$ .

$x^\alpha \leftarrow \mathcal{O}_{exp}^{\mathbf{Real}}(x)$  takes a group element  $x \in \mathbb{G}$  and returns  $x^\alpha$ . The adversary is limited to making up to  $q$  queries.

$\tau \leftarrow \mathcal{O}_{H_{kw}}^{\mathbf{Real}}(\omega)$  hashes the input  $\omega \in \{0, 1\}^*$  to a random  $l$ -bit tag  $\tau \in \{0, 1\}^l$ .

To show that  $\mathbf{Ideal}_q$  and  $\mathbf{Real}_q$  have the same server privacy guarantee, we assume a PPT adversary  $\mathcal{A}$  that interacts with  $\mathbf{Real}_q$  and design a simulator  $\mathcal{S}$  which given black-box access to  $\mathcal{A}$  extracts the same information from the ideal world  $\mathbf{Ideal}_q$ . In Theorem 3, we proved that the MS-PSI is correct. Since MS-PSI is correct and its output is deterministic, we only have to prove the following computational indistinguishability to show that the server privacy in the real and ideal worlds are equivalent:

$$\mathbf{View}_{\mathbf{Real}_q}^{\mathcal{A}}([w_i]_{i \in [q]}, \mathcal{Y}) \stackrel{c}{\equiv} \mathbf{View}_{\mathbf{Ideal}_q}^{\mathcal{S}^{\mathcal{A}, \mathcal{O}_{\mathbf{Ideal}}(\mathcal{Y}, \cdot)}}([w_i]_{i \in [q]}, \mathcal{Y}).$$

Where  $\mathbf{View}_f^P(X, Y)$  is the view of  $P$  in an execution of  $\mathbf{PSI}_{\text{adt}}$  instantiated with  $f$ ,  $X$  is the client's input, and  $Y$  is the server's input.

We start with a high-level overview of the proof. We build simulator  $\mathcal{S}$  by constructing oracles to represent MS-PSI. Afterward, we show that the adversary cannot distinguish simulator  $\mathcal{S}$  from  $\mathbf{Real}_q$ . Oracles  $\mathcal{O}_{H_G}^{\mathcal{S}}$  and  $\mathcal{O}_{exp}^{\mathcal{S}}$  are similar to their real world counterparts. The oracle  $\mathcal{O}_{Pub}^{\mathbf{Real}}$  follows the pre-process phase of the MS-PSI protocol to compute the server's tag collection

**TC** and produces a set of  $\mathcal{N}$  random  $l$ -bit tags generated by the hash function  $H$ . To mimic this, the oracle  $\mathcal{O}_{Pub}^S$  returns  $\mathcal{N}$  random  $l$ -bit tags. To construct the oracle  $\mathcal{O}_{H_{kw}}^S$ , simulator  $\mathcal{S}$  has to extract the adversary's effective input and query it to the ideal oracle  $\mathcal{O}_{Ideal}$  to respond accordingly (i.e. with one of the  $\mathcal{N}$  random outputs of  $\mathcal{O}_{Pub}^S$  for positive and a uniformly random tag for a negative response). The key idea in building this oracle is that the simulator uses the server's secret  $\alpha$  to decrypt queries  $\omega = d||x^\alpha$  to the oracle  $\mathcal{O}_{H_{kw}}^S$  and extract the input  $x$ . After extracting  $\mathcal{A}$ 's input,  $\mathcal{S}$  queries the ideal oracle and responds accordingly. If  $\mathcal{S}$  makes more than  $q$  queries from  $\mathcal{O}_{Ideal}$ , the simulation fails, and the simulator aborts.

First, we prove in Lemma 1 that as long as the simulator does not abort,  $\mathcal{S}$  is indistinguishable from  $\mathbf{Real}_q$ . Second, we show in Lemma 2 that the probability of abort is negligible. The simulator aborts when an adversary queries  $q + 1$  distinct keywords from  $\mathcal{O}_{H_{kw}}^S$  without querying  $\mathcal{O}_{exp}^S$  more than  $q$  times. Informally, this means that the adversary can compute the exponentiation  $r_i^\alpha$  of  $q + 1$  random values  $r_i$  with only  $q$  queries to the exponentiation oracle, which translates into the One-more-Gap-DH problem. We show that if an adversary  $\mathcal{A}$  exists which has a non-negligible chance of forcing an abort, we can build an adversary  $\mathcal{B}$  that can break the One-more-Gap-DH assumption given black-box access to  $\mathcal{A}$ .

We now give details on how to build simulator  $\mathcal{S}$ :

$x \leftarrow \mathcal{O}_{HG}^S(w)$  responds the same way as  $\mathcal{O}_{HG}^{Real}$ ; stores the mapping between each keyword and its matching element.

$TC \leftarrow \mathcal{O}_{Pub}^S()$  chooses a random key  $\alpha$  and generates the tag collection **TC** as a set of  $\mathcal{N}$  uniformly random  $l$ -bit tags. Recall that the simulator receives  $N$  and  $\mathcal{N}$  as input.

$x^\alpha \leftarrow \mathcal{O}_{exp}^S(x)$  same as  $\mathcal{O}_{exp}^{Real}$ . The adversary is limited to making up to  $q$  queries.

$\tau \leftarrow \mathcal{O}_{H_{kw}}^S(\omega)$  the oracle responds to repeated queries consistently. For a new query  $\omega$ , it proceeds as follows:

1. Parse the input  $\omega$  as " $d||z$ " where  $d \in \mathbb{Z}_N$  and  $z \in \mathbb{G}$ . If this fails, respond with a random  $l$ -bit tag  $\tau$ .
2. Use the secret key  $\alpha$  to compute the adversary's effective input element  $x = z^{\alpha^{-1}}$ .
3. If  $x$  is the result of a query to the  $\mathcal{O}_{HG}^S$ , let  $w$  be the corresponding preimage. Otherwise, return a random  $l$ -bit tag  $\tau$ .



4. If  $w$  has not been queried, query  $w$  from the ideal oracle  $\mathcal{O}_{\text{ideal}}(\mathcal{Y}, \cdot)$  and store the response.
5. If simulator  $\mathcal{S}$  has queried the ideal oracle more than  $q$  times, then abort.
6. Respond with a random unused tag  $\tau \in TC$  if  $d \in \mathcal{O}_{\text{ideal}}(\mathcal{Y}, w)$ . Otherwise, respond with a random  $l$ -bit tag  $\tau$ .

**Lemma 1.** The simulator  $\mathcal{S}$  is indistinguishable from the  $\text{Real}_q$  as long as  $\mathcal{S}$  does not abort.

*Proof.* Oracles  $\mathcal{O}_{exp}^{\text{Real}}$  and  $\mathcal{O}_{exp}^{\mathcal{S}}$  are identical, and it is easy to see that  $\mathcal{O}_{HG}^{\mathcal{S}}$  and  $\mathcal{O}_{Pub}^{\mathcal{S}}$  are indistinguishable from their **Real** counterparts as their output is uniformly random.

The MS-PSI protocol uses  $\mathcal{O}_{Hkw}^{\text{Real}}$  to produce  $l$ -bit tags. Theorem 7 proves the correctness of the MS-PSI protocol and shows that a final tag  $\mathsf{T}^{(d)} \leftarrow \mathcal{O}_{Hkw}^{\text{Real}}(\omega)$  is in the server’s tag collection  $\mathsf{TC}$  if  $\omega = “d \parallel \hat{H}(w)^s”$  and  $w \in Y_d$  where  $s$  is the server’s secret key. Otherwise,  $\mathsf{T}^{(d)}$  is a random tag. Similarly, the oracle  $\mathcal{O}_{Hkw}^{\mathcal{S}}(\omega)$  responds with a tag  $\tau \in \mathsf{TC}$  if  $\omega = “d \parallel \mathcal{O}_{HG}^{\mathcal{S}}(w)^\alpha”$  and  $d \in \mathcal{O}_{\text{ideal}}(\mathcal{Y}, w)$  where  $\alpha$  is the simulator’s secret key. Otherwise, the oracle responds with a random  $l$ -bit tag. As long as the oracle  $\mathcal{O}_{Hkw}^{\mathcal{S}}$  correctly detects the effective input  $(w, d)$  and its status  $w \in Y_d$ , the adversary cannot distinguish the oracles  $\mathcal{O}_{Hkw}^{\mathcal{S}}$  and  $\mathcal{O}_{Hkw}^{\text{Real}}$ . There are two possible cases for an incorrect response: false positives and false negatives. Now, we show that the probability of incorrect response is negligible. Let  $q_G$  and  $q_{kw}$  be the number of queries to oracles  $\mathcal{O}_{HG}^{\mathcal{S}}$  and  $\mathcal{O}_{Hkw}^{\mathcal{S}}$  respectively. A false positive happens when there is a collision between the adversary’s input  $z = x^\alpha$  to the oracle  $\mathcal{O}_{Hkw}^{\mathcal{S}}$  and an unintended keyword queried from  $x \leftarrow \mathcal{O}_{HG}^{\mathcal{S}}(w)$ . This event has a probability of  $q_G \cdot q_{kw} / \text{Ord}(\mathbb{G})$  due to the randomness of  $\mathcal{O}_{HG}^{\mathcal{S}}$ . A false negative happens when  $x \leftarrow \mathcal{O}_{HG}^{\mathcal{S}}(w)$  is not known at the time of the  $\mathcal{O}_{Hkw}^{\mathcal{S}}$  query. The probability of  $\mathcal{O}_{HG}^{\mathcal{S}}(w)$  matching one of previous  $\mathcal{O}_{Hkw}^{\mathcal{S}}(\omega)$  queries is  $q_{kw} / \text{Ord}(\mathbb{G})$ , which limits the false negative probability to  $q_G \cdot q_{kw} / \text{Ord}(\mathbb{G})$ .  $\square$

The simulator  $\mathcal{S}$  aborts when adversary  $\mathcal{A}$  queries  $q + 1$  distinct keywords from  $\mathcal{O}_{Hkw}^{\mathcal{S}}$  while making at most  $q$  queries from  $\mathcal{O}_{exp}^{\mathcal{S}}$ . We assume that  $\mathcal{A}$  triggers an abort with the probability of  $\epsilon$ . We state the One-more-Gap-DH assumption and relate it to  $\epsilon$ .

**The One-more-Gap-DH assumption.** informally states that computing CDH is hard even if the adversary has access to a CDH oracle and the DDH problem

is easy.

The adversary  $\mathcal{A}$  in the One-more-Gap-DH assumption gets access to a CDH oracle  $x^\alpha \leftarrow \mathcal{O}_{CDH}(x)$  with the secret  $\alpha$  and a  $DL_\alpha$  oracle  $1/0 \leftarrow \mathcal{O}_{DL_\alpha}(x, z)$  which determines whether a pair of elements  $x, z \in \mathbb{G}$  has a discrete logarithm equal to the oracle's secret, i.e.,  $\alpha = \log_x(z)$ . The  $DL_\alpha$  oracle is a weaker form of the DDH oracle since  $\mathcal{O}_{DL_\alpha}(x, z) = DDH(h, h^\alpha, x, z)$ .

The One-more-Gap-DH assumption states that the adversary has negligible chance in producing  $q+1$  DH pairs  $(x_i, x_i^\alpha)$  given  $M \gg q$  random challenge elements  $Ch = (c_1, \dots, c_M) \in \mathbb{G}^M$  while making at most  $q$  queries to the CDH oracle  $\mathcal{O}_{CDH}$

$$Pr \left[ \{(x_i, x_i^\alpha) \mid x_i \in Ch\}_{i \in [q+1]} \leftarrow \mathcal{A}^{\mathcal{O}_{CDH}(\cdot), \mathcal{O}_{DL_\alpha}(\cdot, \cdot)}(Ch) \right] < \mu.$$

**Lemma 2.** If the adversary  $\mathcal{A}$  has a non-negligible probability  $\epsilon$  in forcing an abort in the simulator  $\mathcal{S}$ , there exists an adversary  $\mathcal{B}$  which has a non-negligible advantage in solving the One-more-Gap-DH problem given black-box access to  $\mathcal{A}$ .

*Proof.* We start with a sketch of the proof. We construct an adversary  $\mathcal{B}$  that simulates  $\mathcal{S}$  to adversary  $\mathcal{A}$  to solve the One-more-Gap-DH challenge. Simulator  $\mathcal{S}$  has two main functions: computing exponentiations with a secret key in  $\mathcal{O}_{exp}^S$  and finding the matching input element  $x = z^{\alpha^{-1}}$  used to query  $\mathcal{O}_{H_{kw}}^S$ . Adversary  $\mathcal{B}$  programs  $\mathcal{O}_{HG}^B$  to fix input elements to challenge points and uses the CDH oracle  $\mathcal{O}_{CDH}$  to respond to  $\mathcal{O}_{exp}^B$  queries. Finally,  $\mathcal{B}$  uses  $\mathcal{O}_{DL_\alpha}$  to detect which challenge point matches the group element  $z$  in the  $\mathcal{O}_{H_{kw}}^B$  query  $\omega = "d||z"$ . If  $\mathcal{B}$  receives  $q+1$  queries corresponding to distinct keywords  $\{w_i\}_{i \in [q+1]}$  in the oracle  $\mathcal{O}_{H_{kw}}^B$ , then  $\mathcal{B}$  can produce  $q+1$  DH pairs from the challenge set without querying  $\mathcal{O}_{CDH}$  more than  $q$  times.

Concretely, we build the adversary  $\mathcal{B}$  as follows:

$x \leftarrow \mathcal{O}_{HG}^B(w)$  responds with a new challenge element  $x \in Ch$  and stores the mapping between each keyword and its matching group element.

$TC \leftarrow \mathcal{O}_{Pub}^B()$  since  $\mathcal{O}_{CDH}$  has its own secret  $\alpha$ , oracle  $\mathcal{O}_{Pub}^B$  does not choose another secret. The oracle creates  $TC$  in the same manner as  $\mathcal{O}_{Pub}^S$ .

$x^\alpha \leftarrow \mathcal{O}_{exp}^B(x)$  uses  $\mathcal{O}_{CDH}(x)$  to respond to up to  $q$  queries.

$\tau \leftarrow \mathcal{O}_{H_{kw}}^B(\omega)$  is similar to  $\mathcal{O}_{H_{kw}}^S$  and responds to repeated queries consistently. Unlike  $\mathcal{O}_{H_{kw}}^S$ , this oracle does not know the secret  $\alpha$  to decrypt the input

element. Instead, it uses  $\mathcal{O}_{DL_\alpha}$  to check  $z$  against all challenge points  $x \in Ch$  and find the corresponding element  $z = x^\alpha$  where  $1 = \mathcal{O}_{DL_\alpha}(x, z)$ .

1. Parse the input  $\omega$  as “ $d||z$ ” where  $d \in \mathbb{Z}_N$  and  $z \in \mathbb{G}$ . If this fails, respond with a random  $l$ -bit tag  $\tau$ .
2. Find challenge point  $x \in Ch$  where  $1 = \mathcal{O}_{DL_\alpha}(x, z)$ . If no such point exists, respond with a random  $l$ -bit tag  $\tau$ .
3. If  $x$  has been queried from the oracle  $\mathcal{O}_{HG}^\mathcal{B}$ , let  $w$  be the corresponding preimage. Otherwise, respond with a random  $l$ -bit tag  $\tau$ .
4. If  $w$  has not been queried, query  $w$  from the ideal oracle  $\mathcal{O}_{\text{ideal}}(\mathcal{Y}, \cdot)$  and store the response.
5. If  $\mathcal{B}$  has queried  $q + 1$  distinct keywords from the ideal oracle, then abort the simulation and solve the One-more-Gap-DH challenge.
6. Respond with a random unused tag  $\tau \in TC$  if  $d \in \mathcal{O}_{\text{ideal}}(\mathcal{Y}, w)$ . Otherwise, respond with a random  $l$ -bit tag  $\tau$ .

In Lemma 3 (below), we prove that the adversary  $\mathcal{A}$  cannot distinguish the simulator  $\mathcal{S}$  from adversary  $\mathcal{B}$ . Therefore, if  $\mathcal{A}$  has a non-negligible chance  $\epsilon$  in forcing an abort in  $\mathcal{S}$ , then with the probability  $\epsilon$  adversary  $\mathcal{A}$  queries  $q + 1$  distinct keywords from  $\mathcal{O}_{H_{kw}}^\mathcal{B}$  while making at most  $q$  queries from oracle  $\mathcal{O}_{exp}^\mathcal{B}$ . Let  $\{\omega_i\}_{i \in [q+1]}$  be the  $\mathcal{O}_{H_{kw}}^\mathcal{B}$  queries corresponding to the distinct keywords  $\{w_i\}_{i \in [q+1]}$ . By the construction of  $\mathcal{O}_{H_{kw}}^\mathcal{B}$ , we know that  $\left\{ \omega_i = “d_i || z_i” \mid 1 = \mathcal{O}_{DL_\alpha}(x_i, z_i) \wedge x_i = \mathcal{O}_{HG}^\mathcal{B}(w_i) \right\}_{i \in [q+1]}$ . Since the oracle  $\mathcal{O}_{HG}^\mathcal{B}$  responds with fresh challenge points, we know that the  $x_i$ s are unique and belong to the challenge set  $Ch$ . Adversary  $\mathcal{B}$  queries the  $\mathcal{O}_{CDH}$  oracle once per  $\mathcal{O}_{exp}^\mathcal{B}$  query. Since adversary  $\mathcal{A}$  makes less than  $q + 1$  queries from  $\mathcal{O}_{exp}^\mathcal{B}$ ,  $\mathcal{B}$  makes at most  $q$  queries from the CDH oracle  $\mathcal{O}_{CDH}$ . Adversary  $\mathcal{B}$  produces  $q + 1$  DH pairs  $\{(x_i, z_i) \mid z_i = x_i^\alpha \wedge x_i \in Ch\}_{i \in [q+1]}$  with at most  $q$  queries to the CDH oracle  $\mathcal{O}_{CDH}$  and solves the One-more-Gap challenge with probability  $\epsilon$ .  $\square$

**Lemma 3.** The adversary  $\mathcal{B}$  is indistinguishable from simulator  $\mathcal{S}$ .

*Proof.* Oracles  $\mathcal{O}_{exp}^\mathcal{B}$  and  $\mathcal{O}_{Pub}^\mathcal{B}$  are identical to their  $\mathcal{S}$  counterparts  $\mathcal{O}_{exp}^\mathcal{S}$  and  $\mathcal{O}_{Pub}^\mathcal{S}$ . The oracle  $\mathcal{O}_{HG}^\mathcal{B}$  responds with challenge points which are indistinguishable from the uniformly random elements used in  $\mathcal{O}_{HG}^\mathcal{S}$ . Oracles  $\mathcal{O}_{H_{kw}}^\mathcal{B}$  and  $\mathcal{O}_{H_{kw}}^\mathcal{S}$  only differ in how they compute  $x = z^{\alpha^{-1}}$  in the step 2. We split the inputs to the oracle  $\mathcal{O}_{H_{kw}}^\mathcal{B}$  based on their inclusion in the challenge set  $ch$ , and show that oracle  $\mathcal{O}_{H_{kw}}^\mathcal{B}$  is indistinguishable from oracle  $\mathcal{O}_{H_{kw}}^\mathcal{S}$  in both cases. As long as the element  $x$  is chosen from the challenge set, i.e.,  $x \in Ch$ , the pair

$(x, z)$  is unique, and both oracles compute the same effective input  $x$  because  $x = z^{\alpha^{-1}}$  is equivalent to  $1 = \mathcal{O}_{DL\alpha}(x, z)$ . On the other hand, when the element  $z$  is generated from an element  $x = z^{\alpha^{-1}}$  which is not in the challenge set, then oracle  $\mathcal{O}_{H_{kw}}^B$  cannot compute  $x$ . Despite the fact that the oracle cannot compute  $x$ , it can determine that  $x$  is not from the challenge set and consequently not a response from oracle  $\mathcal{O}_{H_G}^B$  as  $x \notin \text{Range}(\mathcal{O}_{H_G}^B) = Ch$ . Both oracles  $\mathcal{O}_{H_{kw}}^B$  and  $\mathcal{O}_{H_{kw}}^S$  respond with a random  $l$ -bit tag when the effective input  $x$  is not a response from oracles  $\mathcal{O}_{H_G}^B$  and  $\mathcal{O}_{H_G}^S$ , respectively. We conclude that oracles  $\mathcal{O}_{H_{kw}}^B$  and  $\mathcal{O}_{H_{kw}}^S$  are indistinguishable.  $\square$

## A.2 The limits of document search

We show that even with ideal searches an adversary can recover documents or even extract the whole corpus. We formalize the extraction problem as follows: an adversary receives a list of  $n$  keywords  $U = \{a_1, \dots, a_n\}$  and a search oracle  $\mathcal{O}$  which respond to queries using the server's set of  $N$  documents  $\mathbf{Docs} = \{d_1, \dots, d_N\}$ . The adversary's goal is recovering the document set  $\mathbf{Docs}$ . Since the adversary is only interested in the set  $U$  of keywords, we ignore any keyword outside of this set in our analysis.

### A.2.1 One-bit search extraction

In this section, we consider a 1-bit search oracle  $\mathcal{O}$  which returns a boolean answer for each query which determine whether at least one matching document exists. The oracle supports one operation, **query**, which takes a set of keywords  $P$  as input and returns boolean answer  $0/1 \leftarrow \mathcal{O}.\text{query}(P)$ .

A set of keywords  $R$  represent a document if and only if this set returns a positive search result  $\mathcal{O}.\text{query}(R) = 1$  and adding any other keyword to this set  $R$  results in a negative response  $\forall x \in U, x \notin R : \mathcal{O}.\text{query}(R \cup \{x\}) = 0$ . It is easy to see that a document  $D = \{d_1, \dots, d_m\}$  is represented by  $R = D \cap U$ .

Recall from Section 2.5.3 that uniqueness number  $u_D$  is the smallest number of keywords that uniquely identify a document  $D$ . Moreover, when a document  $D_x$  is included in a larger document  $D_y$ , i.e.  $D_x \subset D_y$ , then its uniqueness number  $u_{D_x}$  is  $\infty$ , and document  $D_x$  cannot be detected. We have discussed that such documents do not have a high impact as they are overshadowed by the larger document. In this section, we assume that all documents

---

**Algorithm 9** Recover the rest of the document given a keyword set  $\{a_k, \dots, a_n\}$ .

Start : RECOVERDOCUMENT( $P$ )

---

```

function RECOVERDOCUMENT( $P$ )
  for  $i \leftarrow k \dots n$  do
    if  $\mathcal{O}.\text{query}(P \cup \{a_i\}) = 1$  then
       $P \leftarrow P \cup \{a_i\}$ 
  return  $P$ 

```

---

have a finite uniqueness number and only recover documents with uniqueness number  $u_D$  less than the uniqueness limit  $u_{\text{lim}}$ .

*Document recovery.* We assume an adversary who has partial knowledge  $P$  about a document  $D$  that is represented by  $m$  keywords. If the adversary wants to recover the rest of document  $D$ , then she needs to ask at least  $t = n - m$  and at most  $n$  queries from the oracle which leads to a  $\Theta(n)$  query complexity. It is important to note that if there is more than one document that contains  $P$ , then recovering *any* of these documents counts as document recovery.

We claim that the adversary needs to ask at least one query for each keyword which is not in the document, i.e., that it must make at least  $t = n - m$  queries. We assume to the contrary that the adversary recovers the document with less than  $t$  queries and then show that there are two possibilities for  $D$  that the adversary cannot distinguish. Since the number of queries is smaller than  $t$ , based on the pigeonhole principle a keyword  $x$  exist which has never been queried without another keyword  $y \notin D$  present in the query. We claim that the adversary cannot distinguish  $D$  from the document  $D \cup \{x\}$  as the oracle's responses to all queries will be consistent for both documents. The queries which do not include  $x$  are not impacted by the inclusion of  $x$  in the document, and queries that include  $x$  include a keyword  $y \notin D$  which ensures a negative answer for both  $D$  and  $D \cup \{x\}$ . Hence, the adversary cannot distinguish  $D$  from  $D \cup \{x\}$  and needs to make at least  $t$  queries. Clearly,  $n$  queries suffice; showing the result.

Algorithm 9 recovers a document with  $n$  queries. Without loss of generality, we re-index the keywords to represent the adversary's known set of keywords as  $P = \{a_1, \dots, a_{k-1}\}$ . The algorithm extends this set with the remaining keywords  $\{a_k, \dots, a_n\}$  as long as the oracle keeps returning 1. Eventually, the algorithm returns a maximal extension of the initial set  $P$ .

*Corpus extraction.* Having a set  $P$ , extracting *one* plausible document is straightforward. However, extracting *all* documents that contain  $P$  is more complex. The reason behind this complexity is that when the adversary adds

a keyword  $a_x$  to the set  $P$  and receives a positive query response, she knows a document  $D$  exists such that  $(P \cup \{a_x\}) \subseteq D$  but cannot determine whether any document  $D'$  exists such that  $P \subset D'$  and  $a_x \notin D'$ . Hence, the adversary needs to expand both cases.

We designed a corpus extraction algorithm that takes care of this uncertainty, see Algorithm 10. This recursive algorithm is called with a set of sets representing the documents  $D$ , the set of keywords  $P$  that the algorithm is considering at this moment, and the index  $k$  into the list of keywords (the keywords with index less than  $k$  have already been considered). To find all documents with respect to the list of keywords  $S = \{a_1, \dots, a_n\}$ , call `EXTRACT( $\emptyset, \emptyset, 1$ )`.

The algorithm is recursive. It considers the current set of keywords  $P$  and tries to extend it with a keyword  $a_i$  ( $k \leq i < n$ ). If the oracle returns 0, clearly there is no document matching  $P \cup \{a_i\}$ . If the oracle returns 1, we cannot distinguish the two cases above, so we recurse along both paths, one for documents that contain  $a_i$ , and the other for documents that do not contain  $a_i$ . When the algorithm finds the `ulim`'th keyword in the set  $P$ , the algorithm can uniquely identify the document and checks whether this document has been extracted before (by calling `IsInDocs`) to prevent duplicates. After reaching a partial set of at least `ulim` keywords, the algorithm only traverses the branch which includes  $a_i$  as only one document exists which contains the set  $P$  since  $|P| \geq \text{ulim}$ . When pursuing only one branch, the algorithm is similar to the `RecoverDocument` function in Algorithm 9. If the algorithm exhausts all possible keywords without branching, it has found a document and after checking for duplicates, the algorithm adds  $P$  as a new document to the current set of documents  $D$  and returns.

We argue that this algorithm finds all documents with uniqueness number  $u_D < \text{ulim}$ . Clearly, the algorithm explores all sets  $P$  of size less than `ulim` for which there exist matching documents. So, eventually, the algorithm will find the unique set for each document, which it will then extend to the corresponding full document.

It is easy to see that the brute-force part, when  $|P| < \text{ulim}$ , requires at most  $\mathcal{O}(n^{\text{ulim}})$  queries. However, the algorithm does not expand keyword sets with negative responses, and on average, document sparsity leads to a significantly lower number of queries. Once  $|P| \geq \text{ulim}$  the algorithm enters a linear exploration, as it stops branching. It runs through this linear phase exactly once for each document. Resulting in a total complexity of  $\mathcal{O}(n^{\text{ulim}} + nd)$ .

---

**Algorithm 10** Extract non-contained documents with an uniqueness number  $u_D$  smaller than  $u_{lim}$  with a one-bit search oracle based on the keyword set  $S = \{a_1, \dots, a_n\}$ .

Start:  $\text{EXTRACT}(\emptyset, \emptyset, 1)$

---

```

function  $\text{EXTRACT}(D, P, k)$ 
  if  $|P| \geq u_{lim}$  then
    if  $\text{ISINDOCS}(P, D) = 1$  then
      return  $D$ 
    for  $i \leftarrow k \dots n$  do
      if  $\mathcal{O}.\text{query}(P \cup \{a_i\}) = 1$  then
         $D \leftarrow \text{EXTRACT}(D, P \cup \{a_i\}, i + 1)$ 
        if  $|P| < u_{lim}$  then
           $D \leftarrow \text{EXTRACT}(D, P, i + 1)$ 
        return  $D$ 
    if  $\text{ISINDOCS}(P, D) = 0$  then
       $D \leftarrow D \cup \{P\}$ 
    return  $D$ 

function  $\text{ISINDOCS}(P, D)$ 
  for all  $d \in D$  do
    if  $P \subseteq d$  then
      return 1
  return 0

```

$\triangleright$  The document is uniquely identifiable.  
 $\triangleright P$  is already extracted.  
 $\triangleright$  No more extension possible.

---

### A.2.2 #doc search extraction

In this section, we consider a #doc search oracle  $\mathcal{O}$  which returns the number of matching documents for each query. The oracle only supports one operation, **query**, which takes a set of keywords  $P$  as input and returns the number of matching documents  $t \leftarrow \mathcal{O}.\text{query}(P)$ .

*Document recover.* Since the 1-bit search oracle's output can be computed from the #doc oracle, the algorithms from the previous section also work against the #doc search oracle. As a matter of fact, when only considering a single document, the behavior of the #doc oracle is equivalent to that of the 1-bit search oracle, thus **RECOVERDOCUMENT** in Algorithm 9 is also optimal for the #doc system in recovering documents.

*Corpus extraction.* The extra information provided by the #doc oracle, however, helps create a much more efficient corpus extraction function. In particular, an attacker is no longer faced with the uncertainty caused by the one-bit oracle. Given an existing set of keywords  $P$ , the attacker can query  $P \cup \{a_x\}$  and see if the number of matching documents changes, or not. If the number of matching documents changes, there were documents that match  $P$  but not  $P \cup \{a_x\}$ . If the number of matching documents stays the same, all documents that match  $P$  also match  $P \cup \{a_x\}$ .

**Algorithm 11** Extract all `#matches` documents which include the partial document  $P$ , with a `#doc` search oracle based on the keyword set  $S = \{a_1, \dots, a_n\}$ .

Start : `EXTRACT( $\emptyset, \emptyset, 1, \infty$ )`

---

```
function EXTRACT( $D, P, k, matches$ )
  for  $i \leftarrow k, n$  do
     $next = \mathcal{O}.query(P \cup \{a_i\})$ 
    if  $next > 0$  then
       $D \leftarrow EXTRACT(D, P \cup \{a_i\}, i + 1, next)$ 
      if  $matches > next$  then  $\triangleright$  At least one doc did not contain  $a_i$ 
         $D \leftarrow EXTRACT(D, P, i + 1, matches - next)$ 
    return  $D$ 
return  $D \cup \{P\}$ 
```

---

Algorithm 11 exploits this principle. It keeps track of the current set of documents represented as  $D$ , the set of keywords  $P$  that it is currently considering, the index  $k$  into the list of keywords (the keywords with index less than  $k$  have already been considered), and the number `matches` of documents that contain the current set of keywords  $P$ . To find all documents with respect to the set of keywords  $S = \{a_1, \dots, a_n\}$ , call `EXTRACT( $\emptyset, \emptyset, 1, \infty$ )`.

Given the current set  $P$  with `matches` matching documents it proceeds as follows. It asks the next keyword  $a_i$ , if there are still matching documents (i.e., `next > 0`) it adds  $a_i$  to  $P$  and continues exploring. If some documents matched  $P$  but did not match  $P \cup a_i$  (i.e., `matches > next`), the algorithm also continues exploring by skipping the keyword  $a_i$ .

In the beginning, the algorithm starts with an empty set and checks every keyword. This requires  $n$  queries, and the algorithm continues with a deterministic document recovery for  $d$  documents. Therefore, this algorithm requires a total of  $\mathcal{O}(nd)$  queries for extracting the corpus.



# APPENDIX B

## Appendix for Private Collection Matching Protocols

### B.1 Extra material

We provide extra materials in this section.

**Summary.** Table B.1 summarizes our notation and asymptotic parameters and Table B.2 summarizes the functionality of our protocols.

**Tversky similarity.** Algorithm 12 shows how to process rational Tversky parameters to enable computing similarity with modular arithmetic.

**BFV parameters.** We report full details of our BFV parameters including their supported multiplicative depth in Table B.3. Next we provide a microbenchmark of basic operations and key sizes in Table B.4. It is possible

---

**Algorithm 12** Process Tversky parameters  $t, \alpha$ , and  $\beta$  to compute integer coefficients  $(a, b, c)$ .

---

```
function TVERSKY-PARAM-PROCESS( $\alpha, \beta, t$ )
  ( $\alpha_1, \alpha_2$ )  $\leftarrow$  ToRational( $\alpha$ )
  ( $\beta_1, \beta_2$ )  $\leftarrow$  ToRational( $\beta$ )
  ( $t_1, t_2$ )  $\leftarrow$  ToRational( $t$ )
   $l \leftarrow$  LCM( $t_1, \alpha_2, \beta_2$ )
   $a \leftarrow l \cdot (t^{-1} - 1 + \alpha + \beta)$ 
   $b \leftarrow l \cdot \alpha$ 
   $c \leftarrow l \cdot \beta$ 
   $g \leftarrow$  GCD( $a, b, c$ )
  return ( $a/g, b/g, c/g$ )
```

---

Table B.1: Notation.

$q, \mathbb{Z}_q, \mathbb{Z}_q^*$	A prime number, a prime ring, and a prime field.
$n$	The security parameter.
$a \leftarrow^{\$} A$	Draw $a$ uniformly at random from the set $A$ .
$[n], \langle a_i \rangle$	Present the set $\{1, \dots, n\}$ and the list $[a_1, \dots, a_m]$ .
$\mathbb{1}[E]$	Function that returns ‘1’ when $E$ is true and ‘0’ otherwise.
<b>HE</b>	An IND-CPA circuit-private homomorphic scheme.
$pk, sk$	The client’s public and private HE keys.
$\llbracket a \rrbracket$	An encryption of $a$ .
$N_{deg}$	The degree of the RLWE polynomial.
$m_{pt}, m_{ct}$	The plaintext and ciphertext modulus of the HE scheme.
$X, \mathcal{Y}, Y_i$	The client’s set, server’s collection, and server’s $i$ ’th set.
$n_c, n_s^i$	The size of client set $ X $ and server’s $i$ ’th set $ Y_i $ .
$N, N_s$	The number of server sets and their total size $N_s = \sum_i n_s^i$ .
$D$	The set input domain.
$T$	The matching threshold $[t_{min}, t_{max}]$ .
$\llbracket z_i \rrbracket$	The client’s encrypted bit-vector $z_i \leftarrow (d_i \in X)$ .
$Q$	The client’s query. Small input: $\langle \llbracket x_i \rrbracket \rangle$ , domain: $\langle \llbracket z_i \rrbracket \rangle$ .
$\llbracket s_i \rrbracket$	An encrypted status determining iff $x_i \in Y_k$ .
$\llbracket \mathbf{ca} \rrbracket$	An encrypted cardinality of intersection $\mathbf{ca} =  X \cap Y_k $ .
$\llbracket \gamma_k \rrbracket$	A matching response. $\lambda = 0$ iff the set $Y_k$ is interesting.
$\llbracket A \rrbracket$	An aggregated collection-wide response.
$\llbracket R \rrbracket$	A term to randomize the output of malicious users.

to reduce the size of the rotation keys in exchange for more costly rotation operations. We report key sizes that provide a balanced computation/communication trade-off.

## B.2 Sum of random $\mathbb{Z}_q^*$ elements

In Section 3.7, we argued that the distribution of the sum  $s = \sum_{i=1}^k x_i$  of  $k$  random  $x_i \leftarrow^{\$} \mathbb{Z}_q^*$  elements is close to uniform when  $q$  is prime. Now we prove that the probability of the sum being zero is bounded by  $1/(q-1)$  and the difference between the probability of the sum being zero vs a non-zero value  $a$  is at most  $1/(q-1)^2$  when  $k$  is larger than one.

Let  $z^{[k]}$  be the probability that the of sum of  $k$  elements from  $\mathbb{Z}_q^*$  is zero and  $p_a^{[k]}$  be the probability that the of sum is a non-zero value  $a$ . When  $k = 1$ , e.g., we sum one element, these probabilities are  $z^{[1]} = 0$  and  $p_a^{[1]} = 1/(q-1)$ . The  $k$  elements  $x_i$  are independent from each other so we choose the value of the

Table B.2: Summary of our protocols. We show the computed functionalities, their output range, and auxiliary input variables in the table.

Protocol	Function	Range	Aux.
PSI	$X \cap Y$	$\{0, 1\}^{n_c}$	
PSI-CA	$ X \cap Y $	$\mathbb{Z}$	
Matching: $\lambda \leftarrow f_M(X, Y)$			
F-Match	$X \subseteq Y$	$\{0, 1\}$	
Th-Match	$ X \cap Y  \geq t$	$\{0, 1\}$	$t$
Tv-Match	$\text{Tv}_{\alpha, \beta}(X, Y) \geq t$	$\{0, 1\}$	$t, \alpha, \beta$
Aggregation: $A \leftarrow f_A(\lambda_1, \dots, \lambda_N)$			
NA-Agg	$\langle \lambda_i \rangle$	$\{0, 1\}^N$	
X-Agg	$\exists i \mid \lambda_i = 1$	$\{0, 1\}$	
CA-Agg	$ \{i \mid \lambda_i = 1\} $	$\mathbb{Z}$	
Ret-Agg	$D_j \mid \lambda_j = 1 \wedge  \{i \mid \lambda_i = 1 \wedge i \in [j]\}  = \kappa$	$\mathbb{Z}$	$D, \kappa$

Table B.3: BFV parameters with 128-bit security

	$N_{deg}$	$m_{pt}$	$\lg(m_{ct})$	Mult. depth
$P_{8k}$	8192	4079617	218-bit	2
$P_{16k}$	16384	163841	438-bit	7
$P_{32k}$	32768	786433	880-bit	16

last element and recursively compute the probability by using the distribution of  $k - 1$  elements.

$$\begin{aligned}
 z^{[k]} &= \sum_{a=1}^{q-1} p_a^{[k-1]} / (q-1) = p_a^{[k-1]} \\
 p_a^{[k]} &= z^{[k-1]} / (q-1) + \sum_{i \in \mathbb{Z}_q^* - \{a\}} p_{a-i}^{[k-1]} / (q-1) \\
 &= z^{[k-1]} / (q-1) + (q-2) \cdot p_a^{[k-1]} / (q-1) \\
 &= p_a^{[k-1]} + (z^{[k-1]} - p_a^{[k-1]}) / (q-1)
 \end{aligned}$$

The probability gap of  $z^{[k]}$  and  $p_a^{[k]}$  gets narrower as the number of elements  $k$  increases as:

$$\begin{aligned}
 p_a^{[k]} - z^{[k]} &= p_a^{[k-1]} + (z^{[k-1]} - p_a^{[k-1]}) / (q-1) - p_a^{[k-1]} \\
 &= (z^{[k-1]} - p_a^{[k-1]}) / (q-1)
 \end{aligned}$$

Table B.4: Cost of basic BFV operations.

	$P_{8k}$	$P_{16k}$	$P_{32k}$
Addition ( $\mu\text{s}$ )	29	115	530
Multiplication (ms)	7.3	36.3	182
Plaintext mult. (ms)	0.97	4.13	18.3
Rotation by 1 (ms)	2.16	10.8	57
Ciphertext (KB)	384	1536	6144
Public key (KB)	512	2048	7680
Relinearization key (MB)	3	12	60
Rotation key (MB)	22	96	510

The highest probability of a zero sum happens when summing two random elements and the probability is bound by  $z^{[2]} = p_a^{[1]} = 1/(q-1)$ . Similarly the highest probability difference happens when  $k = 2$  and is bound by  $z^{[2]} - p_a^{[2]} = (p_a^{[1]} - z^{[1]})/(q-1) = 1/(q-1)^2$ . This means that the probability of false-positive in our approach is bounded by  $1/(q-1)$  while the probability of distinguishing the number of non-zero elements (when at least one non-zero element is present) is  $1/(q-1)^2$ .

### B.3 Privacy proof

We prove the security and privacy of our framework. We throughout assume that honest users do not have any interaction outside our framework that can leak information.

**Roadmap.** We start by formally defining security properties of HE schemes such as IND-CPA security, circuit privacy, and strong input privacy in Section B.3.1. Next, we address our framework's security in a semi-honest setting and use real-world/ideal-world simulation to prove Theorem 5 in Section B.3.2. We study malicious servers in Section B.3.3 and provide a tight reduction from our client privacy to the semantic security of our HE scheme to prove Theorem 6. Finally, we extend the notion of real-world/ideal-world simulation to a new paradigm called cipher-world, which provides better support for simulating HE protocols in a malicious setting. We use this new notion to prove the server privacy of our scheme against malicious clients in a non-standard manner in Section B.3.4.

### B.3.1 Security properties of HE schemes

We formally define the security properties of HE schemes in this section. We start with semantic security (IND-CPA). Next, we discuss noise in HE schemes and define circuit privacy, which ensures that the noise contained in ciphertexts does not leak information about the computation performed on them. Finally, we extend circuit privacy, which only applies in a semi-honest setting, to a malicious setting and introduce strong input privacy.

**Definition 8** (IND-CPA). An encryption scheme is indistinguishable against chosen plaintext attacks if no PPT adversary  $\mathcal{A}$  exists such that:

$$\Pr \left[ \begin{array}{l} p \leftarrow \text{HE.ParamGen}(1^\ell) \\ (pk, sk) \leftarrow \text{HE.KeyGen}(p) \\ (\text{st}, m_0, m_1) \leftarrow \mathcal{A}(pk) \\ b \leftarrow^{\$} \{0, 1\} \\ c_b \leftarrow \text{HE.Enc}(pk, m_b) \\ b' \leftarrow \mathcal{A}(\text{st}, c_b) \end{array} : b = b' \right] > \frac{1}{2} + \epsilon$$

Each BFV ciphertext  $c$  contains noise. The amount of noise increases with each operation and can be measured. As a result, a ciphertext  $c$  contains more information than its decrypted value  $p \leftarrow \text{HE.Dec}(sk, c)$ . If the noise grows beyond the HE parameter's noise budget, then the decryption fails  $\perp \leftarrow \text{HE.Dec}(sk, c)$ . Informally, circuit privacy states that the ciphertext  $c$  should not reveal any information about the computation performed on  $c$  beyond the decrypted result  $p$ . We follow the definition of Castro et al. [223]:

**Definition 9** (Circuit privacy). Let HE be a leveled homomorphic encryption scheme and let

$$\begin{aligned} \text{params} &\leftarrow \text{HE.ParamGen}(q) \\ (sk, pk) &\leftarrow \text{HE.KeyGen}(\text{params}) \\ c_i &\leftarrow \text{HE.Enc}(sk, m_i) \\ M &\leftarrow f(m_1, \dots, m_n, p_1, \dots, p_k) \end{aligned}$$

be an (honestly) generated key pair, ciphertexts, and output of the computation. The scheme HE is  $\epsilon$ -circuit private if a PPT simulator  $\mathcal{S}$  exists such that for all functions  $f$  of depth  $l \leq L$  all PPT distinguisher algorithms  $\mathcal{D}$  are

bounded by

$$\left| \Pr \left[ \mathcal{D}(\text{HE.Eval}(pk, f, \langle c_i \rangle, \langle p_i \rangle), sk, pk, \langle c_i \rangle) = 1 \right] - \Pr \left[ \mathcal{D}(\mathcal{S}(sk, pk, M), sk, pk, \langle c_i \rangle) = 1 \right] \right| \leq \epsilon.$$

The circuit privacy's definition focuses on the semi-honest setting and requires honest generation of HE keys and ciphertexts. We extend this definition by (1) removing the honest generation requirement, which extends the property to the malicious setting, and (2) relaxing privacy by revealing the functionality  $f$  and only hiding the evaluator's private data  $\langle p_i \rangle$ .

**Definition 10** (Strong input privacy). A leveled homomorphic encryption scheme HE is  $\epsilon$ -strong input private if a simulator  $\mathcal{S}_{sip}$  exists such that all PPT adversaries  $\mathcal{A}$  are bounded by:

$$\Pr \left[ \begin{array}{l} (\mathbf{st}, pk, sk) \leftarrow \mathcal{A}(1^\ell) \\ (\mathbf{st}, f, \langle c_i \rangle, \langle p_i \rangle) \leftarrow \mathcal{A}(\mathbf{st}) \\ b \leftarrow_{\mathcal{S}} \{0, 1\} \\ m_i \leftarrow \text{HE.Dec}(sk, c_i) \\ M \leftarrow f(\langle m_i \rangle, \langle p_i \rangle) \\ a_0 \leftarrow \text{HE.Eval}(pk, f, \langle c_i \rangle, \langle p_i \rangle) \\ a_1 \leftarrow \mathcal{S}_{sip}(sk, pk, f, \langle c_i \rangle, M) \\ b' \leftarrow \mathcal{A}(\mathbf{st}, a_b) \end{array} : b = b' \right] \leq \frac{1}{2} + \epsilon$$

Similar to the circuit privacy definition,  $f$  is the computed functionality,  $\langle c_i \rangle$  is the client's encrypted input, and  $\langle p_i \rangle$  is the server's plaintext input while  $\mathbf{st}$  is an state for the adversary for storing information between interactions.

**Ciphertext indistinguishability.** When simulating our protocols, in the next section, the simulator generates ciphertexts. As part of our proof, we need to show that these ciphertexts are indistinguishable from the output of our protocols. Hence, we discuss when a distinguisher  $\mathcal{D}$  can distinguish two ciphertexts  $c$  and  $c'$  in three settings:

*Known public key.* We first consider the case where the distinguisher only knows the public key ( $pk$ ). This scenario directly follows from the IND-CPA property. As long as the HE scheme is *IND-CPA secure* and ciphertexts have the same size  $|c| = |c'|$ , then the distinguisher  $\mathcal{D}(pk, c, c')$  has a negligible chance.

*Semi-honest with known secret key.* Next, we consider the case where the distinguisher knows both the secret and public keys  $(sk, pk)$  in the semi-honest setting, i.e., keys and ciphertexts are honestly generated. In this scenario, the distinguisher can decrypt ciphertexts so IND-CPA is not enough. Decrypting a ciphertext  $p \leftarrow \text{HE.Dec}(sk, c)$  results in a plaintext  $p$  and a measurable noise  $\epsilon$ . This transforms ciphertext indistinguishability to showing two statements: both decrypted plaintexts and ciphertext noises are indistinguishable. While comparing decrypted values  $p$  is straightforward, we rely on circuit privacy to ensure noises are indistinguishable. As long as the HE scheme is *IND-CPA secure* and *circuit-private*, and  $p \leftarrow \text{HE.Dec}(sk, c) \wedge p' \leftarrow \text{HE.Dec}(sk, c') \wedge p \stackrel{c}{\equiv} p'$ , then the distinguisher  $\mathcal{D}(sk, pk, c, c')$  has a negligible chance in a semi-honest setting.

*Malicious with known secret key.* The distinguisher knows both the secret and public keys  $(sk, pk)$  in a malicious setting. We require our scheme to be *IND-CPA secure* and *strongly input private* in this setting and use the simulator of strong input privacy ( $\mathcal{S}_{sip}$ ) to produce indistinguishable ciphertexts.

**Lack of circuit privacy.** The Lattigo library does not provide circuit privacy. This is not surprising since other popular HE libraries such as Microsoft Seal [224] do not provide circuit privacy either. While there are possible mitigations in the semi-honest setting such as noise-flooding [225] (alternatively called noise smudging) to achieve circuit privacy, there is no known mechanism for the malicious setting (i.e., no HE scheme achieves strong input privacy). Noise-flooding is only proven private in a semi-honest scenario where keys are generated honestly and the computation is guaranteed to start on freshly encrypted ciphertexts. Beyond flooding, there is a new line of work [223] that uses careful parameter selection in RNS or DCRT representation of ciphertexts to achieve lightweight circuit privacy. We hope this approach will be adopted by HE libraries.

Our implementation does not add extra defense mechanisms to prevent possible leakage from noise, due to the extra cost associated with these defenses. Despite this leakage, practical attacks using noise are limited. The complexity and depth of our functions make extracting information from this noise more challenging; especially since the size of the server’s private data,  $N_s$ , is significantly larger than the capacity of the noise for storing information.

### B.3.2 Semi-honest security

We use real-world/ideal-world simulation to prove the security of our PCM protocols in the semi-honest setting (see Theorem 5). Our well-formedness checks, computed using (SD-)QUERY-CHECK, have no impact on honest execution as they only add  $\llbracket 0 \rrbracket$  to the result when the query is generated honestly. Therefore, we ignore these functions in this section. Our framework can be instantiated to support different functionalities, but they share a similar structure, which enables us to write a single proof that is customizable depending on the protocol variation. As seen in Definition 4, PCM is a two-party interaction that computes

$$(f_c(X, \mathcal{Y}), \perp) \leftarrow PCM_{f_M, f_A}(X, \mathcal{Y})$$

where  $f_c(X, \mathcal{Y}) = f_A(f_M(X, Y_1), \dots, f_M(X, Y_N))$  is a *deterministic* function selected from the Table B.2. To reason about the properties the matching layer, we allow  $f_A$  to be equal to the identity function. To reason about the PSI layer, we allow  $f_c$  to have the natural PSI and PSI-CA definition.

Our semi-honest scenario has deterministic output functions. Therefore, we can use the simpler formulation of security in Lindell [226] which requires schemes to satisfy two properties to be secure. *Correctness*: the output of parties is correct. *Privacy*: the view of parties can be separately simulated as follows:

$$\begin{aligned} \{\mathcal{S}_C(1^n, X, f_c(X, \mathcal{Y}))\}_{X, \mathcal{Y}, n} &\stackrel{c}{\equiv} \{\text{View}_{\text{client}}(n, X, \mathcal{Y})\}_{X, \mathcal{Y}, n}, \\ \{\mathcal{S}_S(1^n, \mathcal{Y}, \perp)\}_{X, \mathcal{Y}, n} &\stackrel{c}{\equiv} \{\text{View}_{\text{server}}(n, X, \mathcal{Y})\}_{X, \mathcal{Y}, n} \end{aligned}$$

where  $n$  is the security parameter,  $X$  is the client input, and  $\mathcal{Y}$  is the server input. We omit  $n$  in the rest of this section. We assume that the client honestly generates the key pair  $(pk, sk) \leftarrow \text{HE.KeyGen}(\text{params})$  and sends the public key  $pk$  to the server before running the protocol.

**Correctness.** In a semi-honest scenario where both parties follow the protocol specification, showing that Algorithms 3 to 6 compute the functionality described in Table B.2 is straightforward math. We have described the intuition behind these algorithms in Sections 3.6 to 3.8, so we do not repeat the argument here.



**Server privacy.** We simulate the view of clients to ensure server privacy. Let the view of the client be

$$\text{View}_{\text{client}}(X, \mathcal{Y}) = (X, \text{rnd}, Q, R, A = f_c(X, \mathcal{Y}))$$

where  $\text{rnd}$  is the internal random tape,  $Q$  is the encrypted query,  $R$  is the server's encrypted response, and  $A$  is the client's output.

All our protocols start with clients sending an encrypted query  $Q$  to the server and receiving an encrypted response  $R$ . Afterward, clients apply the appropriate reveal function on  $R$  to compute the output  $A \leftarrow \text{REVEAL}(sk, R)$ . Correctness ensures that the output  $A$  computed by the reveal algorithm is equal to the expected output  $f_c(X, \mathcal{Y})$  summarized in Table B.2.

Now we build a simulator  $\mathcal{S}_C$  that given the input  $(X, \text{rnd}, A = f_c(X, \mathcal{Y}))$  simulates the clients view as follows:

1. The simulator uses  $Q' \leftarrow (\text{SD-})\text{QUERY}(pk, X)$  to compute a query depending on the domain size with  $\text{rnd}$  as internal randomness.
2. We categorize the client's output based on the range of  $f_c(X, \mathcal{Y})$  into 3 groups:

$$A \in \begin{cases} \{0, 1\} & \text{For: F-Match, Th-Match, Tv-Match, X-Agg} \\ \{0, 1\}^k & \text{For: PSI, NA-Agg} \\ \mathbb{Z} & \text{For: PSI-CA, CA-Agg, Ret-Agg} \end{cases}$$

Depending on the range category, the simulator computes the server response  $R'$  as follows using the circuit-privacy simulator to ensure equivalent noise levels:

$$R' \leftarrow \begin{cases} \text{HE.Enc}(pk, r \cdot A) & A \in \{0, 1\} \\ \langle \text{HE.Enc}(pk, r_j \cdot A[j]) \rangle & A \in \{0, 1\}^k \\ \text{HE.Enc}(pk, A) & A \in \mathbb{Z} \end{cases}$$

where  $r \leftarrow^{\$} \mathbb{Z}_q^*$  and  $r_j \leftarrow^{\$} \mathbb{Z}_q^*$  are random values.<sup>1</sup>

3. The simulator returns  $(X, \text{rnd}, Q', R', A)$ .

Now we show that the distribution returned from the simulator  $\mathcal{S}_C$  is indistinguishable from  $\text{View}_{\text{client}}$ . The three variables  $X, A, \text{rnd}$  are taken from the

<sup>1</sup>When evaluating F-Match, the simulator chooses random variables from  $\mathbb{Z}_q$  instead of  $\mathbb{Z}_q^*$  to ensure the same false positive probability as the real-world execution.

input and are guaranteed to have the same distribution between the simulation and the view. Therefore, we only need to show that the joint distribution for the query and the server response are indistinguishable – conditional on the common  $X, A$ , and  $\text{rnd}$  variables.

Both the query  $Q$  and response  $R$  are ciphertexts. As discussed in ‘ciphertext indistinguishability’ in Section B.3.1, we are in a semi-honest scenario where the distinguisher knows both the public and private keys  $(pk, sk)$ . Our HE scheme is both IND-CPA and circuit private so we only need to show that these ciphertext pairs,  $(Q, Q')$  and  $(R, R')$ , decrypt to the same value. Queries are computed following  $Q \leftarrow (\text{SD-})\text{QUERY}(pk, X)$ . The query function only encrypts the input set  $X$  as the query so both  $Q$  and  $Q'$  should decrypt to the same value. The server’s response  $R$  depends on the client’s output  $A$  and this relation is specified in our  $\text{REVEAL}$  functions. Our correctness property ensures that the following relation between  $R$  and  $A$  holds in the real-world:

$$R \leftarrow \begin{cases} \llbracket b \rrbracket & b = \begin{cases} 0 & A = 0 \\ \leftarrow^{\$} \mathbb{Z}_q^* & A = 1 \end{cases} \mid A \in \{0, 1\} \\ \langle \llbracket b_j \rrbracket \rangle & b_j = \begin{cases} 0 & A[j] = 0 \\ \leftarrow^{\$} \mathbb{Z}_q^* & A[j] = 1 \end{cases} \mid A \in \{0, 1\}^k \\ \llbracket A \rrbracket & A \in \mathbb{Z} \end{cases} .$$

It is straightforward to see that the underlying plaintext of  $R'$ , produced in step (2) by  $\mathcal{S}_C$ , follows the same distribution.  $\square$

**Client privacy.** We simulate the view of servers to ensure client privacy. Simulating the server’s view is considerably simpler than the client’s as the server only observes ciphertexts encrypted under the client’s key. In this simulation, the adversary is a semi-honest server and the distinguisher does not have access to the client’s secret key. Moreover, our HE scheme is IND-CPA, which simplifies ciphertext indistinguishability to ensuring  $|c| = |c'|$  (see Section B.3.1).

Let the server’s view be  $\text{View}_{\text{server}} = (\mathcal{Y}, \text{rnd}, Q, R)$ . We build a simulator  $\mathcal{S}_S$  that given the input  $(\mathcal{Y}, \text{rnd})$  proceeds as follows:

1. The simulator chooses two random variables  $r_1$  and  $r_2$  with the same size as the query and the server response then encrypts them.

$$Q' \leftarrow \text{HE.Enc}(pk, r_1), R' \leftarrow \text{HE.Enc}(pk, r_2)$$

2. The simulators return  $(\mathcal{Y}, \text{rnd}, Q', R')$ .

The variables  $\mathcal{Y}$  and  $\text{rnd}$  are taken from the input and have the same distribution between the simulation and the view. The two variables  $Q'$  and  $R'$  are encrypted under the client's key. Without the knowledge of the secret key  $Q \stackrel{c}{\equiv} Q'$  and  $R \stackrel{c}{\equiv} R'$  hold independent of their content.  $\square$

### B.3.3 Malicious server

Now, we study the setting where the server is malicious. In this setting, the client is honest and server privacy is not applicable. Our framework provides *no correctness guarantee* in this setting. The malicious server can force a corrupted response that depends on the client input or fix the outcome independent of the query. Despite the lack of correctness guarantee, malicious servers cannot learn any information about the client's private data. We restate client privacy (Definition 6) more formally, then prove Theorem 6 by giving a tight reduction from our scheme's client privacy to the IND-CPA security of our HE encryption scheme. During this proof, we assume that the client's input fits into a single batched ciphertext, which holds throughout our evaluation. Adjusting the proof to support input encrypted in  $k$  ciphertexts is straightforward and gives a  $k$ -fold advantage to the adversary.

**Definition** (Client privacy). A PCM protocol is client private if no PPT adversary  $\mathcal{A}$  exists such that:

$$\Pr \left[ \begin{array}{l} p \leftarrow \text{HE.ParamGen}(1^\ell) \\ (pk, sk) \leftarrow \text{HE.KeyGen}(p) \\ (\text{st}, X_0, X_1) \leftarrow \mathcal{A}(pk) \\ b \leftarrow_{\$} \{0, 1\} \\ Q_b \leftarrow (\text{SD-})\text{QUERY}(pk, X_b) \\ b' \leftarrow \mathcal{A}(\text{st}, Q_b) \end{array} : \begin{array}{l} |X_0| = |X_1| \\ b = b' \end{array} \right] > \frac{1}{2} + \epsilon.$$

*Proof.* Let  $\mathcal{A}$  be an adversary that can break the client privacy property with a non-negligible probability  $\epsilon$ . We build a new adversary  $\mathcal{A}'$  that can break the IND-CPA security (Definition 8) of our HE scheme with the same probability  $\epsilon$ .

1. The adversary  $\mathcal{A}'$  starts an IND-CPA challenge and receives a public key  $\mathcal{A}'(pk)$ .
2. The adversary  $\mathcal{A}'$  calls  $(\text{st}, X_0, X_1) \leftarrow \mathcal{A}(pk)$ .

3. Depending on the domain size,  $\mathcal{A}'$  converts sets  $X_a$  to  $m_a = \langle x_i \rangle$  or  $m_a = \langle z_i \rangle$  following the logic of (SD-)QUERY.
4. The adversary  $\mathcal{A}'$  continues the IND-CPA challenge with  $(\mathbf{st}, m_0, m_1)$  and receives  $\mathcal{A}'(\mathbf{st}, c_b)$ .
5. The adversary  $\mathcal{A}'$  passes the challenge  $b' \leftarrow \mathcal{A}(\mathbf{st}, Q_b = c_b)$  to  $\mathcal{A}$  and returns  $b'$  as the output.

We show that the adversary  $\mathcal{A}$  cannot distinguish interaction with  $\mathcal{A}'$  from our protocol. The adversary  $\mathcal{A}$  has two interactions in the client privacy challenge. The first interaction is getting a fresh public key  $pk$  which is the same between the IND-CPA and client privacy challenges. In the second interaction,  $\mathcal{A}$  receives a query  $Q_b$  produced by (SD-)QUERY. We know that (SD-)QUERY consists of two steps: convert  $X$  into  $p = \langle x_i \rangle$  or  $p = \langle z_i \rangle$  which  $\mathcal{A}'$  performs in (3) and encrypting  $p$  with  $pk$  which is performed as part of IND-CPA challenge. This ensures that  $c_b$  is computed in the same way as  $Q_b$  and follows the same distribution. The adversary  $\mathcal{A}'$  runs one instance of client privacy with  $\mathcal{A}$  and succeeds the IND-CPA as long as the  $\mathcal{A}$  succeeds leading to the same  $\epsilon$  advantage.  $\square$

### B.3.4 Malicious clients

When addressing malicious clients, client privacy is irrelevant and correctness does not apply either as the server has no output. Therefore, we only need to address our protocols' server privacy.

**Theorem 8.** Our protocols provide server privacy against malicious clients if the HE scheme is IND-CPA and strongly input private.

Unfortunately, direct application of real-world/ideal-world simulation on our protocols in the malicious setting is not possible. The biggest challenges in simulation proofs in the malicious setting is that the adversary is not required to use its input and random tape during the execution. Therefore, the simulation needs to extract the effective input that the malicious adversary uses to determine the corresponding output. In our protocols, the server only receives an encrypted query from the client and there is no further interaction, such as having ROM calls in the client, to provide any extraction opportunity. Therefore, the semantic security of our HE scheme prevents extracting the effective inputs. To address this challenge, we adapt the ideal-world paradigm

and introduce a new notion for simulating HE protocols called cipher-world. Cipher-world is inspired by trapdoor commitments and allows the trusted party to decrypt the query and compute the ideal functionality. We first prove that cipher-world provides the same server privacy guarantee as the ideal-world then use it to prove our framework server-private. The introduction of cipher-world as a new paradigm to prove security makes our proof highly non-standard.

## Cipher world

We extend the notion of real-world/ideal-world simulation for two-party HE-based schemes in which exactly one party, called the querier, holds a pair of HE keys while the other party, called the evaluator, performs computation in the encrypted domain. This extension is *one-sided* and only addresses the privacy of the evaluator (i.e., the server in our protocols). We call this extension cipher-world.

Cipher-world assumes that the trusted party defined in the ideal-world accepts encrypted inputs, is *computationally unbounded*, and can extract the secret key  $sk$  of the querier from its public key  $pk$  – breaking the security of the HE scheme in the process. The trusted party uses this secret key to extract the effective input  $X$  of malicious queriers. Unlike the ideal-world, there is no guarantee that the decrypted input of the cipher oracle is valid and follows the input restrictions. The cipher oracle first verifies input restrictions and outputs  $\perp$  if any check fails; otherwise, it computes and outputs  $f(X, Y)$ . Additionally, the cipher-world reveals the secret key  $sk$  to the simulator. This key is solely used for the purpose of simulating ciphertext noise. Figure B.1 shows the structure of the cipher-world and how it compares to the standard ideal-world setting. We define both ideal-world and cipher-world oracles below to highlight their differences:

$$\begin{aligned} f(X, Y) &\leftarrow \text{IDEAL}(n, X, Y) \\ (sk, f(X, Y)/\perp) &\leftarrow \text{CIPHER}(n, (pk, c = \llbracket X \rrbracket), Y). \end{aligned}$$

Recall that here  $Y$  contains the evaluator’s private input.

**Theorem 9.** The cipher-world provides the same privacy guarantee for the evaluator’s private information as the ideal-world.

*Proof.* It is clear that queriers can learn more information from interacting with cipher oracles than ideal oracles. However, we show that this leakage

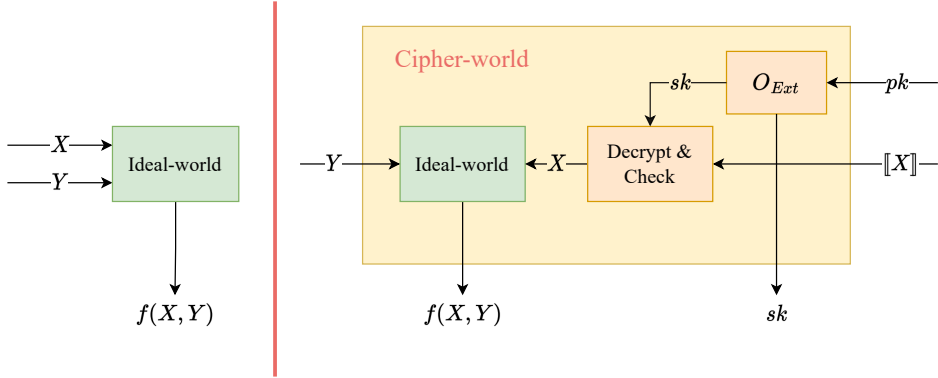


Figure B.1: Structure of the cipher-world and its difference with the ideal-world.

does not impact evaluator's privacy. First, we formally prove that this leakage  $\mathcal{L}$  is bounded to providing an oracle  $sk \leftarrow \mathbb{O}_{\text{Ext}}(pk)$  which extracts the secret key from HE public keys. Next, we prove that the leakage  $\mathcal{L}$  is independent of the evaluator's private data  $Y$ .

We assume that a PPT adversary  $\mathcal{A}$  exists such that  $\mathcal{A}$  gains more advantage from interacting with a cipher oracle instead of an ideal oracle than  $\mathcal{L} = \{\mathbb{O}_{\text{Ext}}\}$ . We build a new adversary  $\mathcal{A}'$  that given an ideal oracle  $\text{IDEAL}(Y, \cdot)$  and an extraction oracle  $\mathbb{O}_{\text{Ext}}$  can simulate the view of  $\mathcal{A}$ .

1. Adversary  $\mathcal{A}'$  initiates a new interaction with  $\mathcal{A}$  and receives  $(pk, c = \llbracket X \rrbracket) \leftarrow \mathcal{A}(\cdot)$ .
2. Adversary  $\mathcal{A}'$  uses the extraction oracle within  $\mathcal{L}$  to extract the secret  $sk \leftarrow \mathbb{O}_{\text{Ext}}(pk)$ .
3. Adversary  $\mathcal{A}'$  decrypts  $\mathcal{A}$ 's query  $X \leftarrow \text{HE.Dec}(sk, c)$ .
4. Adversary  $\mathcal{A}'$  verifies whether  $X$  passes input restrictions.
  - (4.a) If any check fails,  $\mathcal{A}'$  sets  $A \leftarrow \perp$ .
  - (4.b) Otherwise,  $\mathcal{A}'$  interacts with the ideal world oracle with the decrypted input and sets  $A \leftarrow \text{IDEAL}(X, Y)$ .
5. Adversary  $\mathcal{A}'$  finishes the execution  $\mathcal{A}(sk, A)$ .

We need to show that the adversary  $\mathcal{A}$  cannot distinguish  $(sk, A)$  produced in our simulation from the output of the cipher-world. Both the simulation and the cipher-world oracle are directly using the extraction oracle  $\mathbb{O}_{\text{Ext}}$  to produce the secret key  $sk$  which ensures secret key indistinguishability. We study two

cases for  $A$ : (1) The adversary  $\mathcal{A}$  does not follow the input restriction. In this case, the cipher oracle responds with the failure symbol  $\perp$ . The adversary  $\mathcal{A}'$  performs the same input verification process as the cipher oracle which leads to setting  $A \leftarrow \perp$  when one of step (4.a) checks fail. (2) The adversary  $\mathcal{A}$  follows the input restriction. In this scenario, both the cipher and ideal oracles compute the same output ensuring that  $A = f(X, Y)$ . This proves that the leakage of cipher-world can be bound to  $\mathcal{L} = \{\mathbb{O}_{\text{Ext}}\}$ .

Now we need to show that the leakage of the cipher-world is independent of the evaluator's private data. As we bound the leakage to an extraction oracle  $\{\mathbb{O}_{\text{Ext}}\}$ , this independence is clear since extraction is not impacted by changing the evaluator's private data  $Y$ . Note that in our protocol, exactly one party, the querier, generates HE keys, so assuming that HE key extraction is easy has no impact on the evaluator. Therefore, our cipher-world provides the same evaluator privacy guarantee as the ideal world.  $\square$

We showed that our cipher-world provides the same server (evaluator) privacy as the original ideal-world. Note that the cipher-world is one-sided and does not make any security claim about clients (queriers). To prove that our protocol provides server privacy, we have to show that the real-world view can be simulated given access to a cipher-world oracle.

$$\begin{aligned} \{\text{CIPHER}_{\mathcal{S}'_C}^{\text{Client}}(X, \mathcal{Y})\} &\stackrel{c}{\equiv} \{\text{REAL}_{\mathcal{A}}^{\text{Client}}(X, \mathcal{Y})\} \\ &\Downarrow \\ \mathcal{S}'_C(X, \text{rnd}, \mathcal{A}, \text{CIPHER}(Y, \cdot)) &\stackrel{c}{\equiv} \text{View}_{\mathcal{A}}(X, \mathcal{Y}) = (X, \text{rnd}, Q, R, A) \end{aligned}$$

We build a simulator  $\mathcal{S}'_C$  that given a PPT real-world malicious client  $\mathcal{A}$  and a cipher-world oracle  $\text{CIPHER}(\mathcal{Y}, \cdot)$  fixed with the server's input, simulates the real-world as follows:

1. Simulator  $\mathcal{S}'_C$  initiates a new interaction with  $\mathcal{A}$  and receives  $(pk, Q = \llbracket X \rrbracket) \leftarrow \mathcal{A}(\cdot)$ .
2. Simulator  $\mathcal{S}'_C$  interacts with the cipher oracle and learns  $(sk, A') \leftarrow \text{CIPHER}((pk, Q))$ .
3. If the cipher-world detects a malicious query  $Q$  which does not respect input checks (i.e.,  $A' = \perp$ ), the simulator  $\mathcal{S}'_C$  chooses a random response  $t \leftarrow^{\$} \mathbb{Z}_q$ , sets the output accordingly  $A' \leftarrow \text{REVEAL}(sk, \llbracket t \rrbracket)$ , and skips to the step 5.

4. Otherwise,  $\mathcal{S}'_C$  computes the response  $t$  as follows:

$$t \leftarrow \begin{cases} r \cdot A' & A \in \{0, 1\} \\ \langle r_j \cdot A'[j] \rangle & A \in \{0, 1\}^k \\ A' & A \in \mathbb{Z} \end{cases} .$$

Note that the range of  $A$  is determined by functionality  $f$  and is independent of the parties' input.

5. Simulator  $\mathcal{S}'_C$  performs  $R' \leftarrow \mathcal{S}_{sip}(sk, f, Q, t)$ .
6. Adversary  $\mathcal{A}$  only learns the response  $R$  if the decryption succeeds. Therefore, Simulator  $\mathcal{S}'_C$  checks if the decryption  $\text{HE.Dec}(sk, R')$  succeeds. Otherwise,  $\mathcal{S}'_C$  sets  $A' \leftarrow \perp$ .
7. The simulator returns  $(X, \text{rnd}, Q = \llbracket X' \rrbracket, R', A')$ .

Now, we show that the real view  $(X, \text{rnd}, Q = \llbracket X' \rrbracket, R, A)$  is indistinguishable from the simulated view  $(X, \text{rnd}, Q, R', A')$ . The simulator  $\mathcal{S}'_C$  does not use variables  $X$  and  $\text{rnd}$  as there is no guarantee that malicious clients will use their input or random tapes. As  $(X, \text{rnd}, Q)$  are directly taken from the input, we only need to show that  $(R, A) \stackrel{c}{\equiv} (R', A')$  conditional on the common variables. We split our analysis into three cases:

*Malicious queries which do not represent a set.* When the client is malicious there is no guarantee that the query represents a set. In Section 3.6.3, we designed (SD-)QUERY-CHECK functions and proved that they randomize the output of our protocols when the query does not represent a set as long as the HE abstraction holds. Our simulator  $\mathcal{S}'_C$  relies on the cipher oracle to detect when the query does not represent a valid set in step 3 and assigns a uniformly random value  $t$  to be encrypted as the response  $R'$ . To ensure that  $R'$  has an indistinguishable noise from the real response  $R$ , instead of directly using the encryption in step 5,  $\mathcal{S}'_C$  uses the simulator  $\mathcal{S}_{sip}$  from the strong input privacy property (Definition 10). Our simulator follows the same REVEAL process to compute the output  $A$  from the response as the real protocol. Since  $R \stackrel{c}{\equiv} R'$ , we will have  $A \stackrel{c}{\equiv} A'$  as long as the server response  $R'$  decrypts successfully.

*The malicious query decrypts to the set  $X'$ :* When the query represents a set, (SD-)QUERY-CHECK produces an encrypted zero and does not impact the output of the protocol (i.e., adding  $\llbracket 0 \rrbracket$  is neutral). Knowing that the query is an encryption of the set  $X'$ , our protocols ensure that  $A = f(X', \mathcal{Y})$  as long as the decryption of  $R$  succeeds. This guarantee follows from combining (1) our semi-honest correctness from Section B.3.2 and (2) knowing that the



query decrypts to the same value as  $(\text{SD-})\text{QUERY}(pk, X')$ . The simulator  $\mathcal{S}'_C$  follows a similar process to compute  $R$  from  $A$  as our semi-honest simulator  $\mathcal{S}_C$  with the difference of replacing  $\text{HE.Enc}$  with  $\mathcal{S}_{sip}$  (i.e., relying on strong input privacy instead of circuit privacy).

*Failed decryption.* Unlike the semi-honest setting where we know that the client’s query is freshly encrypted, the server may receive queries with a high noise level. This may lead to producing server responses that fail the decryption. The use of  $\mathcal{S}_{sip}$  in step 5 ensures that the response of  $\mathcal{S}'_C$  has the same noise level as our real-world response. Therefore, the decryption of  $R'$  fails if and only if the decryption of  $R$  fails. When the decryption fails, the real-world cannot compute the output ( $A = \perp$ ) while the simulator  $\mathcal{S}'_C$  sets  $A' \leftarrow \perp$  in step 6.

## B.4 Extra benchmarks

In this section, we provide extra details on our evaluation and add more benchmarks. First, we compare our small domain PSI layer to an existing small domain paper. Second, we provide more detail on how we design and evaluate generic solutions with the same privacy and functionality as our document search engine and provide extra performance plots. Third, we compare our document search engine to one of the fastest OT-based PSI protocols which does not satisfy our privacy requirements.

### B.4.1 Small-domain protocols

In this section, we evaluate the performance of our small domain PSI-CA protocol and compare it to existing work. We focus on Ruan et al. [109] in this section. We compare with Shimizu et al. [55] in Section 4.8. We do not consider Bay et al. [103] here, as it focuses on a multi-*party* scenario, which leads to higher costs.

The code for the protocol of Ruan et al. is not available but the paper provides a detailed cost analysis. We use the same scenario and the same CPU (Intel Core i7-7700) to allow us to directly compare performance without requiring us to rerun their protocol from scratch. Moreover, we extend and optimize their approach for a many-set scenario. Ruan et al. use bit-vectors encrypted with ElGamal [227] or Paillier [228] encryption to perform PSI.

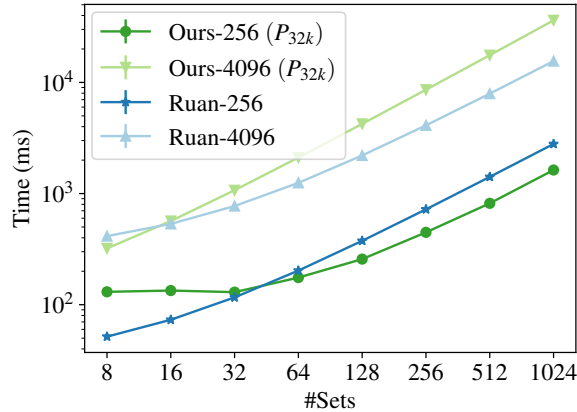


Figure B.2: Computation cost for performing small domain PSI-CA. Two systems provide different security levels: Ruan et al. support 80-bit security while ours provide 128-bit security.

We extend their approach to only compute the query once and apply it to many sets. Their detailed performance benchmark allows us to compute the cost of performing a many-set query with  $N$  sets. In Fig. B.2, we show the computation cost with a fixed input domain size ( $|D| \in \{256, 4096\}$ ) and varied the number of sets. The cost does not include the key exchange. The performance of our scheme is comparable to Ruan et al. – which scheme is performing best depends on the specific scenario.

Despite having the same operating point for both approaches, they have very different security guarantees. Ruan et al. assume a *semi-honest* privacy model and use Pallier keys with 1024-bit RSA primes which only provide *80-bit* security. Extracting more information than cardinality from their protocol is trivial for malicious clients. On the other hand, we provide full *128-bit* security and protect against malformed queries by misbehaving clients (see Section 3.6.3).

#### B.4.2 Circuit-based protocols

In this section, we provide more details on our evaluation of the generic solutions in the document search scenario of Section 3.11.2. We expand on their threat model and properties. Moreover, we report and compare the server computation cost of all approaches.

**Generic SMC.** We use a high-level SMC compiler, EMP tool-kit [119], to design and evaluate circuits providing the same properties as our search engine. More specifically, we use the ‘EMP-sh2pc’ branch of the compiler that provides security in a two-party semi-honest setting and supports garbled circuits.

We encode each input as a 32 bits binary value, which results in a higher false-positive rate due to encoding keywords than our framework, where we encode keywords using 39 or 44 bits. We use a private equality check offered by EMP for comparing encrypted set elements. To determine whether one keyword of the client set has a match in a given document, we perform equality tests against all keywords of a document and perform an OR over the comparison results. To perform the full matching, we do an AND over the matching status of all client keywords. This produces a 1-bit result for each document determining its relevance. The F-Match process does not create any extra false-positive in this approach.

Now we have to aggregate  $N$  1-bit document matching statuses according to our X-Agg and CA-Agg policies. The X-Agg variant is straightforward and we use an OR to check if any document is relevant. To count the number of relevant documents, we first convert binary matching statuses to integers encoding ‘0’ or ‘1’ to enable us to continue the computation with an arithmetic circuit. Then, we compute the sum of these integer statuses.

**Circuit-PSI.** We choose Chandran et al. [81] as a state-of-the-art circuit-PSI paper that is secure against semi-honest adversaries. Circuit-PSI protocols perform an intersection between the sets of two parties and secret share the output among them. This enables using circuits to privately compute arbitrary functions over the intersection. Despite the capability of Chandran et al. to be extended with circuits to compute F-Match matching and X-Agg or CA-Agg aggregation, we decide to not extend their circuit and use the time necessary for computing the intersection as a lower bound on the cost of searching. Since the PSI protocol of Chandran et al. is a single set protocol, we run  $N$  instances of Circuit-PSI sequentially to simulate searching  $N$  documents. We use the default parameters decided by Chandran et al., meaning that each item is encoded using 32 bits with an additional false positive rate of  $2^{-40}$  due to computation.

**Server’s computation cost.** We have already reported the end-to-end latency, communication cost, and client’s computation cost in the main body (Fig. 3.6). We report the server’s computation cost in Fig. B.3. Starting from

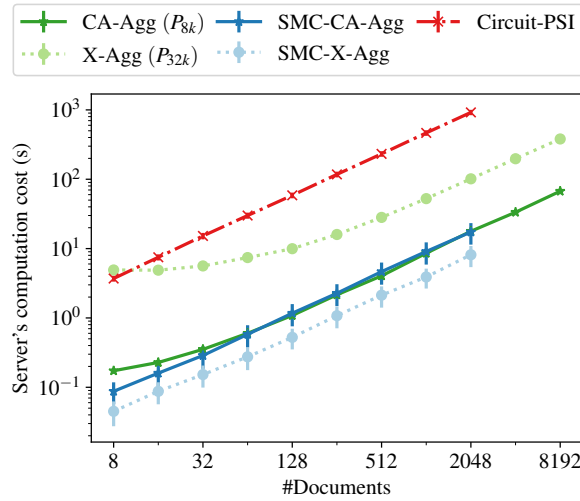


Figure B.3: Server's computation cost for document search.

16 documents, our framework has lower server computation than Chandran et al. [81]. On the other hand, the EMP solutions have better server efficiency than our scheme which is not surprising as our framework outsources the computation load from thin clients to the server. Despite our outsourcing, our CA-Agg search only increases the server cost by a factor of 2.5x and X-Agg by a factor of 30x when searching 1k documents.

### B.4.3 OT-based protocol

There are efficient PSI protocols that are based on the oblivious transfer in both the semi-honest (such as SpOT-light [72]) and the malicious setting (such as PaXoS [92]). As discussed in Section 3.3, this line of research focuses on computing one-to-one equality tests between the client and server which leaks information about each server set and cannot satisfy our privacy requirements. Despite providing a lower privacy guarantee, we compare our approach to the SpOT-light protocol as a baseline cost.

**Document search with SpOT-light.** We follow the document search setting from the Section 3.11.2 and evaluate the cost of using SpOT-light to search  $N$  documents. SpOT offers 128-bit security in a semi-honest setting and accepts 256-bit input elements which bypasses the false-positive rate of mapping keywords. However, SpOT is (1) a single-set protocol and (2) does not support privacy extensions such as computing relevance without leaking the intersection cardinality to the client (i.e., private set matching) or aggregating the

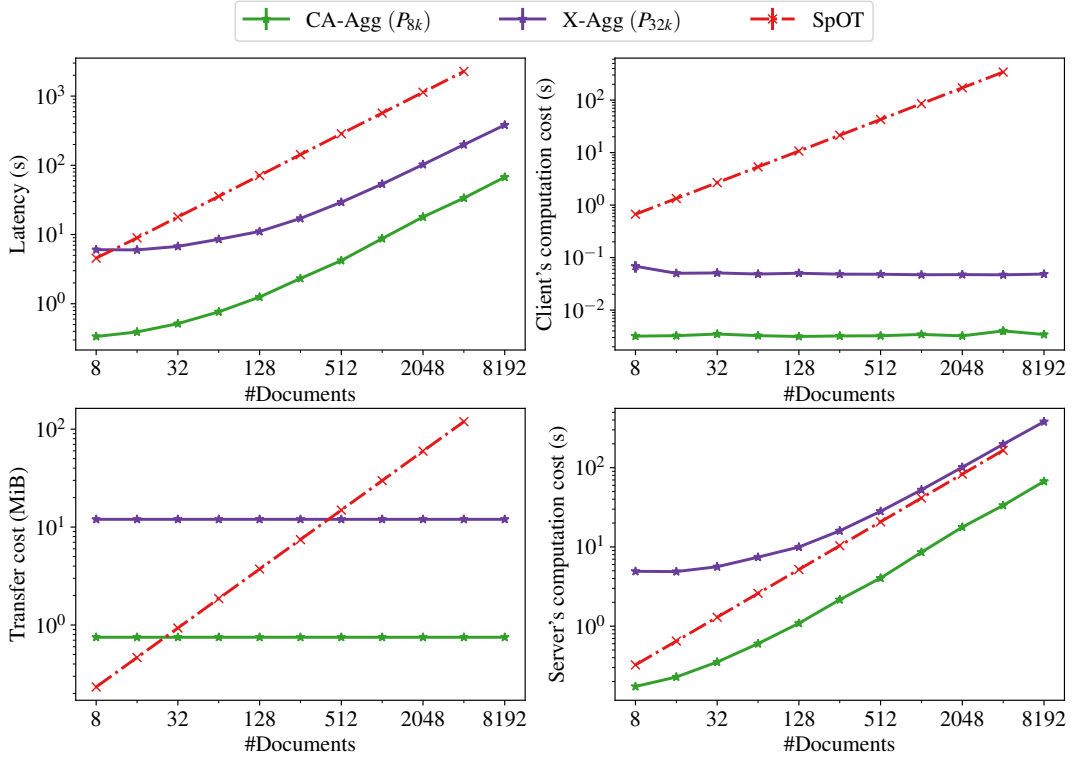


Figure B.4: The end-to-end latency (upper-left), communication cost (lower-left), and client and server computation cost (right) of the document search.

search result of multiple documents similar to our X-Agg and CA-Agg variant. We handle the single-set limitation by running  $N$  sequential PSI interactions to search  $N$  documents, but we do not add any countermeasure for the lack of matching or aggregation functionality. We encountered concurrency issues (with async IO) when running SpOT. This issue gets amplified when repeating the protocol  $N$  times. Instead of directly running the code, we benchmarked the cost of each interaction through multiple runs with  $N = 8$ , then extrapolate the cost for all entries.

We report the end-to-end latency, computation, and communication costs of SpOT-light in Fig. B.4. In the single-set setting and for a small number of documents, SpOT provides better performance. However, as soon as reaching 32 documents, our CA-Agg variant starts to provide better latency, computation, and communication than SpOT despite providing better privacy. When searching 1k documents, our framework improves latency by a factor of 10–65x, communication by a factor of 1.7–27, and client’s computation by a factor of 1800–24,800x depending on the search functionality.

## B.5 Solving matching in mobile apps

We do not separately evaluate the matching in the mobile apps scenario as it is similar to the chemical similarity scenario. The set of attributes to be matched can be represented using a small domain as the number of attributes is limited and they have few possible values. Matching of individual records can be implemented using the threshold matching (Th-Match) protocol. This allows for approximate matches. The results can then be combined using naive aggregation to reveal the matching indices to the querier. Since Th-Match is simpler than the Tv-Match protocol and the threshold for matching is likely smaller than the chemical similarity case, we expect better performance for matching than chemical similarity.

## B.6 PSI-SUM

A benefit of our framework’s modular design is extensibility. To show the ease of adding new functionality, we design a new protocol called PSI-SUM in this section which is getting more popular in the literature due to its use in private ad-monetization systems [83, 84, 86]. In the PSI-SUM protocol, the server assigns a weight to each of its elements and the client wants to compute the sum of weights of common elements, i.e.,  $\text{PSI-Sum}(X, (Y, W)) = \sum_{\{i|y_i \in X\}} w_i$ .

We add this new protocol in our PSI layer following the structure in Fig. 3.3. We define the PSI-SUM algorithms in Algorithm 13. The function `PSI-SUM-PROCESS` computes a binary inclusion status  $\llbracket t_i \rrbracket$  for each *server* element  $y_i$ . The server then proceeds similarly to `ePSI-CA` to compute the encrypted weighted sum. The server can continue processing this value in the next layers, such as checking for a threshold value on the sum, or return  $\llbracket W \rrbracket$  to the client which decrypts it to obtain the answer (see `PSI-SUM-REVEAL`). Our extensions such as ensuring query well-formedness and many-set can directly apply to this protocol without any extra effort.

---

**Algorithm 13** Adding PSI-SUM capabilities.

---

**function** **PSI-SUM-PROCESS**( $pk, Q = \langle \llbracket x_i \rrbracket \rrbracket, Y, W = \langle w_i \rangle$ )  
 $\llbracket t_i \rrbracket \leftarrow \text{HE.IsZero}(pk, \text{HE.IsIn}(pk, y_i, \langle \llbracket x_j \rrbracket \rrbracket))$   
 $\llbracket \mathcal{W} \rrbracket \leftarrow \sum_{i \in [n]} w_i \cdot \llbracket t_i \rrbracket$   
**return**  $\llbracket \mathcal{W} \rrbracket$

**function** **PSI-SUM-SD-PROCESS**( $pk, Q = \langle \llbracket z_i \rrbracket \rrbracket, Y, W = \langle w_i \rangle$ )  
 $\langle \llbracket t_i \rrbracket \rangle \leftarrow \text{PSI-SD-PROCESS}(pk, \langle \llbracket z_i \rrbracket \rrbracket, Y)$   
 $\llbracket \mathcal{W} \rrbracket \leftarrow \sum_{d_i \in D} w_i \cdot \llbracket t_i \rrbracket$   
**return**  $M \leftarrow \llbracket \mathcal{W} \rrbracket$

**function** **PSI-SUM-REVEAL**( $pk, M = \llbracket \mathcal{W} \rrbracket$ )  
**return**  $\text{HE.Dec}(sk, \llbracket \mathcal{W} \rrbracket)$

---





# APPENDIX C

## Appendix for Janus

### C.1 Extended evaluation

We evaluated the performance of Janus in Section 4.8. In this section, we provide more details on the computation cost of SMC-Janus and the impact of the template size on the performance.

**Template size.** We measure the performance of deduplication with different template sizes in Fig. C.1 to study the impact of template size on performance. We use solid lines to show shorter templates (**Finger**<sub>64</sub> and **Iris**<sub>2048</sub>) and dashed lines to show longer templates (**Finger**<sub>640</sub> and **Iris**<sub>10240</sub>). We observe that increasing the template size has a linear impact on the computation of the registration station, but does not impact the computation of biometric provider or the communication. This is in line with our expectation since the size of the template only impacts the HE computation of the distance while the communication and BP's computation cost are dominated by the SMC thresholding.

**SMC-Janus.** Fig. C.2 shows the single-core computation and communication cost of operating SMC-Janus with fingerprint and iris sensors of various sizes over different database sizes. Due to our use of garbled circuits, the computation cost of both parties (the registration station and biometric provider) are similar. Moreover, we observe that the number of biometric samples  $f$ , the template size **TS**, and the number of users registered in the database have a linear impact on all three RS and BP computation and communication costs.

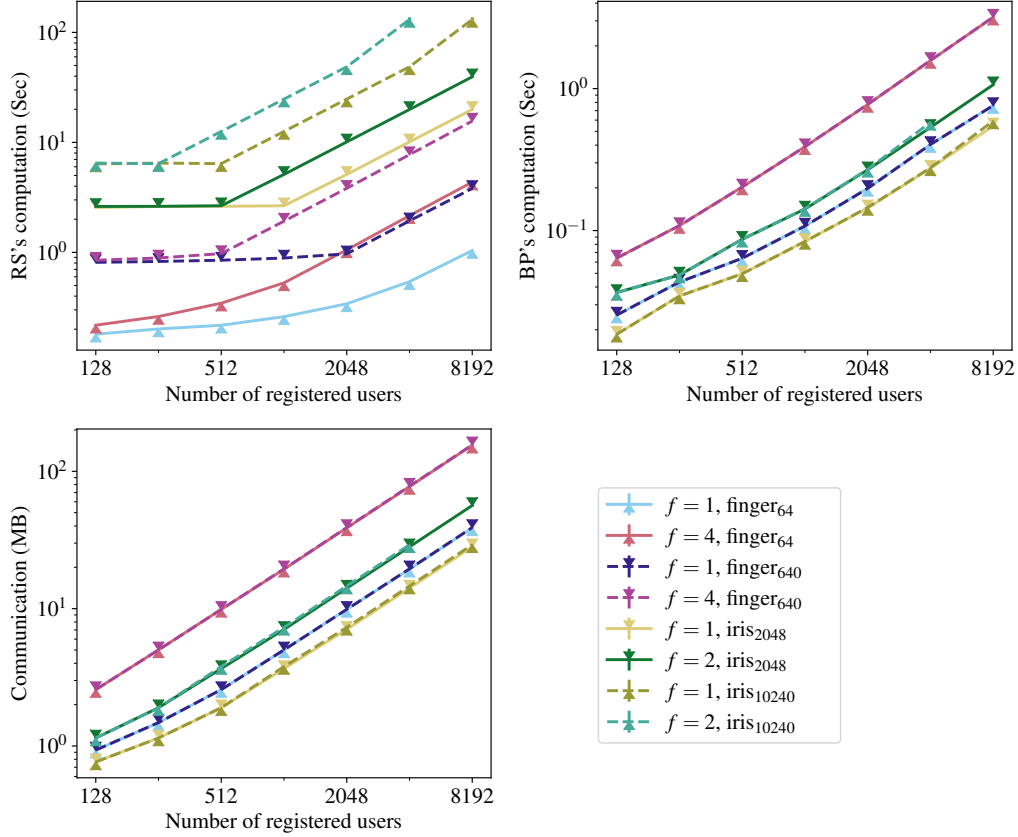


Figure C.1: Evaluating the single-core membership performance of SHE-Janus with  $f$  iris or fingerprint samples of varied sizes.

## C.2 Normalized Hamming distance

In this section, we define Normalized Hamming distance and discuss how we optimize our protocols to compute this distance.

**Normalized Hamming distance.** A biometric template  $X$  can have an associated binary mask  $M_X$  to exclude certain values during comparison. The normalized Hamming distance,  $\text{D.NormHamming}((X, M_X), (Y, M_Y))$ , applies both masks, calculates the Hamming distance, and then normalizes by the count of active comparison bits as follows:

$$\sum_i ((X[i] \oplus Y[i]) \wedge (M_X[i] \wedge M_Y[i])) * \text{TS} / |M_X \wedge M_Y|.$$

We apply the following adaptations to our instantiations:

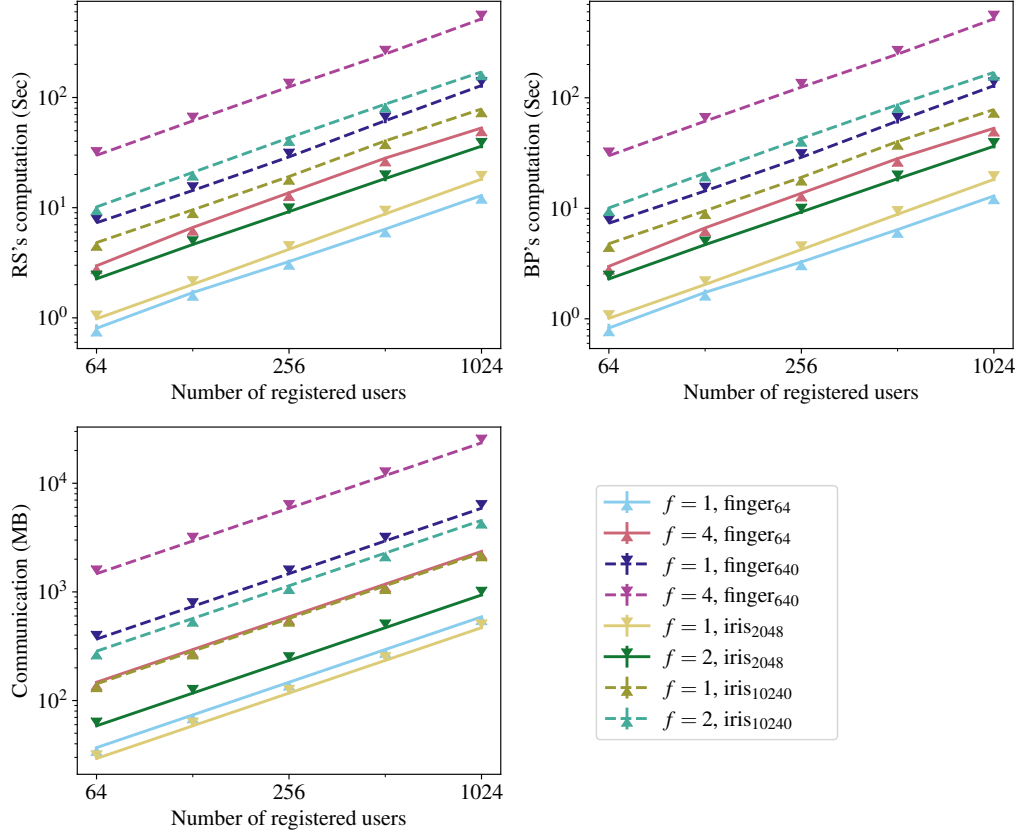


Figure C.2: Evaluating the single-core membership performance of SMC-Janus with  $f$  iris or fingerprint samples over different database sizes.

*TEE-Janus.*: Our TEE-based solution computes the matching in a plaintext domain inside an enclave where supporting normalize Hamming is trivial.

*SMC-Janus.*: We store and handle the mask in the same manner as the biometric data (secret shared between two parties) in our SMC-based solution and adapt Algorithm 7 to apply the masks and scale the final distance based on  $\sum_i (M_X[i] \wedge M_Y[i])$ .

*SHE-Janus.* While SHE schemes can support a multiplication depth beyond 1, increasing the depth impacts the schemes parameter selection leading to higher computation cost for each operation and a larger size for each ciphertext. We aim to have a multiplication depth of 1 in our SHE-Janus solution to be able to use the most efficient parameter with  $N_{deg} = 4096$ . Using multiplication to compute both the masking  $\wedge$  and the hamming  $\oplus$  in the cipher domain leads to a minimum depth of 2. Therefore, we rely on a different representation of the hamming distance, namely  $D.Hamming(X, Y) = \sum_i (X[i] \wedge \neg Y[i]) \vee (\neg X[i] \wedge Y[i])$ , to compute the distance with a lower depth. When adding a

new member, instead of separately encrypting and storing both  $Y$  and  $M_Y$ , we store the following three vectors:  $(\llbracket Y[i] \wedge M_Y[i] \rrbracket, \llbracket \neg Y[i] \wedge M_Y[i] \rrbracket, \llbracket M_Y[i] \rrbracket)$ . When performing the membership, we can compute  $X[i] \wedge M_X[i]$  and  $\neg X[i] \wedge M_X[i]$  in the plaintext domain that allows us to compute the distance as  $D \leftarrow \sum_i (X[i] \wedge M_X[i]) \cdot \llbracket \neg Y[i] \wedge M_Y[i] \rrbracket + (\neg X[i] \wedge M_X[i]) \cdot \llbracket Y[i] \wedge M_Y[i] \rrbracket$ .

Note that while we use a SHE approach to compute the distance, our algorithms only require a linearly homomorphic scheme.

# APPENDIX **D**

## Appendix for Brutus

### D.1 Random Forrest

In Section 5.4.3, we use metadata and features extracted from text to design a ciphertext detector using a Random Forrest model. Now, we provide a detailed description of the features used to train the model in Table D.1.

### D.2 Additional results on generated datasets

In Section 5.5, we perform synthetic experiments to study our detectors' behavior when facing plaintext or cipher domain shifts. When the impact of an experiment is focused on only one of the precision or recall metrics, we present the varying metric in the body and the unaffected measure in the appendix.

As mentioned in Section 5.5.2, our plots report the mean precision or recall of ciphertext detection for each detector on the cross-validation sets. Error bars show the 95% confidence interval. Since we discuss every plot in their original experiment description, we do not add individual explanations here.

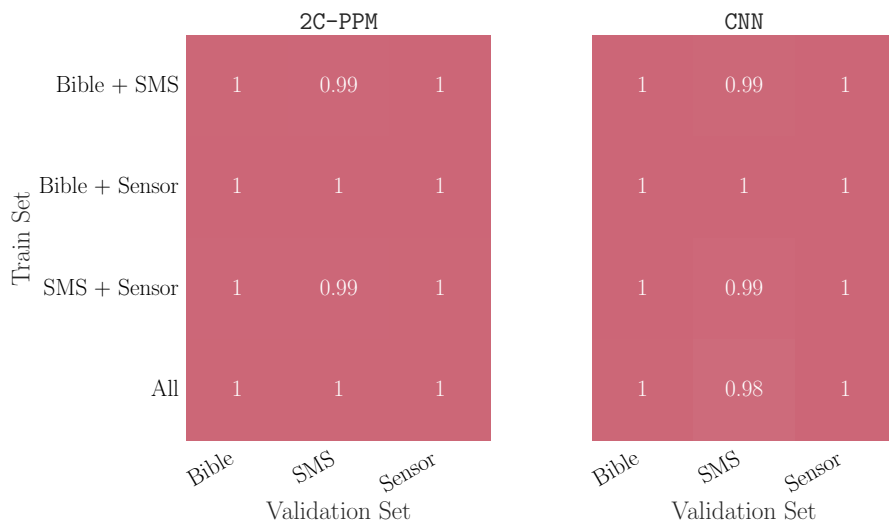


Figure D.1: Recall of cipher detection on different text sources (columns) for models trained on a mix of languages (rows). See Section 5.5.5 for more detail.

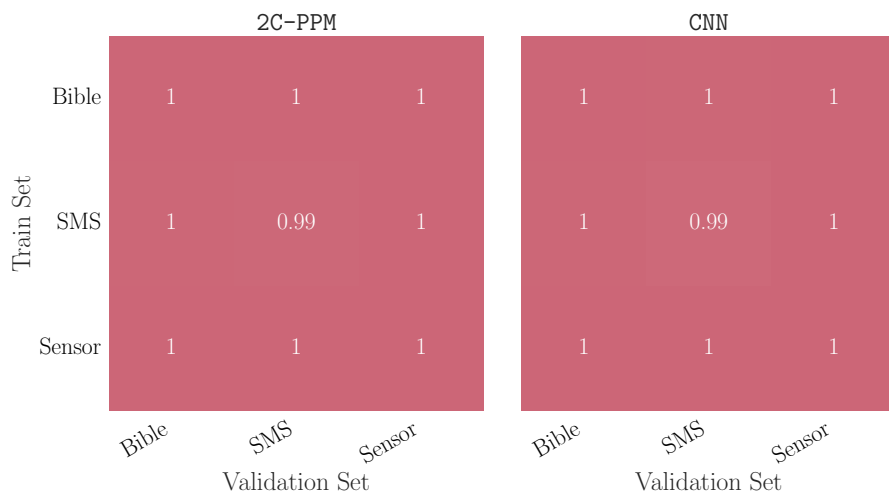


Figure D.2: Recall of cipher detection on different text sources (columns) for models trained on different texts (rows). See Section 5.5.5 for more detail.

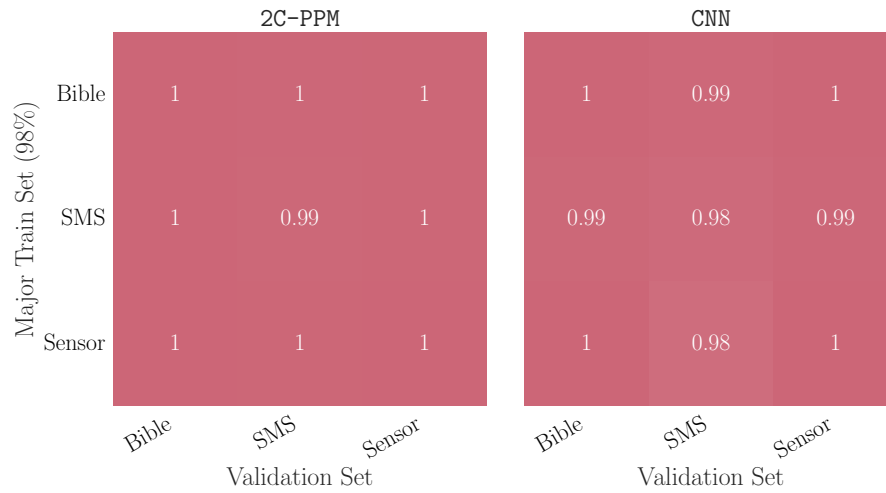


Figure D.3: Recall of cipher detection on different text sources (columns) for models trained on train sets with an unbalanced mix of languages (rows). Row labels indicate the majority source of training samples (98%). See Section 5.5.5 for more detail.

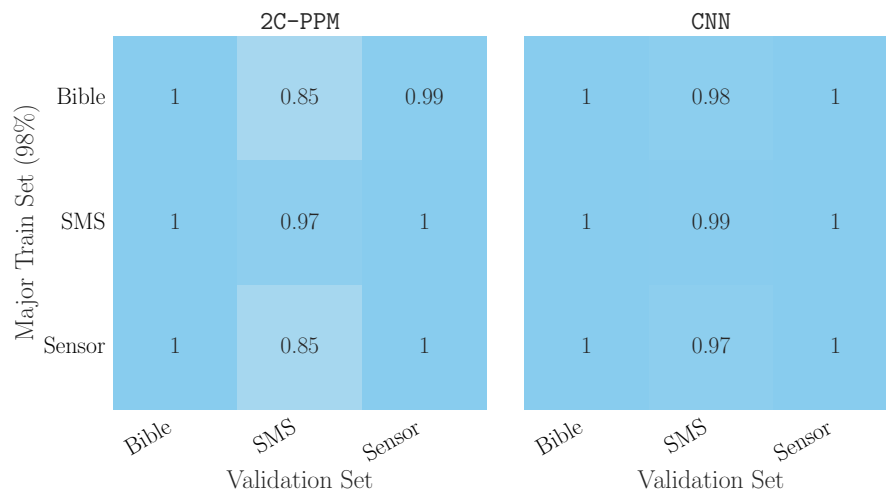


Figure D.4: Precision across validation sets from different text sources (columns) for models trained on train sets with an unbalanced mix of languages (rows). Row labels indicate the majority source of training samples (90%). See Section 5.5.5 for more detail.

Table D.1: Features of the Random Forests classifier. Features marked as M are message metadata fields, those marked as T are features extracted from the message text.

Name	Type	Description
label	M	Message label that allows to route ACARS messages to the correct endpoint in the network
manufacturer	M	Aircraft manufacturer
model	M	Aircraft model
operator	M	Airline operating the aircraft
owner	M	Registered aircraft owner
cases	T	Boolean feature indicating whether both capital and lower-case characters appear in the message text
date	T	Boolean feature indicating whether a date appears in the message text
special dict	T	Fraction of special characters in the message text
	T	Number of matches within the message text with words from the English dictionary

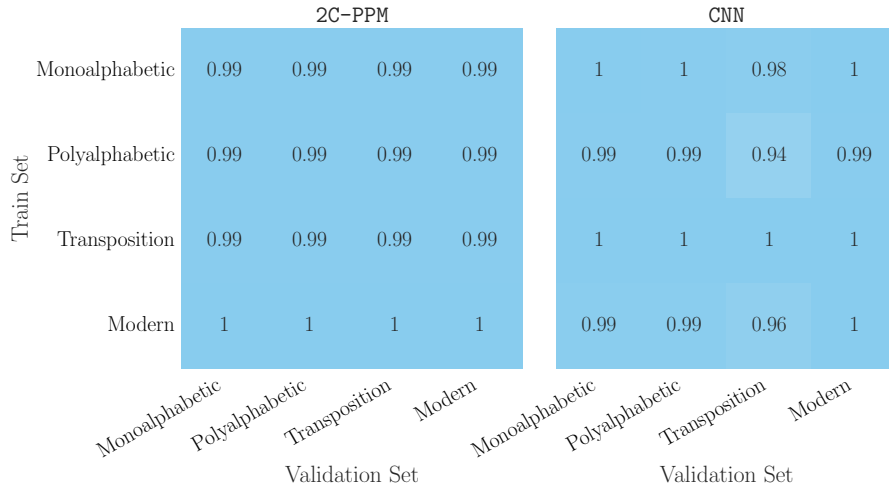


Figure D.5: Precision of cipher detection on the SMS data using different cipher suits. See Section 5.5.6 for more detail.

### D.3 Decision support system

In this section, we present our dashboard and discuss how it helps analysts. Fig. D.6 shows screenshots of our system. We are using real-world messages in our dashboard, so we are blurring information that can be used to identify actors responsible and involved aircraft.



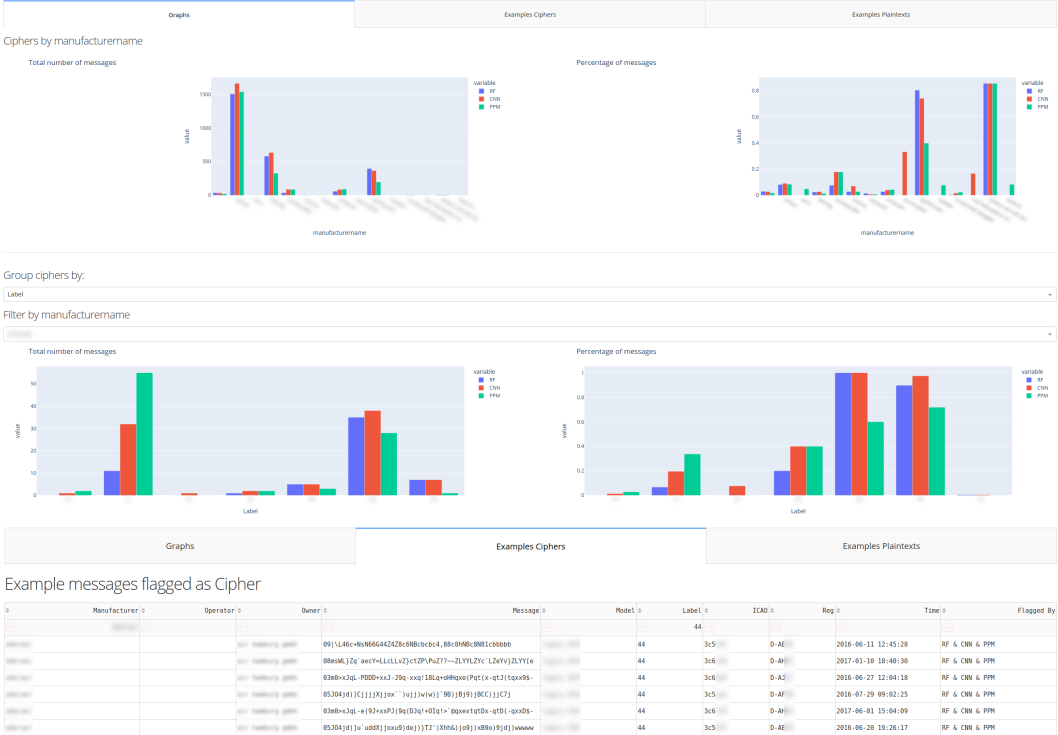


Figure D.6: An overview of our dashboard.

Our dashboard provides three views:

*Graph view.* Our dashboard provides statistics about flagged messages based on metadata and enables analysts to find patterns between messages and identify the actor responsible for the (insecure) cipher. We provide flexibility on how metadata should be used for grouping and filtering messages as the responsible actor may have different roles such as manufacturer or airline.

*Cipher view.* Our dashboard provides detailed information about messages flagged as cipher and enables searching based on metadata. We show a selected set of fields from the original raw message collection. This field selection is based on discussions with our analyst. We extend these raw features with additional aircraft metadata gathered through third-party sources (see Section 5.4.3).

*Plaintext view.* This view is similar to the cipher view but shows messages labeled as `plain`. The analyst may want to compare flagged messages with non-flagged messages with similar metadata. This comparison may have two reasons: (1) using plaintexts as a baseline to identify protocols with this specific metadata and (2) helping with further cryptanalysis as some proprietary

protocols support sending both plain and encrypted messages (with the same metadata).

## Bibliography

- [1] ICIJ. Datashare. <https://datashare.icij.org/>.
- [2] ICIJ. Panama papers. <https://www.icij.org/investigations/panama-papers/>.
- [3] Boston Globe. Church allowed abuse by priest for years. <https://www.bostonglobe.com/news/special-reports/2002/01/06/church-allowed-abuse-priest-for-years/cSHfGkTlrAT25qKGvBuDNM/story.html>, 2002.
- [4] Andrea Louise Carson. *Investigative journalism, the public sphere and democracy: the watchdog role of Australian broadsheets in the digital age*. PhD thesis, University of Melbourne, 2013.
- [5] Susan E. McGregor, Franziska Roesner, and Kelly Caine. Individual versus Organizational Computer Security and Privacy Concerns in Journalism. *PoPETs*, 2016.
- [6] Susan E. McGregor, Polina Charters, Tobin Holliday, and Franziska Roesner. Investigating the computer security practices and needs of journalists. In *USENIX*, 2015.
- [7] Susan E. McGregor, Elizabeth Anne Watkins, Mahdi Nasrullah Al-Ameen, Kelly Caine, and Franziska Roesner. When the Weakest Link is Strong: Secure Collaboration in the Case of the Panama Papers. In *USENIX*, 2017.
- [8] Amos Beimel and Yuval Ishai. Information-Theoretic Private Information Retrieval: A Unified Construction. In *ICALP*, 2001.
- [9] Ian Goldberg. Improving the Robustness of Private Information Retrieval. In *IEEE S&P*, 2007.
- [10] Eyal Kushilevitz and Rafail Ostrovsky. Replication is NOT needed: SINGLE database, computationally-private information retrieval. In *FOCS*, 1997.
- [11] Yan Huang, David Evans, and Jonathan Katz. Private Set Intersection: Are Garbled Circuits Better than Custom Protocols? In *NDSS*, 2012.

- [12] Benny Pinkas, Thomas Schneider, Christian Weinert, and Udi Wieder. Efficient Circuit-Based PSI via Cuckoo Hashing. In *EUROCRYPT*, 2018.
- [13] Benny Pinkas, Thomas Schneider, Gil Segev, and Michael Zohner. Phasing: Private Set Intersection Using Permutation-based Hashing. In *USENIX*, 2015.
- [14] Emiliano De Cristofaro and Gene Tsudik. Practical Private Set Intersection Protocols with Linear Complexity. In *FC*, 2010.
- [15] Brett Hemenway Falk, Daniel Noble, and Rafail Ostrovsky. Private set intersection with linear communication from general assumptions. In *WPES@CCS*, 2019.
- [16] Ágnes Kiss, Jian Liu, Thomas Schneider, N. Asokan, and Benny Pinkas. Private Set Intersection for Unequal Set Sizes with Mobile Applications. *PoPETs*, 2017.
- [17] Benny Pinkas, Thomas Schneider, and Michael Zohner. Faster Private Set Intersection Based on OT Extension. In *USENIX*, 2014.
- [18] Sebastian Angel and Srinath T. V. Setty. Unobservable Communication over Fully Untrusted Infrastructure. In *OSDI*, 2016.
- [19] Benny Chor, Niv Gilboa, and Moni Naor. Private Information Retrieval by Keywords. Technical report, Department of Computer Science, Technion, Israel, 1997.
- [20] Xiao Wang, Samuel Ranellucci, and Jonathan Katz. Authenticated Garbling and Efficient Maliciously Secure Two-Party Computation. In *CCS*, 2017.
- [21] David Lazar, Yossi Gilad, and Nikolai Zeldovich. Karaoke: Distributed private messaging immune to passive traffic analysis. In *OSDI*, 2018.
- [22] Jelle van den Hooff, David Lazar, Matei Zaharia, and Nikolai Zeldovich. Vuvuzela: scalable private messaging resistant to traffic analysis. In *SOSP*, 2015.
- [23] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. Tor: The Second-Generation Onion Router. In *USENIX*, 2004.
- [24] Sebastian Angel, Hao Chen, Kim Laine, and Srinath T. V. Setty. PIR with Compressed Queries and Amortized Query Processing. In *IEEE S&P*, 2018.

- [25] Emiliano De Cristofaro, Paolo Gasti, and Gene Tsudik. Fast and Private Computation of Cardinality of Set Intersection and Union. In *CANS*, 2012.
- [26] Amanda C Davi Resende and Diego F Aranha. Faster unbalanced private set intersection. *FC*, 2018.
- [27] Bin Fan, David G. Andersen, and Michael Kaminsky. Cuckoo Filter: Better Than Bloom. *login.*, 2013.
- [28] Nym project. The nym system. <https://nymtech.net/>.
- [29] Ania M. Piotrowska, Jamie Hayes, Tariq Elahi, Sebastian Meiser, and George Danezis. The Loopix Anonymity System. In *USENIX*, 2017.
- [30] Debajyoti Das, Sebastian Meiser, Esfandiar Mohammadi, and Aniket Kate. Anonymity Trilemma: Strong Anonymity, Low Bandwidth Overhead, Low Latency - Choose Two. In *IEEE S&P*, 2018.
- [31] Masayuki Abe. A secure three-move blind signature scheme for polynomially many signatures. In *EUROCRYPT*, 2001.
- [32] Jan Camenisch, Susan Hohenberger, Markulf Kohlweiss, Anna Lysyanskaya, and Mira Meyerovich. How to win the clonewars: efficient periodic  $n$ -times anonymous authentication. In *CCS*, 2006.
- [33] Stefan A. Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. 2000.
- [34] Foteini Baldimtsi and Anna Lysyanskaya. Anonymous credentials light. In *CCS*, 2013.
- [35] Man Ho Au, Willy Susilo, Yi Mu, and Sherman S. M. Chow. Constant-size dynamic  $k$ -times anonymous authentication. *IEEE Systems Journal*, 2013.
- [36] David Pointcheval and Olivier Sanders. Short Randomizable Signatures. In *CT-RSA*, 2016.
- [37] Dogan Kesdogan, Dakshi Agrawal, and Stefan Penz. Limits of anonymity in open environments. In *Information Hiding*, 2002.
- [38] Fernanda López-Escobedo, Carlos-Francisco Méndez-Cruz, Gerardo Sierra, and Julián Solórzano-Soto. Analysis of stylometric variables in long and short texts. *Procedia-Social and Behavioral Sciences*, 2013.

- [39] G MuthuSelvi, GS Mahalakshmi, and S Sendhilkumar. Author attribution using stylometry for multi-author scientific publications. *Advances in Natural and Applied Sciences*, 2016.
- [40] George Danezis. Petlib: A python library that implements a number of privacy enhancing technologies. <https://github.com/gdanezis/petlib>.
- [41] Rajath Agasthya. cuckoo: Pure python implementation of cuckoo filter. <https://github.com/rajathagasthya/cuckoo>.
- [42] Tor project. Tor metrics - performance. <https://metrics.torproject.org/onionperf-buildtimes.html>.
- [43] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *J. ACM*, 2004.
- [44] Carmit Hazay and Yehuda Lindell. Efficient Protocols for Set Intersection and Pattern Matching with Security Against Malicious and Covert Adversaries. *J. Cryptology*, 2010.
- [45] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *EUROCRYPT*, 2004.
- [46] Benny Pinkas, Thomas Schneider, and Michael Zohner. Scalable private set intersection based on OT extension. *ACM Trans. Priv. Secur.*, 2018.
- [47] Yongjun Zhao and Sherman S. M. Chow. Can You Find The One for Me? 2018.
- [48] Carlos Aguilar Melchor, Joris Barrier, Laurent Fousse, and Marc-Olivier Killijian. XPIR : Private Information Retrieval for Everyone. *PoPETs*, 2016.
- [49] Dawn Xiaodong Song, David A. Wagner, and Adrian Perrig. Practical Techniques for Searches on Encrypted Data. In *IEEE S&P*, 2000.
- [50] Raluca A. Popa, Catherine M. S. Redfield, Nikolai Zeldovich, and Hari Balakrishnan. CryptDB: protecting confidentiality with encrypted query processing. In *SOSP*, 2011.
- [51] Vasilis Pappas, Fernando Krell, Binh Vo, Vladimir Kolesnikov, Tal Malkin, Seung Geol Choi, Wesley George, Angelos D. Keromytis, and Steven M. Bellovin. Blind seer: A scalable private DBMS. In *IEEE S&P*, 2014.

- [52] Mohammad Etemad, Alptekin Küpçü, Charalampos Papamanthou, and David Evans. Efficient Dynamic Searchable Encryption with Forward Privacy. *PoPETs*, 2018.
- [53] Dagmar Stumpfe and Jürgen Bajorath. Similarity searching. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 2011.
- [54] Laufkötter, Oliver and Miyao, Tomoyuki and Bajorath, Jürgen. Large-Scale Comparison of Alternative Similarity Search Strategies with Varying Chemical Information Contents. *ACS Omega*, 2019.
- [55] Kana Shimizu, Koji Nuida, Hiromi Arai, Shigeo Mitsunari, Nuttapong Attrapadung, Michiaki Hamada, Koji Tsuda, Takatsugu Hirokawa, Jun Sakuma, Goichiro Hanaoka, et al. Privacy-preserving search for chemical compound databases. *BMC bioinformatics*, 2015.
- [56] Kasra Edalatnejad, Wouter Lueks, Julien Pierre Martin, Soline Ledésert, Anne L’Hôte, Bruno Thomas, Laurent Girod, and Carmela Troncoso. DatashareNetwork: A Decentralized Privacy-Preserving Search Engine for Investigative Journalists. In *USENIX*, 2020.
- [57] The fork: Restaurant booking system. <https://www.thefork.com/>, .
- [58] Strava rolls out significant new routes feature. <https://www.dcrainmaker.com/2020/03/strava-rolls-out-significant-new-routes-feature.html>, .
- [59] Tinder: An online dating application. <https://tinder.com/>, .
- [60] Emiliano De Cristofaro and Gene Tsudik. Practical Private Set Intersection Protocols with Linear Computational and Bandwidth Complexity. *IACR Cryptol. ePrint Arch.*, 2009.
- [61] Emiliano De Cristofaro, Jihye Kim, and Gene Tsudik. Linear-Complexity Private Set Intersection Protocols Secure in Malicious Model. In *ASIACRYPT*, 2010.
- [62] Benny Pinkas, Thomas Schneider, Oleksandr Tkachenko, and Avishay Yanai. Efficient Circuit-Based PSI with Linear Communication. In *EUROCRYPT*, 2019.
- [63] Lea Kissner and Dawn Xiaodong Song. Privacy-preserving set operations. In *CRYPTO*, 2005.

- [64] Ling Xue, Florence L Stahura, Jeffrey W Godden, and Jürgen Bajorath. Mini-fingerprints detect similar activity of receptor ligands previously recognized only by three-dimensional pharmacophore-based methods. *Journal of chemical information and computer sciences*, 2001.
- [65] Peter Willett, John M Barnard, and Geoffrey M Downs. Chemical similarity searching. *Journal of chemical information and computer sciences*, 1998.
- [66] Adrià Cereto-Massagué, María José Ojeda, Cristina Valls, Miquel Mulero, Santiago Garcia-Vallvé, and Gerard Pujadas. Molecular fingerprint similarity search in virtual screening. *Methods*, 2015.
- [67] Ingo Muegge and Prasenjit Mukherjee. An overview of molecular fingerprint similarity search in virtual screening. *Expert Opinion on Drug Discovery*, 2016.
- [68] Amos Tversky. Features of similarity. *Psychological review*, 1977.
- [69] Paul Jaccard. The distribution of the flora in the alpine zone. *New phytologist*, 1912.
- [70] Okcupid: Free online dating. <https://www.okcupid.com/>,.
- [71] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. Efficient Batched Oblivious PRF with Applications to Private Set Intersection. In *CCS*, 2016.
- [72] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. Spotlight: Lightweight private set intersection from sparse OT extension. In *CRYPTO*, 2019.
- [73] Mike Rosulek and Ni Trieu. Compact and malicious private set intersection for small sets. In *CCS*, 2021.
- [74] Carmit Hazay. Oblivious Polynomial Evaluation and Secure Set-Intersection from Algebraic PRFs. In *TCC*, 2015.
- [75] Dana Dachman-Soled, Tal Malkin, Mariana Raykova, and Moti Yung. Efficient robust private set intersection. In *ACNS*, 2009.
- [76] Hao Chen, Kim Laine, and Peter Rindal. Fast Private Set Intersection from Homomorphic Encryption. In *CCS*, 2017.
- [77] Daniel Kales, Christian Rechberger, Thomas Schneider, Matthias Senker, and Christian Weinert. Mobile Private Contact Discovery at Scale. In *USENIX*, 2019.



- [78] Benny Pinkas, Thomas Schneider, Nigel P. Smart, and Stephen C. Williams. Secure two-party computation is practical. In *ASIACRYPT*, 2009.
- [79] Ferhat Karakoç and Alptekin Küpçü. Linear complexity private set intersection for secure two-party protocols. In *CANS*, 2020.
- [80] Michele Ciampi and Claudio Orlandi. Combining Private Set-Intersection with Secure Two-Party Computation. In *SCN*, 2018.
- [81] Nishanth Chandran, Divya Gupta, and Akash Shah. Circuit-psi with linear complexity via relaxed batch OPRF. *PoPETs*, 2022.
- [82] Peter Rindal and Phillipp Schoppmann. VOLE-PSI: fast OPRF and circuit-psi from vector-ole. In *EUROCRYPT*, 2021.
- [83] Jack P. K. Ma and Sherman S. M. Chow. Secure-computation-friendly private set intersection from oblivious compact graph evaluation. In *AsiaCCS*, 2022.
- [84] Jason H. M. Ying, Shuwei Cao, Geong Sen Poh, Jia Xu, and Hoon Wei Lim. Psi-stats: Private set intersection protocols supporting secure statistical functions. In *ACNS*, 2022.
- [85] Satrajit Ghosh and Mark Simkin. The Communication Complexity of Threshold Private Set Intersection. In *CRYPTO*, 2019.
- [86] Mihaela Ion, Ben Kreuter, Erhan Nergiz, Sarvar Patel, Shobhit Saxena, Karn Seth, David Shanahan, and Moti Yung. Private Intersection-Sum Protocol with Applications to Attributing Aggregate Ad Conversions. *IACR Cryptol. ePrint Arch.*, 2017.
- [87] Yongjun Zhao and Sherman S. M. Chow. Are you the one to share? secret transfer with access structure. *PoPETs*, 2017.
- [88] Shishir Nagaraja, Prateek Mittal, Chi-Yao Hong, Matthew Caesar, and Nikita Borisov. BotGrep: Finding P2P Bots with Structured Graph Analysis. In *USENIX*, 2010.
- [89] Daniel Demmler, Peter Rindal, Mike Rosulek, and Ni Trieu. PIR-PSI: Scaling Private Contact Discovery. *PoPETs*, 2018.
- [90] Marcin Nagy, Emiliano De Cristofaro, Alexandra Dmitrienko, N. Asokan, and Ahmad-Reza Sadeghi. Do I Know You? Efficient and Privacy-Preserving Common Friend-Finder Protocols and Applications. In *ACSAC*, 2013.

- [91] Pierre Baldi, Roberta Baronio, Emiliano De Cristofaro, Paolo Gasti, and Gene Tsudik. Countering GATTACA: efficient and secure testing of fully-sequenced human genomes. In *CCS*, 2011.
- [92] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. PSI from PaXoS: Fast, Malicious Private Set Intersection. In *EUROCRYPT*, 2020.
- [93] Seny Kamara, Payman Mohassel, Mariana Raykova, and Saeed Sadeghian. Scaling Private Set Intersection to Billion-Element Sets. In *FC*, 2013.
- [94] Ronald L Rivest, Len Adleman, and Michael L Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation*, 1978.
- [95] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *STOC*, 1990.
- [96] Andrew Chi-Chih Yao. How to Generate and Exchange Secrets (Extended Abstract). In *FOCS*, 1986.
- [97] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. In *STOC*, 1987.
- [98] Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. In *EUROCRYPT*, 2015.
- [99] Rishabh Poddar, Tobias Boelter, and Raluca Ada Popa. Arx: An encrypted database using semantically secure encryption. *Proc. VLDB Endow.*, 2019.
- [100] Emil Stefanov, Marten van Dijk, Elaine Shi, T.-H. Hubert Chan, Christopher W. Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path ORAM: an extremely simple oblivious RAM protocol. *J. ACM*, 2018.
- [101] Matteo Maffei, Giulio Malavolta, Manuel Reinert, and Dominique Schröder. Maliciously secure multi-client ORAM. In *ACNS*, 2017.
- [102] Anrin Chakraborti and Radu Sion. Concuroram: High-throughput stateless parallel multi-client ORAM. In *NDSS*, 2019.
- [103] Aslı Bay, Zeki Erkin, Mina Alishahi, and Jelle Vos. Multi-party private set intersection protocols for practical applications. In *SECRYPT*, 2021.

- [104] Zhusheng Wang, Karim Banawan, and Sennur Ulukus. Multi-party private set intersection: An information-theoretic approach. *IEEE J. Sel. Areas Inf. Theory*, 2021.
- [105] Vladimir Kolesnikov, Naor Matania, Benny Pinkas, Mike Rosulek, and Ni Trieu. Practical multi-party private set intersection from symmetric-key techniques. In *CCS*, 2017.
- [106] Junfeng Fan and Frederik Vercauteren. Somewhat Practical Fully Homomorphic Encryption. *IACR Cryptol. ePrint Arch.*, 2012.
- [107] Martin Burkhart, Mario Strasser, Dilip Many, and Xenofontas A. Dimitropoulos. SEPIA: privacy-preserving aggregation of multi-domain network events and statistics. In *USENIX*, 2010.
- [108] Sumit Kumar Debnath and Ratna Dutta. Secure and Efficient Private Set Intersection Cardinality Using Bloom Filter. In *ISC*, 2015.
- [109] Ou Ruan, Zihao Wang, Jing Mi, and Mingwu Zhang. New approach to set representation and practical private set-intersection protocols. *IEEE Access*, 2019.
- [110] Giuseppe Ateniese, Emiliano De Cristofaro, and Gene Tsudik. (If) Size Matters: Size-Hiding Private Set Intersection. In *PKC*, 2011.
- [111] Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, Satya Lokam, Daniele Micciancio, Dustin Moody, Travis Morrison, Amit Sahai, and Vinod Vaikuntanathan. Homomorphic Encryption Security Standard. Technical report, HomomorphicEncryption.org, 2018.
- [112] Lattigo. <http://github.com/ldsec/lattigo>, 2020.
- [113] Christian Mouchet, Juan Ramón Troncoso-Pastoriza, and Jean-Pierre Hubaux. Multiparty Homomorphic Encryption: From Theory to Practice. *IACR Cryptol. ePrint Arch.*, 2020.
- [114] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic SIMD operations. *Des. Codes Cryptogr.*, 2014.
- [115] Durant, Joseph L and Leland, Burton A and Henry, Douglas R and Nourse, James G. Reoptimization of MDL keys for use in drug discovery. *Journal of chemical information and computer sciences*, 2002.

- [116] David Mendez, Anna Gaulton, A Patrícia Bento, Jon Chambers, Marleen De Veij, Eloy Félix, María Paula Magariños, Juan F Mosquera, Prudence Mutowo, Michał Nowotka, et al. ChEMBL: towards direct deposition of bioassay data. *Nucleic acids research*, 2019.
- [117] Gaulton, Anna and Bellis, Louisa J and Bento, A Patricia and Chambers, Jon and Davies, Mark and Hersey, Anne and Light, Yvonne and McGlinchey, Shaun and Michalovich, David and Al-Lazikani, Bissan and others. ChEMBL: a large-scale bioactivity database for drug discovery. *Nucleic acids research*, 2012.
- [118] RDKit: Open-source cheminformatics. <http://www.rdkit.org>, .
- [119] Xiao Wang, Alex J. Malozemoff, and Jonathan Katz. EMP-toolkit: Efficient MultiParty computation toolkit. <https://github.com/emp-toolkit>, 2016.
- [120] Boya Wang, Wouter Lueks, Justinas Sukaitis, Vincent Graf Narbel, and Carmela Troncoso. Not yet another digital ID: privacy-preserving humanitarian aid distribution. *IEEE S&P*, 2023.
- [121] The fundamental principles of the international red cross and red crescent movement. [https://www.icrc.org/sites/default/files/topic/file\\_plus\\_list/4046-the\\_fundamental\\_principles\\_of\\_the\\_international\\_red\\_cross\\_and\\_red\\_crescent\\_movement.pdf](https://www.icrc.org/sites/default/files/topic/file_plus_list/4046-the_fundamental_principles_of_the_international_red_cross_and_red_crescent_movement.pdf).
- [122] Human Rights Watch. New Evidence that Biometric Data Systems Imperil Afghans. <https://www.hrw.org/news/2022/03/30/new-evidence-biometric-data-systems-imperil-afghans>.
- [123] Zara Rahman, Paola Verhaert, and Carly Nyst. Biometrics in the humanitarian sector. 2018.
- [124] Hachim El Khiyari and Harry Wechsler. Face verification subject to varying (age, ethnicity, and gender) demographics using deep learning. *Journal of Biometrics and Biostatistics*, 7(323):11, 2016.
- [125] Arun Ross, Sudipta Banerjee, and Anurag Chowdhury. Deducing health cues from biometric data. *Computer Vision and Image Understanding*, 221, 2022.
- [126] Benjamin Tams. Unlinkable minutiae-based fuzzy vault for multiple fingerprints. *IET Biom.*, 2016.
- [127] Ari Juels and Martin Wattenberg. A fuzzy commitment scheme. In *CCS*, 1999.

- [128] Yadigar N. Imamverdiyev, Andrew Beng Jin Teoh, and Jaihie Kim. Biometric cryptosystem based on discretized fingerprint texture descriptors. *Expert Syst. Appl.*, 2013.
- [129] Mahesh Kumar Morampudi, Munaga V. N. K. Prasad, and U. S. N. Raju. Privacy-preserving iris authentication using fully homomorphic encryption. *Multim. Tools Appl.*, 2020.
- [130] Jiawei Yuan and Shucheng Yu. Efficient privacy-preserving biometric identification in cloud computing. In *INFOCOM*, 2013.
- [131] Julien Bringer, Hervé Chabanne, and Alain Patey. Privacy-preserving biometric identification using secure multiparty computation: An overview and recent trends. *IEEE Signal Process. Mag.*, 2013.
- [132] Liehuang Zhu, Chuan Zhang, Chang Xu, Ximeng Liu, and Cheng Huang. An efficient and privacy-preserving biometric identification scheme in cloud computing. *IEEE Access*, 2018.
- [133] Stevens Le Blond, Alejandro Cuevas, Juan Ramón Troncoso-Pastoriza, Philipp Jovanovic, Bryan Ford, and Jean-Pierre Hubaux. On enforcing the digital immunity of a large humanitarian organization. In *IEEE S&P*, 2018.
- [134] The International Committee of Red Cross. *Handbook on Data Protection in Humanitarian Action*. ICRC, 2020. <https://www.icrc.org/en/publication/430501-handbook-data-protection-humanitarian-action-second-edition>.
- [135] The International Committee of Red Cross. *Policy on the Processing of Biometric Data*. ICRC, 2019. <https://www.icrc.org/en/document/icrc-biometrics-policy>.
- [136] Cyber attack on icrc: What we know. <https://www.icrc.org/en/document/cyber-attack-icrc-what-we-know>.
- [137] Belkis Wille. You don't need to demand sensitive biometric data to give aid. The Ukraine response shows how. <https://www.thenewhumanitarian.org/opinion/2023/07/11/you-dont-need-demand-sensitive-biometric-data-give-aid-ukraine-response-shows>.
- [138] The missing persons digital matching project: Faster and better answers. Online: <https://missingpersons.icrc.org/news-stories/missing-persons-digital-matching-project-faster-and-better-answers>, 2023.

- [139] United Nations High Commissioner for Refugees. Biometric Identity Management System. <https://www.unhcr.org/media/biometric-identity-management-system>.
- [140] World Food Program. WFP SCOPE: Know them better, to serve them better. <https://documents.wfp.org/stellent/groups/public/documents/communications/wfp258555.pdf>.
- [141] Human Rights Watch. UN Shared Rohingya Data Without Informed Consent. <https://www.hrw.org/news/2021/06/15/un-shared-rohingya-data-without-informed-consent>.
- [142] Paul Currion. Eyes wide shut: The challenge of humanitarian biometrics. <https://www.thenewhumanitarian.org/opinion/2015/08/26/eyes-wide-shut-challenge-humanitarian-biometrics>.
- [143] IRIN. How a fingerprint can change an asylum seeker's life. <https://www.refworld.org/docid/5a1694724.html>.
- [144] Julien Bringer, Hervé Chabanne, Mélanie Favre, Alain Patey, Thomas Schneider, and Michael Zohner. GSHADE: faster privacy-preserving distance computation and biometric identification. In *IH&MMSec*, 2014.
- [145] Ying Luo, Sen-ching Samson Cheung, Tommaso Pignata, Riccardo Lazzeretti, and Mauro Barni. An efficient protocol for private iris-code matching by means of garbled circuits. In *ICIP*, 2012.
- [146] Yan Huang, Lior Malka, David Evans, and Jonathan Katz. Efficient privacy-preserving biometric identification. In *NDSS*, 2011.
- [147] Mauro Barni, Tiziano Bianchi, Dario Catalano, Mario Di Raimondo, Ruggero Donida Labati, Pierluigi Failla, Dario Fiore, Riccardo Lazzeretti, Vincenzo Piuri, Fabio Scotti, and Alessandro Piva. Privacy-preserving fingercodes authentication. In *MM&Sec*. ACM, 2010.
- [148] Masaya Yasuda, Takeshi Shimoyama, Jun Kogure, Kazuhiro Yokoyama, and Takeshi Koshiba. New packing method in somewhat homomorphic encryption and its applications. *Secur. Commun. Networks*, 2015.
- [149] Vishnu Naresh Boddeti. Secure face matching using fully homomorphic encryption. In *BTAS*, 2018.
- [150] Joshua J. Engelsma, Anil K. Jain, and Vishnu Naresh Boddeti. HERS: homomorphically encrypted representation search. *IEEE Trans. Biom. Behav. Identity Sci.*, 2022.

- [151] Yoongu Kim, Ross Daly, Jeremie S. Kim, Chris Fallin, Ji-Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors. In *ISCA*, 2014.
- [152] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Anders Fogh, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. Meltdown: Reading kernel memory from user space. In *USENIX*, 2018.
- [153] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F. Wenisch, Yuval Yarom, and Raoul Strackx. Foreshadow: Extracting the keys to the intel SGX kingdom with transient out-of-order execution. In *USENIX*, 2018.
- [154] Anil K. Jain, Salil Prabhakar, Lin Hong, and Sharath Pankanti. Fingercodex: A filterbank for fingerprint representation and matching. In *CVPR*. IEEE Computer Society, 1999.
- [155] Christel loic Tisse, Lionel Martin, Lionel Torres, and Michel Robert. Person identification technique using human iris recognition. In *Proc. Vision Interface*, 2002.
- [156] Jucheng Yang. Non-minutiae based fingerprint descriptor. *Biometrics*, 2011.
- [157] Jiankang Deng, Jia Guo, Jing Yang, Niannan Xue, Irene Kotsia, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2022.
- [158] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015.
- [159] Srdan Barzut, Milan Milosavljevic, Sasa Adamovic, Muzafer Saracevic, Nemanja Macek, and Milan Gnjatovic. A novel fingerprint biometric cryptosystem based on convolutional neural networks. *Mathematics*, 2021.
- [160] Anil K. Jain, Salil Prabhakar, Lin Hong, and Sharath Pankanti. Filterbank-based fingerprint matching. *IEEE Trans. Image Process.*, 2000.
- [161] Lifeng Sha, Feng Zhao, and Xiaoou Tang. Improved fingercodex for filterbank-based fingerprint matching. In *ICIP (2)*, 2003.

- [162] Joshua J. Engelsma, Kai Cao, and Anil K. Jain. Learning a fixed-length fingerprint representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2021.
- [163] John Daugman. How iris recognition works. *IEEE Trans. Circuits Syst. Video Technol.*, 2004.
- [164] Karen P. Hollingsworth, Kevin W. Bowyer, and Patrick J. Flynn. The best bits in an iris code. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008.
- [165] Unique Identification Authority of India (UIDAI). Aaadhar dashboard. <https://uidai.gov.in/en/>.
- [166] The US department of state. Safety and Security of U.S. Borders: Biometrics. <https://travel.state.gov/content/travel/en/us-visas/visa-information-resources/border-biometrics.html/>.
- [167] The Multimedia Signal Processing and Security Lab (WaveLab). University of Salzburg Iris Toolkit (USIT) v3. <https://www.wavelab.at/sources/USIT/g>.
- [168] Christian Rathgeb, Andreas Uhl, Peter Wild, and Heinz Hofbauer. Design decisions for an iris recognition sdk. *Handbook of iris recognition*, 2016.
- [169] Xiao Wang, Alex J. Malozemoff, and Jonathan Katz. EMP-toolkit: semi honest two party computation. <https://github.com/emp-toolkit/emp-sh2pc>, 2016.
- [170] EPFL-LDS, Tune Insight SA. Lattigo v4. Online: <https://github.com/tuneinsight/lattigo>, 2022.
- [171] Fortanix. Fortanix enclave development platform. <https://edp.fortanix.com/>.
- [172] Sasa Z. Adamovic, Milan Milosavljevic, Mladen D. Veinovic, Marko Sarac, and Aleksandar Jevremovic. Fuzzy commitment scheme for generation of cryptographic keys based on iris biometrics. *IET Biom.*, 2017.
- [173] Ari Juels and Madhu Sudan. A fuzzy vault scheme. *Des. Codes Cryptogr.*, 2006.
- [174] Valérie Viet Triem Tong, Hervé Sibert, Jérémy Lecoœur, and Marc Girault. Biometric fuzzy extractors made practical: A proposal based on fingercodes. In *ICB*, 2007.



- [175] Yevgeniy Dodis, Leonid Reyzin, and Adam D. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *EUROCRYPT*, 2004.
- [176] A Stoianov, Tom Kevenaar, and Michiel Van der Veen. Security issues of biometric encryption. In *2009 IEEE Toronto International Conference Science and Technology for Humanity (TIC-STH)*, 2009.
- [177] Alex Stoianov. Security of error correcting code for biometric encryption. In *PST*. IEEE, 2010.
- [178] Bashra Alwawi and Ali Althabhawee. Towards more accurate and efficient human iris recognition model using deep learning technology. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, 2022.
- [179] Christian Rathgeb and Andreas Uhl. A survey on biometric cryptosystems and cancelable biometrics. *EURASIP J. Inf. Secur.*, 2011.
- [180] Andrew Teoh Beng Jin, David Ngo Chek Ling, and Alwyn Goh. Biohashing: two factor authentication featuring fingerprint data and tokenised random number. *Pattern Recognit.*, 2004.
- [181] Alessandra Lumini and Loris Nanni. An improved biohashing for human authentication. *Pattern Recognit.*, 2007.
- [182] *Project/programme monitoring and evaluation guide*. 2011. <https://www.ifrc.org/sites/default/files/2021-09/IFRC-ME-Guide-8-2011.pdf>.
- [183] Aeronautical Radio Inc. (ARINC). Aircraft Communications Addressing and Reporting Systems (ACARS). Technical report, Aeronautical Radio Inc. (ARINC), 2012.
- [184] Aeronautical Radio Inc. (ARINC). DataLink Security, Part 1 - ACARS Message Security. Technical report, Aeronautical Radio Inc. (ARINC), 2007.
- [185] Matthew Smith, Daniel Moser, Martin Strohmeier, Vincent Lenders, and Ivan Martinovic. Economy Class Crypto: Exploring Weak Cipher Usage in Avionic Communications via ACARS. In *FC*, 2017.
- [186] Matthew Smith, Daniel Moser, Martin Strohmeier, Vincent Lenders, and Ivan Martinovic. Undermining privacy in the aircraft communications addressing and reporting system (ACARS). *PoPETs*, 2018.

- [187] Aeronautical Radio Inc. (ARINC). Datalink Ground System Standard and Interface Specification. Technical report, Aeronautical Radio Inc. (ARINC), 2014.
- [188] Martin Strohmeier, Ivan Martinovic, and Vincent Lenders. Securing the air-ground link in aviation. In *The Security of Critical Infrastructures*. 2020.
- [189] Bruno Blanchet. Symbolic and computational mechanized verification of the arinc823 avionic protocols. In *CSF*, 2017.
- [190] Friedrich Wilhelm Kasiski. *Die Geheimschriften und die Dechiffirkunst: mit besonderer Berücksichtigung der deutschen und der französischen Sprache*. 1863.
- [191] William Frederick Friedman. *The index of coincidence and its applications in cryptography*. 1922.
- [192] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. 2014.
- [193] Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 1948.
- [194] William F. Smyth and Reihaneh Safavi-Naini. Automated Cryptanalysis of Substitution Ciphers. *Cryptologia*, 1993.
- [195] Jian Chen and Jeffrey S. Rosenthal. Decrypting classical cipher text using Markov chain Monte Carlo. *Stat. Comput.*, 2012.
- [196] Noor R. Al-Kazaz, Sean A. Irvine, and William John Teahan. An Automatic Cryptanalysis of Transposition Ciphers Using Compression. In *CANS*, 2016.
- [197] Noor R. Al-Kazaz, Sean A. Irvine, and William John Teahan. An Automatic Cryptanalysis of Playfair Ciphers Using Compression. In *His-toCrypt*, 2018.
- [198] Noor R. Al-Kazaz, Sean A. Irvine, and William John Teahan. An automatic cryptanalysis of simple substitution ciphers using compression. *Inf. Secur. J. A Glob. Perspect.*, 2018.
- [199] Al-Kazaz, Noor R. *Compression-based Methods for the Automatic Cryptanalysis of Classical Ciphers*. 2019.
- [200] Guang Cheng and Ying Hu. Encrypted traffic identification based on n-gram entropy and cumulative sum test. In *CFI*, 2018.

- [201] Zhengzhi Tang, Xuewen Zeng, and Yiqiang Sheng. Entropy-based feature extraction algorithm for encrypted and non-encrypted compressed traffic classification. *International Journal of ICIC*, 2019.
- [202] Qing Li, Yonghui Ju, Chang Zhao, and Xintai He. An Encrypted Field Locating Algorithm for Private Protocol Data Based on Data Reconstruction and Moment Eigenvector. *IEEE Access*, 2021.
- [203] Petr Velan, Milan Cermák, Pavel Celeda, and Martin Drasar. A survey of methods for encrypted traffic classification and analysis. *Int. J. Netw. Manag.*, 2015.
- [204] Joel Reardon, Álvaro Feal, Primal Wijesekera, Amit Elazari Bar On, Narseo Vallina-Rodriguez, and Serge Egelman. 50 ways to leak your data: An exploration of apps' circumvention of the android permissions system. In *USENIX*, 2019.
- [205] Georg Baselt, Martin Strohmeier, James Pavur, Vincent Lenders, and Ivan Martinovic. Security and privacy issues of satellite communication in the avlatlon domain. In *2022 14th International Conference on Cyber Conflict: Keep Moving! (CyCon)*, 2022.
- [206] Martin Strohmeier, Matthew Smith, Vincent Lenders, and Ivan Martinovic. The real first class? inferring confidential corporate mergers and government relations from air traffic communication. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, 2018.
- [207] Elaine Rich. *Automata, computability and complexity: theory and applications*. 2008.
- [208] John G. Cleary and Ian H. Witten. Data Compression Using Adaptive Coding and Partial String Matching. *IEEE Trans. Commun.*, 1984.
- [209] Ian H. Witten, Radford M. Neal, and John G. Cleary. Arithmetic Coding for Data Compression. *Commun. ACM*, 1987.
- [210] Project nayuki: Reference arithmetic coding. <https://github.com/nayuki/Reference-arithmetic-coding>.
- [211] Thorsten Joachims. Text Categorization with Support Vector Machines. *Proc. European Conf. Machine Learning*, 1998.
- [212] Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems*, 2013.

- [213] Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level Convolutional Networks for Text Classification. In *Advances in Neural Information Processing Systems*, 2015.
- [214] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NeurIPS workshop*, 2011.
- [215] Opensky’s aircraft registry. <https://opensky-network.org/datasets/metadata/>.
- [216] Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E. Scikit-learn: Machine Learning in Python. *JLMR*, 2011.
- [217] Canterbury corpus. <https://corpus.canterbury.ac.nz/>.
- [218] Tao Chen and Min-Yen Kan. Creating a live, public short message service corpus: the NUS SMS corpus. *Lang. Resour. Evaluation*, 2013.
- [219] Acars data collection infrastructure. <https://thebaldgeek.github.io/>.
- [220] Cnil-inria data protection award. <https://www.cnil.fr/en/2021-inria-and-cnil-award-european-researchers-awarded-their-work-privacy-protection>.
- [221] The caspar bowden award for outstanding research in privacy enhancing technologies. <https://petsymposium.org/award/winners.php>.
- [222] Real world crypto symposium. <https://rwc.iacr.org/2023/>, 2023.
- [223] Leo de Castro, Chiraag Juvekar, and Vinod Vaikuntanathan. Fast Vector Oblivious Linear Evaluation from Ring Learning with Errors. In *WAHC@CCS*, 2021.
- [224] Microsoft SEAL (release 4.0). <https://github.com/Microsoft/SEAL>, 2022.
- [225] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, USA, 2009.
- [226] Yehuda Lindell. How to Simulate It - A Tutorial on the Simulation Proof Technique. In *Tutorials on the Foundations of Cryptography*. 2017.
- [227] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inf. Theory*, 1985.

- [228] Pascal Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *EUROCRYPT*, 1999.



# Kasra EdalatNejad

 [kasra.edalat.dev](https://kasra.edalat.dev)  
 [kasra.edalat@epfl.ch](mailto:kasra.edalat@epfl.ch)  
 [keybase.io/kasraedalat](https://keybase.io/kasraedalat)  
 [github.com/kasra-edalat](https://github.com/kasra-edalat)  
 [kasra-edalatnejad](https://kasra-edalatnejad)

## EDUCATION

---

**École polytechnique fédérale de Lausanne** Lausanne, Switzerland  
Ph.D. in Computer Science, Advisor: Prof. Carmela Troncoso 2017–Current  
– Thesis title: *“Bridging the gap between theoretical and practical privacy technologies for at-risk populations”*

**Sharif University of Technology** Tehran, Iran  
B.S. in Computer Engineering - Software Engineering 2011–2016

**Salam Tajrish high school** Tehran, Iran  
Diploma in mathematics and physics 2008–2011

## EXPERIENCE

---

**IMDEA Software Research Institute** Madrid, Spain  
Research intern, Advisor: Prof. Dario Fiore, Dr. Claudio Soriente July 2021–Nov 2021  
– Topic: *“Speeding up privacy-preserving deep learning by enhancing FHE operations with TEE and GPU”*

**University of Michigan** Ann Arbor, Michigan  
Research assistant, Advisor: Prof. Harsha V. Madhyastha 2016–2017  
– Topic: *“Privacy-preserving recommendation sharing”*

**Atomic Energy high school** Tehran, Iran  
Teacher: Combinatory and programming 2011–2015

**Salam Tajrish high school** Tehran, Iran  
Teacher: Algorithm, combinatory, and programming 2011–2012

## PUBLICATIONS

---

1. **Kasra EdalatNejad**, Mathilde Raynal, Wouter Lueks, Carmela Troncoso: *“Private Collection Matching Protocols”*. Proceedings on Privacy Enhancing Technologies, 2023.
2. **Kasra EdalatNejad**, Wouter Lueks, Julien Pierre Martin, Soline Ledéser, Anne L’Hôte, Bruno Thomas, Laurent Girod, Carmela Troncoso: *“DatashareNetwork: A Decentralized Privacy-Preserving Search Engine for Investigative Journalists”*. USENIX Security Symposium 2020.
3. Manu Drijvers, **Kasra EdalatNejad**, Bryan Ford, Eike Kiltz, Julian Loss, Gregory Neven, Igors Stepanovs: *“On the Security of Two-Round Multi-Signatures”*. in IEEE Symposium on Security and Privacy (IEEE S&P) 2019.
4. Manu Drijvers, **Kasra EdalatNejad**, Bryan Ford, Gregory Neven: *“Okamoto Beats Schnorr: On the Provable Security of Multi-Signatures”*. IACR Cryptol. ePrint Arch. 2018: 417 (2018)

- Han Zhang, **Kasra EdalatNejad**, Amir Rahmati, Harsha V. Madhyastha: *“Towards Comprehensive Repositories of Opinions”*. HotNets 2016.

### Contributed talks

- Kasra EdalatNejad**: *“DatashareNetwork: A Decentralized Privacy-Preserving Search Engine for Investigative Journalists”*. Real world crypto (RWC), 2023.
- Kasra EdalatNejad**: *“Automatic detection and decryption of insecure ciphertexts in ACARS”*. Cyber Alp Retreat, 2021.
- Kasra EdalatNejad**, Soline Ledésert: *“DatashareNetwork: The use of Tor in supporting document search for Investigative Journalists”*. Tor Demo Day, September 2020.











## HONORS AND AWARDS

---

- Runner up for CNIL-Inria data protection award 2022
- Runner up for the Caspar Bowden award for outstanding research in privacy enhancing technologies 2021
- EPFL IC distinguished service award 2020
- EPFL teaching assistance award 2019
- EPFL EDIC fellowship for doctoral studies 2017
- **Ministerial award** from ministry of science, research, and technology for outstanding performance in olympiad 2017
- **Gold medal** (Ranked 1st) in the national computer science olympiad for graduate students 2016
- Ranked 6th in the Iranian national graduate examination (Konkur) 2016
- Ranked 23rd (top 1%) in the 9th IEEEExtreme Oct 2015
- **Presidential award** from **Pres. M. Ahmadinejad** for outstanding performance in olympiad 2011
- **International Silver medal** in 23th international olympiad in informatics (IOI), Thailand July 2011
- **Silver medal** in 5th Asia-Pacific informatics olympiad (APIO) May 2011
- **Ministerial award** from ministry of science, research, and technology for outstanding performance in olympiad 2010
- National **Gold medal** (ranked 1st) in Iranian national olympiad in informatics (INOI) Sep 2010
- National **Silver medal** in Iranian national olympiad in informatics (INOI) March 2010
- Member of Iranian national elite foundation since 2011

## PROJECTS

---

- **PCM**: A framework for private collection matching  Repo
- **DatashareNetwork**: A decentralized privacy-preserving search engine
  - Proof of concept for cryptographic primitives  crypto
  - Production code  core,  server,  client
  - For more information visit  kasra.edalat.dev
- **SSCred**: Single show anonymous credentials
  - Source code  Repo
  - Pypi package  sscred
- **KSSH**: An SSH server as a plugin for Sharif HoneyPot  Repo
- **KilliCent**: A micropayment system based on MilliCent  Repo



## SKILLS

---

### Programming languages

- Proficient in: C++, Python, Go, Java, Android, JavaScript
- Familiar with: C, Lisp, Verilog, Haskell, Racket

## TEACHING AND SUPERVISION

---

### Student Supervision

- Pierugo Pace, Master project: “Biometric deduplication in humanitarian aid distribution”. 2023
- Eva Luvison, Master project: “Wallets for Privacy-Preserving Aid Distribution”. 2022
- Lorenzo Carlo Rovati, Master project: “Using smartcards for privacy-friendly aid distribution”. 2022
- Jodok Vieli, Master project: “Automatic decryption of classical ciphers with neural networks”. 2021
- Ma Ke, Master project: “Securing biometric authentication data with trusted hardware”. 2021
- Sacha Kozma, Master project: “Efficiently updatable accumulator”. 2020
- Omid Karimi, Bachelor project: “Automatic cryptanalysis of classical ciphers”. 2020
- Mathilde Raynal, Master project: “Secure Multiparty Computation for PSI”. 2019
- Bradley Mathez, Bachelor project: “An SMC Approach to Multi-Device Key Management”. 2019
- Valentyna Pavliv, Bachelor project: “Efficient Blacklisting of Anonymous Users”. 2019

### Teaching assistant

- CS-523: Advanced topics on privacy enhancing technologies 2020, 2021
- COM-301: Computer security 2018, 2019, 2020
- COM-412: Software security 2019
- MATH-232: Probability and statistics 2018

## ACADEMIC SERVICE

---

### Program committee

- IEEE International Conference on Blockchain and Cryptocurrency (ICBC) 2020
- IEEE International Conference on Blockchain and Cryptocurrency (ICBC) 2019

### External reviewer

- PoPETS 2023
- PoPETS 2022
- EuroCrypt, USENIX Security, OSDI, IEEE Trans. on Information Forensics and Security 2021
- NSDI 2020
- PoPETS, FC 2019