

Privacy–Preserving Data Discovery in Distributed Health Networks with Dynamic Consent

Master Thesis

Michael Herwig

The present work was submitted to the
Chair of Communication and Distributed Systems
RWTH Aachen University, Germany

Advisor(s):

Johannes Lohmöller, M. Sc.

Examiners:

Prof. Dr.-Ing. Klaus Wehrle
Priv.-Doz. Dr. rer. nat. Walter Unger

Registration date: January 17, 2023

Submission date: July 17, 2023

Eidesstattliche Versicherung Statutory Declaration in Lieu of an Oath

Herwig, Michael

Name, Vorname/Last Name, First Name

301 999

Matrikelnummer (freiwillige Angabe)

Matriculation No. (optional)

Ich versichere hiermit an Eides Statt, dass ich die vorliegende ~~Arbeit/Bachelorarbeit/~~
Masterarbeit* mit dem Titel

I hereby declare in lieu of an oath that I have completed the present ~~paper/Bachelor thesis/~~ Master thesis* entitled

Privacy-Preserving Data Discovery

in Distributed Health Networks

with Dynamic Consent

selbstständig und ohne unzulässige fremde Hilfe (insbes. akademisches Ghostwriting) erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

independently and without illegitimate assistance from third parties (such as academic ghostwriters). I have used no other than the specified sources and aids. In case that the thesis is additionally submitted in an electronic format, I declare that the written and electronic versions are fully identical. The thesis has not been submitted to any examination body in this, or similar, form.

Aachen, 17.07.2023

Ort, Datum/City, Date

Unterschrift/Signature

*Nichtzutreffendes bitte streichen

*Please delete as appropriate

Belehrung:

Official Notification:

§ 156 StGB: Falsche Versicherung an Eides Statt

Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

Para. 156 StGB (German Criminal Code): False Statutory Declarations

Whoever before a public authority competent to administer statutory declarations falsely makes such a declaration or falsely testifies while referring to such a declaration shall be liable to imprisonment not exceeding three years or a fine.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt

(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.

(2) Strafflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Para. 161 StGB (German Criminal Code): False Statutory Declarations Due to Negligence

(1) If a person commits one of the offences listed in sections 154 through 156 negligently the penalty shall be imprisonment not exceeding one year or a fine.

(2) The offender shall be exempt from liability if he or she corrects their false testimony in time. The provisions of section 158 (2) and (3) shall apply accordingly.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

I have read and understood the above official notification:

Aachen, 17.07.2023

Ort, Datum/City, Date

Unterschrift/Signature

Abstract

Electronic health data is becoming increasingly important for clinical research and medical treatment, with more use cases emerging to improve public health. The quantity of data significantly impacts its usefulness in these applications, resulting in the necessity of sharing health data across facilities and international boundaries. However, sharing sensitive health information risks patients' privacy due to increasing hacking incidents and data leakages. In addition, governmental regulations impede the process of organizational cooperation to build larger datasets and prohibit the reuse of existing data for evolving use cases. Recent research on dynamic consent has shown that the active involvement of patients fulfills the requirement of informed consent and positively impacts patients' willingness to contribute their data. In this thesis, we adopt state-of-the-art techniques to develop a new distributed health data network for data discovery with dynamic consent that builds upon result aggregation rather than data aggregation to eliminate the need to share patients' health data. The evaluation of our test system shows that we can scale the network to hundreds of institutions and process datasets with more than ten million records within thirty seconds.

Kurzfassung

Elektronische Gesundheitsdaten werden für die klinische Forschung und die medizinische Behandlung immer wichtiger, und es entstehen immer mehr Anwendungsfälle zur Verbesserung der öffentlichen Gesundheitspflege. Die Menge der Daten hat erhebliche Auswirkungen auf ihren Nutzen, was dazu führt, dass Gesundheitsdaten über Einrichtungen und internationale Grenzen hinweg ausgetauscht werden müssen. Die gemeinsame Nutzung sensibler Gesundheitsdaten ist jedoch mit einem großen Risiko für die Privatsphäre der Patienten verbunden, da die Anzahl der Hackerangriffe und Datenlecks zunimmt. Darüber hinaus behindern staatliche Vorschriften den Prozess der organisatorischen Zusammenarbeit zum Aufbau größerer Datensätze und verbieten die Wiederverwendung vorhandener Daten für neue Anwendungsfälle. Jüngste Forschungen im Bereich 'Dynamic Consent' haben gezeigt, dass die aktive Beteiligung der Patienten die Anforderung der informierten Einwilligung erfüllt und sich positiv auf die Bereitschaft von Patienten auswirkt, ihre Daten zur Verfügung zu stellen. In dieser Arbeit setzen wir modernste Techniken ein, um ein neues verteiltes Netzwerk von Gesundheitsdaten mit 'Dynamic Consent' zu entwickeln, das auf der Aggregation von Ergebnissen statt auf der Aggregation von Daten beruht, um die Notwendigkeit der Weitergabe von Gesundheitsdaten der Patienten zu beseitigen. Die Auswertung unseres Testsystems zeigt, dass wir das Netzwerk auf Hunderte von Einrichtungen skalieren und Datensätze mit mehr als zehn Millionen Datensätzen innerhalb von dreißig Sekunden verarbeiten können.

Acknowledgments

I am profoundly grateful to everyone who has contributed to successfully completing my master's thesis. Here, I want to express my gratitude and acknowledgments.

First, I would like to thank Prof. Dr.-Ing. Klaus Wehrle, for allowing me to work on this thesis.

I want to deeply appreciate my supervisor, Johannes Lohmöller. He supported me throughout the thesis with his expertise, advice, and guidance, even during times of high workload. His continuous feedback and criticism helped me to improve my thesis throughout my work.

My thanks go to my second examiner, Priv.-Doz. Dr. rer. nat. Walter Unger's compelling lectures have ignited my passion for computer science and motivated me to persevere in my studies. His lectures have significantly enriched my academic perspective.

I am also thankful to the entire team at COMSYS. The constructive feedback I received while writing this thesis has greatly enhanced my work. Moreover, I want to thank the entire chair for the long-lasting access to a dedicated server for evaluation purposes. Without this access, the results of my thesis would not have been possible.

Finally, I want to thank my relatives and beloved partner, who supported and encouraged me throughout this thesis.

Contents

1	Introduction	1
1.1	Contributions	2
1.2	Outline	3
2	Background	5
2.1	Cryptography	5
2.1.1	Elliptic Curves Cryptography	5
2.1.2	ElGamal – Homomorphic Encryption	8
2.1.3	Stealth Addresses	8
2.1.4	Zero-Knowledge Proofs of Discrete Logarithms	9
2.2	Public Key Encryption with Keyword Search	10
2.2.1	Attack Models	11
3	Related Work	13
3.1	Health Data Sharing	14
3.1.1	Ethics and Requirements for Health Data Sharing	14
3.1.2	Regulatory Frameworks	15
3.1.3	Dynamic Consent	16
3.1.4	Measurements and Technologies for Regulatory Compliance	17
3.2	Privacy-Preserving Queries on PHI	21
3.2.1	Owner-Centric Schemes in Multi-User Environments	21
3.2.2	Horizontal Scalability of Data Providers	23
3.2.3	Achieving Sublinear Efficiency on Queries	24
3.3	Distributed Health Data Networks	26
3.3.1	PCORnet — Privacy-Preserving Cohort Discovery	26
3.3.2	UnLynx — Decentralized Privacy-Conscious Data Sharing	29

4	Problem Statement	33
4.1	Problem Scope	33
4.2	Objectives	34
4.2.1	Functional Objectives	34
4.2.2	Non-Functional Objectives	35
4.3	Summary	35
5	System Design	37
5.1	System Model	37
5.1.1	Conceptional Idea	38
5.1.2	Schematic Overview of Actors and Tasks	40
5.1.3	Entities	41
5.1.4	Network Architecture	44
5.2	Building Blocks	45
5.2.1	Structure of Electronic Health Records	45
5.2.2	Bucketization	46
5.2.3	A suitable PEKS Scheme	48
5.2.4	Query Data Structure	52
5.2.5	Efficient Linkability of Anonymous Messages	54
5.3	Protocols	55
5.3.1	Organization Data Exchange with the Data Owner	56
5.3.2	Data Import	56
5.3.3	Query Deployment	60
5.3.4	Query Processing	65
5.3.5	Result Reception	69
6	Implementation	73
6.1	Tools and Technologies	73
6.1.1	Rust: Concurrency, Networking, and Cryptography	73
6.1.2	Frameworks	74
6.1.3	Cryptographic Primitives	76
6.2	Software Composition	77
6.2.1	Components	78

6.2.2	Binaries	80
6.3	Implementation Details	80
6.3.1	Non-Blocking Runtime for Network Communication	81
6.3.2	Database Structure and Algorithms	81
6.4	Verification and Testing	83
6.4.1	Correctness of Query Evaluation	83
7	Evaluation	85
7.1	Methodology	85
7.1.1	Testbed	85
7.1.2	Test Data and Queries	87
7.1.3	Performance Analysis	89
7.2	Performance Analysis	91
7.2.1	Query Duration in Correlation with Number of Records	91
7.2.2	Analysis of Hot Spots	94
7.2.3	Vertical and Horizontal Scaling of Test-Servers	97
7.2.4	Bandwidth Usage and Message Size	98
7.3	Scalability	100
7.3.1	Horizontal Scalability of Collective Authority Members	101
7.3.2	Horizontal Scalability of Data-Providers	102
7.3.3	Impact of Unavailability and Infrastructure Changes	103
7.3.4	A Use Case Scenario on German University Hospitals	104
7.4	Privacy and Security	105
7.5	Trade-Offs on Performance and Privacy	106
7.5.1	Tracing Overhead	107
7.5.2	Feasability of Larger Queries	107
7.5.3	Artificial Data	108
7.6	Discussion	109
7.6.1	Over the Edge of the Threat Model	110
8	Conclusion	111
8.1	Future Work	112
8.1.1	A Data Owner Interface	112
8.1.2	Structural Adoption of ICD Codes	112
8.1.3	A Custom-Made PEKS Scheme	113
8.1.4	Enhanced Query Expressiveness	113

Bibliography	115
A Appendix	133
A.1 List of Abbreviations	133
A.2 Network Messages	136
A.3 PEKS Algorithms	140
A.4 University Hospitals Data	142

1

Introduction

Health data, in the form of electronic health records (EHRs), are widely accepted to facilitate clinical research in various fields [CSK⁺13, DMTB21], with more use cases emerging [CBC⁺17]. Its quantity and quality have a significant impact on its usefulness [BHCW10, KRN⁺19]. Therefore, sharing data is essential to improve public health [WB11, SKMS13, CBP14]. Nevertheless, this comes at great risk to patients' privacy. The California data breach report [HG16] and reports of the U.S. Department of Health and Human Services [oHfCR21] show an increase of protected health informations (PHIs) affected by data breaches, with more than thirty-seven million individuals affected in 2020. Moreover, patients are aware of these risks and less willing to donate their data, especially due to marginal control of their data usage [CH12]. This threat is also reflected by the increasing amount of regulations on data sharing that institutions must fulfill by law, including the necessity of informed consent of patients for any use of their data, inducing problems in new use-cases that require reinform patients about the evolving usage of their data [SS11, SKMS13].

One health data application important to clinical research and observational studies is cohort discovery [BHCW10, FMH⁺21]. Here groups of individuals are identified and assembled based on shared characteristics. To locate cohorts and to address a hypothesis, researchers use data discovery to search for subsets of shared characteristics in large datasets that are applicable for further investigation. Therefore cohort discovery directly benefits from larger datasets [FMTP⁺20].

For the last decade, multiple systems were presented, employed, and tested to support data discovery for shared health data based on Privacy-Preserving Technologies

(PPT) in real-world use cases. Yuan et al. [YMM⁺17] published a system of a central data hub that evaluates search requests on encrypted data adopting research in the cryptographic field of asymmetric searchable encryption. This system allows providers to share data in an encrypted format, protecting their data even in a breach. In addition, their implementation utilizes advances in encrypted databases to achieve fast performance in evaluating search requests. However, stored data is still vulnerable to brute-force attacks by the data hub, and the contribution to this

system still requires us to outsource patient data in encrypted form. The UnLynx system [FES⁺17] followed another approach with a distributed setup of multiple institutions evaluating queries on encrypted data without a central hub. In their system, the data hub is replaced by a collective authority that ensures data safety in a more robust any-trust security model, further securing patients' data against offline keyword-guessing attacks. While these systems enhance the adoption of medical data sharing by strengthening the privacy and security of the data, other systems have developed that focus on compliance with policies and regulations and increase the willingness of patients to collaborate in clinical research. This research became known as dynamic consent and focused on adopting interactive interfaces, such as websites and mobile devices, to fulfill the necessity of informed consent.

From our previous brief introduction to existing research in health data sharing, we know it is possible to build distributed networks collaborating in clinical research with great security and privacy guarantees for data owners and users by utilizing PPT. Especially the UnLynx system showed that data can be processed by a collection of members rather than a central entity. However, still, institutions are required to share their data in encrypted form, inducing problems with institutional policies and regulations, thus limiting its scalability in international collaboration [CWJ⁺16]. Furthermore, patients need to be more actively involved in the usage of their data as research has shown that this solves problems by fulfilling the necessity of informed consent and increases the willingness of patients to contribute their data by offering more granular options on whom to share their data. We conclude there is a gap between research and its adoption in distributed health data networks (DHDNs), limiting the scalability and disposability of health data.

1.1 Contributions

In the context of this thesis, we tackle the challenges of collaborative research on protected health information (PHI) by developing our own distributed health network. The problems we solve include patient privacy and security issues and data sharing constraints, including its limited scalability for evolving use cases. To achieve these goals, we adapt advances in the research of distributed health data network (DHDN), encrypted databases, dynamic consent, and searchable encryption. More particularly, we achieve the following contributions:

- We combine the benefits of the DHDN architectures from Yuan et al. [YMM⁺17] and Froelicher et al. [FES⁺17] such that we can exploit the performance benefits of a central entity and encrypted databases with the security guarantees of a distributed collective authority.
- By adopting recent advances in asymmetric searchable encryption, we put more control on the patient and promote the usage of dynamic consent to involve the patient in the usage of their data, giving them the decision-making power of whom to contribute their data.
- We achieve the previous two goals in a distributed system that scales more efficiently with the number of participating institutions and is more performant than a central data hub, suitable for practical use in real-world scenarios.

By achieving our results, we enhance the privacy and security guarantees for patients and researchers, promote the contribution of more health data in various applications, including clinical research, and improve public health for our society.

1.2 Outline

To show the previous results of our contribution, we structure our thesis as follows: We introduce our notation and some background on cryptographic primitive and searchable encryption in Chapter 2. This background serves as a more technical introduction to some of the technologies we deploy that we consider as non-fundamental knowledge in cryptography and Privacy-Preserving Technologies (PPT), including the basics of asymmetric searchable encryption. Following, we give a more comprehensive view of related work and the context of this thesis in Chapter 3. Therefore, we start by sensitizing the topic of health data sharing, covering ethical principles and the resulting governmental policy frameworks that must be fulfilled. Further, we elaborate on existing technologies that solve the challenges of ethical and governmental obligations before finally reviewing the design of existing distributed health data network (DHDN) and how they deploy those technologies. After reviewing the current state of research and its deployment, we identify the research gap and determine our thesis objective in alignment with our previously set research contributions in our problem statement in Chapter 4. We propose a system design in Chapter 5 of a distributed health network with dynamic consent supporting privacy-preserving data discovery and fulfilling our objective goals. Therefore, we first introduce a schematic view of the technologies and concepts we deploy before comprehensively describing every protocol deployed, including a discussion on trade-offs and security and privacy characteristics. After that, we highlight some of our implementation details of a prototype in Chapter 6 before assessing its conformance with our set objectives in our evaluation. In Chapter 7, we show that our system is not limited to adopting new technologies to facilitate data sharing but also does so efficiently. We show the simulation of our network in a distributed system of twenty institutions on a single machine and still get competitive results compared to other DHDNs. Finally, in Chapter 8, we conclude this thesis with an overview of achieved goals and research contribution before discussing future work to improve our proposed system.

2

Background

This chapter describes the background of fundamental cryptographic primitives deployed throughout this thesis. Therefore, we start with some basic knowledge on elliptic curve cryptography (ECC), introduce our notation, and gradually advance to more specialized protocols. After that, we look into a specific field of asymmetric searchable encryption (ASE) called public key encryption with keyword search (PEKS) which is of major importance in the context of this thesis.

2.1 Cryptography

This section starts with a basic introduction to our notation for elliptic curve cryptography (ECC), followed by an overview of protocols for establishing shared secrets and generating digital signatures. We define bilinear pairings, a widely used primitive in the asymmetric searchable encryption (ASE) field. After that, we look into three larger cryptographic protocols that can be implemented using ECC. More particularly, we look into stealth addresses that enable us to link messages without revealing identities and homomorphic encryption to aggregate encrypted data. For the last protocol, we highlight discrete logarithm proofs that empower us to prove the correctness of computations involving secrets without revealing the secrets.

2.1.1 Elliptic Curves Cryptography (ECC)

ECC is a form of asymmetric encryption used in public-key cryptography and has been proposed as an alternative to the widely adopted Rivest-Shamir-Adleman (RSA) cryptosystem. One of its advantages is that using shorter key lengths compared to RSA results in more efficient computation for basic operations. To accomplish this, ECC utilizes the mathematical structure of elliptic curves to map finite field elements onto a plane curve. We denote the set of finite field elements or scalars as $\mathbb{Z}_p^*$ and

the set of points on the elliptic curve as $\mathbb{G}$. For readability, we use lowercase letters for scalars and capital letters for points.

Scalars form a field that supports the typical arithmetic operations of addition (+) and multiplication ($\cdot$). Moreover, additive ($-s$) and multiplicative (s^{-1}) inverses exist, enabling subtraction and division to be naturally defined. On the other hand, the elliptic curve forms an abelian group under addition (+), with inversion of elements ($-P$). In addition, elliptic curves have an exponentiation operation ($\cdot$) that raises a point to the power of a scalar $s.P = \sum_1^s P$.

Importantly, the exponentiation serves as a trapdoor function in this context. For given points P and B , it is assumed computationally infeasible to find a scalar s that satisfies $P = s.B$. This problem is known as the Elliptic Curve Discrete Logarithm Problem (ECDLP).

In ECC, the secret key corresponds to a scalar $\text{sk}_A \in \mathbb{Z}_p^*$, and the public key is a corresponding point on the curve $\text{PK}_A = \text{sk}_A.B$. The point B on the elliptic curve, referred to as the base point, acts as a generator for the elliptic curve group. Importantly, B is a parameter of the cryptographic scheme chosen over an elliptic curve and is publicly known to all parties.

2.1.1.1 Elliptic Curve Diffie–Hellman

ECDH serves as the elliptic curve variant of the Diffie–Hellman key agreement protocol, with the main objective of establishing a shared secret over an unsecured channel via public-key cryptography. Consequently, mutual knowledge of public keys among participants is required. As an example, given $(\text{PK}_A, \text{sk}_A)$ and $(\text{PK}_B, \text{sk}_B)$ as Alice’s and Bob’s key pairs, respectively, the shared secret can be calculated as follows:

$$\text{SS} = (\text{sk}_A \cdot r) \cdot \text{PK}_B = (\text{sk}_B \cdot r) \cdot \text{PK}_A = (\text{sk}_A \cdot \text{sk}_B \cdot r) \cdot B \text{ with } r \leftarrow \mathbb{Z}_p^*$$

A primary advantage of this scheme is the ability to compute a shared secret without any prior interaction with the recipient. Consequently, the protocol is frequently employed to transmit a shared secret alongside a ciphertext, which is encrypted using symmetric encryption with a key derived from the shared secret via a hash function [KE10].

2.1.1.2 Digital Signatures

Elliptic curves find another key application in digital signature schemes. A digital signature is a cryptographic tool that validates the authenticity of messages. In essence, a digital signature scheme comprises two operations:

- $\text{Sign}_{\text{sk}_S}(M) \rightarrow S$: The signing operation authenticates a message on behalf of the sender by generating a signature.
- $\text{VerifyPK}_S(M, S) \rightarrow \{\text{true}, \text{false}\}$: This operation confirms whether the sender indeed issued a given signature for a specific message.

Typically, message authenticity is ensured through secure communication channels [RD08]. However, in secure channel free (SCF) environments, these schemes can authenticate messages and protect their integrity. In the field of ECC-based signatures, two well-established schemes exist: Elliptic Curve Digital Signature Algorithm (ECDSA) and Edwards-curve Digital Signature Algorithm (EdDSA) [BBJ⁺08]. While the latter is designed as a faster alternative to ECDSA, it requires a specific type of elliptic curve known as the twisted Edwards curve.

One issue with traditional signature schemes is the need to know the signer of a message to validate the signature. The following section will introduce a different signature scheme that addresses this issue.

Linkable Spontaneous Anonymous Group Signatures

Anonymous group signatures enable a group member to sign a message on behalf of a group without revealing their identity [LWW04]. These group signatures can be further categorized based on their group member structure. In spontaneous groups, the signer can select a set of co-signers from a list of eligible candidates, forming a random group of signers. Notably, collaboration from a co-signer is not required for the signer to create a valid signature.

However, the anonymity of these signatures can induce issues in certain applications. For instance, consider a scenario involving a petition where signers prefer to keep their identities hidden. They could choose co-signers from a list of eligible candidates, sign the document for themselves and each co-signer, and due to anonymity, no one could detect this misuse.

To address the issue of double signing, linkable group signatures introduce two additional constructs: a message tag and a key image. The key image is an additional cryptographic entity generated by the sign operation, and it is deterministic and unique to the signer's secret key and message tag, regardless of the chosen group of co-signers. Consequently, if someone signs two messages with the same message tag, they must have the same key image; otherwise, the signature must be invalid [A⁺20]. The key image does not reveal any information about the signers' identity.

2.1.1.3 Bilinear Pairing

A bilinear pairing, in the context of ECC, is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_t$ of two elliptic curves $\mathbb{G}_1, \mathbb{G}_2$ to a third curve $\mathbb{G}_t$ fulfilling the following properties:

Bilinearity. $\forall a, b \in \mathbb{Z}_p^*, P \in \mathbb{G}_1, Q \in \mathbb{G}_2 : e(a.P, b.Q) = e(P, Q)^{a \cdot b}$

Non-Degeneracy. $e \neq B_t^0$

Computability. The pairing function e is efficiently computable.

For $\mathbb{G}_1 = \mathbb{G}_2$, we call e a symmetric pairing, otherwise asymmetric. Notably, asymmetric pairings are of higher efficiency at the same security level [BCM⁺15]. Thus, we generally work on asymmetric pairings with the previously introduced

notation to differentiate their groups. In addition, we write B_1, B_2 and $B_t = e(B_1, B_2)$ for the base points of the respective groups.

While bilinear pairings are an active field of research, they have various applications, including signatures [BLS01] and identity based encryption (IBE) [BF01]. One field of particular interest in the context of this thesis is searchable encryption, where bilinear pairings are a state-of-the-art technique to provide privacy-preserving searches on encrypted data [GY15, WMCZ16, LLZ19].

2.1.2 ElGamal – Homomorphic Encryption

Homomorphic encryption refers to encryption schemes with homomorphic ciphertext compositions that correspond to their plaintext compositions. Thus, one can compute the aggregation of encrypted values without knowing them using homomorphic encryption schemes. One of these homomorphic encryption schemes is ElGamal, which uses asymmetric key encryption and works with ECC as defined in Algorithm 2.1.

$\text{ElGamal}_{\text{PK}_A}(M)$	$\text{ElGamal}_{\text{sk}_A}^{-1}((C_1, C_2))$
return $(r.B, (\pi(M) \cdot r).PK_A), r \leftarrow \mathbb{Z}_p^*$	return $\pi^{-1}(C_2 - \text{sk}_A.C_1)$

Algorithm 2.1: Definition of the ElGamal encryption scheme using ECC. Here π is a reversible mapping of the message space to the elliptic curve.

According to the definition of the ElGamal encryption, we can aggregate ciphertext corresponding to their plaintext's addition.

$$\begin{aligned}
& \text{ElGamal}_{\text{sk}_A}^{-1}(\text{ElGamal}_{\text{PK}_A}(\pi(a)) + \text{ElGamal}_{\text{PK}_A}(\pi(b))) \\
&= \text{ElGamal}_{\text{sk}_A}^{-1}(\text{ElGamal}_{\text{PK}_A}(a.B) + \text{ElGamal}_{\text{PK}_A}(b.B)) \\
&= \text{ElGamal}_{\text{sk}_A}^{-1}((r_a.B + r_b.B, (r_a + a).PK_A + (r_b + b).PK_A)) \\
&= \text{ElGamal}_{\text{sk}_A}^{-1}(((r_a + r_b).B, (r_a + a + r_b + b).PK_A)) \\
&= \pi^{-1}(\text{sk}_A \cdot (r_a + a + r_b + b).B - \text{sk}_A \cdot (r_a + r_b).B) \\
&= \pi^{-1}(a + b).B = a + b
\end{aligned}$$

One challenge of this encryption scheme is the mapping from message to and from an elliptic curve element [FJT13]. However, for a feasible small message space, we can iterate all possible messages and store their reverse mapping into a hashmap. This small message space allows us to use a one-way function for our mapping and still be able to decrypt messages. In the case of Algorithm 2.1, we chose a one-way function for small integers such that $\pi(a) = a.B$.

2.1.3 Stealth Addresses

Stealth addresses are a method developed in blockchains to conceal a receiver's identity where transaction details are publicly visible [CM17]. The sender generates a random public key and includes additional information in the transaction to enable the receiver to derive the secret key.

- $\text{StealthAddress}_{\text{PK}}(\text{PK}_R) \rightarrow (\text{PK}_{\text{SA}}, E)$: Generates a new random address for a receiver PK_R and outputs a random public key PK_{SA} and some extra information E to be sent to the receiver.
- $\text{StealthAddress}_{\text{sk}}(\text{sk}_R, (\text{PK}_{\text{SA}}, E)) \rightarrow \text{sk}_{\text{SA}}$: Derives the secret key sk_{SA} for a stealth address PK_{SA} that targets the receiver $\text{PK}_R = \text{sk}_R.B$.

The public part $(\text{PK}_{\text{SA}}, E)$ must be unlinkable to the receiver's public key PK_R . Stealth addresses are closely related to the ECDH protocol and, indeed, the first stealth address protocol used ECDH to establish a shared secret that is hashed to a secret key [CM17]. The problem with this approach is that both the sender and receiver can derive the secret key sk_R . Therefore the scheme was further adopted to prevent impersonation [VS18].

$\text{StealthAddress}_{\text{PK}}(\text{PK}_R)$

$r \leftarrow \mathbb{Z}_p^*$
 $E = r.B$
 $s = H(r.PK_R)$
 $\text{PK}_{\text{SA}} = \text{PK}_R + s.B$
return $(\text{PK}_{\text{SA}}, E)$

$\text{StealthAddress}_{\text{sk}}(\text{sk}_R, (\text{PK}_{\text{SA}}, E))$

$s = H(\text{sk}_R.E)$
if $s.B + \text{PK}_R \stackrel{?}{=} \text{PK}_{\text{SA}}$ **then**
 return $s + \text{sk}_R$
fi
return $\perp$

Algorithm 2.2: An improved stealth address scheme is preventing the impersonation of the receiver by the sender [VS18].

2.1.4 Zero-Knowledge Proofs of Discrete Logarithms

In 1997, Camenisch et. al. [CS97] published a proof system for general statements about discrete logarithms. Here we summarize a small portion of their system and its capabilities that fit our usage needs. Their system enables us to prove the knowledge of a discrete logarithm $w \in \mathbb{Z}$ of $Y \in \mathbb{G}$ to the base $X \in \mathbb{G}$. In the simplest form, this corresponds to proving the knowledge of a private key satisfying a statement $\text{DL}(Y, X, w) := Y = w.X$. However, the system is also capable of proving more general statements on the knowledge of linear expressions

$$\text{LE}(Y, X_1, \dots, X_n, w_1, \dots, w_n) := Y = \sum_i^n w_i.X_i$$

with $Y, X_1, \dots, X_n \in \mathbb{G}$ and $w_1, \dots, w_n \in \mathbb{Z}$. Moreover, these formulas can be combined to prove multiple statements consistently. One use-case of such statements is the proof of equality of two logarithms which will serve as an example.

$$\text{DLogEq}(Y_1, Y_2, B, U, w) := \text{DL}(Y_1, B, w) \wedge \text{DL}(Y_2, U, w)$$

Here the witness w is used twice and must be consistent to yield valid proof. In the following, these statements are summarized as a tuple $(V_{\text{pub}}, V_{\text{sec}}, T)$ of public variables V_{pub} , variables of the witnesses V_{sec} and a set of linear expressions T . The statement is the conjunction of all terms of T and is limited to using previously

defined variables. For convenience, the explicit statement of unbound variables may be skipped by passing a formula to the statement constructor DLogStmt , and bindings to the curve generator B can be skipped. Using this notation, the previous example can be summarized as follows:

$$\begin{aligned} \text{DLogEq} &:= (\{Y_1, Y_2, U\}, \{w\}, \{Y_1 = w.B, Y_2 = w.U\}) \\ &= \text{DLogStmt}(Y_1 = w.B \wedge Y_2 = w.U) \end{aligned}$$

A proof of this statement can then be constructed by $\text{proof} = \text{DLogProof}(\text{stmt}, \text{var}) \rightarrow \mathbb{G}^m \times \mathbb{Z}^n$ where stmt is a statement tuple and var a mapping from V_{pub} to $\mathbb{G}$ and V_{sec} to $\mathbb{Z}$ respectively. The proof consists of $m = |F|$ points for each linear expression and $n = |V_{\text{sec}}|$ scalars for each witness. Similarly such proof can be verified using $\text{DLogVerify}(\text{stmt}, \text{proof}, \text{var}) \rightarrow \{\text{true}, \text{false}\}$. Here var only requires mappings for V_{pub} .

2.2 Public Key Encryption with Keyword Search

Cloud computing offers users an economical and scalable platform to outsource services and data, but besides its benefits, users' adoption is hindered by the security risk of data breaches [HKA19]. One solution to this problem is to store data encrypted in the cloud such that a data leakage reveals no information. In 2000, Song et al. [SWP00] published a cryptographic scheme allowing users to query encrypted data on a semi-trusted, third-party server. With the original formulation only considering symmetric encryption, Boneh et al. [BCOP04] adapted the scheme to asymmetric encryption introducing the notion of public key encryption with keyword search (PEKS). This section introduces the basic framework of PEKS schemes.

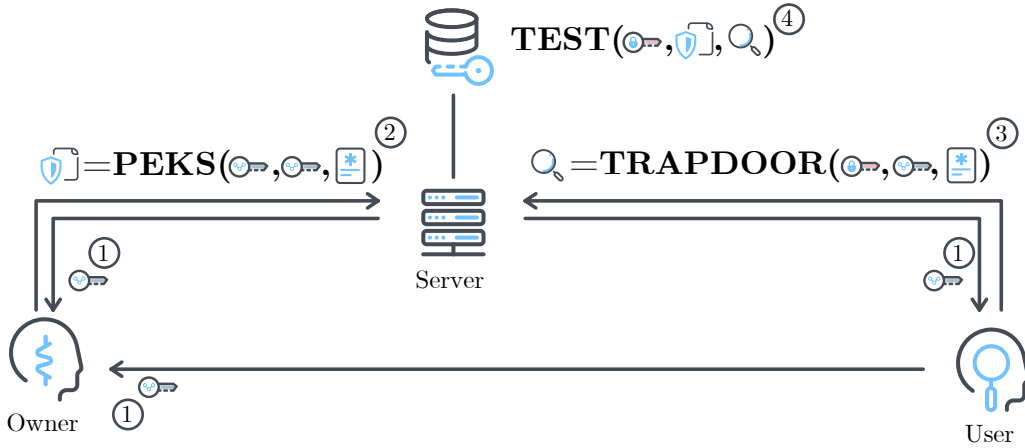


Figure 2.1 Exemplary PEKS architecture without result retrieval. 1) Exchange of global parameters. 2) Patient encrypts keywords for a particular user and server. The index is forwarded to the server and stored in a database. 3) The user creates a trapdoor to search for a specific keyword. 4) The server matches the trapdoor with patients' indices.

In a PEKS scheme, an owner encrypts their data for a specific user and then stores the encrypted keyword on a semi-trusted cloud server. After that, the authorized

user can issue an encrypted search request to that server, matching the data without letting the server infer any information on the data and search. Commonly the encrypted data is called index ($\mathcal{I}$), and the encrypted search request trapdoor ($\mathcal{T}$). Further, we name the corresponding functions **PEKS**, **Trapdoor**, and **Test** to create the index, trapdoor, and match the data on the server, respectively.

2.2.1 Attack Models

Although PEKS schemes strengthen security and privacy guarantees, they are vulnerable to attacks. This section briefly introduces the two most considered attacking models on PEKS schemes [NAG22].

Keyword Guessing. Keyword guessing is one of the most prominent attacks on PEKS schemes. In a keyword-guessing attack, the attacker tries to guess the content of an index or trapdoor, compromising the owner's and user's privacy. Commonly this induces a brute-force method where the attacker benefits from a small keyword space. Therefore the attacker uses access to the test function to repetitively generate all possible indices until it matches a trapdoor, revealing the keyword searched. Similarly, an attacker can abuse the capability to create trapdoors to iteratively create all possible trapdoors until it matches an index. Further keyword-guessing attacks are distinguished by the method they are carried out. In an external keyword guessing attack, the attacker cannot perform the required brute-force operation by themselves and uses other vulnerabilities, e.g. non-secure communication channels, to replace the search requests of other users. On the other hand, if the attacker can perform the test and an additional operation, we categorize them as an internal attacker.

File Injection. In a file injection (FI) scenario, an attacker injects false data into a system to obtain information on the query. For example, the attacker could observe the access pattern on injected data to get insights into the keyword of a trapdoor [NAG22] using statistical analysis. While Zhand and Zhang [ZZ11] were the first to show the feasibility of FI attacks, Ahmed and Barkat Ullah [ABU18] have shown that this attacking vector is increasingly executed, including in the context of health data.

3

Related Work

Since the introduction of electronic health records, many different uses have been explored to increase public health. Data sharing has become an increasingly important component with increased opportunities and benefits. However, due to the high sensitivity of the data, these systems must always find a compromise between usability and privacy. In this chapter, we will take a closer look at the ethical principles and technology used to accomplish them in modern systems.

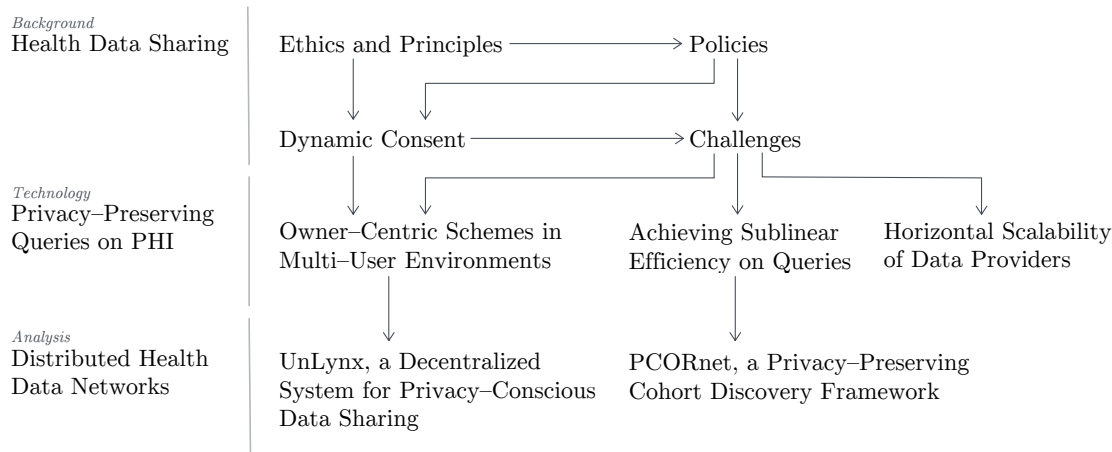


Figure 3.1 A structural overview of the related work chapter, categorizing the content into multiple stages that progressively build upon each other. Arrows indicate the logical flow, where sections serve as a coherent follow-up to the preceding one.

Sensitizing for sensitive health data, we elaborate on electronic health care in the context of data sharing in Section 3.1. We introduce vital aspects of collecting, storing, and analyzing EHRs, pointing out problems in usability and collaboration and modern concepts to overcome those. Following, in Section 3.2, we review asymmetric searchable encryption as a state-of-the-art technology for privacy-preserving data exploration. We cover this area’s most common research directions, highlighting key features and drawbacks of health data-sharing challenges. Finally, based on

the taxonomy introduced, we classify existing systems with privacy-preserving in Section 3.3. Therefore, we elaborate on current systems exploration capabilities, analyze them, and work out standard metrics for evaluating such systems.

3.1 Health Data Sharing

Health data, in the form of Electronic health records (EHRs), are widely accepted to facilitate clinical research in various fields [CSK⁺13, DMTB21], with more use cases emerging [CBC⁺17]. Its quantity and quality have a significant impact on its usefulness [BHCW10, KRN⁺19]. Therefore, sharing data is essential to improve public health more efficiently [WB11, SKMS13, CBP14]. However, due to the high sensitivity of health data, strict standards have been developed to balance the need for data sharing and patient privacy [KHT⁺11, ADW⁺18]. As this thesis will work on such data, it is of great interest to elaborate on these ethical standards and their principles to carve out commonly used techniques, trade-offs, and characteristics.

3.1.1 Ethics and Requirements for Health Data Sharing

In 1979, the Belmont report set the de-facto standard of fundamental ethical principles providing a clear guideline to protect humans as a research subject [ftPoHSoBR79]. With the increasing adoption of EHRs, researchers investigated these ethical principles in the context of electronic data-sharing systems. Their goal was to balance the rights and responsibilities of those who produce and consume data in a sustainable way [WB11] that will increase public health in the long term. The elaborated principles were later subclassified into groups of principles and procedures by Knoppers et al. [KHT⁺11] as an international data-sharing Code of Conduct. These principles served as a basis for governmental frameworks and research in this area, stating critical requirements for data-sharing systems.

Quality. Shared data should be of high quality and peer-reviewed. Achieving higher quality results in more efficient and comparable results in collaborative research.

Accessibility. Data should be stored and shared securely so that data providers can maximize their value. Data usage and generation should be balanced between user and owner so that donors benefit from participating.

Responsibility. Governance should be shared across all parties, encouraging interoperability.

Security. Data storage access should be authorized, and data providers and users should be traceable to be held accountable for violating their responsibilities.

Transparency. Participating network entities should be transparent, publishing all about their research intent, funding, and progress.

Accountability. Participants should be held accountable and stay under continuous monitoring and governance of third-party agencies.

Integrity. Derivation of datasets from other primary sources should retain integrity and acknowledge authors of the original data.

In the following, we will elaborate on the challenges to satisfy these requirements and regulatory frameworks. These will give a comprehensive view of the problems we solve in this thesis, including common techniques to face those.

3.1.2 Regulatory Frameworks

Due to the sensitivity of the information in EHRs and the increasing public awareness of the risks of sharing health data [MC11, KBK⁺16], governments continuously regulate its usage to protect patient privacy while partially retaining usability [ADW⁺18]. Regulations like the US Department of Health and Human Services Common Rule, Health Insurance Portability and Accountability Act (HIPAA), or the EU Data Protection Directive set legal frameworks for obtaining, storing, and processing personal data. Some critical aspects of fulfilling regulations include, but are not limited to, de-identification and consent.

De-identification. This method modifies records to impede the re-identification of individuals. Here the challenge is to find a balanced amount of modifications to retain the quality of the data. For example removing too much sensitive data decreases usability for specific research areas [ADW⁺18]. Finding the right amount of de-identification to ensure anonymization is an elaborated task with high risks and implies extra cost [Ohm09, LLM⁺16]. Furthermore, de-identification shall impede re-identification but not prevent, leading to vague requirements [Swe02, CSK⁺13]. HIPAA states that there shall at most be a “small risk” to reidentifying patients, lacking a precise formulation leading to adoptions vulnerable to re-identification [ADW⁺18]. For example, Latanya Sweeney [Swe02] showed that linking multiple data sources makes re-identification possible. In that particular case, voter registrations were linked with patient-specific data based on individuals’ day of birth, sex, and ZIP code leading to unambiguous identification of individuals. Benitez et al. [BM10] investigated the risk of re-identification on datasets applying the HIPAA privacy rules. Their results show that applying policies does not guarantee privacy and can leave organizations at the risk of reidentifying up to 60% of the individuals based on the organizations’ metrics and policy chosen. Similar results were surveyed outside the context of clinical research by El Emam et al. [EEJAM11], highlighting publications attacking de-identification with evidence of high re-identification rates. Notably, Ochoa et al. [ORRS01] described a method of reidentifying 75% of homicide victims in Chicago, across databases, based on an improved matching approach on geographic data to link individuals.

We conclude that re-identification is a prominent risk, challenging protecting patients’ privacy, even if government policies are applied. Later, in Section 3.1.4.3, we will investigate more into research enhancing privacy guarantees while retaining usability.

Consent. In the consent model, data providers ask patients for agreement to share their data in a pseudonymized form. Regulations provide general requirements for such a consent formula, including the necessity to state the research intent in a compact structure. This so-called informed consent protects the patients’ autonomy and self-determination, enforcing ethical standards [CSK⁺13, ADW⁺18]. The informed consent can be very restrictive, especially for later reuse or inter-organizational research, as this often requires gaining retrospective consent, which

is not always possible [CDD⁺17] and is cost intensive [DCK⁺17]. Broad consent tries to solve the problems of data reuse in the broad consent model in the case of uncertainty in future research. This model partially solves the problem of continuously required re-consent [SS11] but at the cost of transparency and donors' freedom of choice [SKMS13], leading to controversial discussions on the ethical aspects of broad consent [HLP12, SKMS13, KWL⁺15]. Consequently, the recent development of the regulatory framework General Data Protection Regulation (GDPR) includes the necessity of informed consent [LMAS20]. Another drawback is the decreasing willingness to contribute data because donors are less likely to consent to broad agreements, having no granular control over their data usage [CH12, KBK⁺16].

We have seen regulatory frameworks obligatory to share health data that balance usability and owners' privacy. In the case of de-identification, we highlighted the vague and insufficient formulation leaving data at risk of re-identification. On the other hand, the necessity of owner consent leads to problems in maintainability and usability and decreases patients' willingness to contribute their data to research. For the latter problem, researchers have come up with a modern solution, namely dynamic consent.

3.1.3 Dynamic Consent

With the progress in information technology (IT), emerging technologies like smart-phone apps, and Internet of things (IoT) devices, the amount of data collected and its impact on electronic healthcare is increasing rapidly [KRN⁺19, VHVJ20]. Traditional consent models struggle to fit these new requirements of rising numbers of available records [KWL⁺15], evolving use cases [SKMS13], and international co-operation [VPPE⁺14, BLTK⁺17]. Researchers proposed the dynamic consent model utilizing these new technologies to create an interactive communication channel between researchers and patients [KWL⁺15]. This model enables researchers to contact patients more efficiently, ask for (re-)consent, actively inform them about their data usage, and share data cross-border at a larger scale [BLTK⁺17]. Additionally, it shifts data access governance to its owner, allowing patients to administer their shared data access. Thus, dynamic consent promises to solve multiple of our elaborated problems with classical consent schemes, but it is relatively new, requiring practical experience in its application.

iCONCUR [KBK⁺16] put dynamic consent to the test. In their study, participants could configure preferences of which data to share with whom. The results show that a dynamic consent management system makes patients feel more comfortable sharing their data and increases their willingness to participate. Furthermore, it spotlights vital characteristics of research usage intent and data categorization that influence participation, highlighting the importance of transparency. Notably, patients were more likely to share their data with affiliated institutions and non-profit organizations and would like to be informed about their data usage. Similar results have been published by Page et al. [PMM16], showing that patients care about being informed about their data usage. Additionally, patients would like to have the ability to revoke consent.

Although dynamic consent management systems can increase the level of trust and lead to higher participation, the necessity of interaction may lead to decreasing

numbers of shared records, as passive participants may be discouraged by the burden of active upkeep [SKMS13, KWL⁺15]. Therefore, consent management systems should be designed as opt-in and easy to configure, allowing patients to dive deeper if desired and as broad as possible simultaneously. Implementing such a system is a challenging task and may increase the cost compared to traditional solutions in the short term because of high development costs, but they are likely to be more cost-efficient in the long term [BLTK⁺17].

3.1.4 Measurements and Technologies for Regulatory Compliance

So far, we have seen requirements and regulations to comply with ethical principles and governmental regulations. However, to review existing schemes in the context of our thesis, we need to gain knowledge of comparable standards commonly used to match introduced obligations. In this section, we fill this gap by reviewing research that elaborates on techniques and characteristics to facilitate and assess networks for health data sharing.

3.1.4.1 Quality

In order to use EHRs efficiently and gain reliable and comparable results, it is crucial to rely on high-quality data [BHCW10, CSK⁺13]. Researchers [BHCW10, KHT⁺11, CSK⁺13] measure such quality in terms of validity, completeness, and accuracy. Achieving these goals can be very challenging and costly, especially in the case of new diseases where data availability is changing rapidly [LUM⁺20]. To ensure reliable and comparable quality, researchers either peer-review data or rely on a trusted third-party certification from organizations like Healthcare Interoperability Testing and Conformance Harmonization (HITECH) [CSK⁺13]. These monitoring requirements show a close relation of quality to responsibility, accountability, and integrity as users need to rely on the data in others' responsibility.

3.1.4.2 Interoperability

Interoperability of heterogeneous systems is considered one of the fundamental challenges in the design of EHR systems [BZ17]. Every computation requires a computer to interpret, compute and present structured data meaningfully. Therefore, researchers developed semantic models to map existing data and their relations into a standardized electronic format.

ICD-10 [Org04], LOINC [MHS⁺03], and SNOMED-CT [D⁺06] all standardize existing disease terminology into codes serving as unique identifiers. While LOINC and SNOMED-CT cover a broader range of other medical terms like diagnosis and general health care as well, ICD-10 does not and is usually combined with International Classification of Functioning, Disability and Health (ICF) or International Classification of Health Interventions (ICHI), as part of the WHO Family of International Classifications (WHO-FIC). These codes provide a discrete representation for non-numerical attributes which can be enumerated. Notably, other non-numerical data types exist, lacking a common standard, like images and genomes, and finding an appropriate

interface of communication is an open problem at the time being [KRN⁺19]. Using these attribute-specific representations, more comprehensive standards are developed.

openEHR publishes one of these more comprehensive standards based on ISO EN 13606 [ISO19]. This non-profit organization develops specifications for longitudinal patient-centric data utilizing existing standards for compatibility while also providing architectural models and extendable formats. Therefore, openEHR suits complex and application-specific architectures requiring longitudinal persistent EHR databases. However, due to its high complexity, openEHR is not suited for all applications [BZ17, RT21].

Fast Health Interoperability Resources (FHIR), by HL7, is a system that focuses more on EHRs as resource rather than part of a relational system and is designed to share documents in short short-term communications. As a result, FHIR gained much adoption for sharing documents and established itself as a de-facto standard for interoperability [SRR19]. The extraction of FHIR documents from existing database models is an active topic of research and is usually implemented as an extra layer of abstraction on top of standards for long-term preservation [BZ17, RT21].

In addition to structural incompatibility, there are also incompatibilities in political governance. Federal policies not synchronized to each other increase data waste and costs of international collaboration [CSK⁺13]. Privacy-preserving technologies can bypass this problem and fulfill even stricter requirements by avoiding data sharing [CSK⁺13, ADW⁺18].

3.1.4.3 Privacy

As mentioned in Section 3.1.2, governmental policies are insufficient to guarantee patients' privacy. In this section, the focus is on methods balancing usability and risks more efficiently and provide provable guarantees. Formerly, we categorize those methods into data anonymity, differential privacy, and encryption. Note that this does not include anonymity and privacy for the issuer of an analysis, which is generally underrepresented in electronic healthcare systems. However, there is more research in that regard, in the context of cryptographic schemes, which we will cover in Section 3.2.

Data Anonymity

This category refers to methodology modifying data in order to prevent identification of individuals. Section 3.1.2 already introduced de-identification, modifying existing data to prevent re-identification. In this section we focus on other methods providing stronger and provable anonymity guarantees.

In the k -anonymity model published by Latanya Sweeney [Swe02] every query on quasi-identifier¹ is required to return at least k records. This guarantees that every record obtained by a single query is indistinguishable from at least $k - 1$ other records. Such indistinguishability is commonly obtained by generalization, as proposed by

¹An attribute is a quasi-identifier, if it can be linked with external resources (e.g., name, date of birth, or tax number).

Kim et al. [KLC17]. Their model of privacy-preserving data cubes describes the methods of global/local generalization and bucketization, each with different trade-offs regarding data size and information loss. Poulis et al. [PLSGD17] proposed a method of generalization for demographics and diagnosis codes. Their specialized algorithms utilize the hierarchical structure of demographics and ICD-10 codes to obtain strong anonymity guarantees for pre-defined utility constraints.

Weaknesses of k -anonymity include unexpected background knowledge or sequential releases of data used to identify individuals based on the low diversity of records returned in sensitive attributes [FWCY10, ADW⁺18]. To circumvent these limitations, Wang et al. [WF06] introduced (X, Y) -privacy as a generalization of k -anonymity, which guarantees k -anonymization of X quasi-identifiers concerning sensitive attributes Y . Notably, Machanavajjhala et al. [MKG07] proposed a similar extension of k -anonymity called ℓ -diversity.

One open problem in k -anonymity-related privacy models is the selection of quasi-identifier to protect [FWCY10]. Wang et al. [WXFW09] showed that finding optimal anonymization with minimal generalization is NP-hard under the assumption that data contains observable quasi-identifiers and sensitive attributes. Despite its problems, anonymization has been successfully tested and deployed in the context of clinical research and is an active topic of research [HRP⁺15, FI10, PLSGD17].

Please observe that the level of modifications and received anonymization should always be chosen with the application in mind. Here we gave a short outline on the research of this topic, but this is far from a general guideline of security schemes to adopt.

Differential Privacy

Even with all enhancing schemes on k -anonymity, none can provably secure data against background knowledge attacks. For that reason, researchers came up with a different security model. Cynthia Dwork [Cyn06] published the notion of ϵ -differential privacy featuring privacy guarantees independent of an attacker's background knowledge and therefore eliminates the risk of linking individuals to external databases. Ensuring that two databases are only distinguishable with at most negligible probability if they only differ in one row, including addition and removal, achieves this property. Therefore, the differential privacy model also protects the privacy of individuals not participating.

Records are modified to achieve this privacy notion by adding noise with negligible observable impact on statistical distribution, which limits its application. Especially for non-numerical data like strings or tree-structured relations, adding noise has no sense, further limiting its practicality [Dwo08]. However, sensitive numerical attributes can also be problematic as slight variations have to high impact on the statistical result. McSherry et al. [MT07] showed that differential privacy could be applied to sensitive attributes if we know the computation of the data in advance [AS00]. Therefore, general data utility is reduced, and privacy parameters must be chosen with a specific application in mind, leading to non-trivial design choices between privacy and usability [ADW⁺18].

Differential privacy outstands due to its low complexity involving no extra overhead during computation when applied to a specific use case. As a result, it achieves

remarkable results on big-data computations, including deep learning [ACG⁺16], real-time health observation [ZLZ⁺17], and distributed aggregation [FES⁺17]. However, this security model is still insufficient to guarantee absolute privacy as the aggregation of small sample groups may still leak information about individuals [CSK⁺13]. Thus, one should not refer to differential privacy as the solution for privacy-preserving health data evaluation but rather as one of the tools in a toolset summarized in this section.

Encryption

While data anonymity and differential privacy focus on preventing getting insights of individuals from a dataset, encryption tries to prohibit viewing the dataset in the first place. Therefore, encryption schemes provide cryptographic primitives for integrity and confidentiality protection of data exchange and preservation. They have provable solid security and privacy guarantees making storage outsourcing to cloud service providers (CSPs) a viable choice. However, encryption does not generally guarantee privacy but transfers data protection responsibilities to the secret owner [QQLM20]. In electronic health care, asymmetric encryption schemes are generally preferred as they do not require secret sharing and thus are usable in open environments without secure channels, and multiuser setups do not require complex key management [HKA19, NAG22]. Nevertheless, these schemes secure encrypted data using the secret of one individual, which requires a high level of trust, not always applicable to sensitive data.

Froelicher et al. [FES⁺17] proposed a collective authority using the (k, n) -threshold scheme of Shamir [Sha79] to strengthen the security model even further. Their model does not require the trust of a single entity holding a key but rather a closed group of entities that stays secure as long as at least one member is not compromised. Such key sharing can then encrypt and process sensitive data securely.

Recent advances in homomorphic cryptography enable such computation of algorithms using encrypted data in asymmetric systems. Additionally, such computations can be (partially) oblivious, increasing the privacy guarantees of the owner and the user [FIPR05, RBLL06, WSLH15]. Such systems are widely adopted as they enable data providers to participate in distributed computation without sharing sensitive data in the first place [CSK⁺13, ADW⁺18]. One such computation with extensive research efforts is the search on encrypted data [BCOP04].

Asymmetric searchable encryption allows privacy-preserving exploration of encrypted clinical data without knowing the exact search query. Therefore, it is considered state-of-the-art practice for data exploration and cohort discovery in distributed health data networks (DHDNs) and widely adopted in active research, and practice [CWJ⁺16, FES⁺17, YMM⁺17, RTPM⁺19, FMTP⁺20]. Additionally, data can be stored and processed by an untrusted CSPs without sharing sensitive information, outsourcing storage and computation in a secure way. Due to the high significance of searchable encryption on this thesis topic, Section 3.2 will cover its current state of research in the context of this thesis.

3.2 Privacy-Preserving Queries on Protected Health Information

The challenges of health data sharing are typical for applications on sensitive data and apply to other areas, such as secure databases, encrypted audit logs, or smart grids [HKA19], to name a few. As a result, extensive research on searchable encryption (SE) exists, but we focus on recently deployed asymmetric schemes [YMM⁺17, FES⁺17]. In particular, we do not incorporate schemes based on symmetric encryption or approximate searching. We align ourselves to established techniques commonly used in health data sharing [CAWW19, HKA19], such as public-key and homomorphic encryption. Further, we exclude techniques impractical for high amounts of data due to computational overhead, such as hidden vector encryption (HVE) [BW07, ZLW18, ZLW19] and multi-party computation (MPC) [RBLL06, Ahl16, ZXLZ17] as well as order-preserving encryption (OPE), due to its problems with privacy [KS12] and efficiency [ZSD⁺22].

To elaborate on the current state of research, we investigated recent surveys [BHJP14, CAWW19, HKA19, NAG22] and analyzed architectures and cryptographic approaches in the context of this thesis. That is the potential of privacy-preserving queries on protected health information (PHI) with owner-centric capabilities. Foundend on extracted approaches, we filtered the publications mentioned and searched for additional research. In the following, we comprehensively view our chosen related work, including the increasingly more assertive notions of security and existing medical applications adopting these technologies.

The organization of the remaining section reflects this systematic approach. First, we highlight different architectures and technologies in publications and evaluate their capabilities in owner-centric multi-user environments. Based on the publications extracted, we continue to review distributed systems' privacy, security, and scalability solutions. Finally, we take a look at additional techniques used in research to achieve higher efficiency and their impact on usability and privacy.

3.2.1 Owner-Centric Schemes in Multi-User Environments

A patient-centric scheme is required to fulfill the obligations of dynamic consent, in which the patient is free to control whom to share their owned data [BLLS11]. Therefore, we alborate multi-user schemes with fine-grained access control, including enrollment and revocation of users [WMCZ16, SYL⁺16, YM16]. In the following, we summarize the directions of research taken, fulfill such obligations, and analyze them concerning security, privacy, and scalability.

IBE-based Solutions. Identity based encryption (IBE) introduced by Shamir [Sha85] simplifies key management in public key infastructures (PKIs) and is closely related to public key encryption with keyword search (PEKS) [BW07]. Inherently IBE is well-suited for multi-user environments, and Wang et al. [WMCZ16] adopts this property to develop a multi-user PEKS scheme. However, their methodology requires an n -degree polynomial index scaling with the number of possible users, making it impracticable for large scaling systems. In the case of min-entropy attributes, Wu et al. [WTT14] scheme has better scalability as their index size linearly depends on the

number of possible keywords. Still, in both schemes, the trapdoor generation requires the users' secret key and the plaintext keyword to query, resulting in the necessity of a secure communication channel [NAG22]. Camenisch et al. [CKRS09] proposed a scheme that neither requires a secure communication channel nor reveals the query, but their scheme does not support multi-user environments. To our knowledge, up to this point in time, no IBE-based scheme exists without those requirements in a multi-user setup.

ABE-based Solutions. The previous limitations do not hold for systems based on ABE introduced by Sahai and Waters [SW05]. ABE encrypts an access policy rather than the document and thus is independent of the number of users [NAG22], resulting in very efficient multi-user schemes. Sun et al. [SYL⁺16] utilize this, proposing a scheme scaling linearly with the number of possible attributes per index, which also supports multiple owners. Nevertheless, to ensure consideration of owners' preferences regarding user access, the owner is constrained to trust the server where their preference is stored as a list of allowed users. Thus, the server must be semi-trusted, and active owner upkeep is required giving access to new users. Zhang et al. [ZZD18] noted that previous ciphertext-policy attribute-based encryption (CP-ABE) schemes might leak sensitive data included in the policy and proposed a scheme in which specific attributes can be privacy protected by using linear secret-sharing (LSS). Other problems of ABE-based schemes are the revocation of user access [LYZ⁺13] and keywords update [LS15], limiting their scalability in terms of updatability. Notably, works [HN11, ZNS12, Yan15] are solving the former problem at the cost of complex key management or user privacy. Nevertheless, it should be mentioned that ABE-based schemes are not designed to enable searches on data but rather receive data. Therefore, these are applicable to enforce policies and access control but not to encrypt patients' data in a searchable way.

Another disadvantage of IBE and ABE-based schemes mentioned is the usage of cryptographic primitives requiring a trusted central authority to manage a master key. In a distributed setup, this leads to a single point of failure, as compromising the master key undermines security and breaks the entire system [LCLS08, CAWW19]. While there are solutions for collective authorities (CAs) providing such primitive cryptographic protocols, such as multi-authority ABE [LCLS08] and (n, t) -IBE [Sia12], these techniques involve MPC or scale exponentially resulting in high overhead in computation and communication costs.

Yang and Ma [YM16] propose a different approach for a multi-user system, adopting Blaze et al. [BBS98] proxy re-encryption protocol. In their scheme, a trusted third party serves re-encryption keys converting ciphertexts from one key to another suitable to perform searches independent of the user. Usually, a separate entity, called a proxy server, is responsible for the re-encryption, and various publications use similar techniques [GLLX13, LZF14, LS15, YM16, YZCT17]. The primary application of these proxies is the utilization as a gateway for fine-grained access control [HKA19, NAG22], such as authorization or time-dependent keys for limited-time access [YM16]. One drawback is the introduction of an additional entity that might harm the system. This entity must be analyzed independently in an appropriate security model [NAG22, YZCT17]. Commonly, publications consider the proxy server to be honest but curious [LS15, YM16], leading to similar security concerns as in the case of cloud service provider (CSP). On the other hand, this approach is less complex

than other approaches achieving similar results and, therefore, can be adopted in environments without trusted authority.

Froelicher et al. [FES⁺17] use a re-encryption technique to map ciphertexts confidentially from a collective key to the key of a user, which they call the “Key Switch” protocol. Here the trusted third party is represented by a CA, requiring collaborative protocols for re-encryption. A group of members encrypts the ciphertext in the CA using a (k, n) -threshold scheme [Sha79]. Therefore, messages are encrypted under a collective key whose members partially and sequentially switch to the user key without any member being able to decrypt the ciphertext themselves. As a result, they increase their network security to an “AnyTrust” model [WCGFJ12], which will only be compromised if every party member of the CA authority is compromised. However, this requires the active involvement of every member for every operation requiring the collective secret key, which makes the protocol less efficient.

Takeaway. Multiple techniques exist to solve the complexity limitations of multi-user and multi-owner environments. Commonly, proposed solutions reduce complexity by mapping ciphertexts to a secret or master key. In the case of IBE and ABE-based solutions, this requires trusted authorities which are difficult to replace. For proxy re-encryption, the mapping can be distributed among several servers, yielding stronger security guarantees at the cost of computational overhead.

3.2.2 Horizontal Scalability of Data Providers

PEKS systems highlighted so far are designed with a client-server architecture in mind, featuring multi-user and multi-owner environments but do not focus on distributed systems with multiple data providers (DPs). Only one of the four surveys we studied considers multiple CSPs [HKA19], not featuring scalability but rather security aspects such as proxy servers and dedicated testers. The following will cover some less common security issues with high significance in scalable distributed systems and how to counter them.

3.2.2.1 Benefits of Secure Channel Free Communication

Scaling DPs or CSPs horizontally can impact capital expense and security captured by secure channel free (SCF) communication. Secure channel free PEKS (SCF-PEKS) schemes were initially proposed by Baek et al. [BSNS08] and define the security model IND-SCF-CKA formerly, covering networks without access to secure channels. Their security model requires indistinguishability of the ciphertext even if the attacker can generate trapdoors of owned keywords. However, IND-SCF-CKA does not cover resilience against keyword guessing (KG) attacks (IND-KGA), which was addressed later by Rhee et al. [RSK09]. Fang et al. [FSGW13] further strengthened both security models to IND-SCF-CKCA and IND-KGA, adding indistinguishability for chosen ciphertext attacks and KG attacks. As a result, adopting SCF-PEKS and a suitable security model removes the necessity of secure channels, making systems more resilient to KG attacks and less vulnerable to data breaches [BSNS08, FSGW13, MMW⁺17]. Notably, it also reduces capital expense [BSNS08, MMW⁺17] and uses of limited resources such as constrained bandwidth [MMW⁺17]. These security models are also

more suitable for scaling cloud server instances, as they can protect a single server's integrity against the compromise of others in the same network.

3.2.2.2 Threat of Information Exposure to Cloud Service Providers

Although SE schemes are designed to enable a designated tester, such as CSP, to search for keywords without knowing the keyword or the document, this does not inherently prevent the server from obtaining any knowledge of documents stored and queries performed. While most publications focus on the standard models of indistinguishability and KG, researchers have shown that this is insufficient to protect data privacy.

An honest but curious server may analyze its' data usage based on evaluating the search predicate. Using this access pattern can lead to leakage of query details [SWP00, BSNS08, CWL⁺14, CGPR15], as the CSP may combine this knowledge with known keyword frequencies [ZDO⁺08]. Cash et al. [CGPR15] formerly specified the required adversary knowledge, showing that an adversary can recover queries by knowing the resulting length. Furthermore, they show that partial knowledge of stored documents can lead to plaintext recovery, tightening the results of Zhang and Zhang [ZZ11], who presented similar results for a malicious server using file injection (FI) attacks. Notably, FI attacks have increased, as Ahmed et al. [ABU18] have shown. Shen et al. [SSW09] analyzed this "Known Background" threat model under *predicate privacy*, requiring tokens not to reveal any information. They conclude that this kind of privacy is inherently impossible in the public-key setting, as it does not require a secret key, and an adversary can encrypt arbitrary plaintexts to match the query. In order to prevent these attacking vectors, researchers most commonly propose to take actions to hide statistical information [SWP00, CWL⁺14, ÖS14, CGPR15, FES⁺17].

To obfuscate the query access pattern, Örencik and Savaş [ÖS14] propose inserting fake entries into the database so that two identical queries may match different entries. However, to prevent the correlation of queries, the number of fake entries must exceed the number of authentic entries. Therefore, this solution introduces much memory and computational overhead. Another approach proposed by Song et al. [SWP00] is the periodical re-encryption of indices to prevent long-term statistics, though this will not hide the number of matches for a particular query leading to the vulnerability of query recovery [CGPR15].

Research shows that CSPs can recover queries and document data using statistical information on access patterns, even in the honest but curious model. Additionally, when outsourcing data, we should consider the increased likelihood of FI attacks. Therefore, it is necessary to hide access patterns and prevent FI by strengthening authorization requirements to protect owner and user privacy.

3.2.3 Achieving Sublinear Efficiency on Queries

Horizontal scalability can improve performance by parallelizing queries among multiple CSPs, but it is limited by the underlying efficiency of the search algorithm. Consequently, in this subsection, we cover privacy-preserving techniques achieving sublinear efficiency in SE. We introduce the notion of deterministic encryption to

analyze its privacy issues and highlight approaches balancing privacy, efficiency, and usability trade-offs.

3.2.3.1 Deterministic Encryption

In deterministic encryption, the encryption algorithm will always yield the same ciphertext for a given key and plaintext. While this is common for some symmetric encryption schemes, it is not for asymmetric encryption, as schemes like Elgamal [Elg85] explicitly use randomness to encrypt data. Bellare et al. [BBO07] were the first to adopt deterministic encryption schemes in public-key encryption to enable efficiently searchable encryption (ESE). In these schemes, random coin tosses are replaced by a pseudo-random generator based on hash functions, yielding comparable deterministic encryption results. The key benefit of those schemes is their efficiency compared to probabilistic encryption.

The proposed framework UnLynx from Froelicher et al. [FES⁺17] (cf. Section 3.3.2) uses deterministic encryption for PHI exploration. In their protocol, a deterministic tag of every attribute queried is derived using deterministic encryption of this attribute to an ephemeral key. Then, the CSP computes the exact derivation for every entry and matches them against the queried tags. Notably, a CA computes this tag in a distributed network. Therefore, their protocol does not benefit from increased efficiency, as the tag computation is time-consuming and must be computed individually for every record.

One vital problem of deterministic encryption schemes is the vulnerability to chosen-plaintext attacks. Suppose an attacker has access to the data and the encryption function; he can brute-force all possible keywords by deterministically encrypting and matching them against the document or index [BG84, SWP00, BBO07, IKK12, BHJP14]. Additionally, Cao et al. [CWL⁺14] mention that such determinism can lead to trapdoor linkability, which a CSP may use to attack keyword privacy using analysis of usage frequencies, as covered in 3.2.2.2. In the public-key encryption setting of Bellare et al. [BBO07], this attack model is much more complicated if the data has a large min-entropy, as the attacker has to attack each public key individually. However, in practice, most data has a small min-entropy making deterministic encryption unsuitable for sensitive data as too much information is leaked about the content of documents and keywords [IKK12, BHJP14].

To circumvent these problems, Song et al. [SWP00] proposed to increase the number of false positives matching a query, yielding indistinguishability among all keywords mapping to the same group, similar to k -anonymity. Researchers adopted this idea, especially in the research area of secure databases, referred to as bucketization [BBO07, Ray13].

3.2.3.2 Bucketization

This application of deterministic encryption is the design of indices with high collision, mapping multiple keywords to the same index. Inherently such indices meet the requirements of Song et al. [SWP00] to increase privacy by hiding the occurrence profile [BBO07]. Furthermore, it reduces the number of documents to process with

a less efficient scheme and is commonly used to increase efficiency [HKA19], but it can also be used as a privacy-preserving technology for k -anonymity [KLC17], as covered in Section 3.1.4.3.

Bellare et al. [BBO07] use bucketization to reach sub-linear search efficiency on encrypted data. They propose deriving deterministic tags on data with low min-entropy to map to specific buckets increasing privacy guarantees. Notably, the proposed bucketization is configurable, leading to a flexible and adjustable trade-off between efficiency and privacy [BBO07, KLC17]. Yuan et al. [YMM⁺17] adopt this procedure with optional tags, referred to as block index. This block index is computed using a part of the attribute as a blocking key and hashing it to the index domain.

Disadvantages of bucketization include a significant increase in data usage and possible loss of information [KLC17] if no additional index for that attribute is stored. On the other hand, if such an index is stored to support more granular queries, additional computations are required to filter false positives [BBO07]. Regarding scalability, updating the indexing structure is difficult, as all participants have to agree and update accordingly and (partially) re-evaluate the bucketization [Ray13].

Takeaway. To achieve sublinear efficiency in SE, researchers take advantage of deterministic encryption, reducing the complexity of searching to the lookup of an encrypted key. Usage of this technique introduces new privacy issues, which are less prominent but also affect PEKS schemes. To circumvent these problems, researchers propose adding false positives that must be filtered using an additional ciphertext or processed indistinguishable from positive matches. We conclude that privacy-preserving ESE is possible but requires careful balancing of privacy, efficiency, and usability trade-offs.

3.3 Distributed Health Data Networks

For the final section, we cover two schemes with privacy-preserving exploration capabilities and classify them according to elaborated taxonomy and security measurements from previous sections. Therefore, we focus on the underlying architecture, technology, and cryptographic primitives rather than the embedding network use case. Notably, other distributed health data networks (DHDNs) [SKD⁺13, CBP14] are not considered because they are applications of schemes covered [RTPM⁺19, FMTP⁺20, FMH⁺21], lack privacy-preserving exploration capabilities [MWM⁺10, KHA⁺14, KHG⁺14, VBL⁺14], or make usage of trusted authorities [CWJ⁺16, VN21] (cf. Section 3.2.1).

3.3.1 PCORnet, a Privacy-Preserving Cohort Discovery Framework

The Patient-Centered Clinical Research Network (PCORnet) was launched in 2014 by the Patient-Centered Outcomes Research Institute (PCORI) as part of a governor-funded initiative to advance the use of electronic health records in clinical research on a national scale [FCC⁺14]. Since then, multiple health systems have been developed to facilitate healthcare providers' networks and nationwide collaboration [KHA⁺14,

FMH⁺21]. One of the current systems, by Yuan et al. [YMM⁺17], supports privacy-preserving cohort discovery, which inherently includes data exploration. In the following, we will analyze this system, highlighting the strengths and weaknesses of the chosen trade-offs, building a basis for decision-making for the system design of our thesis.

PCORnet features a multi-owner single-user client-server model with a trapdoor-less search predicate. In their model, organizations act as data owners contributing data to a centralized data repository called the data hub, which acts as the server. Users can search the data by sending queries to the data hub, which performs the query without knowing the data stored and keywords queried. Figure 3.2 illustrates the entire process and provides an overview of the architecture.

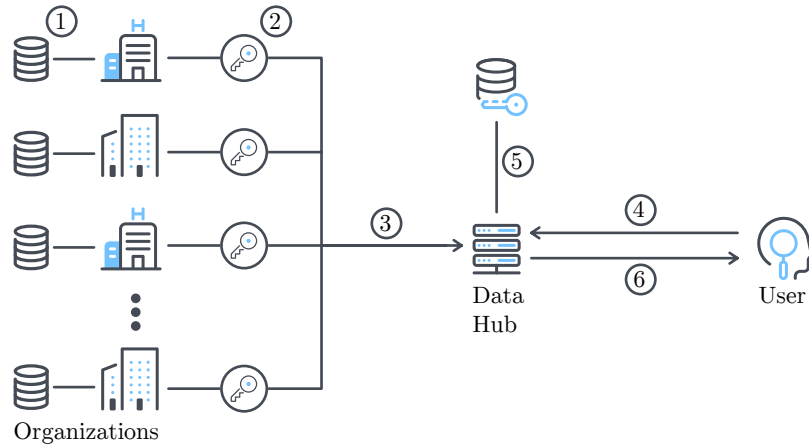


Figure 3.2 Architectural overview of the PCORnet framework. 1) Organizations store plain data on-site. 2) Organizations encrypt and blind their data for the data hub. 3) The data hub stores all encrypted data locally. 4) The user sends a query with encrypted keywords. 5) The data hub matches the query with stored encrypted data. 6) The user receives the query result.

All owners consider their data private and only share them after encrypting with the servers' public key and blinding it with a secret value unknown the data hub does not know. As a result, the data hub cannot decrypt any data an organization provides. That also holds for the representation of a query, yielding no information about the factual keywords. Thus, the server can only match indices against a user-provided predicate and cannot serve any plain data. Technically, a privacy-preserving polynomial evaluation facilitates this computation. In particular, the framework prohibits any disclosure of data to the user, limiting result reports to aggregated count values.

The security analysis considers the data hub to be honest but curious, with the additional requirement not to store or log any queries. This additional restriction prevents the possibility of statistical analysis by design, leaving the question of how to achieve such an obligation open. Nevertheless, it does not prevent keyword guessing (KG) attacks, which the authors mention but consider infeasible due to high computational costs and the possibility of auditing procedures. Entities other than the data hub are potentially malicious. Principally, organizations can perform file injection (FI) attacks to learn about other contributors' data. The publication has no statement about potentially colluding entities. Thus, the data owner and the user may cooperate.

As previously mentioned, result reports are restricted to count aggregations. Consequently, the user is limited to filtering data counted. Therefore, the user can match individual attributes and combine the predicate evaluation with boolean operators AND/OR. Figure 3.3 illustrates query capabilities in a pseudo Structured Query Language (SQL) syntax diagram.

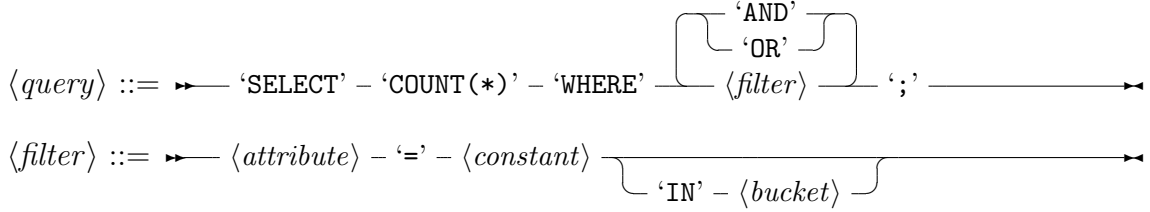


Figure 3.3 Exemplification of PCORnet’s query expressiveness.

For efficiency increase, attributes can optionally make use of block indexing. In this case, users and owners extract a blocking key from keywords, and its’ hash, called block index, is sent along with the predicate. The server uses this block index for bucketization, achieving sublinear efficiency. Notably, this partially reveals information about the entry and the efficiency–privacy trade–off must be considered for every attribute individually based on the sensitivity of data stored, as covered in detail in Section 3.2.3.2 For their framework evaluation Yuan et al. uses three publications [GTA⁺14, HTHB14, CRMC15] with cohort descriptions on the Healthcare Cost Utilization Project (HCUP) Nationwide Inpatient Sample (NIS) dataset [ONL].

Due to the centralized data storage and evaluation on the server, organizations must share their data in encrypted form. As noted in Section 3.1.4.2, this might not be possible, especially in intergovernmental collaboration [CSK⁺13], limiting the system’s scalability. Furthermore, we find that the server must be fully trusted. Considering the statistical analysis covered in Section 3.2.2.2 and that no countermeasures are proposed to either increase the low–entropy attribute’s entropy or hide the access pattern, a KG attack might not be as computationally expensive as the authors expect. Also, the server can perform FI attacks which makes the user vulnerable to query recovery. Colluding organizations and users might perform queries on individuals bypassing security measures that reject queries with low data samples [FES⁺17]. Notably, there is no method of limiting access to specific users from an organizational point of view, and the data hub handles different data owners indistinguishable, making this scheme infeasible for distributed systems with group collaboration [HKA19].

Summary. PCORnet features a multi–owner collaboration framework for privacy–preserving cohort discovery queries. To share data, organizations push encrypted and blinded data to a trusted centralized data repository capable of performing queries using privacy–preserving polynomial evaluation. To increase performance attributes can make use of bucketization. The basis of security guarantees is that the central server cannot decrypt stored data, unspecified auditing procedures, and a more trustful ideal than the honest but curious model.

3.3.2 UnLynx, a Decentralized System for Privacy-Conscious Data Sharing

As previously covered, PCORnet features strong efficiency in a centralized network with solid trust assumptions. Oppositely, UnLynx by Froelicher et al. [FES⁺17] is designed with reliable, nearly trustless security and privacy guarantees at the cost of efficiency and performance. Like the PCORnet publication, UnLynx has practical applications in privacy-preserving data exploration [RTPM⁺19] and cohort discovery [FMTP⁺20]. Consequently, it is convenient to choose as a complementary basis for future decision-making and comparison.

UnLynx features a multi-owner setup constituting a distributed database interconnected through a clique of servers forming a collective authority (CA). Owners are represented as data providers (DPs) storing plaintext data on-site and actively participating in queries. Therefore, the DP connects to a server of his choice, which will be responsible for interfacing the DP with the distributed network. Opposing to multi-CSP approaches, the purpose of this multi-server setup is not to scale computational power horizontally. Instead, the CA acts like a single server managing a collective cryptographic key using secret sharing [Sha79]. Hence every server operation must be performed by a collaborative protocol between all CA members. Figure 3.4 illustrates an exemplary instantiation of the architecture.

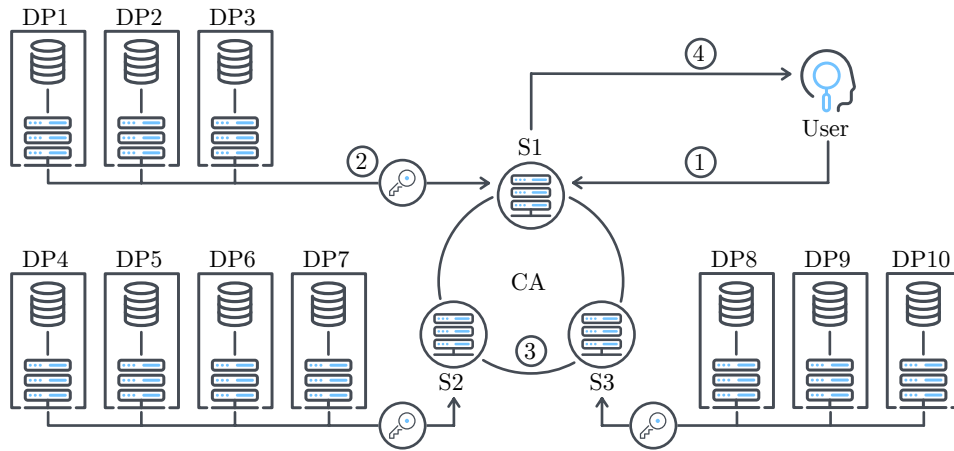


Figure 3.4 Architectural overview of the UnLynx framework. 1) The user connects to a server and announces the query, which the server forwards to all servers and corresponding DPs. 2) Participating DPs respond with data encrypted under the CAs' key. 3) All servers participate in distributed protocols to evaluate, obfuscate and encrypt the result under the users' key. 4) The users' chosen server responds with the encrypted result.

To inquire the distributed database a user chooses a server of his choice and announces his query to the network. In particular, the user sends the query to the connected server, which will then broadcast the query to all data providers directly or indirectly if connected to another server in the CA. DPs will then choose to contribute data or reject participation for the specific query. Notably, users and DPs encrypt data passed under the collective key. That means a server can only decrypt data if every CA member is willing to do so. After receiving data, all servers collaborate in a distributed protocol sequence to evaluate the query. First, servers shuffle their data

to break the link between DP and server. Next, the queried attributes and all data records are re-encrypted under an ephemeral key using deterministic encryption, making ciphertexts of attributes checkable for equality. As a result, each server can match the query predicates with a subset of the data leveraging a partial view of records matching the query. In the case of aggregation queries (cf. Section 3.3.2), the distributed protocol sequence requires two additional steps to aggregate and obfuscate the results using homomorphic encryption and differential privacy. Finally, the solutions' ciphertext is re-encrypted from the ephemeral deterministic encryption to the users' key and transmitted by the users' chosen server. Figure 3.4 illustrates an overview of the query evaluation protocol and participating entities.

UnLynx expresses queries by a pseudo SQL syntax featuring selection, aggregation, and filtering of records. Supported aggregations are **SUM** and **COUNT**. Users can restrict data providers to a subset of those available using a **FROM**-clause to filter data. Additionally, a **WHERE**-clause enables the filtering of records by **AND/OR** compositions of boolean equality checks on records' attributes, identical to the filter expressiveness of PCORnet (cf. Section 3.3.1).

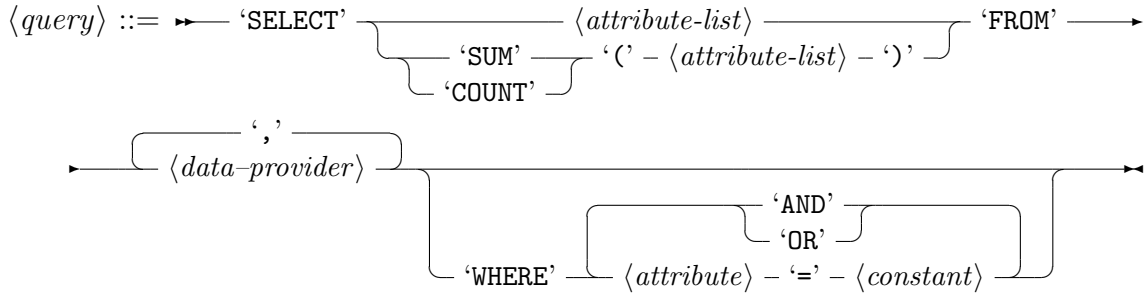


Figure 3.5 Exemplification of UnLynx's query expressiveness.

In addition to query capabilities, data providers can refuse query participation, accelerate query evaluation by transmitting plaintext or increase privacy by providing dummy values. The latter is realized by an additional encrypted flag, marking authentic records oblivious for the server. After the shuffling, records can be filtered by the flag using the same procedure as for other attributes.

Due to the usage of a distributed CA and privacy-preserving evaluation of queries on encrypted data, UnLynx has strong security and privacy guarantees in an AnyTrust model [WCGFJ12], requiring only one server to be none malicious to protect owners' and users' privacy. The verifiable shuffle guarantees the *unlinkability* of data and its corresponding providers. Furthermore, data stays encrypted, preserving data *confidentiality* during the entire process. In the context of searchable encryption (SE), the publication lacks proof of security guarantees in standard security models. However, the system might fulfill the requirements of IND-CKCA and IND-KGA against external attackers because of the non-deterministic and perfectly hiding public-key Elgamal encryption scheme. An attack based on statistical analysis is more challenging to execute, as differential-privacy usage hides concrete results, and servers may get inconsistent results of different data records observed due to the shuffling.

Additionally, DPs might actively reject participation or insert dummy data, leveraging increased security and privacy guarantees in control for the data owner. By rejecting participation, the DP can limit user query expressiveness and forbid using specific attributes or too isolating filters. Using dummy values can hide the data count from the connected server, preventing statistical analysis or framing individuals. However, their publication does not cover the possibility of FI attacks from a server, which is possibly malicious. The data provider only guarantees the integrity and authenticity of records to one server. Therefore, servers may modify or inject additional data to compromise results or perform attacks on observed data. Reusing old data is also possible, as data does not get obsolete, leaving the protocol vulnerable to replay attacks.

In addition to the security concerns named above, the system inherits limitations from the protocols and network architecture chosen. Like PCORnet, UnLynx requires a data provider to share encrypted data to a server, which may limit capabilities due to regulations, as already mentioned. Moreover, the necessity to evaluate most of the protocols under the collective key results in high communication overhead and slows down the system significantly. That is especially troublesome, as the system is not secure channel free (SCF) requiring more communication and computational overhead for data transmission. Using deterministic encryption to ephemeral keys also prevents more efficient search algorithms, not allowing the construction of block indices in advance. Consequently, this scheme misses efficient search capabilities with no additional index for bucketization or any different approach.

Summary. UnLynx is a multi-user and multi-owner data-sharing scheme using a collective authority to strengthen security and privacy guarantees. The schemes utilize cryptographic protocols such as verifiable shuffle, deterministic encryption, and key switching to retain security properties under an AnyTrust model from the data provider to the user at the cost of performance.

4

Problem Statement

Access to a significant amount of electronic health records (EHRs) has increasingly more value in improving public health in areas like systems medicine and clinical research [CSK⁺13, CBC⁺17, KRN⁺19, DMTB21] to facilitate public health [BHCW10, WB11]. At the same time, more and more data is accessible due to digitalization [KRN⁺19, VHVJ20] and the emerging adoption of IoT devices [PCKK17]. However, the sensitivity of healthcare information limits usability and the possibility of collaboratively sharing such data due to high constraints from governmental regulations (Section 3.1.2). As a result, the increasing availability and demand for protected health information (PHI) puts existing systems to the test and requires cautious actions to balance research needs and patients' privacy.

This chapter discusses how we adopt the current state of privacy-preserving health data sharing to tackle these challenges in health data sharing. Therefore, we highlight issues of existing solutions and complement those with recent research in this field. Based on solutions acquired, we then formulate objectives for a system design solving current limitations. In the following, we discuss the current state of research and technologies deployed to identify the research gap and frame our problem scope.

4.1 Problem Scope

In Section 3, we introduced the legal and ethical framework for health data sharing and its resulting challenges with a focus on privacy, security, and scalability. We highlighted elaborated solutions to these problems using dynamic consent and searchable encryption. Furthermore, resulting side-effects and trade-offs have been covered together with actions to address these problems. Finally, we highlighted the research of PEKS systems suitable for protected health information (PHI) with elaborated security models and their trade-offs on performance and privacy. However, existing distributed health data networks (DHDNs) analyzed miss on the adopting these well-studied solutions, leaving a gap between research and adoption in data sharing.

In Section 3.1.4.2, we showed that in addition to incompatible data representation, federal policy incompatibilities limit scalability if the system requires to share PHI. Still, analyzed systems exchange encrypted data in order to explore relevant records. In addition to the resulting security difficulties (Section 3.2.1), it can constrain data discovery to governmental boundaries or require extensive expanses for regulations and contracting to fulfill policy compatibility.

We further discussed elaborated security models and countermeasures to strengthen security privacy guarantees in Section 3.2.2. However, our analysis of existing systems in Section 3.3 showed that those measurements need to be applied and known attacking vectors are practical. Contrarily, those systems rely on the availability of secure channels and auditing mechanisms rather than schemes proven to be secure. Moreover, we have shown in Section 3.1.2 that external databases can re-identify individuals using even highly de-identified data. As a result, not only are patients vulnerable to the increasing amount of hacking and information technology (IT) incidents but also systems demand additional capital expenses further limiting scalability as it requires more effort to securely scale the network (Section 3.2.2).

In Section 3.1.2, we discussed another aspect of scalability covering the problems of the informed consent requirement. We further highlighted the concept of dynamic consent in Section 3.1.3 using the availability of new technologies to solve the limitations of traditional consent systems. The adoption of dynamic consent increases patients' willingness to participate in health data sharing and facilitates researchers to obtain re-consent efficiently [CDD⁺17] for new use cases. Nevertheless, the networks covered needed to have built dynamic consent capabilities into their systems.

Although none of the systems analyzed has privacy-preserving data discovery without data sharing, using schemes proven to be secure in standard taxonomy or dynamic consent capabilities, we reviewed research featuring technologies facilitating such obligations. In particular, we reviewed owner-centric schemes in multi-user environments capable of providing provable secure, distributed data exploration (Section 3.2.1). Additionally, in Section 3.2.2, we introduced the notion of secure channel free (SCF) communication suitable to scale networks with reliable security guarantees and fewer expenses for infrastructure and auditing. Therefore, applying these techniques will result in a more secure DHDN with better scalability while giving the patient more trust, control, and transparency in their data usage.

4.2 Objectives

In the previous section, we identified the system's requirements, challenges, and solutions. Based on this analysis, we can formally specify the capabilities and properties required for a system to address these issues effectively.

4.2.1 Functional Objectives

We divided the desired capabilities of the system into three distinct categories, which we refer to as functional objectives. These categories are independent of one another, representing a unique aspect of the system's functionalities.

F1 Data Discovery. Like existing health data networks, the system should be able to perform queries featuring data selection and filtering. Filtering should be possible on an attribute basis equality check to constant and boolean combinations of those similar to the query model in Section 3.3. In addition, filtering based on a data source identity should be possible, as supported by the UnLynx system (Section 3.3.2).

F2 Distributed System. The system should be distributed so that every data provider can process a query independently.

F3 Dynamic Consent. Protocols should provide basic interfaces facilitating the requirements of dynamic consent. That is the patients' capabilities to dynamically configure their consent, review their data usage and interactively participate in the process. Notably, the users' configuration should transparently reflect the usage intent of users shared with and give fine-granular options on which data to share with whom, as covered in Section 3.1.3.

4.2.2 Non-Functional Objectives

In addition to the system capabilities, the non-functional objectives state desired properties of our system. In contrast to the functional objectives, these properties are not independent and partially conflict with each other. Therefore, assessing conformance to these properties and trade-offs taken is the basis for evaluation.

N1 Security. All protocols should realize tools to achieve *confidentiality* and *integrity* for every operation. In more detail, only authorized users should be able to issue queries, and only legitimate data providers may perform or answer the query. Concerning F3 DYNAMIC CONSENT, this includes a non-trivial access control scheme respecting the patients' choice of whom to share their data.

N2 Privacy. The system should be designed to be privacy-preserving and allow non-organizational individuals to participate authenticated but anonymously. Additionally, state-of-the-art techniques covered in Section 3.2 should be utilized to protect against data recovery and patient re-identification, even in the case of a data breach.

N3 Scalability. In order to scale the system in a distributed network, all operations must scale efficiently with the number of owners, providers, and users. Additionally, the data provider should not share patient data as; otherwise, the system scalability is limited by governmental policies interoperability, as covered in Section 3.1.4.2.

N4 Performance. For practical usage, the system should be performant enough to evaluate queries on real-world data sets in a reasonable time. The system should exploit efficiently searchable encryption (see 3.2.3) and vertical and horizontal scalability, utilizing multi-core architectures and cloud service providers. Notably, this objective contradicts N1 SECURITY and N2 PRIVACY. Therefore, it must be configurable to meet individual and possibly changing security and privacy requirements as new methods and technologies evolve.

4.3 Summary

We highlighted current challenges and issues in existing solutions for health data sharing and recent research addressing those. Based on that, we combined exist-

ing solutions and developed a list of objectives to design a system that puts this combination of features and adoptions to the test. Following Chapter 5, we present contextual assumptions and design decisions framing the environmental conditions of our system. In continuation, we propose a system fulfilling the proposed objectives in Chapter 6, which is then evaluated in Chapter 7.

5

System Design

In this chapter we introduce and formalize a privacy-preserving data discovery distributed health data network with dynamic consent that aligns with the goals set in our problem statement (cf. Section 4.2.1). To build a comprehensive view of the system, we explain the system in a sequence of three progressive sections. In the first section, we start with the general idea and fundamental principles carving out a formal description of required entities, their workflows, and communication network. From that, we continue to introduce data structures and algorithms facilitating the design of our protocols in alignment with our non-functional objectives. In particular, we show how we can adapt existing technologies of bucketization and public key encryption with keyword search (PEKS) schemes to our custom requirements. Finally, we conclude this chapter with an in-depth description of these protocols used to implement workflows in the first section.

5.1 System Model

In this section, we introduce a privacy-preserving system designed to address the functional goals of data discovery (F1), distributed network (F2), and dynamic consent (F3). Therefore, we start from a conceptual idea and specify the resulting actors and tasks with increasing formality. This way, we build a good understanding of the basic principles we apply and introduce the intent for upcoming architectural decisions. We continue with a schematic overview of actors' workflow and state all entities required to fulfill previously worked out principles in the distributed health data network (DHDN). This workflow will show the required communication of entities in the network, thus enabling us to introduce the network architecture in the last section.

5.1.1 Conceptional Idea

Before we detail our design decisions and protocol specifications, we introduce the most fundamental concepts of our model. These concepts are chosen to align with our non and functional goals and will facilitate our system development. In the following we explain each of those concepts in detail.

Search Control. For this principle, we design the data owner to be able to control search access to their data. In other words, a patient should be able to control the usage of their data in network-issued queries actively. To achieve this control, we take advantage of the research in public key encryption with keyword search (PEKS) [RPSL10, YM16] (cf. Section 3.2).

In trapdoor based PEKS schemes, the capability to search encrypted data is in control of the data owner (DO) who shares a secret to the data user (DU) in order to create a trapdoor required to match this data. The owner’s capability of controlling who can search their data aligns with the requirements of dynamic consent. However, the exchange of a secret demands communication between the DU and DO, contradicting scalability. Furthermore, this construction binds the trapdoor to a specific owner and would require trapdoors for each record, resulting in quadratic complexity. We mirror the idea to solve this scaling issue so the user can issue server-dependent trapdoors. However, the server cannot match data without the owner’s corresponding consent. Therefore, the DO actively consent to a query by sending required extra information to match their data on the server. In other words, the server has no access to the encrypted index but rather to a fragment of it and requires the consent of the owner in order to generate an index bound to the query the owner consents. We call the witness required by the DO to generate a consent a consent witness and require it to be different for every data value a DO obsesses. This way, consent must be given for every value individually, and a DO may only consent to a subset of, in their opinion, non-sensitive information [BLTK⁺17, CDD⁺17] aligning with our reviewed research on dynamic consent to facilitate participation (cf. Section 3.1.3).

Notably, such fragments and consent witnesses may be similar in size but have important distinct attributes. The fragment is an encrypted health data value. A consent witness is a secret required to consent the usage of encrypted health data. Hence the DO does not store any health information on a possibly insecure device. In addition, future work may develop new types of indices, e.g. to process genetic data. According to our definition, such adaptation complies with our design of search control if one can extract a fragment and a consent witness.

Ephemeral Receiver. As the previous part states, binding a trapdoor or index with a DO and DU pair results in quadratic complexity. In contrast, encrypting indices for a single DU makes the indices vulnerable to *keyword guessing attacks*. The DU can generate trapdoors and match the indices. Ergo, if an attacker obtains patient data, they generate trapdoors for all possible values and *brute-force* a match. One common way to solve this problem is to bind the trapdoor and index to a test server (TS). Although this is suitable for the DO because the TS and DO are in a 1-to-1 relation, it is not for the DU. The reason is the rising complexity in encrypting a trapdoor for multiple TSs in a distributed network. Our solution is not to bind the trapdoor to any other entity but the DUs themselves. Challenges of this approach include; first, the DU can generate trapdoors for all possible values to perform *keyword guessing*

attacks, and second, anyone with access to the DU public key can generate indices to brute-force the trapdoor. For the former problem, we use a collective authority (CA), creating a new collective ephemeral key for every query. In a sequential protocol, the DU has to dial every CA member to construct trapdoors using that collective key. Thus, as long as at least one member of the CA acts honestly, no one can issue new trapdoors using this key. As a result, when a DO consents to a query, they can be sure that their data stays secure against *keyword guessing attacks* in the any-trust threat model, solving the first problem. Furthermore, we use the proxy server described in the following paragraph to solve the second one.

Proxy Server. We deploy a proxy server as a gateway utilizing re-encryption to forward tasks without revealing sensitive information that may enable other entities to perform certain attacks. This approach has two advantages. First, even if the proxy server can attack a trapdoor, it only acquires information about the query. However, it cannot derive patient information because it holds no patient data and cannot perform queries. Second, the server can simulate DOs for artificial data. That is because the server does not learn the number of records matched and can thus not perform statistical analysis (cf. Section 3.2.2.2). Therefore, the data provider (DP) is a great candidate to manage artificial data on dedicated TSs, which may perform statistical analysis. Notably, the forwarding of a query is conditional. Thus, a proxy server can reject queries to enforce policies.

Artificial Data. In order to solve the problem of low diversity [MKG07], and anonymity [Swe02] for existing datasets and to hide results count to protect against statistical analysis [ZDO⁺08], we deploy artificial data. This kind of data is artificially generated and deployed alongside real patient data. Naturally, to take effect, artificial data must be indistinguishable from real data for the holder, but this indistinguishability needs to be corrected in evaluating results. For this reason, every data record gets an extra stealth bit column indicating whether it is real. Encrypting this column with a non-deterministic encryption scheme under the CA public key prevents the TS from distinguishing real and artificial data, though it is still capable of matching the record. However, telling if a matched record is real would require a collective protocol with the CA inducing quadratic complexity in the number of CA members and records, contrary to our performance goal. Therefore, we aggregate the results beforehand using a homomorphic encryption scheme.

Homomorphic Result Aggregation. This aggregation type folds the number of matched records before sending them to the CA. Here we can use the ElGamal encryption for the stealth bit to add all matching records and get exactly the number of real records but encrypted under the CA public key. Note that this way, neither the proxy server nor the TS can obtain the number of matched records, and the CA will first aggregate the result of all DP before handing the result over to the DU. Hence forwarding any count on data is always secure in the any-trust thread model, and the DU will only learn the accumulation of all DPs.

Summarizing, we introduced five core concepts of our design. Search control allows the DO to take control of the usage of their data by keeping a consent witness required to match their data against a query. The collective ephemeral receiver key ensures that one cannot create new trapdoors once a query is issued, preventing keyword guessing (KG) attacks on indices bound to a query. To enforce query policies and enhance privacy by splitting sensitive data for multiple entities we introduced

a proxy server that is a gateway between the network and TSs. Additionally, we increased privacy guarantees by introducing artificial data that obfuscates the real data indistinguishable in the any-trust thread model. Finally, we have shown that we can still compute results for this obfuscated data set using the homomorphic addition of stealth bits.

Now that we have a conceptual overview of principles to deploy, we can continue with a schematic approach to how certain tasks and workflows may look and which entities are involved.

5.1.2 Schematic Overview of Actors and Tasks

This section presents a conceptual overview of the actors involved in the distributed health data network (DHDN) and their associated tasks. This overview provides a high-level understanding of the network's structure and the roles played by different entities. We divide the common tasks into four workflows, as shown in Figure 5.1, to simplify the understanding of the DHDN. These workflows encompass the various stages involved in the lifecycle of a query within the network.

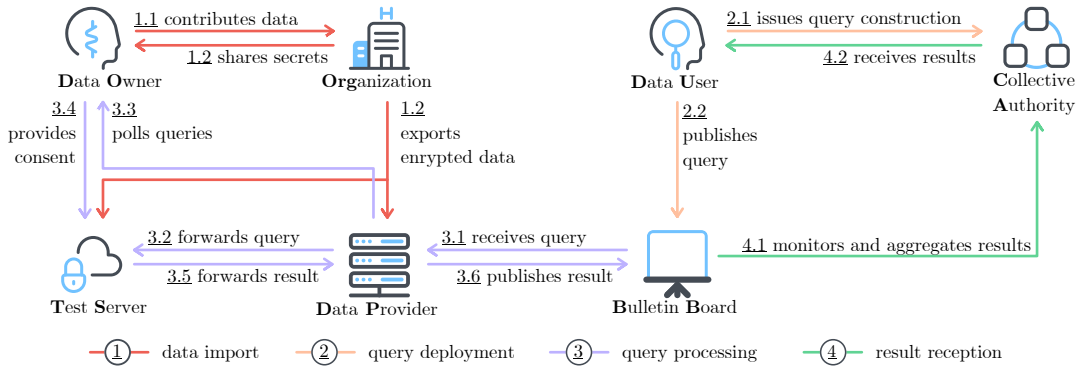


Figure 5.1 Schematic overview of workflows in the DHDN. A separate color indicates every workflow. Note that all entities represent a group of those entities. Arrowheads indicate the direction of data flow.

Data Import. This workflow focuses on preparing health data for processing in the DHDN. Therefore, the DO *contributes data* to an organization that maintains its data. The major task of maintaining data is encrypting it and making it available to the DHDN. Therefore, the organization *exports encrypted data* to dedicated nodes, called DP and TS, which will actively participate in queries issued to the network.

Query Deployment. As mentioned in our conceptual overview, the construction of a query is a collaborative effort involving a researcher, referred to as the DU and the CA. In the query deployment workflow, the DU engages in an interactive protocol that *issues query construction* to all members of the CA to create a valid and authenticated query. Once a valid query is obtained, the DU *publishes the query* to a bulletin board, making it publicly available to all DPs for processing.

Query Processing. The Query Processing workflow begins with individual organization's DPs receiving the queries from a bulletin board. DPs then *forward queries* to

the TSs holding the data. In order to match the records to the query TSs have to wait for DPs to *poll queries* and *provide consent* for them. Once data is processed, or a certain deadline is reached, TSs *forward their results* to the DP. Following, the DP aggregates TSs results and *publishes the aggregated result* back to the bulletin board, completing the processing step of the query. It is important to note that query results will be publicly visible on the bulletin board, emphasizing the need for data encryption and anonymous publication to prevent statistical analysis of specific DP behavior, such as response time.

Query Reception. In this final stage of a query’s lifetime, all members of the CA provide the query result to the DU. Therefore, CA members *monitor and aggregate results* from the bulletin board and make it available for the DU to *receive the results*.

These four workflows outline the key phases involved in the DHDN, from data import to query deployment, processing, and reception of results. In the following sections, we will delve into a detailed exploration of each workflow, providing comprehensive descriptions of the entities involved and explaining the network architecture. Subsequently, we will discuss the protocols, cryptographic techniques, and privacy-enhancing measures employed within the DHDN to ensure health data’s secure and confidential processing.

5.1.3 Entities

Continuing our schematic overview, we elaborated a conceptual view of the entities mentioned and introduced some new entites that reflect organizational structures but are not a direct part of the network. Now we focus on more details of the entities before finalizing the network architecture. Therefore, we describe every entity’s intent, capabilities, responsibilities, and threat model. Additionally, we introduce the conventional notation required to formally state our protocols later.

Organization. Within our design, an organization is a legal institution, such as a doctor’s facility or hospital, which collects and stores plaintext of patient health data. Its’ intent is to contribute this data to the DHDN without sharing it with other organizations while leaving control of data usage to the patient. The organization has access to a dedicated part of the DHDN infrastructure to contribute data to the network. That part comprises a designated DP and one or multiple TSs. Maintaining data on those entities is one of the major organizations’ responsibilities. This node management includes providing encrypted data, shuffling existing data, and hiding access patterns and result counts by providing artificial data. Additionally, the organization must host a service to transfer consent witnesses confidentially to DOs. The fact that the organization holds patients’ plaintext data shows that the DO must fully trust its’ organization. However, this is generally the case and a pre-requisite for treatment. On the other hand, the network classifies the organization as honest but curious.

Data provider (DP). The DP is a node of the DHDN designated to a specific organization but not necessarily hosted by that. As part of the network, it has its’ own identity (PK_{DP}, sk_{DP}) but can also act as part of one or more research networks, as those share a common keypair (PK_{RN}, sk_{RN}). In our system design, the DP is a proxy server. In other words DPs act as a gateway ensuring organization policies and

forwarding data to DOs and TSs without revealing critical secrets owned to compute this data in the first place. Particularly, DOs get the required information on queries to consent, and TSs get a re-encrypted version of queries with the unknown public key. In addition, the DP is responsible for gathering results from all TSs of the organization and publishing them to the DHDN. Please keep in mind that the proxy server (DP) has no access to health data and cannot obtain any intermediate result (cf. Section 5.1.1). Last, the DP must also act as multiple DOs in the network to simulate participation for artificial data. However, an independent server may also achieve the latter and is not particularly bound to the DP. Rather we chose to do so for simplicity, reducing the number of entities involved. From a security perspective, the DP is trusted, although we will consider the case of an honest but curious thread model in our security analysis.

Test server (TS). TSs are organization specific hosts responsible to evaluate queries on DOs' health records and forward aggregated results to the organization DP. Therefore, they store encrypted index fragments and monitor an organization-specific broadcasting channel for DOs' consent. These index fragments are bound to that specific TS, which possesses the keypair (PK_{TS}, sk_{TS}) , required to match any data (cf. Section 5.1.1). Notably, the TS only acts behind the DP gateway. Ergo, those are not part of the DHDN and do not necessarily require access to its network. Consequently, TSs are always assigned to a single DP, and one DP may have multiple TSs attached to it. Nonetheless, considering TSs as part of the DHDN may have a positive impact on the resilience and security of a distributed peer-to-peer (P2P) network [VNM⁺20]. However, this requires additional security analysis because TSs are classified, to be honest, but curious and may monitor networks to obtain insights. In any way, unlike the DP, the TS has no globally known identity aligning with our scalability goal, such that protocols in the DHDN are independent of the number of TSs.

Data owner (DO). The DO entity represents an owner of a particular data record with identity (PK_{DO}, sk_{DO}) . Here we intentionally give a generic definition. That is because a DO can be represented by various actors. In the previous sections, we mentioned two patients and the DP managing artificial data. One problem of this view is the illusion that an individual's identity is bound to a specific DO, but we require DOs to be anonymous and unlinkable by design. For that reason, we distinguish individuals from DOs and view them as a single globally unique identity managing a single record in the entire DHDN. For example, this managing entity can be a patient or another service handling artificial data. Nevertheless, in practice, an individual may be required to manage multiple records. In that case, the individual is responsible for managing multiple DO identities, which are anonymous and unlinkable. For example this can be achieved by a key derivation function (KDF) [KE10] or hierarchical wallet scheme [FTS⁺20] providing anonymity and unlinkability guarantees [C⁺08]. That being said, from now on, we will interchangeably call the owning identity of a record and the individual managing this identity data owner. A DO intends to contribute their data by participating in the DHDN query evaluation. To do so, they store their data in plaintext at an organization and legitimate them to encrypt and contribute their data to the network. After that, the responsibility of the DO is to poll the DP for new queries and provide consent to the TSs broadcast channel as well as poll the organization web service to get updates on their required witnesses. Importantly, in all of these interactions, the DO must be considered possibly malicious, i.e. the DO may monitor all data visible to them or activate abuse protocols to attack the

network. Another point of particular interest is the online capabilities of the DO. We consider the DO managing device as a cold client, i.e. other entities cannot dial the DO, thus the DO must establish every connection. Though DOs are not limited to such devices, we expect this as the bare minimum. In practice, this device might be a smartphone with a designated app to interact with our system.

Research network (RN). Like the organization, a research network is a legal institution that does not actively participate in any network protocols. It represents a group of entities used as escrow to enhance scalability and anonymity among this group. In particular, DPs can organize themselves in RN such that a query can be constructed for a single RN rather than for every individual member. Other beneficiaries of this RN structure are DUs, who can use RNs to authorize themselves as part of that group without revealing their identity. Moreover, RNs has globally visible information attached to its identity. The idea is that DO can have more granular options for whom to consent. Therefore, this information should align with elaborated criteria of high importance for DO decision making [KBK⁺16] (cf. Section 3.1.3).

Data user (DU). One entity that authorizes itself as part of a RN is the DU. They issue queries and gather and analyze their results from the DHDN. Therefore, they have to engage in a multi-round interactive protocol to create a query, publish it to the DHDNs' broadcast channel, and actively poll the network's hosts for results. From a network perspective, the DU is very similar to the DO. Both are not part of the network, must dial hosts to participate in protocols actively, and are considered possibly malicious. One parity of particular importance is the use of an unlinkable identity. That is, the DU identity (PK_{DU}, sk_{DU}) is not an individual's identity that issues a query but an ephemeral identity unique to a query. In other words, the DU generates a new identity for every query and authorizes this identity as part of a research network, e.g. by signing it with sk_{RN} . Consequently, we interchangeably call the authorized identity of a query and the managing individual data user.

Collective authority (CA). The collective authority is a group of s members $CA := \{CA_1, \dots, CA_s\}$ managing a collective key $sk_{CA} = \sum_{t=1}^s sk_{CA,t}$, where each member $CA_t \in \bar{s}$ is the only one with access to the keypair $PK_{CA,t}$. Responsibilities of the CA are the transfer of CA encrypted results to the DU and the creation of trapdoors under an ephemeral cumulative key (cf. Section 5.1.1). Although single CA members may act maliciously, the CA itself is classified in any-trust thread model. In other words, the CA can only act honestly as long as one member acts honestly. Note that this does not prevent single members from acting maliciously, such that consecutive protocols must validate responses from every member. Last, we assume that all CA members are publicly known, such that organizations, DPs and DUs, have access to such a list.

This section concludes the introduction of entities, highlighting every individual's intent, responsibilities, and threat model. It is important to mention that all entities, except the CA, are considered none colluding, In other words, no two entities may contribute to a coordinated attack on the network. Summarizing we have organizations that contribute encrypted data to an organization-specific part of the DHDN, namely a DP and TSs. In addition, the organization transfers consent witnesses to the DOs, which will then be used to legitimate TS to match their data to a bounded query. To coordinate this interaction, the DP acts as a gateway forwarding query information to the DP and TS and aggregating TSs results under a collective

key. The CA manages the required collective key in the any-trust model to ensure the confidentiality of query results and guarantee the construction of trapdoors for ephemeral receivers. Consequently, the DU interacts with the CA to construct queries, issues them to the network, and polls the CA to receive forwarded results.

5.1.4 Network Architecture

For the previous sections, we introduced all network entities, their workflows, and the entities they communicate with. This section will present a network architecture aligning with our previous statements and enabling entities to engage in protocols. This architecture will be the backbone of our system, which we will utilize in the upcoming sections.

To implement such a backbone, we utilize a hybrid distributed architecture that uses Client/Server and P2P networking concepts [Sch01]. We generally differentiate two kinds of communication: calls based on the Client/Server concept and transfers, which use P2P networks.

Authenticated Call. In this model, both entities know each other and can identify themselves using a secure connection protocol, e.g. Transport Layer Security (TLS) [RD08].

Unauthenticated Call. For this approach, one identity dials another using a secure connection protocol. The difference is that the caller stays anonymous, thus, is not willing to identify themselves. Therefore, the network may facilitate this by providing an anonymous overlay network like Tor [DMS04].

Anonymous Transfer. We call the protocol of broadcasting a message transfer. Similar to the unauthenticated call, we expect transfers to be anonymous for the publisher. In this thesis, we denote such broadcast channels as bulletin boards.

Figure 5.2 shows all network communication channels based on previously described notation. Of particular importance is the interconnection of multiple hosts on the bulletin board. Here we use two bulletin boards. The *local bulletin board* is organization-specific and used only by entities related to that organization. On the other hand, the *global bulletin board* facilitates a broadcast channel between all DPs and CA members. Although these appear as different channels in the network, this does not necessarily induce that they are part of a different P2P network. We suggest using a P2P network protocol that supports multiple topics for messages, e.g. GossipSub [VNM⁺20]. However, in this case, entities of both bulletin boards may monitor and analyze traffic to obtain insights, which must be respected in security and privacy analysis.

Summarizing, we showed a hybrid distributed network architecture that facilitates the implementation of protocols for all entities. To achieve this, we presented three types of communication these are un-/authenticated calls and anonymous transfers. Additionally, we highlighted some protocols that enable such communication and enhance security and privacy guarantees. As a result, we have a backbone for our entities which we can further utilize for our protocols. Nevertheless, before introducing these protocols, we still require insights into the required algorithms and data structures facilitating their implementation.

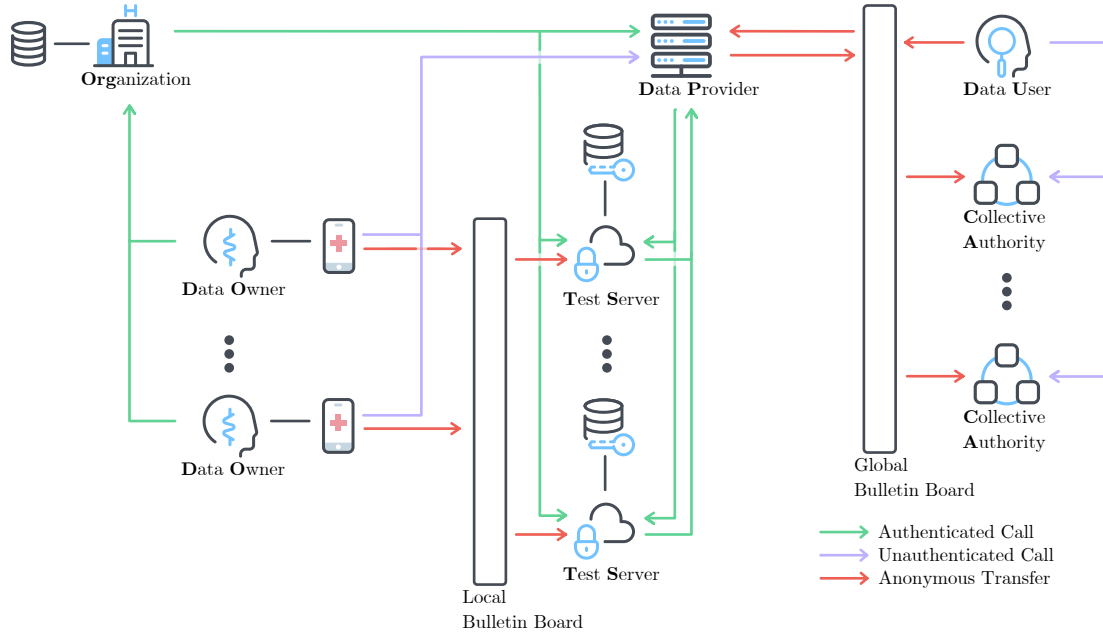


Figure 5.2 Here is an overview of network communication channels for all entities, exemplary for one DP. Arrowheads indicate the callee and direction of data flow for calls and transfers, respectively.

5.2 Building Blocks

Before continuing the formal definition of our system, we have to introduce some data structures and algorithms that are important for implementing our following protocols and unique to our application. First, we still need to include a formal definition of the health data. Continuing, we show two methods to process that data: our adaptation of bucketization and public key encryption with keyword search (PEKS) schemes. Here we also give insights into decision-making and how those adapted algorithms influence security, privacy, and scalability. From that point, we have a ground understanding of the data to exchange and can build data structures to facilitate our functional goals.

5.2.1 Structure of Electronic Health Records

Before we describe our algorithms in detail, we require a more comprehensive view of the stored data structure. Therefore, we formalize the structure of electronic health records (EHRs) aligning to our Nationwide Inpatient Sample (NIS) data sample we use for evaluation. Furthermore, we introduce notation and point out a special case where multiple data fields encode a list of categorized values.

First, we point out that the available formats and methods of storing EHR is a wide field of research with multiple competing standards [oSP20, Int22a, Int22b], not including data extraction. In the scope of this thesis, we expect that data has already been gathered and exported to a tabular structure, similar to the NIS file specification [NISb].

We consider a table as a list of n with every record being a tuple of m values $r_i := (\text{val}_{i,1}, \dots, \text{val}_{i,m})$ where each value corresponds to a unique field $\text{DF}_j :=$

$\text{FieldOf}(\text{val}_{i,j}) : i \in \bar{n}, j \in \bar{m}$ serving as an index. For a fixed data structure, indexing values by order of their specification is sufficient. However, generally, it is possible to have dynamic data fields as long as they stay unique. In addition, every field has an associated type $\text{DT}_j := \text{TypeOf}(\text{DF}_j)$ which corresponds to the domain of possible values for all fields of this type such that $\text{val}_{i,j} \in \text{DT}_j$ holds for all $i \in \bar{n}, j \in \bar{m}$. Loosely speaking, EHRs can be viewed as a table, where records correspond to rows and data fields to columns with associated types. We will now examine the data types stored, providing a closer analysis.

Typically, data types fall into one of two categories: a finite set of integers or strings, which may include additional symbols to encode ill-formed or missing data. These set definitions are typically found within the specification. In the context of this thesis, both data types can be efficiently stored and possess binary representations.

Lastly, multiple fields may be categorized using a common prefix. For example, the NIS dataset lists multiple diagnoses and procedures for a single record. Notably, it is not constructive to query each of these fields individually as their order has no attached meaning, and queries ask for a specific diagnosis or procedure rather than matching a specific index. This usage scenario is matched by disjunctive PEKS schemes where an index is built on a collection of values and can be matched if any of those values match the given trapdoor. However, to our knowledge, no disjunctive PEKS schemes have any efficiency gain over matching every value individually. The reason is that such schemes utilize polynomial equations of the degree of embedded values. Thus, evaluating a single trapdoor requires the evaluation of terms scaling linearly with the number of values. Contrary, the disjunction of multiple values downgrades the efficiency gain of bucketization, as we will see later. Nevertheless, sticking to single-value indices requires us to handle this case explicitly. Therefore, we also distinguish data groups. A data group DG is a list of fields that share the same type and are considered to list multiple values of that group. Note that we limit the possible length of such a list with a fixed upper limit, but the list can be variable in size by encoding null values with $\perp$, which is also used to encode missing or invalid values. Consequently, every field has a unique index it encodes in its group, starting at one, and we compute the group and index of a data field as $\text{DG}_j := \text{GroupOf}(\text{DF}_j)$ and $\text{DI}_j := \text{IndexOf}(\text{DF}_j)$ respectively. For fields that are not part of a list, this field is the only field of a corresponding group, and we further distinguish this edge case by assigning this field the index zero. Additionally, we denote $\mathbb{DG} := \{\text{DG}_j : j \in \bar{m}\}$ as the set of all groups and define $\text{TypeOf}(\text{DG})$ to be the shared type of all fields of that group.

5.2.2 Bucketization

As our performance objective (N4) states, we target sublinear search efficiency in the number of data records n . One method to address this goal commonly used in encrypted databases [MT06, BBO07, Ray13] and also adopted by PCORnet [YMM⁺17] is bucketization. Here we present our adaptation of bucketization specialized for the data structure defined in the previous section. More specifically, we present a variation to take data groups into account.

In bucketization, data values are indexed by a block index surjection $\text{BucketOf}_{\text{DT}} : \text{DT} \rightarrow \mathbb{N}$ mapping values to a block index $\text{BI} \in \mathbb{BI}_{\text{DT}}$. Here $\mathbb{BI}_{\text{DT}}$ refers to the

Field	Type	Group	Index	Description
AGE	$\{0, \dots, 124, \perp\}$	AGE	0	Age in years at admission
FEMALE	$\{0, 1, \perp\}$	FEMALE	0	Indicator of sex
DX_1	$\text{ICD10_CM} \cup \{\perp\}$	DX	1	ICD-10-CM Diagnosis
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
DX_40	$\text{ICD10_CM} \cup \{\perp\}$	DX	40	ICD-10-CM Diagnosis
PR_1	$\text{ICD10_PCS} \cup \{\perp\}$	PR	1	ICD-10-PCS Procedure
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
PR_25	$\text{ICD10_PCS} \cup \{\perp\}$	PR	25	ICD-10-PCS Procedure

Table 5.1 Exemplary data fields of the NIS dataset with their corresponding type and group. Here the types ICD10_CM and ICD10_PCS refer to the official list of International Classification of Diseases, Tenth Revision (ICD-10) codes for diagnosis and procedures, respectively. Descriptions are taken from the official website [NISa].

codomain of mapped indices $\mathbb{BI}_{DT} := \{\text{BucketOf}_{DT}(\text{val}) : \text{val} \in DT\} \subset \mathbb{N}$. The benefit of such block index is that for a given index, we can construct a reverse lookup $\text{ValuesOf}_{DT} : \mathbb{BI}_{DT} \rightarrow \mathcal{P}(DT)$ of values mapped to that index.

$$\text{ValuesOf}_{DT}(\text{BI}) \mapsto \{\text{val} \in DT : \text{BucketOf}_{DT}(\text{val}) = \text{BI}\},$$

Using such a reverse index, we can narrow the search to a subset of values by providing the block index of an encrypted value. Note that this index reveals this information to possible attackers too. Consequently, usage of the bucketization reduces privacy guarantees of an encrypted value to k -anonymity [Swe02] (cf. Section 3.1.4.3). Thus, the level of privacy protection in those groups depends on the choice of BucketOf_{DT} , which we will cover in Section 6.3.2. Consequently, bucketization should only be used with great care for *quasi-identifiers*.

Although the current definition of our reverse lookup suits privacy analysis, we lack a method to index specific records. Moreover, we have to manage data groups efficiently to keep track of the specific values that fall into a bucket. Otherwise, we may associate a record with every bucket of any value of that group. However, if we keep track of specific group values that map to a bucket, we can further reduce the number of values to a subset of all values in that group. Noteworthy is that this optimization would not be possible using a disjunctive PEKS scheme because it is impossible to match different group values independently, requiring the encrypted index to be associated with every bucket for every value it includes. That is why we stated earlier that such a scheme downgrades the efficiency of bucketization.

$$\text{ColumnsOf}_{DG}(\text{BI}, i)$$

$$\mapsto \{\text{IndexOf}(DF_j) : DF_j \in DG \wedge \text{BucketOf}_{\text{TypeOf}(DF_j)}(\text{val}_{i,j}) = \text{BI}\}$$

$$\text{RecordsOf}_{DG}(\text{BI})$$

$$\mapsto \{(i, \text{ColumnsOf}_{DG}(\text{BI}, i)) : i \in \bar{n} \wedge \text{ColumnsOf}_{DG}(\text{BI}, i) \neq \emptyset\}$$

Here, RecordsOf_{DG} is a reverse index mapping to records and contains a set of group indices that match this bucket. The positions correspond to the field DF index in

the group DG . To collect those positions, we use a helper function $\text{ColumnsOf}_{\text{DG}}$ collecting all list positions of values which map to the same bucket, and for specific records, we refer to $\text{ColumnsOf}_{\text{DG}}(i)$ as $\text{Cols}_{\text{DG},i}$.

Takeaway. We formally introduced bucketization and presented an efficient reverse index for our data structure of EHRs. The reverse index $\text{RecordsOf}_{\text{DG}}$ maps a bucket index to a set of possible records that contains any value in this bucket. By adopting the group structure of EHRs, we provided an efficient way to adopt the group structure such that the reverse index also reduces the number of fields to match from a data group. Furthermore, we highlighted the privacy issue using bucketization, which requires further elaboration in our implementation and evaluation.

5.2.3 A suitable PEKS Scheme

We mentioned necessary requirements of the PEKS scheme throughout this chapter. However, we still need a formal description, notably, a PEKS that fulfills this. In the following, we address this issue by writing formal requirements to the PEKS and choosing a suitable PEKS scheme.

5.2.3.1 Requirements and Assessments

We start by formalizing the requirements extracted from previous sections (cf. Section 5.1) to prepare for choosing an appropriate scheme.

- We require the capability to split the index into the index fragment and the consent witness. For the latter, the data owner (DO) must be able to derive consent bound to a specific query.
- Trapdoor construction must be feasible using a cumulative key managed by the collective authority (CA), so new trapdoors for a specific query are unforgeable in the any-trust model.
- The consents and trapdoors are published onto a public bulletin board and, therefore, should fulfill the requirements of SCF-PEKS (cf. Section 3.2.2.1).

Next, we prioritize common used performance evaluation parameters [YM16, MMW⁺17, HKA19, LLZ19] to assess different PEKS schemes. Here we focus on two characteristics, namely computation time and memory size.

Index build time. The index build-time measures the time required to encrypt a single value using the index operation, thus mainly impacting the time of import that has to be repeated for all n records $\Theta(n)$. Note, that this also impacts the time of shuffling data, as this requires a re-import, as we will see later in protocols. Furthermore, a part of the index function must be reused to bind the consent of a data field to a specific query. Thus, the time includes the time of the cryptographic operations for giving consent.

Trapdoor build time. Trapdoors are built by the trapdoor operation for every value queried, directly impacting the query construction time. In addition, part of the

routine must be computed under a cumulative key and repeated for every CA member. Thus, we conclude that this operation scales $\Theta(ps)$ in the number of trapdoors (p) and CA members (s).

Test time. The TS performs the test operation to match queries to records. Because trapdoors query a specific field, the number of matches is limited by the number of values for that field, which is constant. Thus, we can state the worst case of operations performed depending on the number of trapdoors (p) and records (n) by $\mathcal{O}(p \cdot n)$.

Size of index. The index size impacts two parts. First, the database size of TSs as they are required to partially store the index for every value of every record $\Theta(kn) = \Theta(n)$. Second is the network bandwidth because part of the index will be the consent sent to the bulletin board for every query field $\Theta(n \cdot p)$.

Size of trapdoor. The trapdoor size correlates with the query size $\Theta(p)$. Thus, the bandwidth requirements between the data user (DU) and CA members and the published payload on the bulletin board and forwarding queries will be increased.

As highlighted by our distributed health data network (DHDN) use case, a multi-million number of records is a reasonable scenario that should be supported, and the number of queried values in reviewed literature [CRMC15, GTA⁺14, HTHB14] is less than a hundred. Further, the number of CA members can also be considered very low. Concluding, the number of test operations stands out with quadratic complexity in the number of queried fields and data records. Inherently this also holds for the consent part of the index computation and memory footprint, as every test requires individual consent. Thus, the test time, index build time, and index size are the most significant characteristics.

5.2.3.2 Comparison of published Schemes

Based on the elaborated characteristics, we now compare suitable PEKS schemes for this thesis. Therefore, we must find a decent compromise of the orthogonal features security, query expressiveness, and efficiency [BHJP14].

To find a suitable scheme, we first investigated multiple surveys of PEKS schemes and carved out all publications that may fit. From that, we continued with papers referenced by those, especially papers of the past that introduced principles of modern schemes. Last we searched for papers that built upon the publications of fundamental principles (cf. Section 3.2). From a selection of 24 publications, we implemented the eight most relevant due to the application of research and our set requirements. For the implementation, we targeted 128-bit security in conformance with other DHDNs [YMM⁺17, FES⁺17]. On top of that, we built a benchmark suite measuring the most significant computation time of the index, trapdoor, and test operations. Figure 5.3 shows the result of our measurements. In addition, Table 5.2 shows the memory footprint of the schemes for the particular chosen implementation.

From our assessment, the schemes from Rhee et al. [RPSL10] and Lu et al. [LLZ19] stand out and are most applicable to our requirements. However, the latter cannot be adapted to our requirements as the trapdoor is bound to a specific owner, and the index binds a specific user behind a hash function. For a summary of the scheme

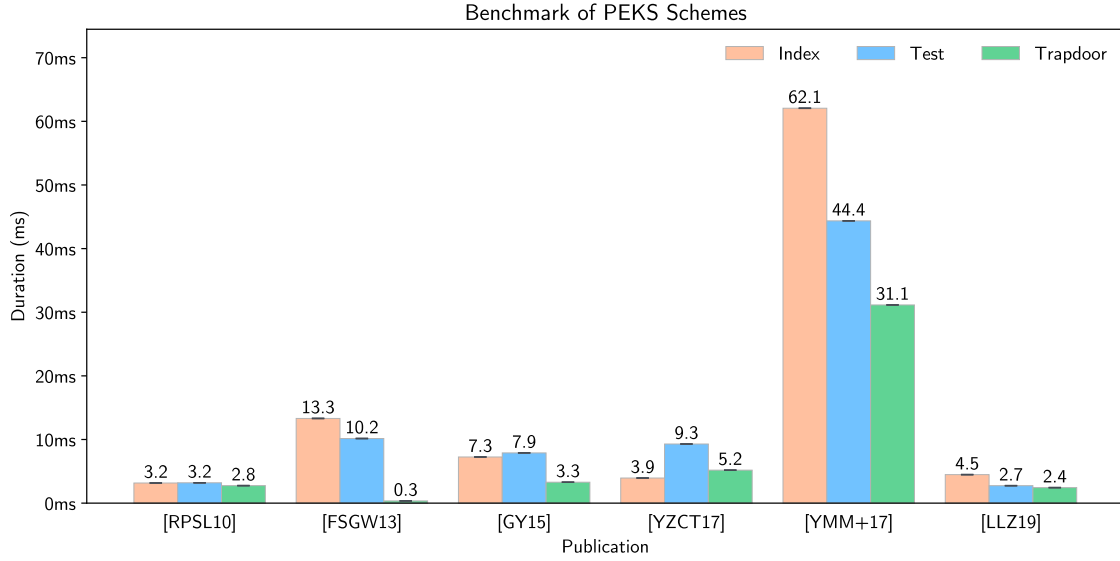


Figure 5.3 Comparison of PEKS schemes based on the computation time with a 99% confidence interval. Benchmarks were performed on the same machine serving evaluation, see Section 7.1.1. Of particular importance is that we implemented all schemes over the same curve, except for [YMM⁺17], which requires a composite order group. All scheme implementations target 128-bit security.

Scheme	Index	$\simeq$	Trapdoor	$\simeq$
[RPSL10]	$ \mathbb{G}_1 + \mathbb{H}_\kappa $	80	$ \mathbb{G}_1 + \mathbb{G}_2 $	144
[FSGW13]	$ \mathbb{Z} + \mathbb{G}_1 + 2 \cdot \mathbb{G}_2 + 2 \cdot \mathbb{G}_t $	1424	$ \mathbb{Z} + \mathbb{G}_1 $	80
[GY15]	$ \mathbb{Z} + \mathbb{G}_1 + \mathbb{G}_2 + \mathbb{H}_\kappa $	208	$ \mathbb{G}_1 + \mathbb{G}_2 $	144
[YZCT17]	$3 \cdot \mathbb{G}_1 + \mathbb{G}_t $	720	$ \mathbb{Z} + 3 \cdot \mathbb{G}_2 $	320
[YMM ⁺ 17]	$3 \cdot \mathbb{G}_{pq} $	54	$3 \cdot \mathbb{G}_{pq} $	54
[LLZ19]	$ \mathbb{G}_1 + \mathbb{H}_\kappa $	80	$ \mathbb{G}_2 $	96

Table 5.2 Memory sizes of the index and trapdoor structure for different PEKS schemes. Note that this only holds for the particular chosen curve BLS12_381. Other curves exist that have similar sizes but balance differently between $\mathbb{G}_1$ and $\mathbb{G}_2$.

showing the non-trivial problems of adoption is listed in Figure A.2. As a result, we decided to adopt the former one because extracting a consent witness and computing trapdoors under a cumulative key is straightforward. Furthermore, the picked scheme allows re-encryption of queries, such that data providers (DPs) can issue test server (TS) specific queries.

5.2.3.3 Adaptation

In the previous section, we compared several PEKS schemes and chose one of them. In this section, we will formally state the algorithms of our adaptation and point out some important characteristics. Those are the relation of the **Index** and **Consent** operation and a deterministically encrypted part of the **Trapdoor** operation. Algorithm 5.1 formally states our adaptation.

Index ($\text{PK}_{\text{TS}}, \text{val}$)	Trapdoor ($\text{sk}_{\text{Q}}, \text{PK}_{\text{RN}}, \text{val}$)
$\text{cw} \leftarrow \mathbb{Z}_p^*$	$r' \leftarrow \mathbb{Z}_p^*$
$\text{DG} = \text{GroupOf}(\text{FieldOf}(\text{val}))$	$\text{DG} = \text{GroupOf}(\text{FieldOf}(\text{val}))$
$\text{IF} = \text{H}_2(e(\text{PK}_{\text{TS}}, \text{cw}.\text{H}_1(\text{DG}, \text{val})))$	$T = \text{sk}_{\text{Q}}^{-1}.\text{H}_1(\text{DG}, \text{val})$
return cw, IF	$T_1 = r'.g$
	$T_2 = T + \text{H}(r'.\text{PK}_{\text{RN}})$
	return (T_1, T_2)
Consent($\text{PK}_{\text{Q}}, \text{cw}$)	Test ($\text{sk}_{\text{TS}}, \text{sk}_{\text{RN}}, \mathcal{I}, \mathcal{T}$)
$\text{IC} = \text{cw}.\text{PK}_{\text{Q}}$	$\mathcal{I} = (\text{IC}, \text{IF}), \mathcal{T} = (T_1, T_2)$
return IC	$T = T_2 - \text{H}(\text{sk}_{\text{RN}}.T_1)$
	return $\text{IF} \stackrel{?}{=} \text{H}_2(e(\text{IC}, \text{sk}_{\text{TS}}.T))$

Algorithm 5.1: A formal definition of our adaptation of the SCF-PEKS scheme from Rhee et al. [RPSL10]. The original formulation (Algorithm A.3) is in the appendix section of this thesis (Appendix A.3). Here H , H_1 and H_2 are domain separated hash functions that map to $\mathbb{G}_2$, $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively.

Index ($\text{PK}_{\text{TS}}, \text{val}$): The **Index** algorithm inputs a TS public key PK_{TS} and a value val as input. Note that binding the fragment IF to a specific server prevents any other server without access to sk_{TS} from matching queries to that fragment. Furthermore, the algorithm returns the consent witness cw . The original version called the following algorithm **Consent** internally to bind the index operation to a specific receiver directly.

Consent ($\text{PK}_{\text{Q}}, \text{cw}$): Here the consent witness cw is used to consent to a query constructed under the public key PK_{Q} . Importantly cw is specific for a single **Index** operation. As a result, the returned consent IC is unique for a specific index fragment IF and query key PK_{Q} , such that consent of the DO is always required.

One more thing to point out is that under the assumption that solving the Elliptic Curve Discrete Logarithm Problem (ECDLP) is hard, consent is computationally indistinguishable from others where the witness is chosen randomly, which we call pseudo-consent. A pseudo-consent can be issued by the DO to hide the information they are unwilling to consent to a particular value.

Trapdoor ($\text{sk}_{\text{Q}}, \text{PK}_{\text{RN}}, \text{val}$): This algorithm creates a secure channel free (SCF) trapdoor for a value val bounded to DPs of a research network with public key PK_{RN} . This variation is an important modification of the original scheme as the following equation shows for two trapdoors $(T_1^{(1)}, T_2^{(1)})$ and $(T_1^{(2)}, T_2^{(2)})$.

$$T^{(1)} = T_2^{(1)} - \text{H}(\text{sk}_{\text{TS}}.T_1^{(1)}) = \text{sk}_{\text{Q}}^{-1}.\text{H}_1(\text{val}) = T_2^{(2)} - \text{H}(\text{sk}_{\text{TS}}.T_1^{(2)}) = T^{(2)}$$

Because of the missing randomization of T , the TS gains access to a deterministically encrypted part of the trapdoor. As a result, two trapdoors of different fields would leak the information that the queried values are equal, which makes statistical analysis more practical, e.g. if age and gender are equal, the records' age is either zero or one because those are the only two values for the gender field in the NIS dataset. Note that even with the field-specific hash, the fact still holds if the user queries the same

group for the same value twice. This recognition would be an issue if multiple queries use the same key sk_Q , but as previously described, this key is randomly generated and for one-time use only.

Other important properties to point out are in the construction of T . Because in order to compute T under a cumulative key it is sufficient to shuffle $H_1(\text{DG}, \text{val})$ multiple times such that $T = \prod_t \text{sk}_{Q,t}^{-1} \cdot H_1(\text{DG}, \text{val})$. Later in Section 5.3.3.1, we will see how to use this property to state a secure query construction protocol. In addition, T is not bound to a specific research network (RN) key. Thus, trapdoors can be partially reused to issue queries to multiple RNs.

Test ($\text{sk}_{\text{TS}}, \text{sk}_{\text{RN}}, \mathcal{I}, T$): This final operation contains two steps by the DP and TS to match fields with trapdoors. First, the DP restores T and forwards the unblinded trapdoors to the TS. After that, the TS loads the fragment **IF** from a local database and pairs it with the DO consent **Consent** to retrieve a valid index $\text{Index} = (\text{IF}, \text{Consent})$.

Notably, everyone with access to an encrypted trapdoor value T and the public key PK_Q can start a *keyword guessing attack* on the query. Therefore, the attacker generates their keypair and repeatedly calls **Index** and **Test** for all possible values. On the other hand, everyone with access to sk_Q and a valid $\mathcal{I}$ for that key can issue any number of trapdoors and test them against the index to perform a *keyword guessing attack* on the DOs' data. For that reason, the encrypted trapdoor values will be encrypted under the ephemeral collective key, such that forgery of new trapdoors is impossible under the any-trust thread model.

Takeaway. Concluding our PEKS scheme adaptation, we have seen four important functions that are capable of meeting our functional requirements. The **Index** operation allows us to extract a consent witness suitable to implement dynamic consent. The trapdoor generation fits the necessity of a collaborative protocol with the CA securing against new trapdoor constructions using the same receiver key. Finally, the test operation aligns with our view of a DP as a gateway between the DHDN and the organizational infrastructure.

5.2.4 Query Data Structure

In the last sections, we have seen the health data structure, bucketization usage, and cryptographic primitives to match data against a search term. In the following, we combine these data structures and algorithms to state the structure and evaluation of a query. In addition, we show how our chosen scheme benefits the expressibility of our query and efficiency in determining the results.

A query consists of a list of trapdoor information and a query evaluation tree, which essentially states a boolean combination of matched trapdoors. Notably, this extra structure of indirection adds complexity to the query. However, it aligns better with our perception of deterministically encrypted values which only require to be encrypted once for every group-specific value. Moreover, the tree structure allows more efficient utilization of bucketization and categorization of results, which we will see after formally stating the structure of trapdoor information and the query evaluation tree.

The extra trapdoor information lists the queried data groups, an optional bucket index, and the cryptographic trapdoor required for the PEKS scheme.

$$(\text{DG}_l, \text{BI}_l, \mathcal{T}_l) : l \in \bar{p}$$

These list elements contain all information to match single values of a record but without any relation. Therefore, a query evaluation tree extends this structure to match arbitrary expressions.

The query evaluation tree (Ω) is a connected directed acyclic graph (DAG) with separate tags for sources, sinks, and inner vertices. Further, we will refer to sources as roots, sinks as leaves, and inner vertices as boolean combinators.

$$\begin{aligned} \Omega &:= (V, E, \text{tag}) : E \subseteq V \times V \\ R &:= (v \in V : \nexists u ((u, v) \in E)) \\ L &:= (u \in V : \nexists v ((u, v) \in E)) \\ B &:= V \setminus (R \cup L) \end{aligned}$$

To interpret this DAG, we tag every node such that induced trees from a root node yield a boolean expression. Therefore, we tag leaves with an index l of provided trapdoor information. Thus, the boolean evaluation of a leaf corresponds to match this trapdoor with a record's data. We use tags of boolean operations for inner vertices, which combine the boolean evaluation of vertices from outgoing edges. Finally, the roots specify how to receive a result from matched records.

$$\begin{array}{ll} \text{tag} : V \longrightarrow \text{TAG with} & \\ \text{tag}(v) \in \{\text{'COUNT(*)'}\} & \text{for all } v \in R \\ \text{tag}(v) \in \{\wedge, \vee\} & \text{for all } v \in B \\ \text{tag}(v) \in \{1, \dots, p\} & \text{for all } v \in L \end{array}$$

The idea of having multiple roots is to evaluate different groups in a query. For example, consider a query counting patients with a specific kind of diagnosis. If the researcher wants to further investigate the distribution of different procedures, they would have to issue multiple queries. However, this would unnecessarily negatively impact the network because the shared diagnosis evaluation must be evaluated multiple times. Hence, we allow sharing common evaluation of subgroups as common node-induced subtrees of Ω . An example of a specific query evaluation tree is shown in Figure 5.4.

The trapdoor information may optionally dismiss either the bucket index or the cryptographic trapdoor. We write a $\perp$ symbol for that missing value in that case. While leaving both out is considered ill-formed, specifying only one can have significant advantages. As previously stated, bucket indices should be optional and not be supported for all data groups, especially in the case of *quasi-identifiers*. Leaving this field out will enhance privacy, making *keyword guessing attacks* more computationally intense and other attacking vectors [MKG07] less likely. Though its absence will result in slower computation as more candidates must be considered. On the other hand, one can leave out a trapdoor to match all entries in a bucket as an efficient way to group results into different categories without the relatively costly evaluation of the **Test** function.

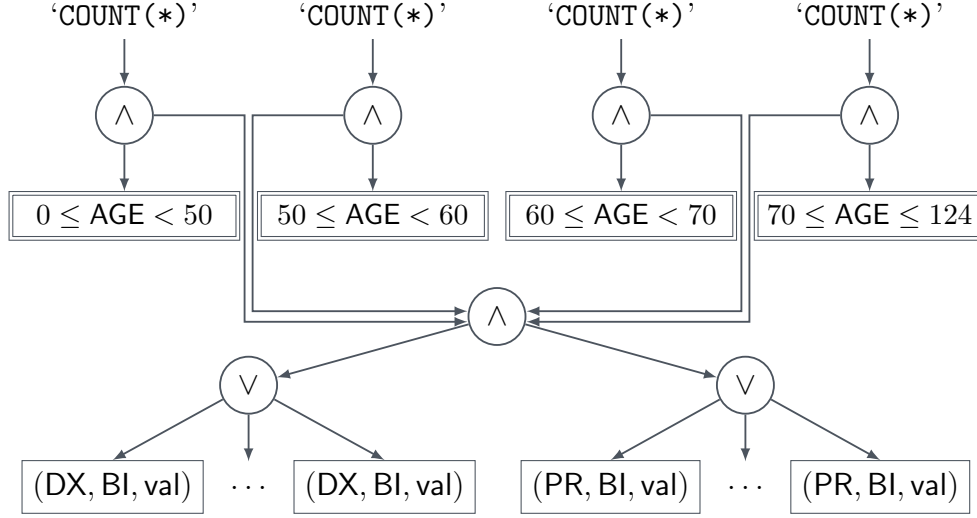


Figure 5.4 An example visualization of a query evaluation tree. Root nodes have no border where circles indicate boolean combinators, and leaves have rectangles. Here the query shares a common evaluation of candidates having one of many diagnoses as well as one of many procedures. These candidates are further categorized into groups based on ranges of their age. Therefore, the double edged placeholders $\text{age}_{lb} \leq \text{age} < \text{age}_{ub}$ represent multiple nodes $(\text{AGE}, \text{BucketOf}_{\text{AGE}}(\text{age}), \perp)$ with $\text{age} \in \{\text{age}_{lb}, \dots, \text{age}_{ub} - 1\}$. Please observe that we are taking advantage of the bucketization of age to categorize results without evaluating the PEKS scheme, leaving the trapdoor out.

Another benefit is the possibility of reducing evaluations of the **Test** function by taking advantage of the tree structure and our data group indices. The idea is to reinterpret leaves as a set of record indices called candidates. Then we can combine those with set operators where all outgoing vertices are operands.

$$\begin{aligned}
 \text{CandidatesOf} : B \cup L &\rightarrow \mathcal{P}(\bar{n}) \\
 (\text{DG}, \perp, \mathcal{T}) &\mapsto \bar{n} \\
 (\text{DG}, \text{BI}, \mathcal{T}) &\mapsto \text{CandidatesOf}((\text{DG}, \text{BI}, \perp)) \\
 (\text{DG}, \text{BI}, \perp) &\mapsto \{i : (i, \text{Cols}_{\text{DG}, i}) \in \text{RecordsOf}_{\text{DG}}(\text{BI})\} \\
 \bigwedge_{v_1, \dots, v_{\delta^+}} &\mapsto \bigcap_{u=1}^{\delta^+} \text{CandidatesOf}(v_u) \\
 \bigvee_{v_1, \dots, v_{\delta^+}} &\mapsto \bigcup_{u=1}^{\delta^+} \text{CandidatesOf}(v_u)
 \end{aligned}$$

Takeaway. We have seen that choosing a DAG for our query structure benefits query expressiveness and efficiency in multiple ways. Results can be categorized efficiently based on bucketization or sharing part of the evaluation for multiple roots. Furthermore, the structure enables an efficient bucketization evaluation, allowing us to reduce possible candidates further.

5.2.5 Efficient Linkability of Anonymous Messages

In Section 5.1.4, we introduced the network architecture, including an anonymous broadcast channel for messages from DOs to TSs. However, this anonymity induces

problems for the TS who has to filter published consent for data owned and link it with the identity of a record. To solve this issue, we introduce a ticket system that utilizes stealth addresses and deterministic encryption to filter and link broadcasted messages efficiently.

The idea is to find a tag for messages that stay unlinkable for any observer but the TS and are deterministically computable by both entities. Then we can use these tags for a hash lookup table that ensures high efficiency in filtering and linking tags to individual records. Therefore, the TS precomputes all tags for expected incoming messages and puts them into a hash table. Note that this requires the TS to know some identifier of incoming messages a priori. Fortunately, this is naturally given by the fact that consent is given as a response to a query announcement. Thus, the TS knows when messages are incoming and a unique associated query identifier id_Q .

One straightforward way to derive such a tag is utilizing the shared secret with a hash function $H(\text{SS}, \text{id}_Q)$. However, this approach has no attached identity. Thus, it needs to be clarified how propagating nodes in the network verify the message without access to SS . Therefore, we rather derive an identity than a hash value. To achieve this, we make use of stealth addresses. Recalling Section 2.1.3 stealth addresses enable us to derive new random public keys (PK_{SA}) for a given receiver (PK_{DO}), such that only the receiver can derive the corresponding secret key. The problem with stealth addresses is their internal randomization. As a result, they cannot be computed deterministically by both entities. We fix this issue by replacing the internal nonce with the previous hash tag definition.

$$\begin{aligned}\text{PK}_{\text{SA}} &= \text{dStealthAddress}_{\text{PK}}(\text{PK}_{\text{DO}}, \text{SS}, \text{id}_Q) = \text{PK}_{\text{DO}} + H(\text{SS}, \text{id}_Q) \cdot B \\ \text{sk}_{\text{SA}} &= \text{dStealthAddress}_{\text{sk}}(\text{sk}_{\text{DO}}, \text{SS}, \text{id}_Q) = H(\text{SS}, \text{id}_Q) + \text{sk}_{\text{DO}}\end{aligned}$$

As a result, we have a suitable tag definition that corresponds to a public key that is unlinkable to other tags and the DO without access to SS , PK_{DO} and id_Q in the random oracle model (ROM). Consequently, the DO signs the payload with sk_{SA} , and every node in the peer-to-peer (P2P) network can check its' integrity. One thing to point out is the usage of an improved stealth address version, where the sender cannot derive the secret key. However, in this particular case, this is not necessarily required as only the TS and DP have access to SS , and the TS has no benefit of masquerading the DO in the honest-but-curious threat model. Thus, one may deploy a simpler version [CM17] that requires fewer operations.

In summary, we have shown that the DO and TS can derive a deterministic tag for a query-specific message that is anonymous and unlinkable to any other observer of the bulletin board. Thus, the tag is suitable to precompute a hash table that maps message tags to candidates of the database, resulting in an efficient method filter and linking arbitrary messages in a shared bulletin board. In addition, we showed that choosing a pseudo-random public key as a tag facilitates network security, allowing every node to validate the message integrity.

5.3 Protocols

We conclude this chapter with a comprehensive view of the design of all protocols. Therefore, we start by stating the initial data preparation required for the data

owner (DO) to share secrets with the data provider (DP) and test server (TS). We briefly cover this, to formalize a foundation of available data that is essential for our protocols. In addition, we highlight some key decisions in managing data that have a major impact on the security aspects of the network. Continuing, we give full particulars on all workflows specifying their protocols and how to compute and validate exchanged messages. This specification includes a deep dive into the security properties of every message concerning the communication channel. Here, we focus mainly on privacy characteristics and give insights into trade-offs taken to accomplish our privacy and security goals.

5.3.1 Organization Data Exchange with the Data Owner

Before any data is available to import, DOs must contribute their data to the organization and legitimate them to manage their data in the distributed health data network (DHDN). We assume that the DO and organization initially set up some secure authenticated communication channel via an organization-hosted service. In this section, we will give a short example of how an organization can achieve such an initial setup and state some required information that must be exchanged during an initial setup.

For initial authentication, the organization can hand out a one time password (OTP) to the DO [LKL⁺10, EAK11, JL13]. For example, such exchange can be achieved by handing out a quick response code (QR code) on registration or sending letters to the DO address. The DO uses this initial communication channel to set up its identity. Therefore, the DO requires a suitable interface; here, we suggest a mobile app (cf. Section 8.1.1) as we already stated the requirement of an interactive device for the DO in Section 5.1.3.

After establishing a secure and authenticated connection, both participants must exchange some necessary information that future protocols will require. Therefore, the DO first generates an initial unique unlinkable identity PK_{DO} as covered in Section 5.1.3 and exchanges it with the organization, which will, in return, send PK_{Org} . Then, both entities exchange a shared secret using the Elliptic Curve Diffie–Hellman (ECDH) protocol.

After that, the organization has access to an unlinkable identity for the data record owner, which they can forward to other entities in the network. Furthermore, the DO and the organization share a secret suitable to share data, e.g. by using AES with Galois/Counter Mode (AES-GCM) [SMC08] or ChaCha20-Poly1305 [NL18]. Finally, we expect the DO to poll for available data periodically.

5.3.2 Data Import

Once DOs contributed their data and set up an identity, the organization can import that data into the DHDN. This section specifies this import process, including the data selection, encryption, and distribution. Importantly, this import procedure sets the foundation for following protocols that ensure secure and partially unauthenticated communication based on the data imported in this protocol.

First, the data import should not start once new data is available. We suggest starting a batched import of multiple records in a certain time interval (e.g. once a day or week at nighttimes) and only if enough data records are available. This delay is because we need to prevent external observers from linking data to individuals by observing facilities [MKG07]. In addition, this makes it possible to schedule the data import with the data shuffling, inducing the re-import of data. Thus, further increasing the number of records to import in a batch, enhancing anonymity among DO in that group. Last but not least, a scheduled update of the records leads to foreseeable data updates for the DOs' device, reducing the number of polls to the organizations' services. Alternatively, we can artificially increase the number of records to import by inserting more artificial data as covered in Section 5.1.1. No matter the strategy, either by fusing the import and re-import or by increasing the number of artificial data, the organization must ensure k -anonymity in the import batch for all quasi-identifiers (cf. Section 3.1.4.3). For the extent of this thesis, we leave the non-trivial problem of finding a good batch open and dedicate a paragraph in our outlook in Section 8.1.2. However, it is worth noting that such batch must be for a single TS. Otherwise, it would undermine the k -anonymity property. In the following, we will assume that a batch of records was selected and is available in plaintext from the organization database, encoded according to our data notation (cf. Section 5.2.1). Moreover, the TS is fixed for that particular batch, leaving us with exactly three entities to import the batch, the DO represented by a web service, the DP, and TS.

5.3.2.1 Preparation of Plaintext Data

Before detailing the specific protocols for importing data into each entity, we begin by discussing the organizational preprocessing of the data to establish shared secrets and encrypt the plaintext of patient records. This preprocessing step must be performed on each record individually. As such, we illustrate the data processing for a single record as an example without always explicitly referring to the index i . Subsequently, we divide the preparation process into four distinct categories.

Shared Secrets. In this context, we establish shared secrets between the DO and the DP, as well as the TS. This variant of the ECDH protocol involves using an additional nonce to conceal the TS's identity from the DO and ensure the freshness of the derived secret. Please note the critical importance of freshness for data re-imports. Refreshing the secret with a random value safeguards the protocol against pre-computation and replay attacks.

$$\begin{array}{lll} SS_{DO,DP}^{DO} = r_2 \cdot PK_{TS} & SS_{DO,DP}^{DP} = r_2 \cdot PK_{DO} & \text{with } r_2 \leftarrow \$ \mathbb{Z}_p^* \\ SS_{DO,TS}^{DO} = r_1 \cdot PK_{TS} & SS_{DO,TS}^{TS} = r_1 \cdot PK_{DO} & \text{with } r_1 \leftarrow \$ \mathbb{Z}_p^* \end{array}$$

It is important to note that the organization cannot access the established shared secrets. Therefore, it cannot act on behalf of the data owner unless it owns sk_{DO} , which can be the case for artificial data.

Bucket Indices. In this scenario, the bucketization process involves precomputing bucket data (BD) on the plaintext and transmitting it alongside the encrypted values.

To collect the bucket data, we iterate through all the groups and gather the bucket indices corresponding to the values within each group.

$$\text{BD}_i := \{(\text{DG}, \{\text{BI}, \text{ColumnsOf}_{\text{DG}}(\text{BI}, i) : \text{BI} \in \mathbb{BI}_{\text{TypeOf}(\text{DG})}\}) : \text{DG} \in \mathbb{DG}\}$$

BD_i provides a comprehensive list of mappings of $\text{RecordsOf}_{\text{DG}}$, allowing the receiver to reconstruct $\text{RecordsOf}_{\text{DG}}$ for that specific record. By utilizing this mapping, the receiver can perform bucketization and efficiently filter this candidate, achieving sublinear efficiency.

Consent Witnesses and Data Fragments. In this step, we utilize our adapted public key encryption with keyword search (PEKS) scheme from Section 5.2.3.3. We encrypt values to fragments and extract the consent witness required to consent to a query.

$$\text{cw}_j, \text{IF}_j = \text{Index}(\text{PK}_{\text{TS}}, \text{val}_j) : j \in \overline{m}$$

Here the consent witness cw is not bound to a DO. Consequently, the organization can create consent for a query, but the organization cannot act on behalf of the DO because they have no access to the shared secrets or sk_{DO} . Hypothetically, the organization can insert a patient record with an owned identity to observe consent requests. However, the organization is fully trusted and not expected to act maliciously. Therefore, this is not an imminent problem. Nevertheless, there is room for improvement here, which we will reference in Section 8.1.3.

Stealth Bit. In this final step, we must state whether the record is real patient-owned or artificially generated. Therefore, we encode a stealth bit SB indicating a real record if set to one. Otherwise, it must be zero. This way, we overload its' purpose, and stealth bits can be summed to obtain the total count of real records.

$$\text{SB} = \text{ElGamal}_{\text{PK}_{\text{CA}}}(b) : b \in \{0, 1\}$$

We adapt our idea of aggregating the result under a cumulative collective authority (CA) key as described in Section 5.1.1. Therefore, we encrypt the stealth bit using an encryption scheme that supports homomorphic addition, e.g., ElGamal.

5.3.2.2 Data Distribution for Entities

In the previous section, we have seen all data prepared to distribute to the three entities involved. Following that, we outline which data is exported, where, and how.

Organization $\rightarrow$ Data owner. As previously noted, the organization cannot dial the DO. Rather a data update is exported to the organizations' web service for every DO in the batch. First is the shared secret exchange for the DP ($\text{SS}_{\text{DO}, \text{DP}}^{\text{DO}}$) and TS ($\text{SS}_{\text{DO}, \text{TS}}^{\text{DO}}$). Those will later be used to ensure authenticated and confidential data transfer on the bulletin board. Second, are the consent witnesses for each field of the DOs' record ($\text{cw}_1, \dots, \text{cw}_m$). Following our PEKS adoption, these witnesses will be used by the DO to consent queries. Additionally, we can facilitate consent decision-making by categorizing field values. Finally, all data will be encrypted using an authenticated symmetric encryption (e.g., ChaCha20-Poly1305 [NL18]) and sent to the web service. We can use the shared secret between the organization and DO for the symmetric encryption.

Organization $\rightarrow$ **DP**. Although the DP shall not process queries, it requires information about the data stored by the TSs. The reason for that is the DP determines DOs that are requested to participate in a query. As a result, the DP can utilize the bucketization to pre-filter records and only give participating DOs a consent request. Note that this also requires the DP to know the identity of a record and let DO authenticate themselves to obtain query information. We will explain more details on this in the corresponding protocols **PUBLISH QUERY** (Section 5.3.3.2) and **ACQUIRE QUERIES** (Section 5.3.4.3). Hence the organization has to send the bucket data (BD) and identity (PK_{DO}) to the DP. Therefore, the DP issues an authenticated call to the DP and transfers data using a request-response based protocol.

Organization $\rightarrow$ **TS**. The TS is the entity storing the encrypted index fragments, monitoring the network for consent, matching queries, and aggregating results. This list also summarizes all data the TS requires. In order to store encrypted index fragments, it requires the data in the first place ($IF_1, \dots, IF_m$). On top of that, the TS applies bucketization to reduce the number of records to match, thus demanding bucket data. Moreover, for result aggregation, the stealth bit is needed. Finally, the TS must be able to relate published consent to a specific record. Therefore, we give the TS access to the record identity and the shared secret exchange $SS_{DO,TS}^{TS}$.

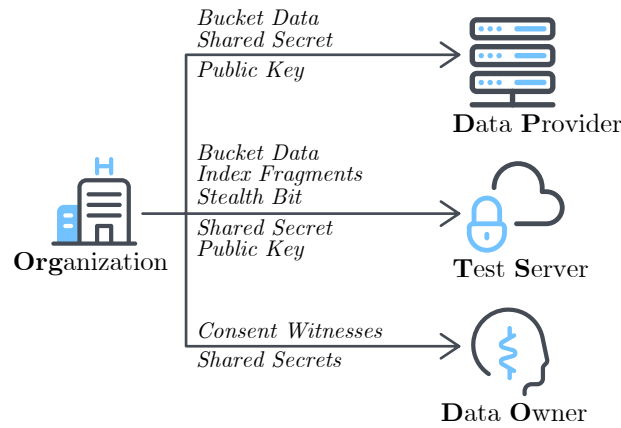


Figure 5.5 Overview of data exported to different entities. Note that the organization cannot push data directly to the DO but rather uses an intermediate web service polled by the DO.

This paragraph concludes the import stage. We introduced all required data for the following protocols and their distribution to the individual entities. Importantly we exchanged shared secrets for secure data transfer over insecure communication channels, shared unlinkable identities for data records, and transferred required data to give consent, match data, and aggregate results.

5.3.2.3 Shuffling of existing Data

We previously noted that shuffling is a substantial part of managing data and strengthening privacy guarantees (cf. Section 3.2.2.2). In this section, we scratch the surface of this topic by explaining what such shuffling implementation may look like.

In the most straightforward, shuffling deletes records of one TS and re-imports it to another. However, in our thread model, the TS is honest-but-curious; thus, it

could keep data and monitor the local bulletin board, which is shared with other TSs. Thus, a curious TS could continue matching new queries on deleted data to obtain more insights.

For this reason, the organization should generate a new anonymous identity for managing the record. Naturally, the organization should not be able to obtain the secret key of that new identity. Therefore, one can utilize stealth addresses (cf. Section 2.1.3). Here the organization generates a new random stealth address from the DOs' public key and re-imports the record using that identity. Note that these records are indistinguishable from their previous encryption due to the IND-CPA security of our PEKS scheme. Afterward, the organization only has to add the stealth address to the DO data deposited on the web service, such that the data user (DU) can derive the secrets of the new identity. Ergo, we can implement the shuffling of data on the same principles of the data import.

5.3.3 Query Deployment

In this first stage of a query's lifetime and the second stage of our schematic overview in Figure 5.1, we describe how to construct and publish a query. Therefore, we go into details of our trapdoor construction and show how it adapts it in a secure and consecutive protocol with all CA members. We also highlight key parts of the query structure that facilitate later protocols. Figure 5.6 shows a more detailed overview of the query construction phase from our schematic overview, containing the two protocols covered in this section.

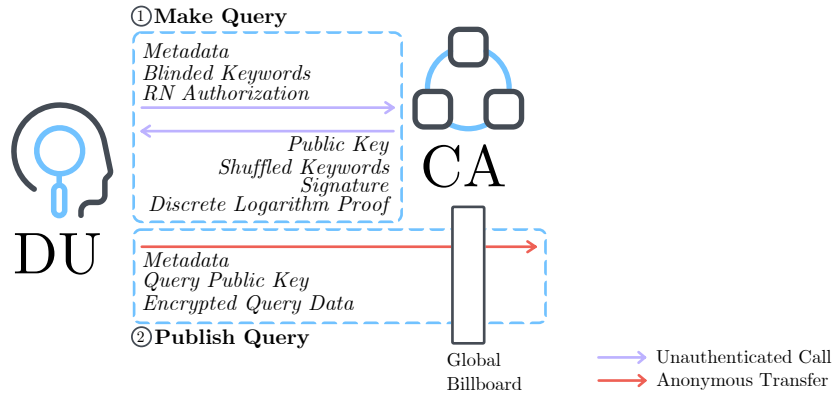


Figure 5.6 Overview of the query construction stage. 1) The DU sequentially authorizes as part of a research network (RN) to all CA members and receives authorized query data. 2) After the creation, the DU pushes the query anonymously under their ephemeral identity PK_{DU} to the global bulletin board.

5.3.3.1 Query Construction (Make Query)

One of the primary goals of the DU is to obtain deterministically encrypted values $val_k : k \in \bar{p}$ under an ephemeral collective key $sk_Q = \prod_{t=0}^s sk_{Q,t}$. Here, sk_Q is constructed by $s + 1$ random keys, where $sk_{Q,0}$ is chosen by the DU and all others by members of the CA. Please note that the specific index of a CA member is given by a linear order. For example, based on the x -coordinate of their public keys.

Collaborative Encryption of Values

First, the DU generates a random key pair $(\text{PK}_{\text{DU}}, \text{sk}_{\text{DU}})$ serving as an ephemeral key for the lifetime of the query. The public key will also be part of the query. This ephemeral key allows the DU to stay anonymous to the DHDN while CA members will authenticate the DU and guarantee entitlement of the anonymous identity. Additionally, query results will be forwarded to that particular key in the ‘Receive Results’ protocol.

To start the construction, we recall part of the **Trapdoor** definition (cf. Section 5.2.3.3).

$$\begin{aligned} T_{0,l} &= (b_l^{-1} \cdot \text{sk}_{\text{Q},0}^{-1}) \cdot \text{H}_1(\text{DG}, \text{val}_l) : l \in \bar{p} \\ T_{t,l} &= \text{sk}_{\text{Q},t}^{-1} \cdot T_{t-1,l} : l \in \bar{p}, t \in \bar{s} \end{aligned}$$

We slightly changed the definition and added a blinding $b_l \leftarrow \mathbb{Z}_p^* : \bar{p}$. If we would not blind the encrypted values CA members could exploit the deterministic encryption and run an *offline keyword-guessing attack* (cf. Section 2.2.1). The reason for that is the necessity of building PK_{Q} simultaneously, which enables the CA to build indices for the intermediate public key $\text{PK}_{\text{Q},t}$.

$$\begin{aligned} \text{PK}'_{\text{Q},0} &= \text{sk}_{\text{Q},0} \cdot B \\ \text{PK}'_{\text{Q},t} &= \text{sk}_{\text{Q},t} \cdot \text{PK}'_{\text{Q},t-1} : t \in \bar{s} \end{aligned}$$

Finally, the necessary query information can be derived by matching our PEKS definition for the ephemeral receiver.

$$\begin{aligned} T_l &= b_l \cdot T_{s,l} \\ &= b_l \cdot (\text{sk}_{\text{Q},s}^{-1} \cdot (\dots (b_l^{-1} \text{sk}_{\text{Q},0}^{-1} \cdot \text{H}_1(\text{DG}, \text{val}_l)))) \\ &= (b_l \cdot b_l^{-1} \cdot \text{sk}_{\text{Q},0}^{-1} \cdot \dots \cdot \text{sk}_{\text{Q},s}^{-1}) \cdot \text{H}_1(\text{DG}, \text{val}_l) \\ &= \text{sk}_{\text{Q}}^{-1} \cdot \text{H}_1(\text{DG}, \text{val}_l) \cdot \text{H}_1(\text{DG}, \text{val}_l) \\ \text{PK}_{\text{Q}} &= \text{PK}'_{\text{Q},s} = (\text{sk}_{\text{Q},0} \cdot \text{sk}_{\text{Q},s}) \cdot B \end{aligned}$$

We have seen a consecutive construction protocol for a valid query according to our PEKS adaption. Nonetheless, currently, the DU has no guarantee of CA members following the protocol, and neither has anyone guaranteed that the DU followed the protocol. Consequently, we will show how both entities can prove and verify correctness. Therefore, to proof correctness, we use proofs of general statements about discrete logarithms (cf. Section 2.1.4).

$$\text{ShuffleStmt} := C_t = \text{sk}_{\text{Q},t}^{-1} \cdot B_2 \wedge T_{t,1} = \text{sk}_{\text{Q},t}^{-1} \cdot T_{t-1} \wedge \dots \wedge T_{t,p} = \text{sk}_{\text{Q},t}^{-1} \cdot T_{t-1}$$

It should be noted that the statement is not bound to a particular CA member. However, we used the notation to align with our previous definitions, simplifying understanding and explicitly stating the variable assignment to construct or verify the proof. One additional binding, yet to be mentioned, is C_t . Highlighting the fact that the correctness of $\text{PK}'_{\text{Q},t}$ is yet to be proven, but trapdoors are elements of $\mathbb{G}_2$, and the usage of $\text{sk}_{\text{Q},t}$ must be consistent across both groups, which is not supported by our proof system. Therefore, we make use of the bilinear pairing, and the DU can check the consistency inverting the shuffle of the public key $e(\text{PK}'_{\text{Q},t}, C_t) \stackrel{?}{=} \text{PK}'_{\text{Q},t-1}$.

For the DU, they not only have to prove their computation's correctness but also provide some authenticated credentials. As seen in Section 5.1.3, the CA is responsible for attesting the eligibility of the credentials to not reveal the DU identity to the network. Still, without knowing the specifics of the credentials, we can specify how they are authenticated. Therefore, the DU packs credentials into the query's metadata (M). This metadata, which we will define later, is sent along to every CA member. Then every member signs their partial view of the query as part of a chain of signatures.

$$\begin{aligned} SC &= (SC_1, \dots, SC_s) \\ SC_t &= \text{EdDsaSign}_{sk_{CA,t}}(M \parallel PK'_{Q,t-1} \parallel PK'_{Q,t}) \end{aligned}$$

It is important to mention that the metadata is not modified after construction, thus validated from every CA member. This CA usage reflects our any-trust threat model, and other DHDN participants can trust the metadata as long as at least one member of the CA acts honestly. Another important fact is the continuation of the chain in the public key construction, which we will refer to as a public key chain. This way, the chain only remains valid if intermediate public keys are passed sequentially as the protocol demands. In order to validate the chain, the DU must send the signature and public key chain along with the query. Nevertheless, it is not required to send the discrete logarithm proofs which are required by the DU. As a result, the number of signatures and public keys only scales with the number of CA members but also legitimates the DU to exchange some trapdoors. This kind of *weak existential forgery* effectively qualifies the DU to drop parts of their query after construction. The reason is that the DU cannot forge new encrypted values under the query key PK_Q . Consequently, they can copy or replace another value with a random one. In the first case, there is no real benefit for the DU because a trapdoor is bound to the group it queries (cf. Section 5.2.3.3), thus copying it to query another group would invalidate the encrypted value which effectively results into a trapdoor that never matches. This invalid trapdoor derivation is also the case for the second variant. However, even if we signed the entire construction, we could not guarantee valid encrypted values because the DU could have used invalid ones from the beginning. In order to prevent such invalid construction, we would require a new kind of argument that convinces CA members about the validity of the evaluated hash function $H_1(DG, val_i)$. While there is progress in systems solving this issue in the field of (*zero-knowledge*) *arguments of knowledge* [GPR21, GKR⁺21], those systems are yet relatively slow and require a trusted setup. For that reason, we leave it up for future work to find a proving scheme that guarantees the validity of the trapdoors. Still allowing this kind of *weak existential forgery* enables three following problems.

First, the DU could issue multiple queries to spam the network or to query subsets of his initial formulation. Every network individual can spot this because PK_Q cannot be changed, and consequently, we track $id_Q = H(PK_Q)$ as a unique identifier. One drawback of this approach is the requirement to store all identifiers. To solve this, we include a timestamp in the signed metadata and define a time window to accept queries. Thus, entities only have to keep track of previous queries for the time window. Note, this also protects against *replay attacks*.

Second, the DU could issue multiple queries simultaneously to different nodes, such that the query pretends to query multiple DOs, but only one gets valid trapdoors.

Thus, any policy of DOs that enforces a certain threshold of contributors is undermined. This scenario is dangerous, especially with *background knowledge attacks*, e.g., if an attacker knows a particular person who visited that medical facility. An attack could abuse this information to evaluate queries on that particular individual and have increasingly strong evidence and success chances if they can reduce the queried records to that specific facility. To resolve this issue, all DOs must have a consistent view of queries published and only start processing those if all received the same query. Consequently, the DHDN requires a broadcast channel that fulfills this requirement, e.g., using a blockchain-based bulletin board [Wür18]. On the other hand, this behavior can be spotted in our chosen GossipSub protocol [VNM⁺20]. The reason for that is the existence of a trapdoor-independent unique identifier of the query, namely its' ephemeral key PK_Q . Ergo, nodes that receive multiple query variants can link them and report abuse. Depending on the method used by the DU to authenticate themselves to CA members, this misbehavior may be linked to an individual.

Third and last, the DU could pretend a different query structure than the evaluated one because chosen trapdoors never match. As a result, policies on the structure of a query can only partially be enforced. In other words, one may reject specific queries that are likely to spot individuals, but a malicious DU could hide that fact by using an or clause in which only one trapdoor is valid. For this third problem, we need a straightforward solution, and future work is required to prevent spotting individuals in the network.

Takeaway. We introduced a sequential protocol for query construction, with security and privacy guarantees in the any-trust thread model. This protocol enables the DU to create trapdoors of our PEKS scheme under an ephemeral key (PK_Q) while also guaranteeing the correctness of its computation. The CA authorizes an ephemeral identity PK_{DU} such that the DU can publish queries anonymously. In addition, the authenticated payload contains extra metadata to embed additional information, signed by the CA to strengthen the query security properties. Following, we utilize this metadata with further extensions required by subsequent protocols.

Additional Query Fields

Before we describe a query's publication, we have to introduce some additional fields. In the previous section, we already stated some essential parts, that are, the metadata (M), which includes the ephemeral public key PK_{DU} and a timestamp, as well as the ephemeral public key of the query PK_Q . However, we are still missing three parts, the authorized research network (RN) on which behalf the DU issues the query, the selection of queried DPs, as well as the query data.

RN Authenticity. Here the DU authorize themselves as part of RN to the CA members. The RN on which behalf the DU issues the query is part of the metadata. Therefore, the metadata includes a unique identifier of the RN.

DP Selection. As stated by our data discovery objective in Section 5.1.3 the DU should be able to query a selection of specific DPs. In our description of the DP (Section 5.1.3), we stated that DPs organize themselves in RNs and have access to a shared secret sk_{RN} . Following this idea, the DU chooses a data network to query and

selects a subset of DPs they chose to query. Therefore, they add the data research network identifier to the query metadata. In addition, we can further restrict the DPs by adding an explicit list of DPs' identifiers to the metadata. If this field is empty, we consider all DPs of the RN network.

Query Data. In Section 5.2.4, we have introduced the required query data. To construct these, the DU chooses a query evaluation tree and has to specify the trapdoor information. For the latter, the DU has to use the previously encrypted values T_l and has to set up trapdoors according to our PEKS definition (cf. Section 5.2.3.3).

$$\mathcal{T}_i = (r_l.B_1, T_l + H(r_l.PK_{RN})) : l \in \bar{p} \quad \text{with } r_l \leftarrow \mathbb{Z}_p^*$$

This trapdoor is from a secure channel free PEKSs (SCF-PEKSs) scheme and is indistinguishable against a chosen keyword attack (IND-CPA) under the assumption that Hash Diffie-Hellman (HDH) is intractable [RPSL10]. Here, the encrypted values T_l are blinded with the secret $r_l.PK_{RN}$ binding trapdoors to a research network. Ergo, everyone with access to sk_{RN} can recover encrypted values. When choosing a subset of DPs from a RN, this should be considered. We will later see that this is sufficient to guarantee that only selected DPs can submit a result, but this will not prevent them from recovering the encrypted values.

In addition, the DU can specify a bucket index for every eligible data group. Although the decision is optional, it is important to note that the bucket indices are not encrypted or blinded, which also holds for the query evaluation tree. The reasoning is to enable every node in the network to provide information for global query policies based on the number of candidates a query yields. Still, this feature is left for future work, and without the usage of this capability, it would also be a reasonable choice to encrypt more data of the query for a specific RN.

We conclude the construction of a query. We described an interactive and consecutive protocol to construct encrypted trapdoor values under a collective key prohibiting selective forgery and securing DOs' records against *keyword guessing attacks*. Following, we showed how the DU could obtain an anonymous but authenticated identity and issue queries on behalf of a RN for a specific set of DPs. With this, everything is prepared to publish the payload of a query announcement to the DHDN.

5.3.3.2 Query Transfer (Publish Query)

Previously, we showed how to choose and obtain all the necessary data to state a valid query. In this section, we publish this query to the DHDN and summarize the payload. Additionally, we shortly address a final signing step to guarantee the authenticity and integrity of the message.

For authenticity, the DU has access to the key pair (PK_{DU}, sk_{DU}) , which is ephemeral but legitimized by the CA signed metadata. Thus, signing the query publication message with sk_{DU} is sufficient. In other words the CA acts like a certificate authority and signs the temporary DU key. This signature is, in turn, used like a certificate chain packed into the query and finally signed by the authorized key. After that, the payload is pushed to the global bulletin board and ready for processing.

Field	Description
Ω	Query evaluation tree stating how to match data and compute results.
BD	Bucketization data to achieve sublinear efficiency.
$\mathcal{T}$	Blinded and encrypted values for the query.
PK_Q	The ephemeral collective public key data owners have to consent.
PK_{DU}	An ephemeral unlinkable key of the DU authorized by the CA.
DU RN	The authorized reasearch network of the DU.
DP RN	The shared research network of all DPs queried.
DP List	A subselection of the DPs to query.
Timestamp	A replay and forgery protection in a pre-configured time window.
Key Chain	Chain of shuffled public keys.
Sig. Chain	Chain of CA signatures for shuffle and metadata validation.
Signature	Outer signature from the authorized DU key PK_{DU} .

Table 5.3 Summarization of all fields published as part of a query publication.

5.3.4 Query Processing

In the previous section, we have seen the actual payload pushed to the global bulletin board. Now every DP has to forward the evaluation to the organization-specific entities in the network. This forwarding breaks down into three essential parts. Firstly, received, preprocessing, and forwarding of query data received to TSs. Second, the DOs observation of new queries and utilization of dynamic consent. Third and finally, the backpropagation from TSs aggregated results to the DP and its' accumulated publication to the DHDN.

5.3.4.1 DP Query Receival and Forwarding

All DPs regularly monitor the global bulletin board for new queries. Once a query is received that targets the DP, they start processing. We subdivide this processing into three steps, validation, preprocessing, and forwarding. While the former should be executed upfront, the latter two may run parallel.

Query Validation. In this pre-step, the DP validates all fields of the query according to their definitions (cf. Section 5.3.3.1). That is the outer signature of the payload signed by PK_{DU} , the chain of signatures signed by individual CA members. Further, the DP verifies the timestamp against the configured time window and checks that no other query under the same public key has recently been published.

Preprocessing of DO Data. As described in Section 5.3.2, the DP gets the bucket data of individual records. We use this data to filter all query candidates (cf. Section 5.2.4). This usage of bucketization enables us to filter out DOs before they give consent and reduces the number of requested fields to consent. Ergo, we can drastically shrink the number of DOs and consents, leading to less bandwidth and computational power usage of the network and DPs, respectively. After that, the DP derives a

pseudo-random message tag for every candidate and builds a lookup table of required fields for consent. Here we use the same method as the TS, which we covered in Section 5.2.5. In essence, the DP builds an efficient lookup table that maps deterministic tags to DO specific data fields that are required to match the query against the corresponding record.

Forwarding of Query Data (Forward Query). To process the query test servers requires the encrypted trapdoor values $T_l = T_l^2 - H(\text{sk}_{\text{RN}}.T_l^1)$, that must be forwarded by the DP. Here, because the DO can call every TS in an authentic and confidential, we do not require a secure channel free (SCF) payload anymore. Thus, the DP has to decrypt the trapdoor values within the query data and send them to the TSs. Please remember that the ephemeral query public key (PK_Q) is not part of the query data, ergo not forwarded to the TS. This condition is of particular importance. Otherwise, the TS can brute-force the trapdoors and gain partial knowledge on data stored (cf. Section 5.2.3.3). Therefore, we chose the query identifier as the hash of PK_Q . As a result, we can safely share this identifier with TSs, which is required to compute the linkable message tags (cf. Section 5.2.5).

In addition, the DP keeps track of all queries in progress using a list of query identifiers ordered by the time of reception. This list will later index new queries not yet observed by DO. However, it is important not to update the list until the TS finished processing the forwarded query. That is due to the necessity to precompute the message tags. If a DO consents before these are available, the TS cannot know whether the consent targets them.

5.3.4.2 TS Query Receival

This section covers the counterpart of the previous one. Here, the TS receives the query data and a id_Q of a newly issued query from the DP, as previously described. It has to precompute all message tags for expected consents (cf. Section 5.2.5) to filter and link consent messages to data records. Therefore, the TS utilizes the bucketization first to filter all candidates of the local database and derive all deterministic message tags $\text{PK}_{\text{SA}} = \text{dStealthAddress}_{\text{PK}}(\text{pk}_{\text{DO}}, \text{SS}, \text{id}_Q)$. Those tags will further be associated with the corresponding record index and required fields to match. In addition, the TS prepares an initial accumulator for result aggregation. That is a zero value encrypted under the collective CA key $\text{ElGamal}_{\text{PK}_{\text{CA}}}(0)$. It should be noted that this value cannot be cached or reused. Otherwise, aggregating the same records will always yield the same ciphertext. Ergo, the initial accumulator serves as an important randomization step.

Finally, the TS sets up a deadline for consent receipt, as it is not expected or required that all candidates will publish consent. Consequently, we need to finish the query at some point, even if not all DOs participated. We call this duration from the query received till the deadline the consent time window. Here, we chose a fixed time window for all queries, but putting an extra field into the query metadata is also reasonable. In that case, the DU can choose a trade-off between faster or more complete results.

5.3.4.3 DO Query Observation and Consent Publication

After all TSs received the forwarded query, the DP unveils the query identifier to all DOs. In order to observe those, DOs have to poll the DP actively.

Query Discovery (Discover Queries). In this protocol, DOs repeatedly issues an anonymous call to the DP to discover new active queries. The frequency of this call is freely choosable by the DO and we call it query poll interval. Choosing a long interval may result in missing short-living queries, while a short interval may result in a higher computational load and energy consumption.

For our system design, we do not consider the identifier of active queries as sensible data. Thus, everyone can query new identifiers without being authorized. Therefore, the DO sends the last observed identifier and receives all subsequent query identifiers back. If the DO has not observed any, they receive a list for all active queries. As a result, the DO can poll all active queries and minimize message size by only including new query identifiers. Highlighting the fact that this request is unauthorized, the DP cannot monitor the polling behavior of individual DOs. One downside of this approach is the impracticality that DOs cannot be linked to the query candidates, thus not being filtered yet, requiring each DO to issue a second follow-up request for each new query.

Acquisition of Consent Requests (Acquire Queries). Once the DO receives new query identifiers, they need to experience if their consent is required and, if so, which fields will be matched. Remember that each field has its consent witness cw_j and that required fields depend on the individual DO bucket data. Consequently, the next step of the DP is to request the required data for giving from the DP. Here we describe the protocol for one new query identifier for simplicity, although it should be batched to reduce the number of calls.

The DO starts by computing a query-specific stealth address of their identity $\text{sk}_{\text{SA}} = \text{dStealthAddress}_{\text{PK}}(\text{sk}_{\text{DO}}, \text{SS}, \text{id}_Q)$, $\text{PK}_{\text{SA}} = \text{sk}_{\text{SA}}.B$ and signing an empty message $\text{EdDsaSign}_{\text{sk}_{\text{SA}}}()$. Following, we send the public key PK_{SA} and the signature to the DP, which utilizes the deterministic message tag PK_{SA} to filter and link the DO to possible candidates. Due to the DP preprocessing step, this only requires a dictionary lookup. Thus, we can efficiently filter candidates and obtain required fields for that DO as a result of the bucketization evaluation. In response, the DP sends the public key to consent to (PK_Q), the set of data field indices that require consent ($\{\text{cr}_1, \dots, \text{cr}_r\} \subseteq \overline{m}$) and the DU authorized research network universally unique identifier (UUID).

Providing Consent (Publish Consent). At this stage of the query processing, the DOs can decide if they participate in the query by giving consent. Therefore, DOs decides for every field individually aligning with our concept of search control (cf. Section 5.1.1). In addition, the DO knows the RN used to issue the query. Therefore, they can use publicly available information for the RN (cf. Section 5.1.3) to restrict participation, e.g. non-profit organizations only. Condensing, the DO has three individual decisions for every consent request $\text{cr}_w : w \in \bar{r}$.

$$\text{IC}_{\text{cr}_w} = \perp$$

$$\text{IC}_{\text{cr}_w} = r.B \text{ with } r \leftarrow \$ \mathbb{Z}_p^*$$

$$\text{IC}_{\text{cr}_w} = \text{cw}_{\text{cr}_w}.\text{PK}_Q$$

Here, the first two equations correspond to no consent given, but the second one hides this fact using pseudo-consent (cf. Section 5.2.3.3). Note, we index by data field indices; this will simplify the matching for corresponding consent in upcoming equations. Although the pseudo-consent here prevents analysis of the DO preferences, it should be noted that this also prohibits relative statements to the number of participants because the TS cannot distinguish whether the DO is willing to participate or hides their denial. Note, that the DO can also ignore the query and give no consent information. In that case, their record is ignored after the consent time window passed.

Assuming the DO chose to participate, the next step is publishing the consent. Following our principle of efficient linkable anonymous messages, the DO computes the message tag $PK_{SA} = sk_{SA}.B$ with secret $sk_{SA} = dStealthAddress_{PK}(sk_{DO}, SS, id_Q)$. Remember that the message tag and consent are indistinguishable for outsiders, thus eligible for SCF communication. However, if we publish $(IC_{cr_1}, \dots, IC_{cr_r})$ unencrypted observers of the network can obtain insights on the number of required fields per DO, including denied consents encoded as $\perp$. For that reason we utilize the shared secret SS and encrypt $AES_{SS}(\{(cr_1, IC_{cr_1}), \dots, (cr_r, IC_{cr_r})\})$. It is generally optional to include the field indices, but rather a design decision to avoid problems in rolling database structure updates. Finally, the DO signs the encrypted consent lookup with sk_{SA} and combines the message tag, encrypted consent, and signature, pushing them to the local bulletin board.

5.3.4.4 Result Aggregation and Publication

While consent is published, the network starts processing data records. In the following, we show how the query is evaluated against those records and results are aggregated and published.

TS Result Aggregation (Forward Result). TSs actively monitor the local bulletin board for the publication of consent. If one explores a new consent that is part of their precomputed lookup table, they start processing the payload. First, they validate the integrity of encrypted consent, discarding the message on failure. On success, consent is decrypted, and provided data field indices are matched against the required ones locally precomputed based on the bucket data. After that, the TS loads corresponding data fragments $((IF_{cr_1}, \dots, IF_{cr_r}))$ and the stealth bit (SB) from the database to evaluate the query for this particular record. Therefore, we start by matching all query leafs (DG, BI, T) .

$$\bigvee_{j \in \text{Cols}_{DG}} IF_j \stackrel{?}{=} H_2(e(IC_j, sk_{TS}.T))$$

Here Cols_{DG} contains all relevant data group fields filtered by the bucketization, which we precomputed and associated to records earlier. At this stage, the outer conjunction ($\bigvee$) highlights the fact that our system evaluates a *has*-relation rather than a *is*-relation. We can match the record by evaluating the query evaluation tree (cf. Section 5.2.4). For a match, the query result accumulator gets incremented by SB . Once messages for all expected tags have been received or the deadline is reached, the TS closes consent submission and forwards the result to the DP using

an authenticated call (cf. Section 5.1.4). That is the query identifier (id_Q) and the aggregated sum of stealth bits (agg_{TS}) encrypted under the collective CA key.

DP Result Publication (Publish Result). Meanwhile, the DP waits for partial result submission of all assigned TSs and aggregates them. Afterward, the aggregated result must be submitted to the global bulletin board. This submission should be anonymous. Otherwise, monitoring the duration from query announcement until result submission may reveal sensitive insights of an individual DP. To achieve anonymity while also providing authenticity and integrity, we utilize the Back Linkable Spontaneous Anonymous Group (bLSAG) signature scheme (cf. Section 2.1.1.2). Here the rings' public keys are given by the DP selection that is part of the query metadata. In addition, we choose $\text{id}_Q = H(\text{PK}_Q)$ as the signature tag, such that key images are unique for every signer and query, which also protects against *replay attacks*. Finalizing, the DP packs the aggregated result and signature together with a query-specific identifier and pushes them anonymously to the global bulletin board.

This scheme secures the protocol beyond our threat model, and other weaker schemes may be computationally less intense and have a smaller memory footprint. In particular, a DP may not push multiple results in the honest but curious threat model. Ergo, there is no necessity to prevent this, and a classical ring signature would be sufficient. In addition, the bLSAG provides linkability across different groups, but the group is currently statically defined by the query metadata. The reasoning is that if the group gets too large, DPs could take a random subset of members for the ring.

With the results' publication, the query processing stage finishes. Due to the number of protocols introduced and their complexity Figure 5.7 summarizes all protocols and breaks their payload down to their most essential parts.

5.3.5 Result Reception

We have seen how a query is constructed and published in the previous sections. Continuing, we describe how the DU can obtain the final result for this final stage of a query's lifetime. The challenge here is the re-encryption of the result from the CA key (PK_{CA}) to the DU key (PK_{DU}) while preserving confidentiality and integrity. Therefore, we adopt the key switch protocol from Yuan et al. [YMM⁺17], which constitutes the main part of this section.

First, all members of the CA monitor the global bulletin board for result publications. Every time they observe one, they validate the signature and check whether the key image has yet to be observed. If this succeeds, the CA member updates an internal accumulator of all published query results and the set of observed key images. Note that here the accumulator cannot be randomized by an initially encrypted zero. The reason for that is the necessary consistency of the encrypted value across members in the CA, required by a later introduced key switch protocol.

Continuing, all CA members wait for query result publications until the size of the key image set equals the number of queried data providers. At this point, the aggregated value agg equals the ElGamal encryption of the query result res .

$$\text{agg} = \text{ElGamal}_{\text{PK}_{\text{CA}}}(\text{res}) = (C_1, C_2) = (r.B, r.\text{PK}_{\text{CA}} + \text{res}.B)$$

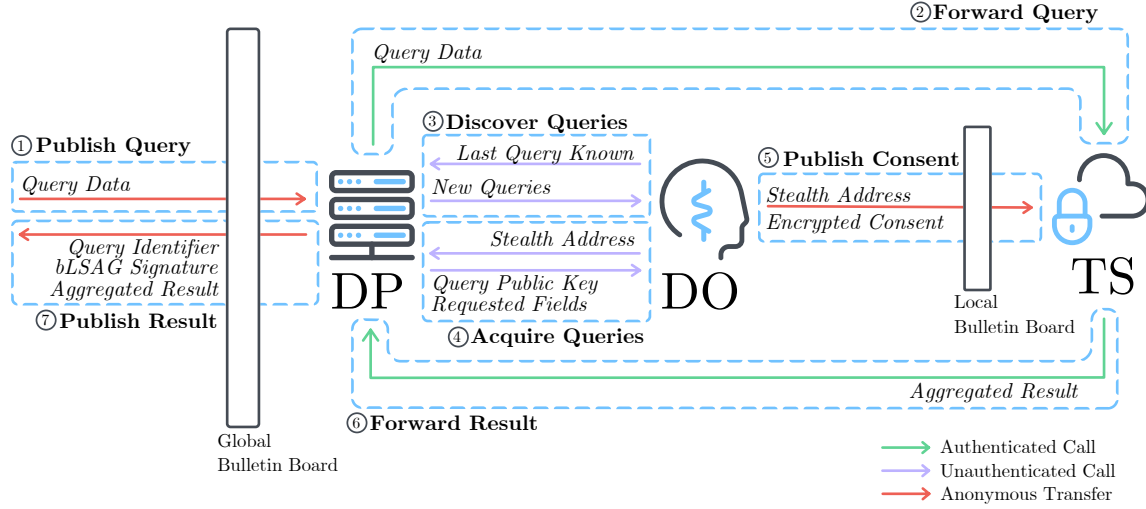


Figure 5.7 Overview of the query processing stage. 1) The DP receives the query and precomputes candidates with associated stealth address tags and fields that require consent. 2) Query data containing unblinded but encrypted trapdoor values are forwarded to all TSs. TSs precompute candidates with associated stealth addresses and monitor the local bulletin board for consent. 3) DOs poll DPs for new queries. 4) For new queries, DOs use a stealth address to ask the DP if they are part of the candidates and which fields require consent. 5) Consent is published to the local bulletin board and linked by a TS using the stealth address. TSs link local data fragments with given consent to match against the query, incrementing the stealth bit counter if they match. 6) After all consents are received or the deadline is reached, TSs forward the sum of stealth bits to the DP. 7) The DP collects the aggregated result of all TSs and publishes it to the global bulletin board.

Now this result is available for the DU to poll. Accordingly, the DU authenticates itself using sk_{DU} and requests the result from every CA member. In response, they will get the aggregated result agg , a key switch payload ks , and a proof of correctness for its computation. Here agg is independent of the particular CA member but checked for consistency across all members by the DU aligning with the any-trust model.

$$\text{ks}_t = (K_{t,1}, K_{t,2}) = (r'_t.B, r'_t.\text{PK}_{\text{DU}} - \text{sk}_{\text{CA},t}.C_1) \text{ with } r \leftarrow \mathbb{Z}_p^*$$

Here, ks_t is independent of any particular order of the CA and can be processed in parallel. Similar to the query construction (cf. Section 5.3.3.1), we can formalize a statement to prove the correctness of ks using a generalized zero-knowledge proof of discrete logarithms.

$$\text{KeySwitchStmt} := K_{t,1} = r'_t.B \wedge \text{PK}_{\text{CA},t} = \text{sk}_{\text{CA},t}.B \wedge K_{t,2} = r'_t.\text{PK}_{\text{DU}} - \text{sk}_{\text{CA},t}.C_1$$

Notably, the binding to $\text{sk}_{\text{CA},t}.B$ is given by the members' public key. Thus, upon receiving ks_t , the DU has all the required information to validate the discrete logarithm proof, ensuring the correctness of its construction. After the DU received and

validated all results, they can compute the encrypted result by evaluating the key switch. The following formula shows this evaluation and its' correctness.

$$\begin{aligned}
 & \left(\sum_t^s K_{t,1}, C_2 + \sum_t^s K_{t,2} \right) \\
 &= \left(\sum_t^s (r'_t.B), r.PK_{CA} + \text{res}.B + \sum_t^s (r'_t.PK_{DU} - \text{sk}_{CA,t}.C_1) \right) \\
 &= \left(r'.B, r.PK_{CA} + \text{res}.B + r'.PK_{DU} - \sum_t^s (\text{sk}_{CA,t}.B) \right) \text{ with } r' = \sum_t^s r'_t \\
 &= (r'.B, r.PK_{CA} + \text{res}.B + r'.PK_{DU} - r.PK_{CA}) \\
 &= (r'.B, r'.PK_{DU} + \text{res}.B) \\
 &= \text{ElGamal}_{PK_{DU}}(\text{res})
 \end{aligned}$$

Finally, the DU computes $\text{res}.B = \text{ElGamal}_{sk_{DU}}^{-1}(\text{ElGamal}_{PK_{DU}}(\text{res}))$ and looks up the result in a pre-computed hash table.

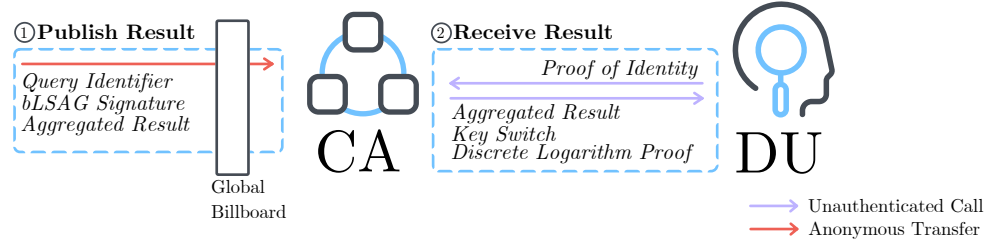


Figure 5.8 Overview of the query result reception stage. 1) CA members receive partial query results from DPs and aggregates them. The bLSAG signature is used to authenticate the DP as one of the queried DPs. 2) The DU polls all CA members for the result received and gets partial key switches used to map the encrypted aggregated result from the CA to the DU key.

6

Implementation

In the previous chapter, we introduced a system design of a distributed health data network (DHDN) that meets the requirements of our functional objectives (cf. Section 4.2.1). However, we must implement a prototype that features every part of our system to assess its conformance with our non-functional goals and test its practicality. Thus, this section introduces key aspects of our system implementation to assess its correctness and give insights into some design decisions.

Therefore, we first introduce the tools and technologies we used to implement our prototype, including adopted programming paradigms and libraries that facilitate the performance of our implementation. Following, we give an overview of our software components before going into detail on how we solved the most challenging problems. Finally, we give more insights into how we verified the performance and correctness of our application.

6.1 Tools and Technologies

Before going into details of our implementation, we introduce important design decisions in the choosing of the programming languages and adoptions of frameworks and libraries to facilitate our development process. In addition, we give insights into the rationale of these decisions, including considered alternatives.

6.1.1 Rust: Concurrency, Networking, and Cryptography

Rust [MK14] is a modern programming language that attracted more attention in recent years due to its unique balance of high-level language features, memory safety, and performance. This section states valuable features required for our implementation and argues how Rust fulfills these being a reasonable choice for our implementation.

Built-In Concurrency. Rust has a built-in concurrency model featuring powerful abstractions for concurrent code, including a dedicated syntax for asynchronous programming. The latter is of major importance for our thesis because we incorporate interactive network protocols, and by adopting an asynchronous runtime, we can utilize idle times of pending requests. In addition, Rust has an ownership and borrowing system that enforces memory safety in concurrent environments, eliminating the risk of race conditions. These features allow us to efficiently write concurrent code to exploit multi-core architectures, which is more reliable software for less time investment.

High-Level Network Stack. One of the core elements of our system design is the peer-to-peer (P2P) network. In Chapter 5, we introduced multiple high-level protocols that require secure Transport Layer Security (TLS) communication and the implementation of request-response and publish-subscribe messaging patterns. Thus, we require a moderate ecosystem for network applications to implement these protocols efficiently on top of pre-implemented software layers. Here Rust offers multiple frameworks for network programming, adopting the built-in concurrency for efficiently handling thousands of connections.

Low-Level Cryptography. In order to implement our public key encryption with keyword search (PEKS) adaptation, we require low-level access to cryptographic primitives. These should exploit hardware-specific instruction sets and optimized algorithms to implement those primitives in a performant manner. Within recent years, Rust became more and more attention in the crypto ecosystem due to its adoption in blockchain applications. In addition, Rust can interface existing C and C++ code allowing us to adopt existing low-level libraries that have established themselves for decades as well optimized and performant cryptographic libraries.

Notably, other programming languages have similar features. Here we highlight the programming language Go which also includes built-in concurrency and has a large ecosystem of performant and reliable network and cryptographic frameworks. However, we, the authors, have longtime experience with C and C++, thus ultimately deciding on Rust due to its similarity to C, easing the learning curve for a new programming language.

6.1.2 Frameworks

In the previous section, we introduced Rust as our choice of programming language. We also mentioned some valuable features and how the Rust ecosystem facilitates our implementation. This section will look more into this Rust ecosystem and present the fundamental frameworks we adopted to implement a prototype.

6.1.2.1 Tokio: An asynchronous Rust runtime

Tokio is a widely adopted asynchronous runtime in the Rust ecosystem, primarily designed for network applications. Its event driven-design facilitates concurrent and asynchronous programming at a high level. Therefore, Tokio incorporates low-level operating system-specific API calls to wrap I/O operations on sockets and the filesystem. Internally these are scheduled by a runtime with fine-granular

configuration options, including the capability to separate different execution runtimes from each other and assign limited resources. Utilizing these runtimes, we can strengthen the availability of our node by encapsulating the processing of network requests from the runtime of long-living computations.

6.1.2.2 libp2p: A modular network stack

One framework that integrates the Tokio runtime is libp2p, part of the IPFS [Ben14] project. libp2p is a modular software stack for P2P networks available in multiple programming languages, including Rust. In particular, libp2p has four modules of major importance for our implementation.

Encrypted Connections. To guarantee network connection confidentiality, integrity, and authenticity, libp2p implements multiple security protocols, including TLS. These protocols enable us to securely connect to hosts of known identity and engage in further protocols. In addition, libp2p integrates these identities into an auto-discovery module.

Peer Discovery and Routing. In a dynamically changing distributed network infrastructure, confining connection parameters and identities for all network hosts can be challenging. Therefore, libp2p incorporates mDNS [CK13] and the Kademlia protocol [MM02] for peer discovery and routing. Using these protocols, we can bootstrap our network and connect to each node by its peer identifier.

Protocol Multiplexing. In order to scale a P2P network while retaining connectivity and efficiency, it is important to process multiple protocols simultaneously. libp2p has built-in support of an additional muxing layer that enables simultaneous execution of multiple protocols without opening multiple connections. Especially in the case of simulating data owners (DOs), this is very helpful because we have to simulate thousands of entities, each engaging in their sequence of protocols. With the usage of protocol multiplexing, we can achieve this without the necessity of opening thousands of connections.

Request-Response and Publish-Subscribe. libp2p comes with two modules for message patterns which we require. These are an implementation of a request-response and the Gossipsub protocol [VNM⁺20]. Their implementation of these protocols also incorporates the previous modules for protocol multiplexing, peer discovery, and routing.

With all these modules enabled, we have a reliable, efficient, and scaling network solution we can adapt to focus on implementing our protocols on top of the request-response and publish-subscribe modules. In addition, we highlight that libp2p is transport agnostic; in the context of this thesis, all connections are established using TCP/IP, but we feel confident that with the choice of libp2p future work may test different transport layers without the necessity to change large portions of the codebase.

6.1.2.3 RocksDB: A persistent key-value store

For data management, we required a solution that is simple to deploy and scale and, at the same time, optimized for read operations. These are all criteria where key-

value storages outclass relational databases. Additionally, we have no requirement on relations of data records making key-value storages the perfect fit for our application. Here we decided on RocksDB, a fast key-value store developed by Facebook. Many alternatives exist, but these other implementations were more complex to set up, had no Rust bindings, or did not support one of the core features we require. These features are multi-threaded access, columns-based storage for different tables, and sorted iteration of keys stored. In Section 6.3.2, we will describe how we utilize these features to evaluate bucketization efficiently.

6.1.3 Cryptographic Primitives

We introduced various cryptographic protocols in Section 2.1 and throughout Chapter 5. However, to implement those, we require some cryptographic primitives; in particular, we need a concrete hash function and elliptic curve. In the following two sections, we will present arguments supporting our chosen instantiations and discuss their suitability regarding capabilities and security.

6.1.3.1 Hash Function

With the release of SHA-3 [Dwo15], we have the first hash functions with arbitrary output lengths in the SHA family. Shortly after, the committee released additional work standardizing SHA-3 derived functions [KjCP16] that use this new construction type to increase security. For our implementation, we use the methodology of these derived functions but use a different hash algorithm of the Keccak family. Here we decided to use Kangaroo Twelve [BDP⁺18] due to its better performance. In particular, we adopt customizable Secure Hash Algorithm **KECCAK** (cSHAKE) for customizable domain separation tags and TupleHash for unambiguously hashing tuples of messages. According to the targeted 128-bit security level, we choose κ for a hash output size of 256 bits [BDP⁺18].

6.1.3.2 Elliptic Curve

According to our PEKS definition (cf. Section 5.2.3.3), we require a pairing-friendly curve that targets 128-bit security and has efficient algorithms to hash to that curve. Therefore, we considered curves that have fast pairing and exponentiation and followed the evaluation of Khandaker et al. [KNG⁺17] choosing the faster and more studied Barreto-Lynn-Scott (BLS12) [BLS03] curve. In compliance with the 128-bit security goal, we took a relatively conservative approach of a 381-bit wide prime number [MSS17]. Another reason for this choice is the wide adoption of BLS12-381, which led to more research in optimizing its computation [Bow19, BC21] and high-quality software implementations. One of these works of particular interest is to hash to this curve in constant time [WB19].

One point of particular interest in Table 6.1 is the unbalance in the cost of exponentiation between $\mathbb{G}_1$ and $\mathbb{G}_2$. Recalling our PEKS definition, these are performed by our most called operations **Consent** and **Test**, respectively. Here we could swap both curve definitions to weight the computational load of DOs and test servers (TSs)

Operation	Group	Symbol	Measurement (ms)	Deviation (%)
Addition	$\mathbb{Z}_p^*$	τ_{zadd}	< 0.01	+0.04, -0.03
	$\mathbb{G}_1$	τ_{gadd_1}	< 0.01	+0.14, -0.12
	$\mathbb{G}_2$	τ_{gadd_2}	< 0.01	+0.10, -0.09
	$\mathbb{G}_t$	τ_{gadd_t}	< 0.01	+0.11, -0.08
Negation	$\mathbb{Z}_p^*$	τ_{zneg}	< 0.01	+0.14, -0.08
	$\mathbb{Z}_p^*$	τ_{zinv}	< 0.01	+0.08, -0.05
	$\mathbb{G}_1$	τ_{gneg_1}	< 0.01	+0.14, -0.13
	$\mathbb{G}_2$	τ_{gneg_2}	< 0.01	+0.13, -0.12
	$\mathbb{G}_t$	τ_{gneg_t}	< 0.01	+0.15, -0.13
Exponentiation	$\mathbb{G}_1$	τ_{gexp_1}	0.34	+0.10, -0.08
	$\mathbb{G}_2$	τ_{gexp_2}	1.03	+0.12, -0.11
	$\mathbb{G}_t$	τ_{gexp_t}	1.46	+0.09, -0.07
Hash to Group	$\mathbb{Z}_p^*$	τ_{hs}	< 0.01	+0.09, -0.09
	$\mathbb{G}_1$	τ_{hp_1}	0.10	+0.08, -0.08
	$\mathbb{G}_2$	τ_{hp_2}	0.50	+0.08, -0.08
Sample Scalar	$\mathbb{Z}_p^*$	τ_{sample}	< 0.01	+0.15, -0.15
Pairing	-	τ_{pair}	1.29	+0.03, -0.03

Table 6.1 Table of the computational costs of selected cryptographic primitives on the elliptic curve. Here τ_{zneg} is the additive inverse, while τ_{zinv} corresponds to the multiplicative inverse. Deviations of the results are given in a 99% confidence interval. Please note that the deviation is given in percent (0.1 corresponds to one thousandth).

differently. On the other hand, this also impacts the payload sizes for consents and trapdoors, as shown in Table 6.2. However, as seen in Section 5.2.3.1, the number of consents widely exceeds the number of trapdoors sent. For that reason, we chose to stick with this definition as it lowers the required bandwidth of the network and puts less computational load on the most powerless DO devices.

Element	Group	Symbol	Memory Size (bytes)
Hash	-	$ \mathbf{H}_\kappa $	32
Point	$\mathbb{G}_1$	$ \mathbb{G}_1 $	48
	$\mathbb{G}_2$	$ \mathbb{G}_2 $	96
Scalar	$\mathbb{Z}_p^*$	$ \mathbb{Z} $	32

Table 6.2 Memory sizes of cryptographic elements for our chosen elliptic curve.

6.2 Software Composition

In this section, we introduce the software developed as part of our prototype and how it facilitates the evaluation of our system. Therefore, we first introduce an overview

of all components developed and then follow up with a description of applications that utilize these components.

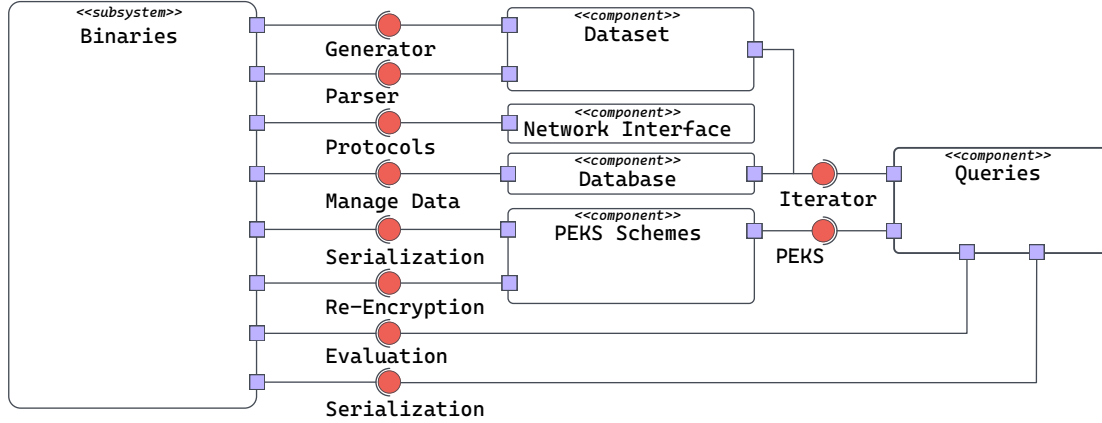


Figure 6.1 Overview of the software components developed for the prototype.

6.2.1 Components

To thoroughly test and benchmark all aspects of our prototype, we recognized the importance of code reusability and modularization. Therefore, we divided our codebase into multiple components, each serving a specific functionality that can be reused across different scenarios. This approach promotes code organization and maintainability and enables us to focus on individual components during testing and benchmarking, ensuring comprehensive evaluation. In the following, we describe each component in more detail.

6.2.1.1 Dataset

The primary task of our dataset component is the supply of data. Therefore, it supplies three different interfaces, two of which correspond to different data sources, and the third is a compatibility wrapper for query evaluation without a database setup.

Generator. The generator is a random data source to create artificial datasets with the same distribution as our real-world dataset (cf. Section 7.1.2) used for testing. Its purpose is to promote comparability and transparency of our results even if the dataset we used is unavailable. In addition, it is useful for testing because we can compile the generator into our program to generate test data on the fly without infringing on data protection rules.

Parser. To import datasets into our system, we use the second interface, which is the parser. The parser takes as input a data file and layout specification to yield all data records contained. Therefore, one can provide different layouts for different data sources as long as the columns match some predefined datatype supported by the parser.

Iterator. This final interface of the Dataset component is a compatibility wrapper for the database Iterator. Here we can configure the data source as an iterator for

data processing without setting up a database. Hence, we can test queries without needing to import data first, especially in the absence of any database optimization, which may include bugs.

6.2.1.2 Database

The database is a simple key-value storage to manage data records and some additional associated information required for our protocols, namely the record's public key and shared secret. In addition, it offers an optimized Iterator interface to loop data records efficiently utilizing bucketization, which is covered in Section 6.3.2. Notably, all data stored is locally and specific for the entity that creates a database. In other words, no instance shares data with another entity.

6.2.1.3 Network Interface

This component is the entity's interface to interact with the peer-to-peer (P2P) network. Therefore, it manages all modules of the libp2p library internally and offers a set of protocols to be configured by a specific entity. A protocol is an interface that wraps a request-response or publish-subscribe message pattern. Adding protocols to the network interface component will enable the entity to send, receive, or respond to messages of that protocol. Hence, we describe the payload and identifier of a specific protocol once and, using entities, add them to the network interface.

6.2.1.4 PEKS Schemes

These are our cryptographic primitives that have to conform to a specific interface. In Section 5.2.3.3, we showed the evaluation of multiple PEKS schemes. We used the PEKS interface to implement all schemes for testing purposes, including a Null scheme with no encryption. For our specific adaption of a PEKS, we require additional interfaces unsupported by traditional PEKS schemes. Therefore, this component has a second interface Re-Encryption offering our design-specific requirements of shuffling and re-encryption. Finally, we must persist and send the cryptographic primitives to the database and over the network. We require the Serialization interface that offers binary conversion of cryptographic primitives agnostic to the chosen PEKS scheme.

6.2.1.5 Queries

Our last shared module is the Queries component. Its purpose is to wrap the implementation of different queries independent of the data source and PEKS scheme in use. Therefore, it requires a specific Iterator and PEKS implementation and offers Evaluation and Serialization of queries. Like the PEKS Schemes component, the Serialization interface sends query data over the network to evaluate or encrypt the query on another node. Thus, other nodes in the network can reconstruct queries using the Serialization interface, connect them with their specific PEKS initiation and data source and evaluate the query locally. In summary, the Queries component ensures that every query is suitable for our system design but can also be evaluated

on simpler PEKS schemes and data sources, which we will use for benchmarking and testing purposes (cf. Section 6.2.2).

Takeaway. We introduced a set of interfaces offered by shared components of our software architecture. These include interfaces for importing data, interacting with the network, managing persistent databases, and evaluating queries locally or over the network.

6.2.2 Binaries

We developed multiple applications to run, maintain, and test our implementation. In this section, we will briefly introduce some of them that are of particular interest for evaluating and verifying the prototype.

Data Hub. The data hub binary is an adoption of the architecture of Yuan et al. [YMM⁺17] (cf. Section 3.3.1). Here we use a single application that combines the functionality of all our components except the network interface. Due to the PEKS scheme agnostic implementation of the database and queries, we can import and query data using different PEKS schemes. Therefore, we used the data hub as a macro benchmark for different schemes and as a testing tool for query evaluation without requiring a network setup.

Tooling. The tooling application is primarily designed to maintain the database. In particular, we can export and import data dumps, delete custom ranges of data records, and optimize the database for read operations. These methods are used throughout our evaluation to simplify the deployment of changing architectures.

Entities. Every entity of our system has a corresponding binary to start that entity. Generally, these servers are running nodes in the network and waiting for a query to be issued. On the other hand, the data user (DU) application is the entry point for testing. This application will always run a single query according to the DU workflow and stop after receiving the result.

Remote Control. The remote control is similar to the tooling binary but maintains the running network. It offers interfaces to ping entities, issue shutdown signals, or import data for a specific organization. In addition, it has access to a tracing protocol that broadcasts commands to all entities in the network, instructing the data collection of runtime metrics.

6.3 Implementation Details

Throughout the implementation of our prototype, we faced several challenges. While most of these are general software development tasks, we briefly recap two specific challenges that likely significantly impact overall performance. These are the simultaneous processing of thousands of requests and implementing a database structure that facilitates our usage of bucketization.

6.3.1 Non-Blocking Runtime for Network Communication

For the data provider (DP), test server (TS), and data owner (DO), we expect thousands of requests that have to be managed asynchronously without blocking newly incoming connections. On the other hand, we have to limit the usage of available resources such that multiple nodes do not interfere with each other. Thus, we had to find a balance that efficiently utilizes computational resources in limited ways to measure the impact of allocating more threads and simultaneously prevent CPU exhaustion and unavailability of nodes due to high workloads. Therefore, we implemented three different runtimes.



Figure 6.2 Visualization of message passing between the three runtimes and the network.

Network Runtime. The network runtime is the runtime of the Network Interface component. Its only job is to process incoming network traffic and send outgoing messages. Therefore, it does not process the protocols; it just forwards messages between the network interface and client implementation. Note that this has the least computational load, and due to our usage of asynchronous processing, it is sufficient to allocate one thread for this runtime.

Client Runtime. The client runtime is responsible for processing protocols. Therefore, it receives messages from the network interface and must process them according to the protocol specification. However, it outsources heavy computation to the task runtime to not block the client. Furthermore, the client can issue new network requests, usually only if a specific task requires it. In other words, the client runtime orchestrates tasks in the task runtime and bridges network requests to the network runtime. Similar to the network runtime, this orchestration requires no heavy computations. Thus, one thread is sufficient.

Task Runtime. The task runtime is where all heavy workloads are pushed from the client. Here we can configure the runtime to use a fixed number of threads with a work-stealing queue. This configuration gives us a fine granular control on thread usage, and even if some task in the runtime hangs, it does not block the network or client interface.

6.3.2 Database Structure and Algorithms

As covered in Section 5.2.4, we can use the structure of the query evaluation tree to filter candidates based on the union and intersection of buckets. Efficiently adopting this scheme was the main challenge in our database design. In this section, we introduce our approach to structuring and iterating our database to evaluate query-specific bucketization efficiently.

Recalling our definition (cf. Section 5.2.2), a bucket is a data group-specific index that maps records to a set of columns, and each column corresponds to the index of a field in that data group that matches the bucket. In our database implementation, we

store each bucket in a separate table with a key that uniquely identifies each record in the database. We store a packed bitfield encoding the indices of the matching data fields in the data group for the values. In other words, we set the i -th bit of the record value if column i matches the table-specific bucket.

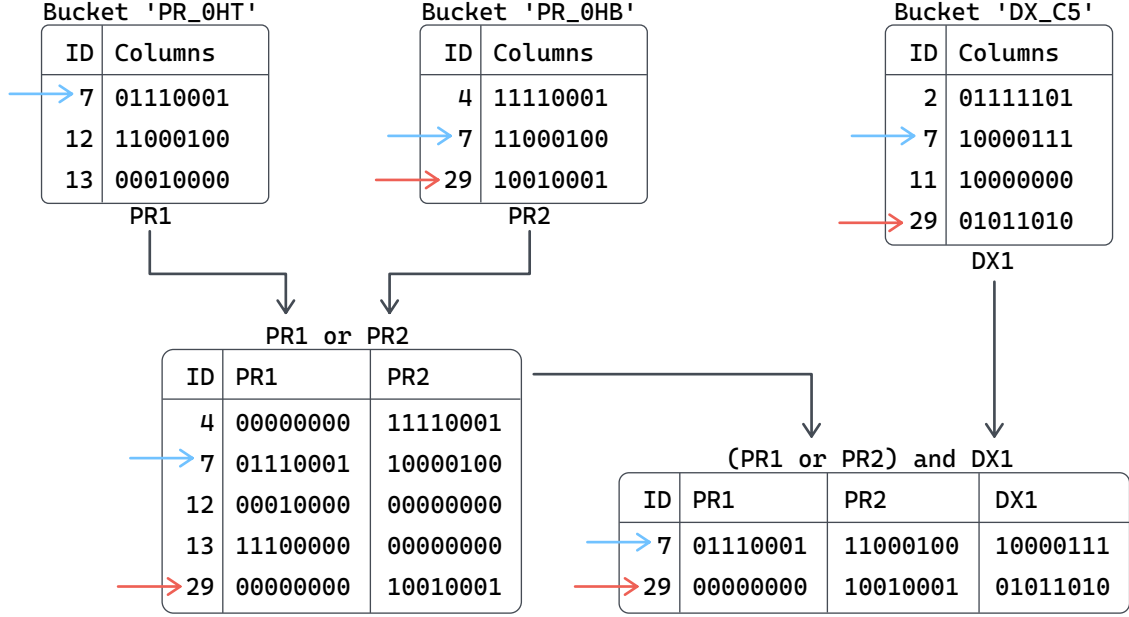


Figure 6.3 Exemplary evaluation of the set of candidates for the intersection of one bucket with the union of two other buckets. The blue and red arrows indicate the state of the current and next iterator, respectively. Notably, in the union of two buckets, the bitfield is set to zero if no corresponding record exists in one bucket.

Whenever we evaluate the bucketization of the boolean expression, we represent each leaf with a set of candidates. For that, we use an iterator that yields all records of the corresponding bucket table; here, our database implementation must support efficient iterators ordered by their key. The specific ordering of the key is inconsequential. However, it must be consistent across tables. That is due to the usage of our union and intersection implementation. Here we use a reflexive structure of an iterator that iterates the union or intersection of existing iterators. Utilizing the ordering of the key, we can consecutively iterate the dataset such that we will always traverse the iterator in a forward direction. Furthermore, we can exploit the availability of seek operations to skip a large chunk of record identifiers. That is, if one of the iterators of an intersection yields a high record identifier, we can seek all other iterators to the maximum observed key. Note that this way, we never allocate the entire set of candidates but rather iterate them record by record, thus having less memory consumption. One challenge in implementing the iterators was the requirement of generic values. This necessity is due to the reason that we require the individual fields to match each leaf. Thus, each combination of two iterators yields the cartesian product of the input iterator values.

Conclusion. In this section, we have shown how we efficiently evaluate a set of candidates for arbitrary query evaluation expression using bucketization. Therefore, we described the organization of buckets in a table format that efficiently utilizes the ordering of keys to iterate candidates in consecutive order. This way, we were

not limited to calculating the set of candidates but also never required to load all candidates at once.

6.4 Verification and Testing

Throughout the implementation and evaluation of our prototype, we maintained a set of tools for continuously testing and benchmarking our implementation. This section introduces tools deployed in our continuous integration pipeline to fix bugs and performance issues as early in the development cycle as possible.

Compilation and Static Code Analysis. In a development cycle, it is important to spot bugs as early as possible to prevent them from occurring in production and save time debugging the application. From that perspective, the best bugs are those that do not compile. Therefore, we always compiled with the highest warning level and treated every warning as an error. In addition, we utilized static code analysis tools to find error-prone parts of our code before we ran them in our test environment.

Unit Tests and Test Coverage. We wrote unit tests for all critical software sections and ensured over 90% code coverage of all cryptographic primitives. Here we aimed especially for all cryptographic protocols and the evaluation of queries, including major parts of the database with a coverage of more than 50%. Therefore, we used the random data generator and set up temporary databases. In sum, we covered major parts of the code base with good coverage to confidently state that our implementation works as expected.

Microbenchmarks. We implemented a benchmark suite for all cryptographic protocols, including benchmarks for the most fundamental hash computations and arithmetic operations on the elliptic curve. Using these benchmarks, we could compare our protocol implementations' expected and measured performance and spot divergencies where we had to improve our implementation.

6.4.1 Correctness of Query Evaluation

To validate the correctness of our results, we have three programs that use different methods to evaluate queries. Each program adds another layer of indirection, enabling us to increase complexity step by step and narrow bugs down to a specific feature. Throughout the implementation and evaluation of our prototype, we tested our queries and datasets to yield consistent results across all methods.

The first program is part of our toolset, takes a raw dataset file as input, and computes the query for every record without any encryption, database usage, or bucketization. This way, we avoid as many bug sources as possible. Notably, this method is implemented from scratch and does not share any source code with the query evaluation used by our prototype or data hub.

The second program is our data hub. As noted in Section 6.2.2, the data hub implementation is public key encryption with keyword search (PEKS) scheme agnostic. Thus, we can use this implementation to check the consistency of all eight PEKS schemes we implemented. Here the data hub uses the database implementation,

including the bucketization. To test the database without encryption, we have an additional PEKS scheme implementation called Null that does not use encryption.

Third is our prototype with the entire network stack and usage of re-encryption. To test our adaptation of the Rhee et al. [RPSL10] PEKS scheme without the necessity of network virtualization, we have also added our thesis scheme to the data hub. To summarize, we have five methods to test different layers of our query and PEKS implementation. All of these methods yield consistent results throughout the evaluation of our prototype. As a result, we are confident that the evaluation of queries works correctly.

7

Evaluation

In order to assess the feasibility and conformance of our system design for practical use and in alignment with our objectives (cf. Section 4) we need to systematically evaluate our system following our goals. Consequently, in this chapter, we elaborate on the compliance of our system to every non-functional objective: performance, scalability, privacy, and security. For this purpose, we first introduce our research methodology and then subdivide the evaluation in correspondence to our objectives. Based on the observations made, we will then continue to assess some of the performance and privacy decisions made before discussing on the characteristics of our distributed health data network (DHDN).

7.1 Methodology

This section outlines the process and materials used in our performance analysis. Therefore, we start with the setup of our testbed, including network simulation and entity configuration. Following, we introduce a real-world dataset and queries used for evaluation in related work and this thesis. Further, we highlight some insights on upgrading test queries to newer International Classification of Diseases (ICD) standards. After that, we summarize our methodology for collecting, processing and analyzing performance metrics.

7.1.1 Testbed

To obtain consistent and comparable results that are also representable for real-world use cases, we start by stating the setup of our test environment used throughout the thesis. For that, we performed all tests and benchmarks on the same machine. That is a two-socket server with 96 simultaneous threads and 512 GiB of RAM.. The hard drive is PCIe connected NVMe SSD, and for the operating system, we chose Ubuntu 22.04.

7.1.1.1 Network Simulation

To simulate representative use-cases we have to use a reasonable simulation of deployed real-world networks. Mininet [LHM10] is a software that is explicitly designed for that purpose, creating realistic virtual networks on a single machine. These virtual networks are categorized by three components, namely a controller, switches and hosts. Here the controller and switches build the backbone of our network which is a point of presence (PoP) level abstraction of the interconnection of geographical locations. Following hosts, are connected to the backbone and run the network specific application, which are the entities in our case. In the context of this section we will only cover the connectivity of hosts to the backbone and follow up with the configuration of entities running on these hosts in the next section.

Backbone. For analysis of real-world PoP level networks Knight et al. [KNF⁺11] released a collection of topology data for more than 140 existing networks, called ‘The Internet Topology Zoo’. Two years later, Großmann and Schuberth [GS13] proposed Auto-Mininet, a method to automatically generate testing environments based on the ‘The Internet Topology Zoo’ dataset using Mininet. In this thesis, we adapt their methodology with two slight variations. First, to efficiently route messages and simplify network management, we utilize the Software Defined Networking (SDN) interface OpenFlow [MAB⁺08] to connect a different external controller. Here we chose Onos [BGH⁺14] because the developer of Mininet encourages to use of either Onos or OpenDaylight for network topologies with loops, and according to research, Onos has better performance [SHV⁺15, RQIB22].

Host Connectivity. Another variation to Auto-Mininet is choosing additional connectivity parameters to connect hosts. Because we are linking hosts directly to the backbone, we need to model the peering links from internet service providers (ISPs) to customers. Here we distinguish the data owner (DO) connectivity from other hosts, as we consider DO to use a mobile device. For choosing the bandwidth, we go with the most available coverage in Germany, DSL for fixed and G2 for mobile networks. Accordingly, we choose 0.1 Mbit for DOs and 16 Mbit for all other hosts. To simulate the connection overhead of routing systems to the backbone, we chose a delay according to measured round trip times of IoT [GS18] and mobile devices [RVGH13]. Here we set the delay to 50 and 30 milliseconds for DOs and other hosts, respectively.

7.1.1.2 Entity Configuration

While our network setup yields all necessary hosts to run our entities, we have to configure the entities additionally. Although this configuration includes their relationship and connectivity, these are naturally given by the design of our network and omitted here. Here, we highlight two configurations of importance for evaluating vertical and horizontal scaling. For thread usage, we need to specify an exact number of utilized threads to evaluate the impact on vertical scaling and prevent exhaustion of the CPU, which would impact the performance of other nodes. Next, we need to distribute data across multiple instances to evaluate horizontal scaling.

Sharding. In order to split data across multiple instances, we make use of sharding. We divide the dataset into multiple equally sized shards and assign each entity a different one. One problem of this method is the incompatibility of stored data

to a specific host identity. For example, test servers (TSs) stores index fragments bound to the TS specific secret key. If we would share this fragment across multiple servers with different keys, they would not be able to match their data. To solve this incompatibility, we force all entities to have the same key pair for all cryptographic computations on the public key encryption with keyword search (PEKS) scheme. We tested the system without this unified key compilation flag, and it neither impacts correctness nor performance measurements.

7.1.1.3 Simulation of Data Owners

One challenge of the test environment is the virtualization of DOs. Simulating six million individual entities would exhaust our server capacities. Thus, we had to come up with two optimizations for testing purposes. First, we developed a single node that is responsible for simulating multiple DOs, which can also be used for artificial data (cf. Section 5.1.1). Due to our asynchronous task scheme, we can spawn multiple DOs simultaneously that share the same CPU resources but can wait simultaneously on network traffic. Thus, we can prevent CPU exhaustion and utilize idle times to spawn more DO to poll for query status updates. In our implementation, we configured the DO node to simulate up to 2,048 DOs simultaneously. Accordingly, we increase the bandwidth of the running host connectivity to 204.8 MiB. The second optimization is an optional flag that prevents the DO from polling for new queries but waiting to be notified with a list of candidates that are part of a new query. This way, we do not have to wait for most of the simulated DO to receive an empty **ACQUIRE QUERIES** response but only simulate those that are part of the query.

Takeaway. We presented a method to create a representative virtual environment simulating real-world networks. To allocate our host machine’s limited resources and assess the impact of scaling computational power, we introduced the configuration for thread utilization. We utilized sharding to distribute data among multiple instances of an entity. Finally, we have shown that we can simulate a multiple thousand DOs with a specialized entity.

7.1.2 Test Data and Queries

In addition to the simulation of networks, we want to assess our system based on real-world patient data and use cases to perform queries that are part of practiced research. Therefore, we chose to use the same dataset source and queries as the work of Yuan et al. [YMM⁺17] to promote reproducibility and comparability. The Healthcare Cost Utilization Project (HCUP) Nationwide Inpatient Sample (NIS) dataset is publicly available with more than six million anonymized real patient data from the United States. In contrast to the work of Yuan et al. we decided on a newer version of the HCUP NIS dataset from 2020, with 6.47 million records. While this recent release keeps our work up to date with the recent dataset development, it comes with a newly updated encoding of diagnosis and procedure codes from ICD-9 to ICD-10. As these encodings are incompatible, we must adapt the query formulation and bucketization strategy.

Query Formulation. To convert queries, we used an online conversion tool that maps ICD-9 to ICD-10 codes using an equivalence mapping released by the Centers

for Medicare & Medicaid Services. Note that this is a one-to-many mapping, and some diagnoses or procedures have multiple equivalent codes in ICD-10 due to more granular distinctions [Car13], e.g. the severity of a diagnosis.

Bucketization Strategy. For our adaption of the bucketization strategy, we develop a configuration that yields a comparable number of candidates for each query. As the length of codes increased from ICD-9 to ICD-10, creating more specific buckets is required. Here we chose to use the first two and three letters from diagnosis (e.g. ‘S8’ for ‘S89102S’) and procedures (e.g. ‘DG2’ for ‘DG20HZZ’) respectively.

To assess our conversion, we compare the results of Yuan et al. [YMM⁺17] using the 2013 HCUP NIS database with our adapted queries using the version from 2020. Table 7.1 compares the number of candidates after applying bucketization and the results of query evaluation.

Work	Nam	[YMM ⁺ 17] Cand.	Our Cand.	[YMM ⁺ 17] Result	Our Result
[CRMC15]	Chung	3,709	3,371	1,097	1,404
[GTA ⁺ 14]	George	8,046	6,529	397	552
[HTHB14]	Habermann	15,385	7,698	10,918	41

Table 7.1 The number of candidates and results from Yuan et al. [YMM⁺17] in comparison to our evaluation on a newer dataset with converted ICD codes.

Our conversion seems reasonable for the first two queries, with slight variations expected for different datasets. However, the Habermann query matches in our adaptation significantly less than in the work of Yuan et al. despite having a comparable number of candidates. This difference is because the corresponding procedures have become much more granular. To overcome this issue, we created extended versions that include additional codes reasonably considered as new distinctions of their ICD-9 predecessors. Note that we have no medical background, and this extended version only measures the impact of more granular procedures and diagnosis codes on performance.

Work	Name	Conv. Cand.	Ext. Cand.	Conv. Result	Ext. Result
[CRMC15]	Chung	3,317	4,787	1,404	1,789
[GTA ⁺ 14]	George	6,529	12,011	552	577
[HTHB14]	Habermann	7,528	10,918	41	4,632

Table 7.2 Comparison of the number of candidates and results between the converted and extended ICD-10 adaptation of the test queries.

Takeaway. Using the HCUP NIS database, we utilize a real-world dataset of more than six million records for our evaluation. We convert existing queries of related work to a recent release of our dataset with different encoding for procedures and diagnosis, achieving similar results compared to our related work.

7.1.3 Performance Analysis

In the previous section, we focused on setting up a testing environment, virtualizing a network, configuring hosts, and providing test data and queries. For this section, we focus on how to measure and analyze runtime execution within our testing environment to assess overall performance and the computational workload distribution of smaller code segments.

7.1.3.1 Measurement of Runtime Metrics

To measure the performance of our application, we use two different methods to collect metrics of running entities. These are event logging and code instrumentation.

Event Logging. In this method, we log certain events during the execution of the program. Every log is associated with a specific process and contains a timestamp and arbitrary message that may encode additional data. Therefore, this method is suitable to measure all times of certain stages of execution, with little to no overhead to the runtime, if used sparsely. However, the disadvantages of this approach are its low accuracy in determining the performance of smaller code sections and the incapability to measure the effective CPU time of asynchronous tasks.

Code Instrumentation. To compensate for the disadvantages of event logging, we incorporate code instrumentation. We explicitly call a tracing API for some code segments whenever we enter or leave that segment, measuring its CPU time. Thus, we can examine the CPU time of small code segments at the cost of some inaccuracy due to the induced overhead of the tracing library. Another benefit is the cooperation with asynchronous runtimes. Here we instrument the runtime to call the tracing library whenever a task resumes monitoring busy and idle timings separately. In addition, code instrumentation is not limited to time measurements. We record the size of individual messages and instrument network sockets to trace the bandwidth usage above the TCP-IP layer. Notably, the induced overhead is one major disadvantage of this method. Therefore, we will examine its impact in Section 7.5.1. Nevertheless, due to its impact on the measurements on wall times, we only enable code instrumentation if we require to inspect the extra measurements.

7.1.3.2 Data Analysis Workflow

For the previous sections, we described how we set up a testing environment and collect runtime measurements. We will use methods during our evaluation to gather data and analyze different network setups. Consequently, we require an automated and timesaving system capable of running tests, collecting measurements, and analyzing the results. Therefore, we managed the entire workflow from test configuration to the generation of charts in Python scripts. For this section, we introduce this workflow stating our evaluation methodology.

Our workflow is divided into three consecutive stages, data collection, preparation, and evaluation. The first one is responsible for running the test environment and saving measured data in a structured way such that the second stage can parse this

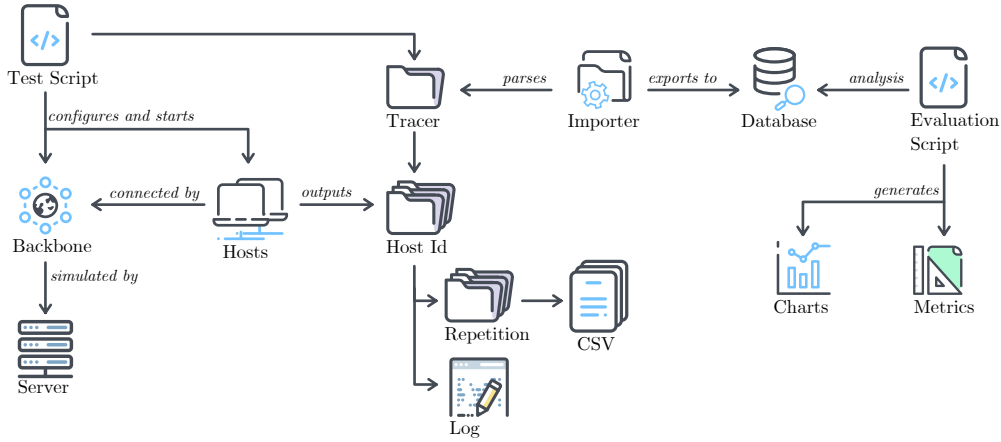


Figure 7.1 Schematic visualization of our evaluation workflow, with tasks being executed in order from left two right.

structure and export it to a database. Finally, the third and last stage evaluates the data utilizing this database and generates metrics and charts for this thesis.

Data Collection. The data collection is executed by a test-specific script responsible for setting up the testing environment, commanding the network to perform specific tasks, and outputting measured data in a structured way. Setting up the testbed is mostly covered in the previous sections. Accordingly, the test script loads an initial configuration of the testbed, which includes a pointer to a ‘The Internet Topology Zoo’ data file for the backbone and then connects and configures all hosts required for included configuration of entities. Here the script is not limited to a static network; especially for testing scalability, the script may create different network architectures to test. In order to ensure that the parser can link measurements in the database with the corresponding test case, we also have to store collected data in a structured way. First, the script must configure every host to log into a specific directory that acts as a tracer. This tracer is the connection to the parser, encoding the specific test case and its testing parameters so that we can uniquely identify a specific test script run in our database. Second, entities log into a host-specific directory that encodes a unique identifier and contains a numeric list of subdirectories for each repetition of the same test. This way, we can later distinguish multiple runs of the same test to get more accurate and stable results. Therefore, we ran every test fifty times for all data measurements in this thesis.

Data Preparation. In this stage, an importer parses the output structure of tests to export all measurements into a database. Here we distinguish between traces that uses code instrumentation and store their data into Comma-Separated Values (CSV) files and event logs. In the latter case, the importer parses the log file to extract all times. Finally, we store all gathered data in test-specific prefixed database tables.

Data Evaluation. For the last stage, we use a mixed solution of Structured Query Language (SQL) scripts and Python libraries for data analysis and chart generation to evaluate the test measurements contained by our database. Once we carve out some noticeable results, we put some final cleanup efforts to present our findings coherently and export them for use in this thesis.

Takeaway. To evaluate the feasibility of our system, we require a realistic testbed with real-world test cases and the possibility of running tests for changing environments.

Therefore, we introduced our methodology stating the usage of network virtualization software, adopting real-world data in a testing environment that can be run and configured programmatically. In the next section, we exploit this system to measure the performance and scalability of our system design. Therefore, we first use an artificial network architecture where every node is connected to a single backbone switch, lowering the possibility of untracked environmental influence and promoting comparability and consistency of our test cases. In continuation to these artificial cases, we will then fully use our virtualization stack with a real-world use case scenario in Section 7.3.4.

7.2 Performance Analysis

For this evaluation, we assess the conformance of our implementation to the non-functional performance objective. In other words, we review whether our system is practical for usage on real-world data and whether we can utilize horizontal and vertical scalability of test servers (TSs). Therefore, we start by analyzing the time to query data in correlation with the number of records. We then continue to spot the most time-consuming code regions and inspect their composition. Based on the elaborated hot spots, we then test the capability of our network to utilize multi-core architectures and cloud service providers (CSPs). Finally, we repeat the previous steps similarly for analyzing bandwidth usage.

7.2.1 Query Duration in Correlation with Number of Records

We sequentially imported batches of one hundred thousand records to evaluate the impact of an increasing number of data records on the query duration. After each import, optimize the database for read operations and then repetitively perform all test queries. For the query evaluation, we located a single host for each entity in one city with a threads assignment of sixty-four threads for the data provider (DP) and TS each, thirty-two threads for the data owner (DO), and eight threads for the collective authority (CA) and data user (DU). Note that we assigned more threads than available, but according to our system design, the time window of overlapping computation for the TS and DP should be very short. We then logged the wall times for every entity from its first observation until the entity performed all tasks for that query. In the case of the DUs, this wall time corresponds to the entire query lifetime. Consequently, these measurements have an inclusive relation to dependent entities. For example, the data provider cannot forward its result to the distributed health data network (DHDN) until every TSs reports its result. Hence, measured wall times are not suited to analyze concurrent tasks but rather the time of sequential stages in the queries' lifetime.

Figure 7.2 shows the query wall times for each entity. Here we point out three notable findings.

First, the overall duration for all queries grows linearly, and none exceeds thirty seconds. Therefore, and in comparison to other DHDNs (cf. Section 3.3), we can confirm that our implementation is performant enough to evaluate queries on real-world data meeting the first subgoal of our performance objective.

Query wall time for increasing number of records

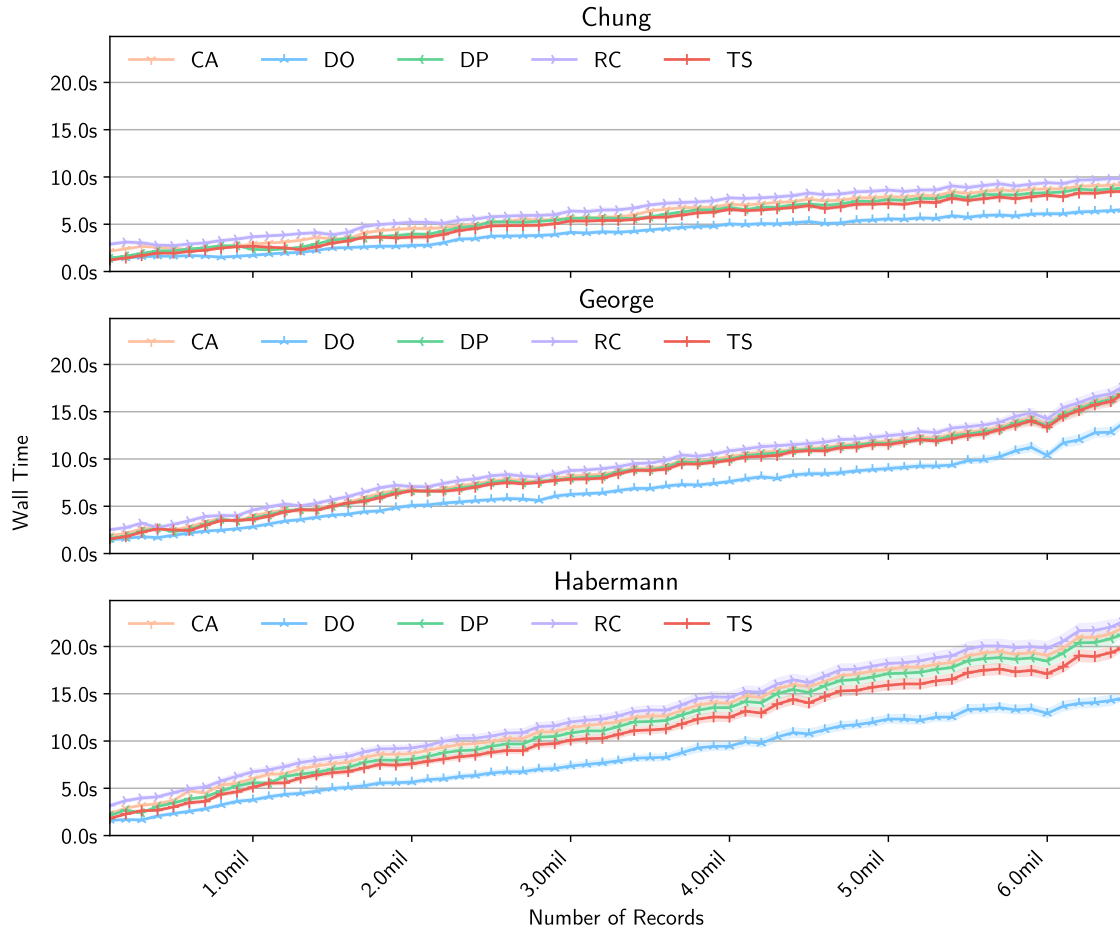


Figure 7.2 Observable queries lifetime from the perspective of different entities in correlation with the total number of records. 99% confidence intervals are displayed by a slightly transparent border of the corresponding lines' color.

Second, the gap between entities finishing after the TS is relatively small compared to the DO and bottom gap. That suggests that the computational load is mostly on the TS and DO. In addition, the weight of the TS seems to be higher as the gap between TS and DO increases with the number of records.

Third, gap sizes between entities finishing after the TS are not increasing, with the number of records reassuring that the aggregation of results is independent of the number of total records. However, query construction should also be independent of this figure, which we have yet to assess.

Please note that the outliers of Figure 7.2, reducing the time from the previous iteration, are due to some internal database optimization algorithms. These outliers correspond to faster database read operations of the DO. Although we could lower the height difference by running the database optimization programmatically, this optimization works differently for some thresholds and yields better results. Thus, we can trace the outliers back to the database optimization, but we did not further investigate different options for better optimization.

To further reassure our observations, we next analyze the CPU time of every entity spent to evaluate a query.

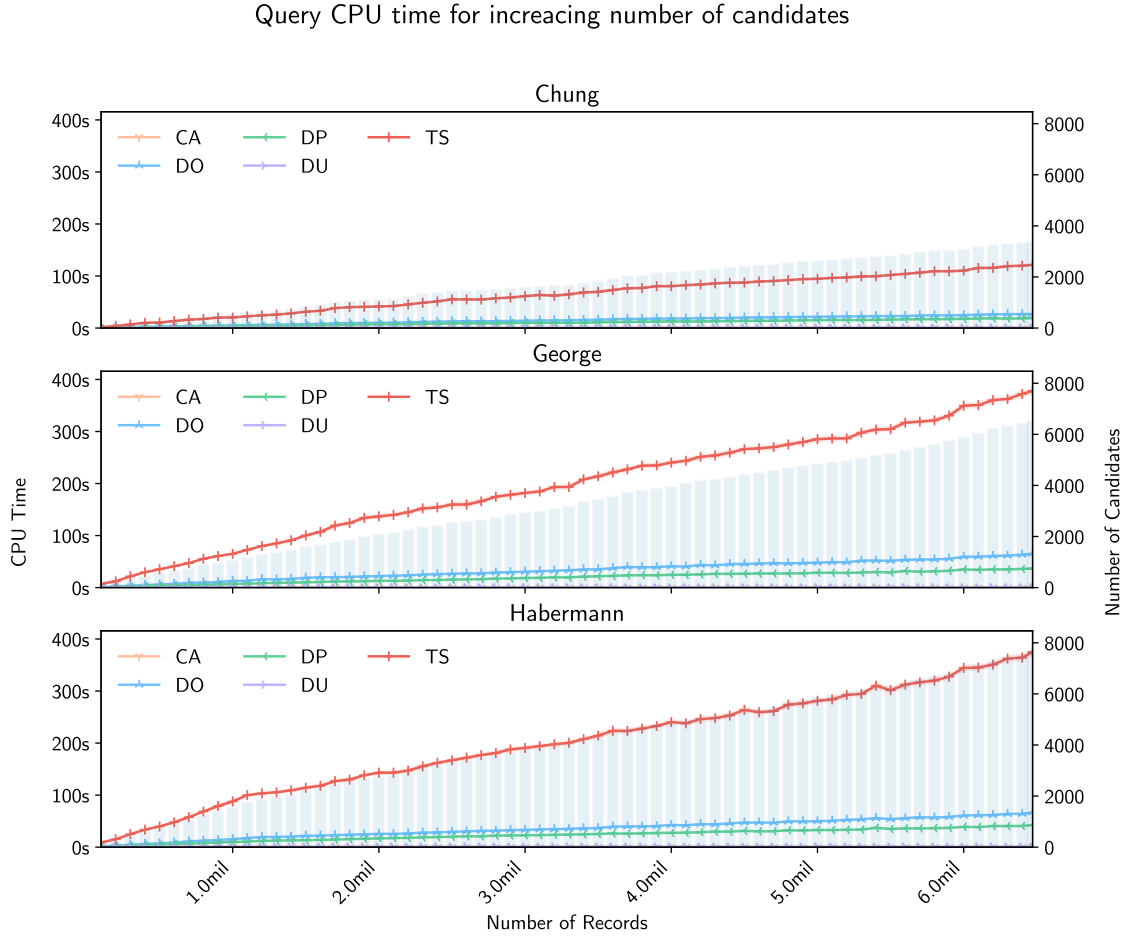


Figure 7.3 Total CPU time for entities to evaluate a query in correlation with the total number of data records and filtered candidates. 99% confidence intervals are displayed by a slightly transparent border of the corresponding lines' color.

Figure 7.3 confirms that most of the computation is done by the TSs and scales linearly with the number of records. Alternatively, to be more precise, the number of candidates grows linearly with the number of records, highlighting that the import has no particular order on criteria relevant to the queries. Notably, DU and CA constantly stay around zero seconds. Thus, we can conclude with high evidence that all of their computations are independent of the number of data records.

The chart also shows a scaling CPU time for DPs and DOs. Naturally, the DO simulation workload increases with the number of candidates as more DO must be simulated. Thus, every DO has to poll the DP for new queries and acquire requested consent which inherently increases the DP load.

In order to ensure that we are aware of all workloads scaling in the number of candidates and to benchmark their share on CPU consumption, we further evaluate CPU time spent for every candidate. Therefore, we analyze the average CPU time of every candidate-based computation and accumulate these corresponding to the entity that performs this computation.

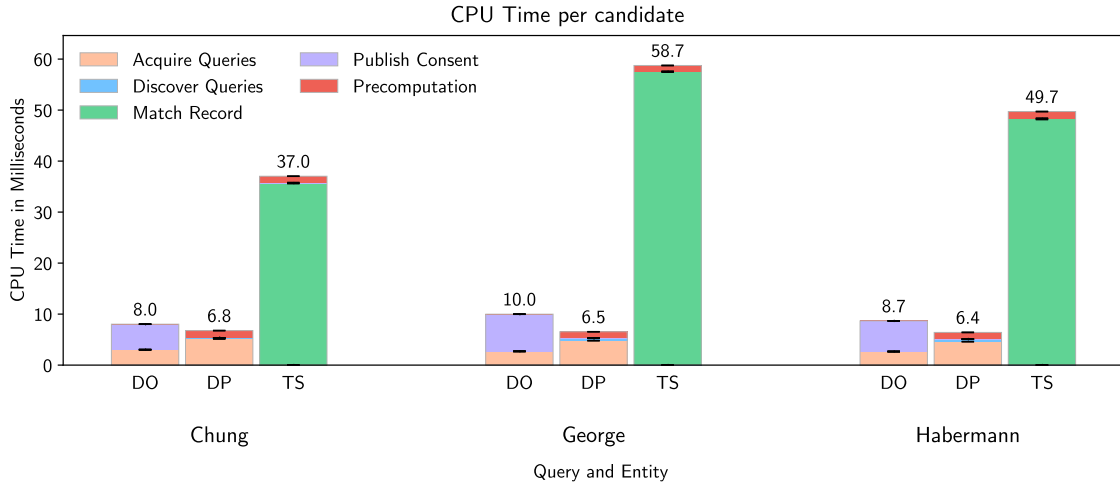


Figure 7.4 Aggregation of the CPU time spent on average for every candidate across all protocols for each query. Error bars are shown for a 99% confidence interval.

The fact that Figure 7.4 reflects the slopes of Figure 7.3 with high accuracy shows that we could identify all significant scaling workloads. In addition, we highlight two important details for our further investigation.

First, the discovery queries operation takes an insignificant amount of time to filter a candidate. This operation is the only protocol to be computed for every record if we would not simulate DOs for candidates only. Our linkability of anonymous messages reduces computational overhead to the number of candidates and enables efficient filtering.

Second, only the publish consent and match record operation vary considerably for individual queries and are the most time-consuming computations for the DO and TS, respectively. As chosen in our design chapter, the most computational power is required by TSs, and the less powerful DO devices only require a fraction of that performance. Following, we investigate further into these two operations to evaluate the distribution of CPU time in different categories spotting specific parts of the implementation which would benefit the most from further optimizations.

7.2.2 Analysis of Hot Spots

In the previous section, we analyzed the query evaluation time and computational workload for increasing data records. We showed that two operations represent the most CPU time spent: the DO consent publication and the TS record matching.

One point of particular interest is the difference in CPU time of publishing consent and matching records for different queries. Here we found a direct correlation with the average number of requested consent and indices tested by the TS (shown in Table 7.3), which seems likely as both scales in the number of trapdoors.

One marginalia on the comparison of requested consents and performed public key encryption with keyword search (PEKS) tests in Table 7.3 is that the latter tends to be much higher than the former. The reason for that is the structure of ICD codes, and that consent is only required for a particular field. In other words, if a DU looks

Query	Characteristic	Avg. Number	Deviation
Chung	Requested Consent	2.7	± 0.007
	Index Tests	10.2	± 0.033
George	Requested Consent	5.0	± 0.004
	Index Tests	16.4	± 0.023
Habermann	Requested Consent	3.6	± 0.004
	Index Tests	14.4	± 0.045

Table 7.3 Comparison of the average number of consents requested and tests performed by the TS for each record.

for a specific procedure with multiple case distinctions that do not matter to them, they may search for eight different procedure codes, but it is likely that a patient only had one of those procedures. As a result, the bucketization will yield the one required field that matches the prefix of all procedures queried. However, from that, which variation is stored in that field is indistinguishable, and the TS has to match every procedure code against the one consented field.

That these characteristics are indeed the cause of the differences for each query is further assured by our analysis that evaluates these two operations in greater detail. Therefore, we split all computations into three categories, cryptographic operations, database transactions, and other computations. The latter categorizes all CPU times that are not assigned to one of the first two groups. We further distinguish different operations in one category by a specific identifier embedded into our code instrumentation. In the following, we use these measurements to distinguish further the computational load of the two most significant operations.

7.2.2.1 Match Record

In this operation, the TS matches the published consent to a corresponding data record. This operation is performed for every candidate of the query. Figure 7.5 shows a comprehensive evaluation for the match record operation.

We observe that over ninety percent of the time is spent evaluating the PEKS Test function in all queries. As this code segment corresponds to a direct call on the cryptographic primitive, we conclude that there is not much opportunity for optimizations rather than to improve the elliptic curve cryptography (ECC) implementation or choose a different PEKS scheme. This also affirms our previous assumption that the number of PEKS tests result in different average CPU times of this operation.

One minor optimization point are the two database read operations, one for the required code fragments and a second for the stealth bit. While the former is batched into a single transaction, the latter is not. The reasoning was to conditionally load stealth bits only if the record matches; however, due to its high overhead, it most certainly would benefit the performance to read the stealth bits together with the consent data.

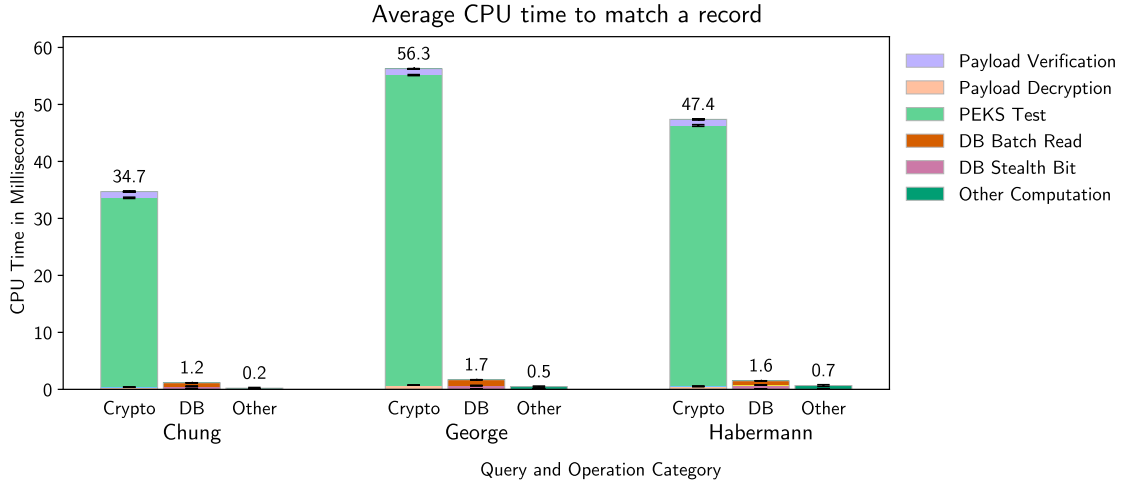


Figure 7.5 Distribution of the index operation CPU time to all called subroutines.

7.2.2.2 Publish Consent

Like the match record operation, this one is performed for every query candidate, but by the DO. Figure 7.6 shows the comprehensive evaluation for the match record operation.

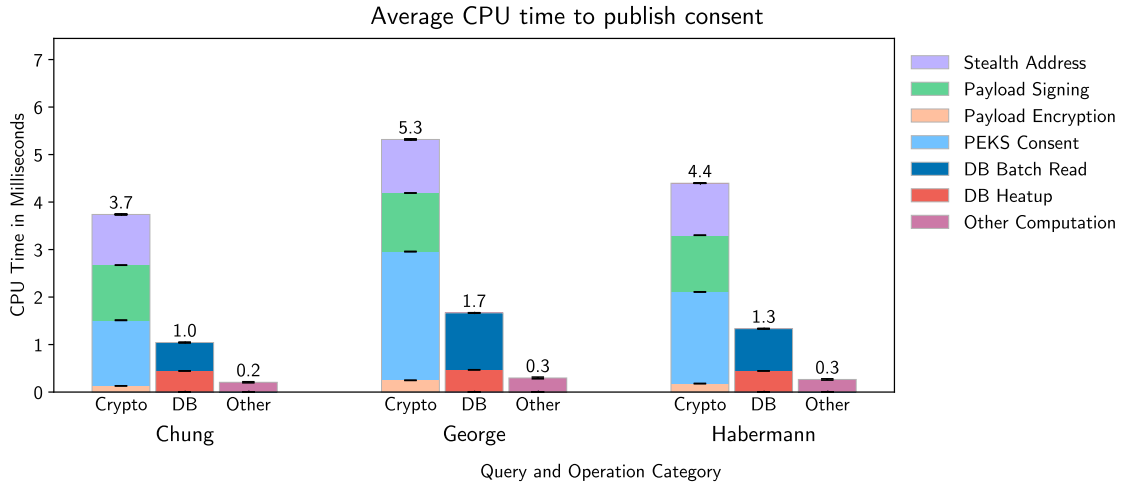


Figure 7.6 Distribution of the publish consent operation CPU time to all called subroutines.

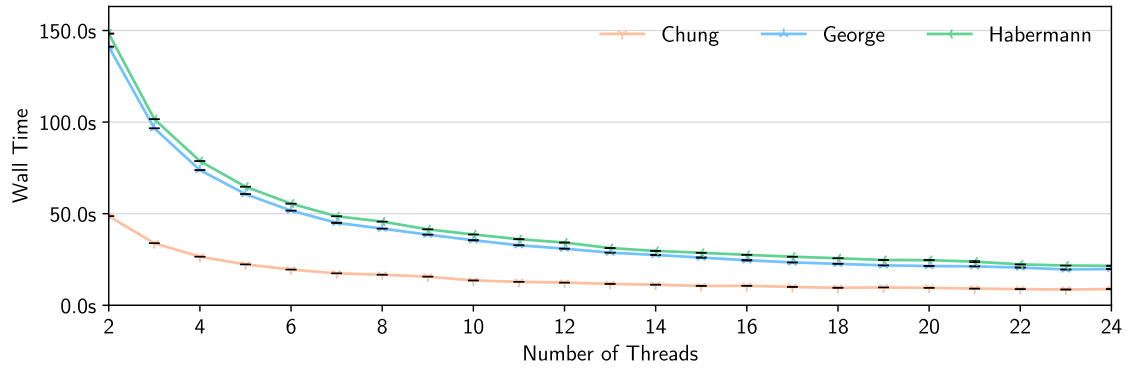
Here the database operations have similar timings to the match record operation. Both load equally sized blobs from the database because the consent witness and fragment size are the same in our implementation. We can also observe some room for optimization, as the heat-up routine takes a noticeable amount of time but could be cached for all operations.

On the other hand, the PEKS operations are the only crypto operations that scale for a noticeable amount concerning different queries but are relatively close to other cryptographic primitives. Thus, we spent much of the time on stealth address integrity protection, which may be optimized by choosing a different scheme or protocol. However, although these portions are relatively high, the overall duration is quite low for cryptographic schemes based on asymmetric encryption and sufficient for our application.

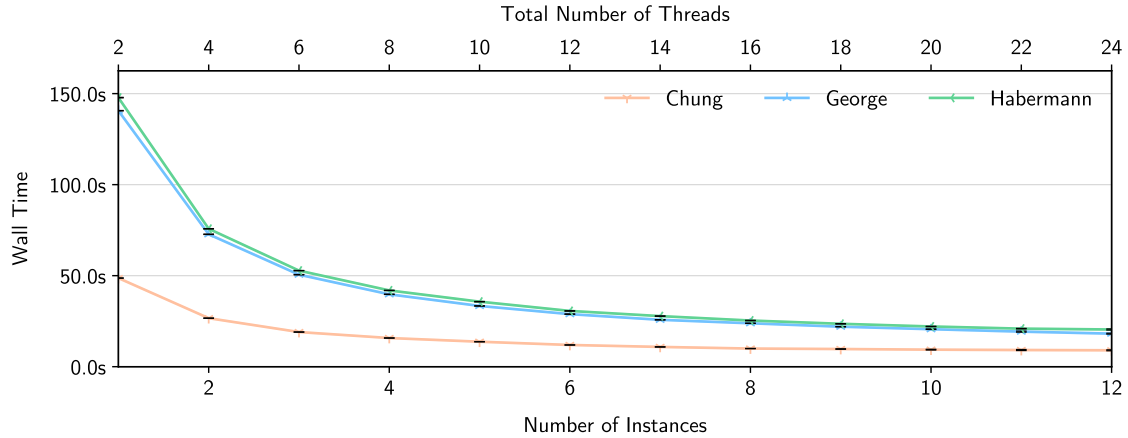
Takeaway. This section shows that our design can perform practically used queries on real-world data in a reasonable time. We further investigated how our system scales with increasing records, highlighting hot spots of our implementation. These two hot spots are performed by the TS and DP that are chosen to scale by design. While we already implemented our DO to simulate multiple DOs in parallel, we have yet to asses whether we can also scale the TSs instances efficiently.

7.2.3 Vertical and Horizontal Scaling of Test-Servers

In the last section, we have elaborated that most of the query evaluation is performed by the TS. Here, we build upon the previous test setup to test the scalability of TSs. We ran the queries only for the full dataset but with dynamically changing thread allocation and number of TSs for vertical and horizontal scalability, respectively. We then observed the maximum wall time of all instances.



(a) TS wall time per query for increasing number of threads.



(b) TS wall time per query for increasing number of instances

Figure 7.7 Comparison of query duration in correlation to vertial (left) and horizontal (right) scaling of the TS. Error bars are shown for a 99% confidence interval.

Figure 7.7 shows the results of our tests. For smaller thread allocations, we observe the desirable effect of duration bisection. However, we get a diminishing return for an increasing number of threads. The reason for that is the limit of our DO simulation, limiting the TS computation to available consent messages. Thus, increasing computational power causes us to cap the available data and wait for the DO. We

will see that we can further decrease the query time once we introduce multiple DP, each with its instance of a simulating DO, in Section 7.3.2. Another noticeable observation is the negligible difference between the vertical and horizontal scaling. Thus, the DP is free to choose between monolithic and distributed distribution of their computing capacity.

7.2.4 Bandwidth Usage and Message Size

Another aspect of scalability that is unattended so far is bandwidth usage. While this is inherently included in the analysis of all times, we cannot anticipate if we might hit a threshold by further scaling the system due to our bandwidth limitation. This section repeats our previous course of action and investigates bandwidth usage. Therefore, we start by analyzing the total bandwidth usage for an increasing amount of data and then break down total usage into different messages.

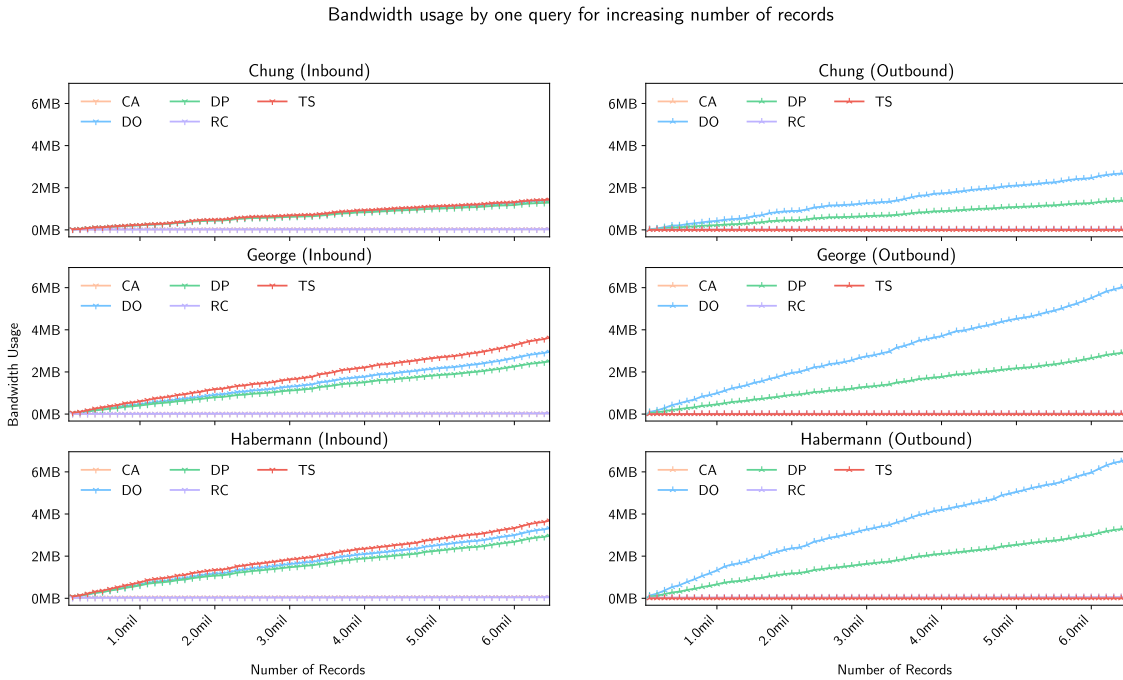


Figure 7.8 Confidence intervals are omitted as

Figure 7.8 shows the total bandwidth of entities in correlation to an increasing amount of data for different queries. Here we have three observations we like to share.

First, the bandwidth usage only scales for DP specific entities, namely the DP, TS, and DO. This observation reassures our design works as intended with query construction and result aggregation scaling independent of the number of records. Second, while TS and DO are designed to scale, the DP is currently considered a monolithic server but has a noticeable bandwidth usage. Thus, for a production-level system, we recommend an internet connection suitable for industrial applications, e.g. a synchronous 1 Gbps fiber connection. Last, the TS has the highest inbound usage in all cases but not outbound bandwidth, indicating that the consent messages induce the highest load on the network. This hypothesis is supported by the fact that the DP and DO seem to share a large portion of their mutual bandwidth. However, the

DO has a significantly higher outbound usage, which could correspond to the consent publication. To further assess the bandwidth usage of messages, we accumulate the sizes of all sent messages in Figure 7.9.

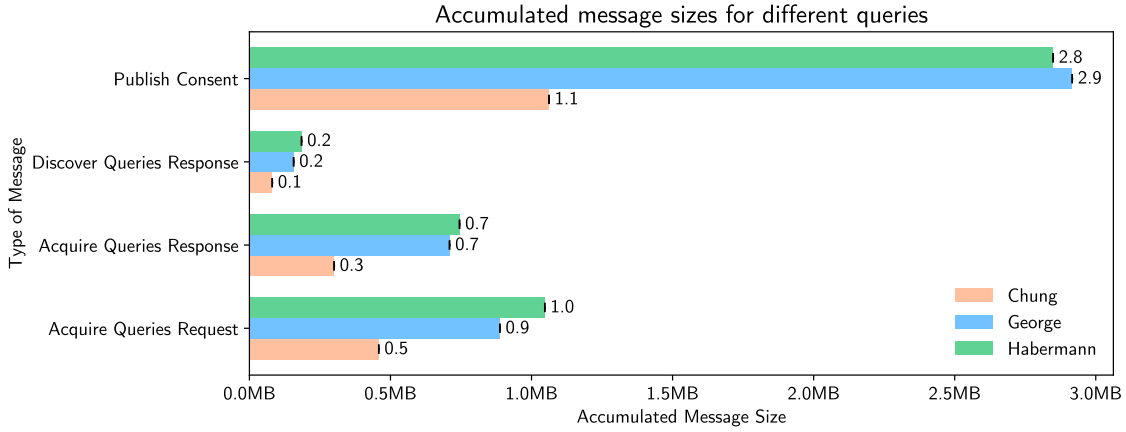


Figure 7.9 Accumulated size of all messages sent for specific protocols during the evaluation of a query. Note that request and response messages are listed separately, and message sizes are measured on top of the request-response protocol layer. Error bars are shown for a 99% confidence interval.

As predicted, Figure 7.9 shows that the consent publication takes the majority of bandwidth, followed by the messages between DP and DO. One unexpected observation is that the sum of all messages only reflects about seventy percent of our total bandwidth usage. Thus, the underlying protocol layers take a significant amount of our overall usage. For further assesment, we inspect the individual sizes of single messages.

Figure 7.10 shows the average size of messages sent. Notably, the messages with the smallest average size are the largest accumulated due to their high emergence scaling with the number of candidates. This case explains the gap between observed message sizes and bandwidth usage, as every message sent has a small overhead due to lower layer protocols that are relatively large compared to the small size of the message. Nevertheless, all messages sent and received by the data owner are sufficiently small for UMTS connections as their sum does not exceed one kilobyte, which is one-hundredth bandwidth per second.

Furthermore, we can review the size of messages that do not scale with the number of data records and were lost in Figure 7.9 due to their negligible accumulated size. Similar to our previous evaluation of CPU time, we observe that the message sizes differ significantly for queries. Here this difference correlates to the number of trapdoors of that specific query, which is decisive for the overall query size. Nevertheless, all message sizes are reasonable to be used in interactive protocols and should not impede the overall performance of the query as their sizes are far from the corresponding bandwidth limitation of the entities that send or receives them.

One last minor observation is the constant size of result propagation. Thus, these message sizes are independent of the number of records and the query. However, we notice that the **PUBLISH RESULT** message is slightly larger than the **FORWARD QUERY RESULT** message due to the Back Linkable Spontaneous Anonymous Group (bLSAG) signa-

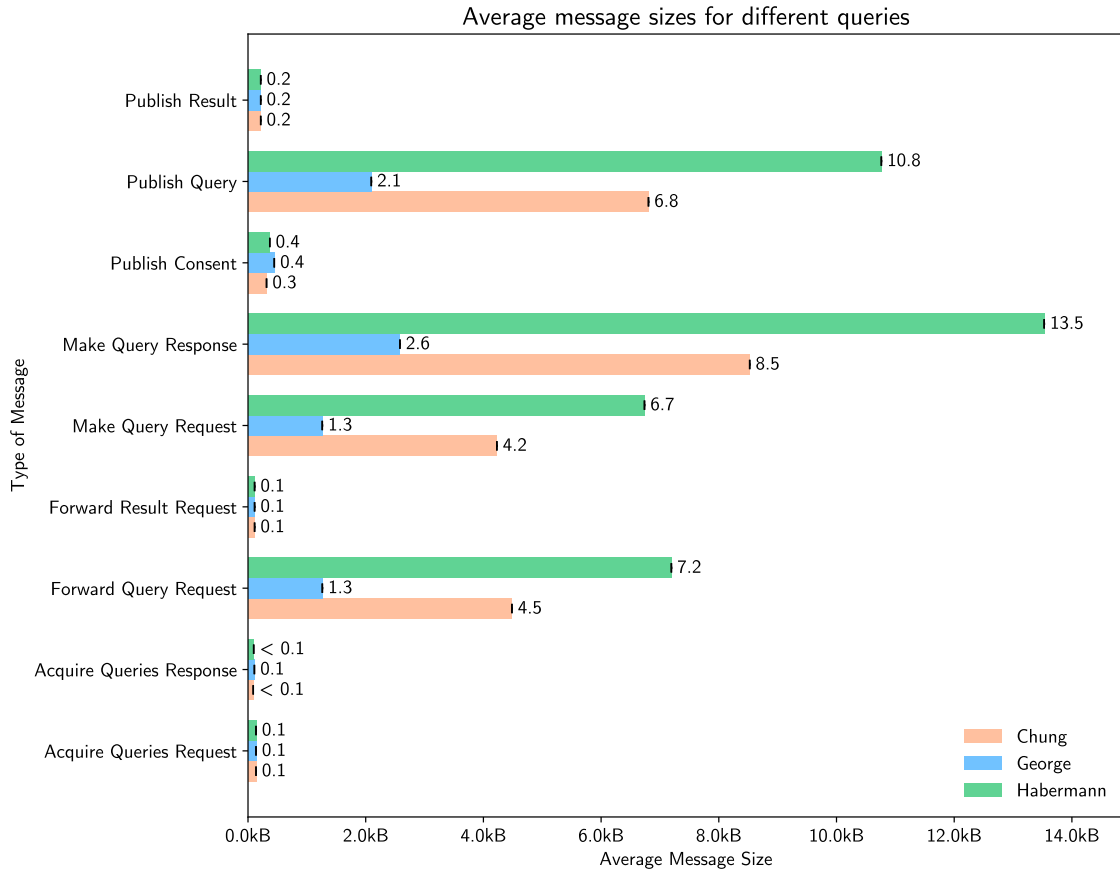


Figure 7.10 Average size of all messages sent for specific protocols during the evaluation of a query. Note that request and response messages are listed separately, and message sizes are measured on top of the request-response protocol layer. Error bars are shown for a 99% confidence interval.

ture. Consequently, we should review this particular message size once scaling the number of DPs.

Takeaway. Our DHDN fulfills all requirements set by our performance objective. We showed that the system scales efficiently using limited resources: computational power and available bandwidth. Further, we can bisect the query duration by exploiting multi-core architectures and CSPs.

7.3 Scalability

In the previous section, we have shown that our system scales efficiently in the number of data records and can utilize multi-core architectures and cloud service providers (CSPs) to reduce query runtime by scaling the test servers (TSs). Thus, we have already seen that our network meets the scalability requirements concerning data owners (DOs) and TSs. However, to assess whether we meet the requirements of our scalability objective, we have to investigate further the impact of increasing the number of data providers (DPs) and collective authority (CA) members. Finally, we conclude this section with a use case scenario that utilizes the distributed architecture of our network to establish a distributed health data network (DHDN) among all

German university hospitals, demonstrating that our system operates efficiently on a larger scale.

7.3.1 Horizontal Scalability of Collective Authority Members

The CA is the entity with the least active participation in the network and query evaluation. It is not part of any protocol in the evaluation stage. Thus, we can neglect the overall query time but have a detailed view of the metrics of query construction and result reception.

Bandwidth Usage. Except for the query publication, all message sizes are independent of the number of CA members. Here we require an additional signature and public key for the signature chain (cf. Section 5.3.3.1). In evaluating our test case, we verify that the message size increases exactly one hundred twenty-eight bytes of each CA member. This increment corresponds to the additional signature and ephemeral public key ($128 = 2 \cdot |\mathbb{G}_1| + |\mathbb{Z}|$). Thus, we can increase the number of CA members with reasonable overhead for each query, increasing the bulletin board load. On the other hand, the data user (DU) has to engage in the **MAKE QUERY** and **RECEIVE RESULT** protocols with every CA member, further increasing the bandwidth usage of participating entities, but not increasing the overall load on the peer-to-peer (P2P) network. We already covered the individual message sizes in Section 7.2.4 with less than twenty kilobytes for the query construction, which must be performed consecutively, thus not increasing the bandwidth usage per second for the DU. In contrast, the result received can be processed in parallel but has less than 100 bytes in total, which we consider negligible bandwidth usage.

Impact on Query Duration. Notably, increasing the number of CA members impacts the query construction, result receipt, and query verification only. For the latter two, the impact on overall duration is negligible. To be more precise, the overhead of validating an additional element of the signature chain is 0.73 milliseconds, and we could not measure any increase in the duration of the result received, as those are requested in parallel, and results are already prepared by the CA members. Thus, we only assess the query construction, which we measured in our artificial setup of a single geolocation, but with a scaling number of collective authority members. Figure 7.11 shows the result of our measurements.

Outstanding is the increased duration for query construction due to the consecutive nature of the protocol. Thus, the DU has to perform every protocol in a sequence leading to an observed linear scaling. However, this also shows that the individual time for each request does not increase, affirming the independence to the total number of CA members. Nevertheless, the query construction time is relatively high compared to the overall query lifetime from other tests. We did not exploit multi-core architectures in implementing the CA and allocated only one thread for each member in our test scenario. Consequently, future work may parallelize the trapdoor shuffle and discrete logarithm proof to improve the overall performance of query construction.

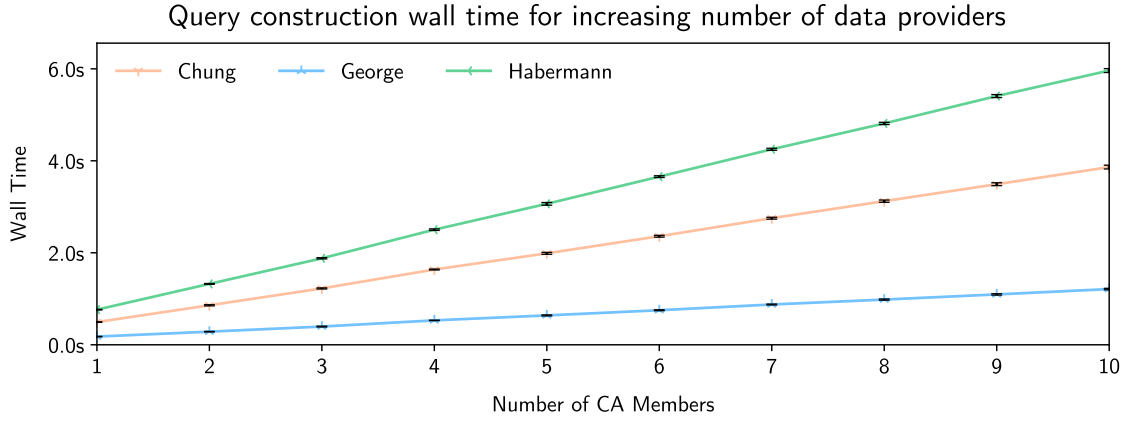


Figure 7.11 Query construction duration in correlation to the number of CA members, with 99% confidence interval. The linear increase reflects the sequential evaluation of the protocol with every member of the CA.

7.3.2 Horizontal Scalability of Data-Providers

In our previous section, we evaluated the impact on scaling TSs, but this gives us only insights into the scalability to process data of a single organization. However, our goal is an environment with multiple organizations collaborating in a distributed network. Consequently, this section will evaluate the performance for increasing DPs. Therefore, we built upon our previous test setup (cf. Section 7.2) but reduced the thread allocation for each organization-specific entity. Here we chose two threads for every TS aligning with our previous scalability test of TSs and six threads for the DP and DO each. Accordingly, we distributed the entire dataset among all instances using. This way, we ensure that the results of all evaluations stay comparable to each other.

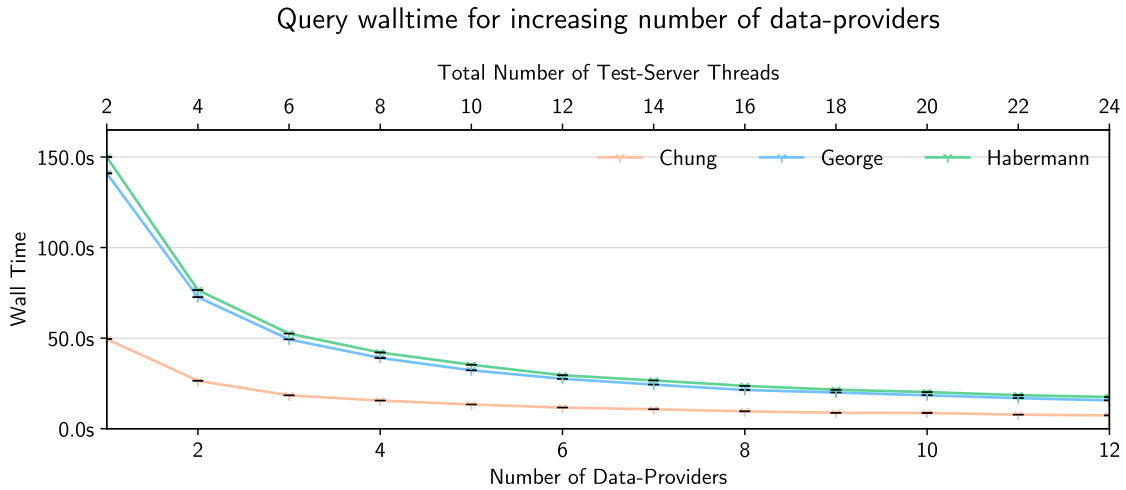


Figure 7.12 Query duration in correlation with increasing DPs with equally distributed data shards of the full dataset. Error bars are shown for a 99% confidence interval.

Figure 7.12 shows the result of our scaling DP test. Compared to Section 7.2.3, we achieve slightly faster evaluation than exploiting the scalability of TSs. This improvement confirms our hypothesis that the scalability of TS is currently bottlenecked by

our DO simulation (cf. Section 7.1.1.3). Nevertheless, achieving comparable results yields evidence that our system can scale with a high number of DPs. In Section 7.3.4, we will put this to the test in a large real-world use case scenario.

Besides the impact on query duration, we also analyzed the impact on bandwidth usage. Here we observed a constant increase of thirty-two bytes per DP for the size of the **PUBLISH RESULT** message. This increment corresponds to the extra scalar required for every ring member chosen for the Back Linkable Spontaneous Anonymous Group (bLSAG) signature. This approach is sufficient for a smaller number of DPs but may not scale indefinitely, as this induces a quadratic increase of total bandwidth usage because every DP must include a scalar for every member of the ring. As mentioned in Section 5.3.4.4, we recommend randomly selecting a fixed number of ring candidates among all queried DPs if the ring size hits a certain threshold. In other words, choosing a maximum ring size for the bLSAG signature would result in a constant overhead, independent of the number of queried DPs. However, it should be noted that this comes with privacy trade-offs for the DP identity [WLSL18].

7.3.3 Impact of Unavailability and Infrastructure Changes

Another aspect of scalability is the dynamic change of its infrastructure and resilience to the unavailability of individual entities. We shortly address this aspect for the entities of the DHDN.

Test Server. TS are only part of the organization-specific network. Thus, any change does not directly impact the DHDN. Though if a TS leaves the system, its data cannot be processed by other TSs because the fragments are bound to that particular server, requiring a reimport of patient data (cf. Section 5.3.2). This case is similar to adding a new TS, where existing data cannot be moved from another TSs but must be reimported under the new TS public key. Consequently, changes on the TS infrastructure should be scheduled with the data shuffle and import routine, as covered in Section 5.3.2.3. A TS outage does not significantly impact the DHDN. Affected DOs would still publish their consent to the bulletin board. However, no TS would process their consent, leaving DOs unable to participate in queries during the outage period.

Data Provider. In the case of dynamically changing DPs, there is no problem with adding or removing new DPs as long as this change is publicly visible for all other members of the DHDN. This publicity requirement is because ring signatures include all DPs of a research network (RN). Thus, their identities must be publicly available to construct the ring of public keys. More problematic is a DP blackout. Here the CA will never receive all results it is waiting for. Naturally, we could configure a timeout and provide the result without submitting that DP, but this may undermine the policies of other DPs queried. For example, an attacker could include unavailable DPs to secretly narrow their search on a single DP (cf. Section 5.3.3.1).

Collective Authority Members. Changing the CA is the most impactful infrastructure change, as any change of the collective public key PK_{CA} invalidates all stealth bits. In this case, every DP will have to partially reimport all data unless they keep track of DOs identities that are artificially generated. Otherwise, they must reimport everything. Another problem occurs if one CA member is unavailable, leading to the

unusability of the DHDN as no new queries can be created or results received. One way to solve these issues is to use a (k, n) -threshold scheme [Sha79] for the CA key. Hence, CA operations can still be performed as long as at least k out of n members are available. Furthermore, this would also be sufficient to delay changes on the cumulative key till maintainability schedules of the entire network.

7.3.4 A Use Case Scenario on German University Hospitals

So far, we built artificial test scenarios that scale all entities connected to a single switch of our backbone. Although this facilitates comparability between our test cases, it does not reflect real-world use cases. In this section, we meet this obligation by presenting a practical application of our system.



Figure 7.13 Visualization of the university hospitals distribution connected to the ‘Deutsches Forschungsnetz’ (DFN) network. Each DP point stands representative for all organization-specific entities (DP, TS and DO).

Therefore, we gathered the data from all university hospitals in Germany and picked the top twenty with the most yearly cases. For the backbone, we chose the ‘The Internet Topology Zoo’ data of the ‘Deutsches Forschungsnetz’ (DFN) network, which is a network designed for research purposes in Germany. We then built a DHDN of these hospitals as DPs and distributed our dataset according to their number of reported cases. Here we set up a single TS and DO for each DP and split all available threads among those according to their share of records managed. A list of all chosen university hospitals and their thread allocation can be found in Section A.4. Additionally, we chose the top six hospitals to be part of the CA.

Query	Records	DFN	Records	Single
Chung	10,860,749	14.54s ± 0.12	6,471,165	9.15s ± 0.22
George		32.09s ± 0.20		14.12s ± 0.39
Habermann		37.56s ± 0.28		19.51s ± 0.46

Table 7.4 Results of the German university hospitals scenario in comparison to our artificial testbed with a single DP. The test ran without a collection of runtime metrics, and deviations are given with 99% confidence.

In total, we simulated sixty-six entities managing 10,860,749 records. The results of our measurements are displayed in Table 7.4. Although the timings look reasonable, we must consider that it does not reflect the performance we would expect from our previous analysis. Therefore, we looked into the wall times of the individual DPs, which vary drastically. For example, in case of the Habermann query DPs finished within 1.18 up to 34.33 seconds. On average, every DP required 12.21 seconds till publishing the result. This imbalance shows an inadequate distribution for the number of threads. Consequently, this use case needs to fit the assessment of the impact of particular changes. However, it suffices to show that we can scale our network to real-world use cases and achieve practical results, even under the limited availability of computational resources.

Takeaway. Our system scales at most linearly with every increasing entity of our network and, most important sublinear in the number of data records due to the usage of bucketization. In addition, we scaled the system to a real-world use case with more than ten million records and still got results sufficient for daily use. As a result, we conclude that our systems meet the requirements of our scalability objective, and we continue to assess our remaining non-functional goals, privacy, and security.

7.4 Privacy and Security

In the last two sections, we have shown that our system conforms to our non-functional performance and scalability goals. It is left to show whether our system also fulfills the requirements set by our privacy and security objective. This assessment has progressively been stated in Chapter 5, where we discussed the most important design decision and rationales to conform to our privacy and security goals. Table 7.5 shortly summarizes the key methods used to secure integrity, confidentiality, authority, and anonymity.

Although this only partially covers all highlighted cases in Section 5.3, it is sufficient to argue that every protocol is security protected and preserves the privacy of the data user (DU), data owner (DO) and data provider (DP). However, to meet our privacy and security goals, we must also show that we deployed a non-trivial access control scheme for dynamic consent and implemented state-of-the-art techniques to protect data recovery and patient-identification.

Search Control. We utilized the consent witnesses to enable the DO for search control. From a cryptographic standpoint, no one except someone with access to that witness can consent to a request, and no one without access to the consent can

Protocol	Integrity	Confidentiality	Authority	Anonymity	TLS
Make Query	–	–	Part of RN	PK _{DU}	✓
Publish Query	EdDSA	IND-CPA	CA Sig. Chain	PK _{DU}	✗
Forward Query	–	–	sk _{DP}	N/A	✓
Discover Queries	EdDSA	–	Stealth Address	Stealth Address	✓
Acquire Queries	EdDSA	–	Stealth Address	Stealth Address	✓
Publish Consent	EdDSA	AES	Stealth Address	Stealth Address	✗
Forward Result	–	ElGamal _{PK_{CA}}	sk _{TS}	N/A	✓
Publish Result	bLSAG	ElGamal _{PK_{CA}}	bLSAG	bLSAG	✗
Receive Result	–	Key Switch	sk _{DU}	PK _{DU}	✓

Table 7.5 Overview of methods used to achieve privacy and security goals in every protocol. For secure Transport Layer Security (TLS) connections without additional protection for a specific goal, the corresponding cell is left blank.

match encrypted data. We further utilized the collective authority (CA) to ensure that a query uses a one-time receiver, ensuring a strong binding between the consent and query, such that no new search request can be issued for the same query. In addition, we used key-derived identities for the DO that are unlinkable to a specific patient. As a result, the patient has access to an anonymous identity that has full control over the processing of their data using the consent witnesses, fulfilling our requirements of an access control scheme respecting patients' privacy.

Data Owner Privacy. Additionally, we deployed multiple techniques to protect the patient's privacy further. In Section cf. Section 5.1.1, we covered the integration of artificial data to achieve k -anonymity and further protect against statistical analysis by hiding the result count (cf. Section 5.3.2.1). Finally, we showed how to deploy data shuffling (cf. Section 5.3.2.3) to hide statistics on individual records further. In summary, we deployed state-of-the-art techniques covered in our related work to protect patients' privacy further, fulfilling privacy goal obligations.

With the last two observations, we conclude that we meet all goals set by our privacy and security goals, finalizing the non-functional analysis of our system design. We have shown that all requirements for performance, scalability, privacy, and security are met, fulfilling conformance to all non-functional goals set by our problem statement (cf. Section 4). For the remainder of this chapter, we will evaluate the trade-offs taken to achieve these goals.

7.5 Trade-Offs on Performance and Privacy

Until this section, we evaluated the conformance of the system design and implementation to our non-functional objectives. However, these objectives are partially orthogonal, requiring decisional trade-offs balancing usability, performance, and privacy against each other. For this section, we assess the impact of design decisions on our system performance. We start by observing the overhead induced by our code instrumentation.

7.5.1 Tracing Overhead

In Section 7.1.3.1, we mentioned that the tracing code instrumentation has an observable overhead to the running time. We add a compiler flag that turns off all code instrumentation to ensure that this measurement does not substantially tamper our all times. We then repeated our use case scenario (cf. Section 7.3.4) to measure the induced overhead of our measurement. The results in Table 7.6 show that our comprehensive measurements significantly alter our results but still give a reasonable estimation of the runtime magnitude. Thus, our analysis is based on an upper bound of the actual CPU time, sufficient to test the feasibility of our system, but future work may improve on the data collection to give more accurate results.

Query	With Tracing		Without Tracing		Overhead	
Chung	15.58s	± 0.11	14.25s	± 0.76	9.36%	± 0.79
George	32.59s	± 0.13	31.45s	± 1.67	3.64%	± 0.42
Habermann	42.05s	± 0.30	37.56s	± 0.28	11.97%	± 0.79

Table 7.6 Comparison of query wall times in the use case scenario of german university hospitals. Data was measured with and without code instrumentation to evaluate the overhead of code instrumentation. Deviations are given for a 99% confidence interval.

7.5.2 Feasibility of Larger Queries

With the introduction of ICD-10, encodings of diagnosis and procedures became more granular. We observed this in Section 7.1.2 converting queries from ICD-9 to ICD-10, showing that our conversion suffices to reach a comparable number of candidates but needs to be completed. Therefore, we presented extended versions of our query to solve the problems with the increasing granularity of ICD codes by including additional codes of new case distinctions. In this section, we review these extended query versions to grasp the impact on our system and answer whether our system can still perform larger queries.

Table 7.7 shows metrics of query structure for the extended version compared to those created using the stated conversion method. Here the extended version has very different characteristics regarding the increase of candidates and results. Especially the George query stands out with an 84% increase of candidates but only 5% more matches. On the other hand, the Habermann query adds more codes but yields comparably fewer extra candidates with a massive increase in results. Next, without tracing, we evaluated the extended queries in our performance evaluation setup (cf. Section 7.2.1) and compared the durations against the converted counterparts.

We observe no clear correlation between the increase in granularity and the duration of a query. Especially the Habermann query shows that a large increase of codes and candidates can match much more data without raising the query time to unacceptable heights. The granularity in ICD codes does not necessarily blow up our query times drastically, and our system is still feasible even for queries with a comprehensive list of codes to match. Nevertheless, we leave it up for future work to assess the

Query		Procedure Codes	Diagnosis Codes	Canidates	Result
Chung	Converted	24	18	3,317	1,404
	Extended	108	22	4,787	1,789
	Increase	350%	22%	44%	27%
George	Converted	6	4	6,529	552
	Extended	47	5	12,011	577
	Increase	683%	25%	84%	5%
Habermann	Converted	57	10	7,528	41
	Extended	267	63	10,918	4,632
	Increase	368%	530%	45%	11198%

Table 7.7 Comparison of converted and extended versions of our test queries. A record matches a query if one of their procedures and one of their diagnosis matches any of the provided procedure and diagnosis codes, respectively.

Query	Converted		Extended		Increase	
Chung	9.15s	± 0.22	13.48s	± 0.06	47.35%	± 0.66
George	14.12s	± 0.39	30.59s	± 0.08	116.67%	± 0.58
Habermann	19.51s	± 0.46	34.23s	± 0.18	75.49%	± 0.91

Table 7.8 Comparison of query durations for the converted ICD-10 queries to their extended counterparts. Deviations are given for a 99% confidence interval.

correlation in more detail and come up with solutions to utilize the structured encoding of ICD codes in a more sophisticated way to enhance query expressiveness and performance (cf. Section 8.1.2).

7.5.3 Artificial Data

Although we stated the methodology of artificial data using stealth bits (cf. Section 5.1.1) and how we can use them to increase anonymity, we did not explicitly evaluate its impact on performance. The reason for that is that artificial data behaves exactly like normal data. Thus, assessing its impact equals increasing the number of records covered in Section 7.2.1. Additionally, we designed our data owner (DO) implementation to act as a simulator for artificial data. Thus, the impact of artificial data comes down to the anonymity and diversity one would like to achieve. However, in the context of this thesis, we do not cover techniques to generate additional data to ensure these properties but rather develop the tools to deploy data that has been generated. Nevertheless, this is an important topic that needs to be addressed and is part of our outlook (cf. Section 8.1.2).

7.6 Discussion

We presented a distributed health data network (DHDN) that scales linearly with the number of stored records and can utilize distributed computational power without significant overhead. Our results confirm that the most significant task is the **Test** operation and that overall performance directly corresponds to the computational cost of it. This correlation is reaffirmed by the close relation of overall performance to the number of available test server (TS) threads. Additionally, in our implementation of a data hub according to Yuan et al. [YMM⁺17], we can observe the same increase in performance by replacing the public key encryption with keyword search (PEKS) scheme. This congruence shows a close relation between both designs, as our data providers are similar to their data hub. Moreover, the findings of Yuan et al. [YMM⁺17] align with our observations on vertical and horizontal scalability. As a result, we are confident that the main performance increase of our system compared to theirs is caused by adopting a different PEKS scheme and, more particularly, a different elliptic curve implementation. However, it was unexpected that the overall performance of our system aligns closely to the performance of a single data hub using the same PEKS scheme, reassuring that other protocols do not contribute significantly to the overall query time.

Further comparing our results to related work, our implementation shows an increased ratio of candidates matching a query (cf. Section 7.1.2). In other words, we get higher results for the same number of candidates. This observation was expected due to our adaption in matching records based on data groups rather than fields, thus checking multiple fields for a specific diagnosis or procedure. However, due to the usage of a different dataset, we cannot assess if we match more records or if some other variation in the dataset changed the ratio. Therefore, in future research, one may stick to the same dataset regardless if it is outdated to promote comparability or develop an artificial test suite to compare the performance of DHDN regardless of the dataset used for testing real-world applications.

Additionally, we emphasize that our test deployment result should be interpreted cautiously because of the data owner (DO) simulation that was necessary to prevent exhaustion of the host machines' network virtualization but does not reflect a real-world deployment. Normally, all DO would have to request updates on queries regardless of whether they are part of the candidates. Although we have shown that the computation and bandwidth usage to filter candidates is minor compared to all other operations, those may sum up to a considerable amount. Therefore, we suggest to setup a larger test environment with enough computation power to simulate millions of DOs simultaneously and assess the impact of high workloads on the data provider (DP). In addition, a review of the results may give a good basis for evaluating the necessity of stealth address usage in the **ACQUIRE QUERIES** protocol. If a single DP can process all incoming DO requests, it would not be necessary to distribute stealth addresses for load balancing, thus removing the necessity to compute them.

After reviewing the time a DO spends in computing their consent and the negligible size of **PUBLISH CONSENT** messages, we would rather suggest swapping the curve implementations in our PEKS formulation. This recommendation is mainly due to the unexpectedly high amount of **Test** operations for every record compared to the

Index operation. Here we underestimated the scaling of trapdoors and data fields, leading to four times as much **Test** than **Index** operations. By design, DO devices are the most scaling entity in the system, and redistributing a small amount of workload from TSs to all DO devices may lead to a significant performance boost for TSs and thus reduce overall query time. However, assessing this change in our test environment may not yield accurate results, as we are bound to simulate all DO on the same machine, and this change would benefit from available resources across multiple devices. This limitation reaffirms the need for a larger testbed where DOs are simulated on additional machines.

7.6.1 Over the Edge of the Threat Model

In this last part of our discussion, we briefly overview one important practical problem beyond our threat model. However, due to its significant impact on data user (DU) privacy, we shortly introduce the problem and give directions for future work or possible deployments.

Everyone with access to the query public key PK_Q and the encrypted trapdoor values can brute-force the queried values by generating new indices and consent to them. One entity capable of performing this offline keyword-guessing attack is the DP, and that's why they must be trusted in our system. In addition, the same could be achieved by a colluding TS and DO entity, which is likely in practice as the system benefits from a lot of contributors, which may lead to low barriers for an attacker to be part of the DOs. Thus, an attacker with access to a compromised TS and DO may brute force the entire query and obtain knowledge on matching fields for all DO that consent to that query.

We recommend implementing a third round after the **ACQUIE QUERIES** protocol to prevent this kind of attack. Utilizing that round, the DP can send all DOs a blinded version of the consent key PK_Q and use the additional round to unblind their consent without ever revealing the PK_Q itself. Notably, this will drastically increase the workload on the DP and may contradict our observation that swapping the group definitions of our PEKS definition is beneficial in performance. However, in the context of this thesis, the DP is trusted, and entities are not colluding. Thus, we will leave the implementation and asses of such an extension for future work.

8

Conclusion

Given the increasing deployment of electronic health record (EHR) [BHCW10] and evolving use for clinical research [CSK⁺13, KRN⁺19] sharing health data becomes increasingly valuable to improve public health [WB11]. On the other hand, cyber-attacks on medical facilities are increasing [HG16, oHfCR21], opposing patients to higher risks of leakage of their protected health information (PHI). In this thesis, we presented the design of a new distributed health data network that adopts advances in the research of searchable encryption to build a system with privacy-preserving data discovery and dynamic consent to promote sharing of health data.

Current systems lower the risk of data leakage by encrypting data and sharing it with a central entity that processes encrypted data using privacy-preserving technologies. However, these approaches still depend on sharing data in encrypted form, limiting their scalability due to governmental policies and regulatory frameworks. In addition, these systems have no capabilities of dynamic consent, lowering patients' willingness to contribute their data and the scope of applications their data can be used for due to the requirements of informed consent for evolving use cases.

To solve these limitations, we adapted two systems with orthogonal features regarding performance and privacy. We adopted encrypted database technologies and privacy-preserving result aggregation to create a system that runs sublinear in the number of records and retains the confidentiality of results in the any-trust model. In addition, we combined the distributed nature of multiple data providers with the performance benefits of a central data hub. Our system features independent organizational networks that are not required to share patient data. With the evaluation of queries on more than ten million records distributed over twenty organizations within thirty seconds, we conclude it is possible to combine the benefits of performance, scalability and privacy of the system from Yuan et al. [YMM⁺17] and Froelicher et al. [FES⁺17].

In addition, we adopted advances in the research of searchable encryption to embed the capabilities of dynamic consent in our system, cryptographically binding the consent to the test of a trapdoor. As a result, we returned control of data processing to the owner in a manner that respects and enforces their consent and did this in a

privacy-preserving way that respects the opaque choice of the owner at every step. Concluding, we have shown that asymmetric searchable encryption can be used to implement dynamic consent.

We tested a prototype implementation of our system, showing that our network scales efficiently with the number of contributing organizations. Moreover, our system utilizes the resources of additional data providers to achieve faster query times with little to no overhead for result aggregation and query processing, respectively. Therefore, it is reasonable to deploy our system with hundreds of organizations and achieve similar, if not faster, results. In addition, we showed that organizations can scale efficiently by utilizing cloud service providers and multi-core architectures offering various options to scale economically regarding organizations interest. Further, our initial performance evaluation results also show high competitiveness to the results of related work. Compared to our related work, we could evaluate queries on real-world datasets at a fraction of the time using larger datasets and comparable hardware. Overall, we consider our results as practical feasible, benefiting from distributed environments, and we conclude that our network scales efficiently and is performant enough to be deployed for practical use in real-world scenarios.

8.1 Future Work

During the development of our system, we encountered several aspects that deserve their attention but need to be included, being outside the scope of the context of this thesis. Here, we highlight some of these directions that we found of particular interest and may be adopted by future work.

8.1.1 A Data Owner Interface

One of the next consequential steps to assess the system in a more practical series of tests is the development of a data owner (DO) interface. Although our developed application is sufficient to simulate artificial data and maybe bridge some existing data with given broad consent to an actively running system, it does not reflect the intended usage, nor is our system functional complete without the implementation of a DO interface. In addition, developing such an app could benefit other research areas, especially in dynamic consent [CH12, KBK⁺16] and biomedical research [BLTK⁺17, CDD⁺17].

8.1.2 Structural Adoption of ICD Codes

In Section 7.1.2, we highlighted some of the structural properties of ICD codes. Further in Section 7.5.2, we have seen how the structural changes from ICD-9 to ICD-10 influence the performance of our system. Therefore, it is important to investigate the structure of ICD codes and associated characteristics to assess the feasibility of splitting single ICD codes into tuples of indices. Such adaptation would enable researchers to query for a wider range of values without specifying all case distinctions resulting in fewer trapdoors and, thus, faster query times. However, we also note

that this may come with a trade-off in privacy, especially in the case of bucketization usage. Thus, evaluating the diversity and anonymity requirements for bucketization on ICD codes is of particular interest. The deployed prefix strategy reflects the hierarchical structure of ICD codes, benefiting the classification of related codes in a single bucket. Nevertheless, one may find an optimized version that balances the number of candidates in buckets and the number of required artificial data records for certain diversity and anonymity goals more efficiently.

8.1.3 A Custom-Made PEKS Scheme

For the context of this thesis, we searched for a public key encryption with keyword search (PEKS) scheme that is eligible for adaptation in our application. However, this scheme is not a perfect match. Notably, the consent witness is not bound to a specific secret of the DO, and everyone with access to that witness can create valid consent. To prevent unauthorized consent we use stealth addresses but strictly speaking, this does not invalidate the consent. In addition, the definition of consent is strongly bound to the definition of an index. However, future work may improve the index definition to support more complex data types, like genomics. Decoupling the consent process from the index may result in a more modular approach that is more applicable for changing use cases of different data types. Finally, we could improve performance, especially using relative slow exponentiations in $\mathbb{G}_2$. Here an optimized version of a PEKS scheme may find a more suitable definition to balance the cost of operations according to their occurrence.

8.1.4 Enhanced Query Expressiveness

In addition to supporting new types of indices, we can also improve query expressiveness by implementing new methods of aggregation. In the following, we highlight three arguably valuable extensions for the network.

Threshold Policies. Here we want to present one possible extension that would yield stronger protection for the privacy of individuals in the network. In particular, we require the possibility to prove that a result is higher than a constant threshold, strengthening the network resilience against queries spotting individuals in the network. The challenge is collectively using the collective authority (CA) without revealing the result to any members.

Result Aggregation. In this thesis, we only support obtaining the number (`COUNT(*)`) of matching data. However, Froelicher et al. [FES⁺17] showed that it is possible to implement collective aggregation for sum and average too. Furthermore, they did so in a privacy-preserving manner using distributed result obfuscation. Therefore, a consequential improvement on the query expressiveness is to adapt their methodology into our system. We also emphasize that their work includes a method for deduplication that is of moderate interest for practical use [CAWW19]. Although this protocol induces a very high workload on the CA, finding a method to remove duplicates from the result aggregation may be worth it.

Hierarchical Aggregation. The data user (DU) and data provider (DP) are similar in aggregating results. Utilizing this similarity, we could implement a new entity type

that acts as a bridge between two networks. In other words, a DP would take a query, authorize it on behalf of another CA and publish it to the corresponding network. Once the result is available, that result is published back to the former network as the result of that DP. This separation may be of interest for the organization intern architecture, as this would benefit the security of an organization-specific network with the advantages of an entire distributed health data network (DHDN). Especially in the case of larger organizations with multiple locations in different countries, this would enable international collaboration within the organization and promote the contribution of their results to other networks.

Bibliography

- [A⁺20] Kurt M Alonso et al. Zero to monero. *Zero to monero*, 2020.
- [ABU18] Mohiuddin Ahmed and Abu S. S. M. Barkat Ullah. False data injection attacks in healthcare. In Yee Ling Boo, David Stirling, Lianhua Chi, Lin Liu, Kok-Leong Ong, and Graham Williams, editors, *Data Mining*, pages 192–202, Singapore, 2018. Springer Singapore.
- [ACG⁺16] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS ’16, page 308–318, New York, NY, USA, 2016. Association for Computing Machinery.
- [ADW⁺18] April Moreno Arellano, Wenrui Dai, Shuang Wang, Xiaoqian Jiang, and Lucila Ohno-Machado. Privacy policy and technology in biomedical data science. *Annu Rev Biomed Data Sci*, 1:115–129, July 2018.
- [Ahl16] Rudolf Ahlswede. *Founding Cryptography on Oblivious Transfer*, pages 337–344. Springer International Publishing, Cham, 2016.
- [AS00] Rakesh Agrawal and Ramakrishnan Srikant. Privacy-preserving data mining. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD ’00, page 439–450, New York, NY, USA, 2000. Association for Computing Machinery.
- [BBJ⁺08] Daniel J. Bernstein, Peter Birkner, Marc Joye, Tanja Lange, and Christiane Peters. Twisted edwards curves. In Serge Vaudenay, editor, *Progress in Cryptology – AFRICACRYPT 2008*, pages 389–405, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.
- [BBO07] Mihir Bellare, Alexandra Boldyreva, and Adam O’Neill. Deterministic and efficiently searchable encryption. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007*, pages 535–552, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.
- [BBS98] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In Kaisa Nyberg, editor, *Advances in Cryptology — EUROCRYPT’98*, pages 127–144, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

- [BC21] Utsav Banerjee and Anantha P Chandrakasan. A low-power bls12-381 pairing cryptoprocessor for internet-of-things security applications. *IEEE Solid-State Circuits Letters*, 4:190–193, 2021.
- [BCM⁺15] Paulo S. L. M. Barreto, Craig Costello, Rafael Misoczki, Michael Naehrig, Geovandro C. C. F. Pereira, and Gustavo Zanon. Subgroup security in pairing-based cryptography. Cryptology ePrint Archive, Paper 2015/247, 2015. <https://eprint.iacr.org/2015/247>.
- [BCOP04] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *International conference on the theory and applications of cryptographic techniques*, pages 506–522. Springer, 2004.
- [BDP⁺18] Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche, Ronny Van Keer, and Benoît Viguier. Kangarootwelve: Fast hashing based on KECCAK- p . In Bart Preneel and Frederik Vercauteren, editors, *Applied Cryptography and Network Security*, pages 400–418, Cham, 2018. Springer International Publishing.
- [Ben14] Juan Benet. IPFS - content addressed, versioned, P2P file system. *CoRR*, abs/1407.3561, 2014.
- [BF01] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In Joe Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, pages 213–229, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.
- [BG84] Manuel Blum and Shafi Goldwasser. An efficient probabilistic public-key encryption scheme which hides all partial information. In *Workshop on the theory and application of cryptographic techniques*, pages 289–299. Springer, 1984.
- [BGH⁺14] Pankaj Berde, Matteo Gerola, Jonathan Hart, Yuta Higuchi, Masayoshi Kobayashi, Toshio Koide, Bob Lantz, Brian O’Connor, Pavlin Radoslavov, William Snow, and Guru Parulkar. Onos: Towards an open, distributed sdn os. In *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, HotSDN ’14, page 1–6, New York, NY, USA, 2014. Association for Computing Machinery.
- [BHCW10] Taxiarchis Botsis, Gunnar Hartvigsen, Fei Chen, and Chunhua Weng. Secondary use of ehr: data quality issues and informatics opportunities. *Summit on Translational Bioinformatics*, 2010:1, 2010.
- [BHJP14] Christoph Bösch, Pieter Hartel, Willem Jonker, and Andreas Peter. A survey of provably secure searchable encryption. *ACM Comput. Surv.*, 47(2), aug 2014.
- [BLLS11] Mrinmoy Barua, Xiaohui Liang, Rongxing Lu, and Xuemin Shen. Espac: Enabling security and patient-centric access control for ehealth

- in cloud computing. *International Journal of Security and Networks*, 6(2-3):67–76, 2011.
- [BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *International conference on the theory and application of cryptology and information security*, pages 514–532. Springer, 2001.
- [BLS03] Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. Constructing elliptic curves with prescribed embedding degrees. In Stelvio Cimato, Giuseppe Persiano, and Clemente Galdi, editors, *Security in Communication Networks*, pages 257–267, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.
- [BLTK⁺17] Isabelle Budin-Ljøsne, Harriet J. A. Teare, Jane Kaye, Stephan Beck, Heidi Beate Bentzen, Luciana Caenazzo, Clive Collett, Flavio D’Abramo, Heike Felzmann, Teresa Finlay, Muhammad Kassim Javaid, Erica Jones, Višnja Katić, Amy Simpson, and Deborah Mascalonzi. Dynamic consent: a potential solution to some of the challenges of modern biomedical research. *BMC Medical Ethics*, 18(1):4, Jan 2017.
- [BM10] Kathleen Benitez and Bradley Malin. Evaluating re-identification risks with respect to the HIPAA privacy rule. *Journal of the American Medical Informatics Association*, 17(2):169–177, 03 2010.
- [Bow19] Sean Bowe. Faster subgroup checks for bls12-381. Cryptology ePrint Archive, Paper 2019/814, 2019. <https://eprint.iacr.org/2019/814>.
- [BSNS08] Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. Public key encryption with keyword search revisited. In Osvaldo Gervasi, Beniamino Murgante, Antonio Laganà, David Taniar, Youngsong Mun, and Marina L. Gavrilova, editors, *Computational Science and Its Applications – ICCSA 2008*, pages 1249–1259, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.
- [BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In Salil P. Vadhan, editor, *Theory of Cryptography*, pages 535–554, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.
- [BZ17] Abdelali Boussadi and Eric Zapletal. A fast healthcare interoperability resources (fhir) layer implemented over i2b2. *BMC Medical Informatics and Decision Making*, 17(1):120, Aug 2017.
- [C⁺08] Lily Chen et al. *Recommendation for key derivation using pseudo-random functions*. US Department of Commerce, National Institute of Standards and Technology, 2008.
- [Car13] Donna J. Cartwright. Icd-9-cm to icd-10-cm codes: What? why? how? *Advances in Wound Care*, 2(10):588–592, 2013. PMID: 24761333.

- [CAWW19] Shekha Chenthara, Khandakar Ahmed, Hua Wang, and Frank Whitaker. Security and privacy-preserving challenges of e-health solutions in cloud computing. *IEEE Access*, 7:74361–74382, 2019.
- [CBC⁺17] Martin R Cowie, Juuso I Blomster, Lesley H Curtis, Sylvie Duclaux, Ian Ford, Fleur Fritz, Samantha Goldman, Salim Janmohamed, Jörg Kreuzer, Mark Leenay, et al. Electronic health records to facilitate clinical research. *Clinical Research in Cardiology*, 106(1):1–9, 2017.
- [CBP14] Lesley H Curtis, Jeffrey Brown, and Richard Platt. Four health data networks illustrate the potential for a shared national multipurpose big-data network. *Health affairs*, 33(7):1178–1186, 2014.
- [CDD⁺17] Daniele Carrieri, Sandi Dheensa, Shane Doheny, Angus J. Clarke, Peter D. Turnpenny, Anneke M. Lucassen, and Susan E. Kelly. Recontacting in clinical practice: an investigation of the views of healthcare professionals and clinical scientists in the united kingdom. *European Journal of Human Genetics*, 25(3):275–279, Mar 2017.
- [CGPR15] David Cash, Paul Grubbs, Jason Perry, and Thomas Ristenpart. Leakage-abuse attacks against searchable encryption. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS ’15, page 668–679, New York, NY, USA, 2015. Association for Computing Machinery.
- [CH12] Kelly Caine and Rima Hanania. Patients want granular privacy control over health information in electronic medical records. *Journal of the American Medical Informatics Association*, 20(1):7–15, 11 2012.
- [CK13] Stuart Cheshire and Marc Krochmal. Multicast DNS. RFC 6762, February 2013.
- [CKRS09] Jan Camenisch, Markulf Kohlweiss, Alfredo Rial, and Caroline Sheedy. Blind and anonymous identity-based encryption and authorised private searches on public key encrypted data. In Stanisław Jarecki and Gene Tsudik, editors, *Public Key Cryptography – PKC 2009*, pages 196–214, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.
- [CM17] Nicolas T Courtois and Rebekah Mercer. Stealth address and key management techniques in blockchain systems. In *ICISSP 2017- Proceedings of the 3rd International Conference on Information Systems Security and Privacy*, pages 559–566, 2017.
- [CRMC15] Thomas K. Chung, Eben L. Rosenthal, J. Scott Magnuson, and William R. Carroll. Transoral robotic surgery for oropharyngeal and tongue cancer in the united states. *The Laryngoscope*, 125(1):140–145, 2015.
- [CS97] Jan Camenisch and Markus Stadler. Proof systems for general statements about discrete logarithms. *Technical Report/ETH Zurich, Department of Computer Science*, 260, 1997.

- [CSK⁺13] P. Coorevits, M. Sundgren, G. O. Klein, A. Bahr, B. Claerhout, C. Daniel, M. Dugas, D. Dupont, A. Schmidt, P. Singleton, G. De Moor, and D. Kalra. Electronic health records: new opportunities for clinical research. *Journal of Internal Medicine*, 274(6):547–560, 2013.
- [CWJ⁺16] Feng Chen, Shuang Wang, Xiaoqian Jiang, Sijie Ding, Yao Lu, Jihoon Kim, S Cenk Sahinalp, Chisato Shimizu, Jane C Burns, Victoria J Wright, Eileen Png, Martin L Hibberd, David D Lloyd, Hai Yang, Amalio Telenti, Cinnamon S Bloss, Dov Fox, Kristin Lauter, and Lucila Ohno-Machado. PRINCESS: Privacy-protecting Rare disease International Network Collaboration via Encryption through Software guard extensionS. *Bioinformatics*, 33(6):871–878, 12 2016.
- [CWL⁺14] Ning Cao, Cong Wang, Ming Li, Kui Ren, and Wenjing Lou. Privacy-preserving multi-keyword ranked search over encrypted cloud data. *IEEE Transactions on Parallel and Distributed Systems*, 25(1):222–233, 2014.
- [Cyn06] Dwork Cynthia. Differential privacy. *Automata, languages and programming*, pages 1–12, 2006.
- [D⁺06] Kevin Donnelly et al. Snomed-ct: The advanced terminology and coding system for ehealth. *Studies in health technology and informatics*, 121:279, 2006.
- [DCK⁺17] Sandi Dheensa, Daniele Carrieri, Susan Kelly, Angus Clarke, Shane Doheny, Peter Turnpenny, and Anneke Lucassen. A 'joint venture' model of recontacting in clinical genomics: challenges for responsible implementation. *European Journal of Medical Genetics*, 60(7):403–409, 2017.
- [DMS04] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. Technical report, Naval Research Lab Washington DC, 2004.
- [DMTB21] Arianna Dagliati, Alberto Malovini, Valentina Tibollo, and Riccardo Bellazzi. Health informatics and EHR to support clinical research in the COVID-19 pandemic: an overview. *Briefings in Bioinformatics*, 22(2):812–822, 01 2021.
- [Dwo08] Cynthia Dwork. Differential privacy: A survey of results. In Manindra Agrawal, Dingzhu Du, Zhenhua Duan, and Angsheng Li, editors, *Theory and Applications of Models of Computation*, pages 1–19, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.
- [Dwo15] Morris Dworkin. Sha-3 standard: Permutation-based hash and extendable-output functions, 2015-08-04 2015.
- [EAK11] Mohamed Hamdy Eldefrawy, Khaled Alghathbar, and Muhammad Khurram Khan. Otp-based two-factor authentication using

- mobile phones. In *2011 Eighth International Conference on Information Technology: New Generations*, pages 327–331, 2011.
- [EEJAM11] Khaled El Emam, Elizabeth Jonker, Luk Arbuckle, and Bradley Malin. A systematic review of re-identification attacks on health data. *PloS one*, 6(12):e28071, 2011.
- [Elg85] T. Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.
- [FCC⁺14] Rachael L Fleurence, Lesley H Curtis, Robert M Califf, Richard Platt, Joe V Selby, and Jeffrey S Brown. Launching PCORnet, a national patient-centered clinical research network. *Journal of the American Medical Informatics Association*, 21(4):578–582, 05 2014.
- [FES⁺17] David Froelicher, Patricia Egger, João Sá Sousa, Jean Louis Raisaro, Zhicong Huang, Christian Mouchet, Bryan Ford, and Jean-Pierre Hubaux. Unlynx: A decentralized system for privacy-conscious data sharing. *Proc. Priv. Enhancing Technol.*, 2017(4):232–250, 2017.
- [FI10] FP6-IST. Advancing clinico-genomic trials on cancer: Open grid services for improving medical knowledge discovery. <https://cordis.europa.eu/project/id/026996>, 2006-2010.
- [FIPR05] Michael J. Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. Keyword search and oblivious pseudorandom functions. In Joe Kilian, editor, *Theory of Cryptography*, pages 303–324, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.
- [FJT13] Pierre-Alain Fouque, Antoine Joux, and Mehdi Tibouchi. Injective encodings to elliptic curves. In *Information Security and Privacy: 18th Australasian Conference, ACISP 2013, Brisbane, Australia, July 1-3, 2013. Proceedings 18*, pages 203–218. Springer, 2013.
- [FMH⁺21] Christopher B. Forrest, Kathleen M. McTigue, Adrian F. Hernandez, Lauren W. Cohen, Henry Cruz, Kevin Haynes, Rainu Kaushal, Abel N. Kho, Keith A. Marsolo, Vinit P. Nair, Richard Platt, Jon E. Puro, Russell L. Rothman, Elizabeth A. Shenkman, Lemuel Russell Waitman, Neely A. Williams, and Thomas W. Carton. Pcornet® 2020: current state, accomplishments, and future directions. *Journal of Clinical Epidemiology*, 129:60–67, 2021.
- [FMTP⁺20] David Froelicher, Mickaël Misbach, Juan R Troncoso-Pastoriza, Jean Louis Raisaro, and Jean-Pierre Hubaux. Medco2: privacy-preserving cohort exploration and analysis. In *Digital Personalized Health and Medicine*, pages 317–321. IOS Press, 2020.
- [FSGW13] Liming Fang, Willy Susilo, Chunpeng Ge, and Jiandong Wang. Public key encryption with keyword search secure against keyword guessing attacks without random oracle. *Information Sciences*, 238:221–241, 2013.

- [ftPoHSoBR79] The National Commission for the Protection of Human Subjects of Biomedical and Behavioral Research. The belmont report ethical principles and guidelines for the protection of human subjects of research, 1979.
- [FTS⁺20] Chun-I Fan, Yi-Fan Tseng, Hui-Po Su, Ruei-Hau Hsu, and Hiroaki Kikuchi. Secure hierarchical bitcoin wallet scheme against privilege escalation attacks. *International Journal of Information Security*, 19:245–255, 2020.
- [FWCY10] Benjamin C. M. Fung, Ke Wang, Rui Chen, and Philip S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Comput. Surv.*, 42(4), jun 2010.
- [GKR⁺21] Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schafneggler. Poseidon: A new hash function for zero-knowledge proof systems. In *USENIX Security Symposium*, volume 2021, 2021.
- [GLLX13] Lifeng Guo, Bo Lu, Xiangyu Li, and He Xu. A verifiable proxy re-encryption with keyword search without random oracle. In *2013 Ninth International Conference on Computational Intelligence and Security*, pages 474–478, 2013.
- [GPR21] Lior Goldberg, Shahar Papini, and Michael Riabzev. Cairo – a turing-complete stark-friendly cpu architecture. Cryptology ePrint Archive, Paper 2021/1063, 2021. <https://eprint.iacr.org/2021/1063>.
- [GS13] Marcel Großmann and Stephan JA Schuberth. Auto-mininet: Assessing the internet topology zoo in a software-defined network emulator. *Messung, Mellierung un Bewertung von Rechensystemen (MMBnet)*, 7:1–10, 2013.
- [GS18] Kevin Grünberg and Wolfram Schenck. A case study on benchmarking iot cloud services. In Min Luo and Liang-Jie Zhang, editors, *Cloud Computing – CLOUD 2018*, pages 398–406, Cham, 2018. Springer International Publishing.
- [GTA⁺14] Erin M. George, Ana I. Tergas, Cande V. Ananth, William M. Burke, Sharyn N. Lewin, Eri Prendergast, Alfred I. Neugut, Dawn L. Hershman, and Jason D. Wright. Safety and tolerance of radical hysterectomy for cervical cancer in the elderly. *Gynecologic Oncology*, 134(1):36–41, 2014.
- [GY15] Lifeng Guo and Wei-Chuen Yau. Efficient secure-channel free public key encryption with keyword search for emrs in cloud storage. *Journal of Medical Systems*, 39(2):11, Jan 2015.
- [HG16] Kamala D Harris and Attorney General. California data breach report. Retrieved August, 7:2016, 2016.

- [HKA19] Rohit Handa, C. Rama Krishna, and Naveen Aggarwal. Searchable encryption: A survey on privacy-preserving search schemes on encrypted outsourced data. *Concurrency and Computation: Practice and Experience*, 31(17):e5201, 2019. e5201 cpe.5201.
- [HLP12] Chris Hempel, Geoffrey Lomax, and Steve Peckman. Broad consent in biobanking. *Nature Biotechnology*, 30(9):826–826, Sep 2012.
- [HN11] Junbeom Hur and Dong Kun Noh. Attribute-based access control with efficient revocation in data outsourcing systems. *IEEE Transactions on Parallel and Distributed Systems*, 22(7):1214–1221, 2011.
- [HRP⁺15] Raymond Heatherly, Luke V Rasmussen, Peggy L Peissig, Jennifer A Pacheco, Paul Harris, Joshua C Denny, and Bradley A Malin. A multi-institution evaluation of clinical profile anonymization. *Journal of the American Medical Informatics Association*, 23(e1):e131–e137, 11 2015.
- [HTHB14] Elizabeth B. Habermann, Kristine M. Thomsen, Tina J. Hieken, and Judy C. Boughey. Impact of availability of immediate breast reconstruction on bilateral mastectomy rates for breast cancer across the united states: Data from the nationwide inpatient sample. *Annals of Surgical Oncology*, 21(10):3290–3296, Oct 2014.
- [IKK12] Mohammad Saiful Islam, Mehmet Kuzu, and Murat Kantarcioglu. Access pattern disclosure on searchable encryption: ramification, attack and mitigation. In *Ndss*, volume 20, page 12. Citeseer, 2012.
- [Int22a] Health Level Seven International. Fhir specification, 2022.
- [Int22b] SNOMED International. Snomed ct release file specifications, 2022.
- [ISO19] ISO/TC. Health informatics — electronic health record communication — part 1: Reference model. <https://www.iso.org/standard/67868.html>, 2019.
- [JL13] Xiangyang Jiang and Jie Ling. Simple and effective one-time password authentication scheme. In *2013 2nd International Symposium on Instrumentation and Measurement, Sensor Network and Automation (IMSNA)*, pages 529–531, 2013.
- [KBK⁺16] Hyeoneui Kim, Elizabeth Bell, Jihoon Kim, Amy Sitapati, Joe Ramsdell, Claudiu Farcas, Dexter Friedman, Stephanie Feudjio Feupe, and Lucila Ohno-Machado. iCONCUR: informed consent for clinical data and bio-sample use for research. *Journal of the American Medical Informatics Association*, 24(2):380–387, 09 2016.
- [KE10] Hugo Krawczyk and Pasi Eronen. Hmac-based extract-and-expand key derivation function (hkdf). Technical report, 2010.

- [KHA⁺14] Rainu Kaushal, George Hripcsak, Deborah D Ascheim, Toby Bloom, Jr Campion, Thomas R, Arthur L Caplan, Brian P Currie, Thomas Check, Emme Levin Deland, Marc N Gourevitch, Raffaella Hart, Carol R Horowitz, Isaac Kastenbaum, Arthur Aaron Levin, Alexander F H Low, Paul Meissner, Parsa Mirhaji, Harold A Pincus, Charles Scaglione, Donna Shelley, Jonathan N Tobin, and on behalf of the NYC-CDRN. Changing the research landscape: the New York City Clinical Data Research Network. *Journal of the American Medical Informatics Association*, 21(4):587–590, 05 2014.
- [KHG⁺14] Abel N Kho, Denise M Hynes, Satyender Goel, Anthony E Solomonides, Ron Price, Bala Hota, Shannon A Sims, Neil Bahroos, Francisco Angulo, William E Trick, Elizabeth Tarlov, Fred D Rachman, Andrew Hamilton, Erin O Kaleba, Sameer Badlani, Samuel L Volchenboum, Jonathan C Silverstein, Jonathan N Tobin, Michael A Schwartz, David Levine, John B Wong, Richard H Kennedy, Jerry A Krishnan, David O Meltzer, John M Collins, Terry Mazany, and for the CAPriCORN Team. CAPriCORN: Chicago Area Patient-Centered Outcomes Research Network. *Journal of the American Medical Informatics Association*, 21(4):607–611, 05 2014.
- [KHT⁺11] Bartha Maria Knoppers, Jennifer R. Harris, Anne Marie Tassé, Isabelle Budin-Ljøsne, Jane Kaye, Mylène Deschênes, and Ma'n H. Zawati. Towards a data sharing code of conduct for international genomic research. *Genome Medicine*, 3(7):46, Jul 2011.
- [KjCP16] John Kelsey, Shu jen Chang, and Ray Perlner. Sha-3 derived functions: cshake, kmac, tuplehash and parallelhash, 2016-12-22 00:12:00 2016.
- [KLC17] Soohyung Kim, Hyukki Lee, and Yon Dohn Chung. Privacy-preserving data cube for electronic medical records: An experimental evaluation. *International Journal of Medical Informatics*, 97:33–42, 2017.
- [KNF⁺11] Simon Knight, Hung X. Nguyen, Nickolas Falkner, Rhys Bowden, and Matthew Roughan. The internet topology zoo. *IEEE Journal on Selected Areas in Communications*, 29(9):1765–1775, 2011.
- [KNG⁺17] Md. Al-Amin Khandaker, Yuki Nanjo, Loubna Ghammam, Sylvain Duquesne, Yasuyuki Nogami, and Yuta Kodera. Efficient optimal ate pairing at 128-bit security level. In Arpita Patra and Nigel P. Smart, editors, *Progress in Cryptology – INDOCRYPT 2017*, pages 186–205, Cham, 2017. Springer International Publishing.
- [KRN⁺19] Ellen Kim, Samuel M. Rubinstein, Kevin T. Nead, Andrzej P. Wojcieszynski, Peter E. Gabriel, and Jeremy L. Warner. The evolving use of electronic health records (ehr) for research. *Seminars in Radiation Oncology*, 29(4):354–361, 2019. Big Data in Radiation Oncology.

- [KS12] Vladimir Kolesnikov and Abdullatif Shikfa. On the limits of privacy provided by order-preserving encryption. *Bell Labs Technical Journal*, 17(3):135–146, 2012.
- [KWL⁺15] Jane Kaye, Edgar A. Whitley, David Lund, Michael Morrison, Harriet Teare, and Karen Melham. Dynamic consent: a patient interface for twenty-first century research networks. *European Journal of Human Genetics*, 23(2):141–146, Feb 2015.
- [LCLS08] Huang Lin, Zhenfu Cao, Xiaohui Liang, and Jun Shao. Secure threshold multi authority attribute based encryption without a central authority. In Dipanwita Roy Chowdhury, Vincent Rijmen, and Abhijit Das, editors, *Progress in Cryptology - INDOCRYPT 2008*, pages 426–436, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.
- [LHM10] Bob Lantz, Brandon Heller, and Nick McKeown. A network in a laptop: Rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, New York, NY, USA, 2010. Association for Computing Machinery.
- [LKL⁺10] Young Sil Lee, Nack Hyun Kim, Hyotaek Lim, HeungKuk Jo, and Hoon Jae Lee. Online banking authentication system using mobile-otp with qr-code. In *5th International Conference on Computer Sciences and Convergence Information Technology*, pages 644–648, 2010.
- [LLM⁺16] Xiaoping Liu, Xiao-Bai Li, Luvai Motiwalla, Wenjun Li, Hua Zheng, and Patricia D. Franklin. Preserving patient privacy when sharing same-disease data. *J. Data and Information Quality*, 7(4), oct 2016.
- [LLZ19] Yang Lu, Jiguo Li, and Yichen Zhang. Scf-pepcks: Secure channel free public key encryption with privacy-conserving keyword search. *IEEE Access*, 7:40878–40892, 2019.
- [LMAS20] Xabier Larrucea, Micha Moffie, Sigal Asaf, and Izaskun Santamaria. Towards a gdpr compliant way to secure european cross border healthcare industry 4.0. *Computer Standards & Interfaces*, 69:103408, 2020.
- [LS15] Kaitai Liang and Willy Susilo. Searchable attribute-based mechanism with efficient data sharing for secure cloud storage. *IEEE Transactions on Information Forensics and Security*, 10(9):1981–1992, 2015.
- [LUM⁺20] Siddique Latif, Muhammad Usman, Sanaullah Manzoor, Waleed Iqbal, Junaid Qadir, Gareth Tyson, Ignacio Castro, Adeel Razi, Maged N. Kamel Boulos, Adrian Weller, and Jon Crowcroft. Leveraging data science to combat covid-19: A comprehensive review. *IEEE Transactions on Artificial Intelligence*, 1(1):85–103, 2020.

- [LWW04] Joseph K Liu, Victor K Wei, and Duncan S Wong. Linkable spontaneous anonymous group signature for ad hoc groups. In *Information Security and Privacy: 9th Australasian Conference, ACISP 2004, Sydney, Australia, July 13-15, 2004. Proceedings 9*, pages 325–335. Springer, 2004.
- [LYZ⁺13] Ming Li, Shucheng Yu, Yao Zheng, Kui Ren, and Wenjing Lou. Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption. *IEEE Transactions on Parallel and Distributed Systems*, 24(1):131–143, 2013.
- [LZF14] Zhiquan Lv, Min Zhang, and Dengguo Feng. Multi-user searchable encryption with efficient access control for cloud storage. In *2014 IEEE 6th International Conference on Cloud Computing Technology and Science*, pages 366–373, 2014.
- [MAB⁺08] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, mar 2008.
- [MC11] Nir Menachemi and Taleah H Collum. Benefits and drawbacks of electronic health record systems. *Risk Manag Healthc Policy*, 4:47–55, May 2011.
- [MHS⁺03] Clement J McDonald, Stanley M Huff, Jeffrey G Suico, Gilbert Hill, Dennis Leavelle, Raymond Aller, Arden Forrey, Kathy Mercer, Georges DeMoor, John Hook, et al. Loinc, a universal standard for identifying laboratory observations: a 5-year update. *Clinical chemistry*, 49(4):624–633, 2003.
- [MK14] Nicholas D Matsakis and Felix S Klock. The rust language. *ACM SIGAda Ada Letters*, 34(3):103–104, 2014.
- [MKG⁺07] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. 1(1), mar 2007.
- [MM02] Petar Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In *International Workshop on Peer-to-Peer Systems*, pages 53–65. Springer, 2002.
- [MMW⁺17] Yinbin Miao, Jianfeng Ma, Fushan Wei, Zhiquan Liu, Xu An Wang, and Cunbo Lu. Vcse: Verifiable conjunctive keywords search over encrypted data without secure-channel. *Peer-to-Peer Networking and Applications*, 10(4):995–1007, Jul 2017.
- [MSS17] Alfred Menezes, Palash Sarkar, and Shashank Singh. Challenges with assessing the impact of nfs advances on the security of pairing-based cryptography. In Raphaël C.-W. Phan and Moti Yung, editors, *Paradigms in Cryptology – Mycrypt 2016. Malicious and Exploratory Cryptology*, pages 83–108, Cham, 2017. Springer International Publishing.

- [MT06] Einar Mykletun and Gene Tsudik. Aggregation queries in the database-as-a-service model. In Ernesto Damiani and Peng Liu, editors, *Data and Applications Security XX*, pages 89–103, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
- [MT07] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 94–103, 2007.
- [MWM⁺10] Shawn N Murphy, Griffin Weber, Michael Mendis, Vivian Gainer, Henry C Chueh, Susanne Churchill, and Isaac Kohane. Serving the enterprise and beyond with informatics for integrating biology and the bedside (i2b2). *Journal of the American Medical Informatics Association*, 17(2):124–130, 03 2010.
- [NAG22] M. H. Noorallahzade, R. Alimoradi, and A. Gholami. A survey on public key encryption with keyword search: Taxonomy and methods. *International Journal of Mathematics and Mathematical Sciences*, 2022:3223509, Mar 2022.
- [NISa] Nis description of data elements. <https://hcup-us.ahrq.gov/db/nation/nis/nisdde.jsp>. Accessed: 2023-06-21.
- [NISb] Nis file specifications. <https://hcup-us.ahrq.gov/db/nation/nis/nisfilespecs.jsp>. Accessed: 2023-06-21.
- [NL18] Yoav Nir and Adam Langley. ChaCha20 and Poly1305 for IETF Protocols. RFC 8439, June 2018.
- [oHfCR21] U.S. Department of Health and Human Services Office for Civil Rights. Reports to congress on breach notification program. <https://www.hhs.gov/hipaa/for-professionals/breach-notification/reports-congress/index.html>, 2013–2021.
- [Ohm09] Paul Ohm. Broken promises of privacy: Responding to the surprising failure of anonymization. *UCLA l. Rev.*, 57:1701, 2009.
- [ONL] National (nationwide) inpatient sample (nis) database. <https://www.hcup-us.ahrq.gov/db/nation/nis/nisdbdocumentation.jsp>.
- [Org04] World Health Organization. *International Statistical Classification of Diseases and related health problems: Alphabetical index*, volume 3. World Health Organization, 2004.
- [ORRS01] Salvador Ochoa, Jamie Rasmussen, Christine Robson, and Michael Salib. Reidentification of individuals in chicago’s homicide database: A technical and legal study. *Massachusetts Institute of Technology*, 2001.
- [ÖS14] Cengiz Örencik and Erkey Savaş. An efficient privacy-preserving multi-keyword search over encrypted cloud data with ranking. *Distributed and Parallel Databases*, 32(1):119–160, Mar 2014.

- [oSP20] openEHR Specification Program. Reference model (rm) component, 2020.
- [PCKK17] Shubaham Purri, Tanupriya Choudhury, Nirbhay Kashyap, and Praveen Kumar. Specialization of iot applications in health care industries. In *2017 International Conference on Big Data Analytics and Computational Intelligence (ICBDAC)*, pages 252–256, 2017.
- [PLSGD17] Giorgos Poulis, Grigorios Loukides, Spiros Skiadopoulos, and Aris Gkoulalas-Divanis. Anonymizing datasets with demographics and diagnosis codes in the presence of utility constraints. *Journal of Biomedical Informatics*, 65:76–96, 2017.
- [PMM16] Stacey A. Page, Kiran Pohar Manhas, and Daniel A. Muruve. A survey of patient perspectives on the research use of health information and biospecimens. *BMC Medical Ethics*, 17(1):48, Aug 2016.
- [QQLM20] Han Qiu, Meikang Qiu, Meiqin Liu, and Gerard Memmi. Secure health data sharing for medical cyber-physical systems for the health-care 4.0. *IEEE Journal of Biomedical and Health Informatics*, 24(9):2499–2505, 2020.
- [Ray13] Tracey Raybourn. *Bucketization Techniques for Encrypted Databases: Quantifying the Impact of Query Distributions*. PhD thesis, 2013.
- [RBLL06] Hyun Sook Rhee, Jin Wook Byun, Dong Hoon Lee, and Jongin Lim. Oblivious conjunctive keyword search. In Joo-Seok Song, Taekyoung Kwon, and Moti Yung, editors, *Information Security Applications*, pages 318–327, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
- [RD08] Eric Rescorla and Tim Dierks. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, August 2008.
- [RPSL10] Hyun Sook Rhee, Jong Hwan Park, Willy Susilo, and Dong Hoon Lee. Trapdoor security in a searchable public-key encryption scheme with a designated tester. *Journal of Systems and Software*, 83(5):763–771, 2010.
- [RQIB22] Alex Rodriguez, Jeanette Quiñones, Yuzo Iano, and Marco A. Q. Barra. A comparative evaluation of odl and onos controllers in software-defined network environments. In *2022 IEEE XXIX International Conference on Electronics, Electrical Engineering and Computing (INTERCON)*, pages 1–4, 2022.
- [RSK09] Hyun Sook Rhee, Willy Susilo, and Hyun-Jeong Kim. Secure searchable public key encryption scheme against keyword guessing attacks. *IEICE Electronics Express*, 6(5):237–243, 2009.
- [RT21] Eugenia Rinaldi and Sylvia Thun. From openehr to fhir and omop data model for microbiology findings. In *Public Health and Informatics*, pages 402–406. IOS Press, 2021.

- [RTPM⁺19] Jean Louis Raisaro, Juan Ramón Troncoso-Pastoriza, Mickaël Misbach, João Sá Sousa, Sylvain Pradervand, Edoardo Missiaglia, Olivier Michielin, Bryan Ford, and Jean-Pierre Hubaux. Medco: Enabling secure and privacy-preserving exploration of distributed clinical and genomic data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 16(4):1328–1341, 2019.
- [RVGH13] Haakon Riiser, Paul Vigmostad, Carsten Griwodz, and Pål Halvorsen. Commute path bandwidth traces from 3g networks: Analysis and applications. In *Proceedings of the 4th ACM Multimedia Systems Conference*, MMSys '13, page 114–118, New York, NY, USA, 2013. Association for Computing Machinery.
- [Sch01] R. Schollmeier. A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In *Proceedings First International Conference on Peer-to-Peer Computing*, pages 101–102, 2001.
- [Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, nov 1979.
- [Sha85] Adi Shamir. Identity-based cryptosystems and signature schemes. In George Robert Blakley and David Chaum, editors, *Advances in Cryptology*, pages 47–53, Berlin, Heidelberg, 1985. Springer Berlin Heidelberg.
- [SHV⁺15] Alexandru L. Stancu, Simona Halunga, Alexandru Vulpe, George Suciu, Octavian Fratu, and Eduard C. Popovici. A comparison between several software defined networking controllers. In *2015 12th International Conference on Telecommunication in Modern Satellite, Cable and Broadcasting Services (TELSIKS)*, pages 223–226, 2015.
- [Sia12] Amar Siad. Anonymous identity-based encryption with distributed private-key generator and searchable encryption. In *2012 5th International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–8, 2012.
- [SKD⁺13] Lisa M Schilling, Bethany M Kwan, Charles T Drolshagen, Patrick W Hosokawa, Elias Brandt, Wilson D Pace, Christopher Uhrich, Michael Kamerick, Aidan Bunting, Philip RO Payne, et al. Scalable architecture for federated translational inquiries network (saftinet) technology infrastructure for a distributed data network. *EGEMS*, 1(1), 2013.
- [SKMS13] Kristin Solum Steinsbekk, Bjørn Kåre Myskja, and Berge Solberg. Broad consent versus dynamic consent in biobank research: Is passive participation an ethical problem? *European Journal of Human Genetics*, 21(9):897–902, Sep 2013.
- [SMC08] Joseph A. Salowey, David McGrew, and Abhijit Choudhury. AES Galois Counter Mode (GCM) Cipher Suites for TLS. RFC 5288, August 2008.

- [SRR19] Rishi Saripalle, Christopher Runyan, and Mitchell Russell. Using hl7 fhir to achieve interoperability in patient health record. *Journal of Biomedical Informatics*, 94:103188, 2019.
- [SS11] Kristin Solum Steinsbekk and Berge Solberg. Biobanks—When is Re-consent Necessary? *Public Health Ethics*, 4(3):236–250, 11 2011.
- [SSW09] Emily Shen, Elaine Shi, and Brent Waters. Predicate privacy in encryption systems. In Omer Reingold, editor, *Theory of Cryptography*, pages 457–473, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.
- [SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, pages 457–473, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.
- [Swe02] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International journal of uncertainty, fuzziness and knowledge-based systems*, 10(05):557–570, 2002.
- [SWP00] Dawn Xiaoding Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000*, pages 44–55, 2000.
- [SYL⁺16] Wenhai Sun, Shucheng Yu, Wenjing Lou, Y. Thomas Hou, and Hui Li. Protecting your right: Verifiable attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud. *IEEE Transactions on Parallel and Distributed Systems*, 27(4):1187–1198, 2016.
- [VBL⁺14] Joshua Vogel, Jeffrey S Brown, Thomas Land, Richard Platt, and Michael Klompas. Mdpnet: secure, distributed sharing of electronic health record data for public health surveillance, evaluation, and planning. *American journal of public health*, 104(12):2265–2270, 2014.
- [VHVJ20] Raju Vaishya, Abid Haleem, Abhishek Vaish, and Mohd Javaid. Emerging technologies to combat the covid-19 pandemic. *Journal of clinical and experimental hepatology*, 10(4):409–411, 2020.
- [VN21] Narasimha Raghavan Veeraragavan and Jan Franz Nygård. Decansec: A decentralized architecture for secure statistical computations on distributed health registry data. In *Proceedings of the 16th International Conference on Availability, Reliability and Security, ARES 21*, New York, NY, USA, 2021. Association for Computing Machinery.
- [VNM⁺20] Dimitris Vyzovitis, Yusef Napora, Dirk McCormick, David Dias, and Yiannis Psaras. Gossipsub: Attack-resilient message propagation in the filecoin and eth2.0 networks, 2020.
- [VPPE⁺14] Willem G Van Panhuis, Proma Paul, Claudia Emerson, John Grefenstette, Richard Wilder, Abraham J Herbst, David Heymann, and

- Donald S Burke. A systematic review of barriers to data sharing in public health. *BMC public health*, 14(1):1–9, 2014.
- [VS18] Nicolas Van Saberhagen. Cryptonote v 2.0 (2013). 2018.
- [WB11] Mark Walport and Paul Brest. Sharing research data to improve public health. *The Lancet*, 377(9765):537–539, 2011.
- [WB19] Riad S Wahby and Dan Boneh. Fast and simple constant-time hashing to the bls12-381 elliptic curve. *Cryptology ePrint Archive*, 2019.
- [WCGFJ12] David I Wolinsky, Henry Corrigan-Gibbs, Bryan Ford, and Aaron Johnson. Scalable anonymous group communication in the anytrust model. Technical report, NAVAL RESEARCH LAB WASHINGTON DC, 2012.
- [WF06] Ke Wang and Benjamin C. M. Fung. Anonymizing sequential releases. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 414–423, New York, NY, USA, 2006. Association for Computing Machinery.
- [WLSL18] Dimaz Ankaa Wijaya, Joseph Liu, Ron Steinfeld, and Dongxi Liu. Monero ring attack: Recreating zero mix-in transaction effect. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/Big-DataSE)*, pages 1196–1201, 2018.
- [WMCZ16] Xiao-Fen Wang, Yi Mu, Rongmao Chen, and Xiao-Song Zhang. Secure channel free id-based searchable encryption for peer-to-peer group. *Journal of Computer Science and Technology*, 31(5):1012–1027, Sep 2016.
- [WSLH15] Bing Wang, Wei Song, Wenjing Lou, and Y Thomas Hou. Inverted index based multi-keyword public-key searchable encryption with strong privacy guarantee. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 2092–2100. IEEE, 2015.
- [WTT14] Tsu-Yang Wu, Tung-Tso Tsai, and Yuh-Min Tseng. Efficient searchable id-based encryption with a designated server. *annals of telecommunications - annales des télécommunications*, 69(7):391–402, Aug 2014.
- [Wür18] Tim Würtele. Entwurf und analyse eines sicheren und effizienten bulletin board auf basis von blockchains. Master’s thesis, 2018.
- [WXFW09] Ke Wang, Yabo Xu, Ada W. C. Fu, and Raymond C. W. Wong. Ff-anonymity: When quasi-identifiers are missing. In *2009 IEEE 25th International Conference on Data Engineering*, pages 1136–1139, 2009.

- [Yan15] Yang Yang. Attribute-based data retrieval with semantic keyword search for e-health cloud. *Journal of Cloud Computing*, 4(1):10, May 2015.
- [YM16] Yang Yang and Maode Ma. Conjunctive keyword search with designated tester and timing enabled proxy re-encryption function for e-health clouds. *IEEE Transactions on Information Forensics and Security*, 11(4):746–759, 2016.
- [YMM⁺17] Jiawei Yuan, Bradley Malin, François Modave, Yi Guo, William R. Hogan, Elizabeth Shenkman, and Jiang Bian. Towards a privacy preserving cohort discovery framework for clinical research networks. *Journal of Biomedical Informatics*, 66:42–51, 2017.
- [YZCT17] Yang Yang, Xianghan Zheng, Victor Chang, and Chunming Tang. Semantic keyword searchable proxy re-encryption for postquantum secure cloud storage. *Concurrency and Computation: Practice and Experience*, 29(19):e4211, 2017. e4211 cpe.4211.
- [ZDO⁺08] Sergej Zerr, Elena Demidova, Daniel Olmedilla, Wolfgang Nejdl, Marianne Winslett, and Soumyadeb Mitra. Zerber: R-confidential indexing for distributed documents. In *Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '08, page 287–298, New York, NY, USA, 2008. Association for Computing Machinery.
- [ZLW18] Yu Zhang, Yin Li, and Yifan Wang. Conjunctive and disjunctive keyword search over encrypted mobile cloud data in public key system. *Mobile Information Systems*, 2018:3839254, May 2018.
- [ZLW19] Yu Zhang, Yin Li, and Yifan Wang. Secure and efficient searchable public key encryption for resource constrained environment based on pairings under prime order group. *Security and Communication Networks*, 2019:5280806, May 2019.
- [ZLZ⁺17] Jiajun Zhang, Xiaohui Liang, Zhikun Zhang, Shibo He, and Zhiguo Shi. Re-dpactor: Real-time health data releasing with w-day differential privacy. In *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pages 1–6, 2017.
- [ZNS12] Fangming Zhao, Takashi Nishide, and Kouichi Sakurai. Multi-user keyword search scheme for secure data sharing with fine-grained access control. In Howon Kim, editor, *Information Security and Cryptology - ICISC 2011*, pages 406–418, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
- [ZSD⁺22] Yu Zhan, Danfeng Shen, Pu Duan, Benyu Zhang, Zhiyong Hong, and Baocang Wang. Mdope: Efficient multi-dimensional data order preserving encryption scheme. *Information Sciences*, 595:334–343, 2022.

- [ZXLZ17] Rui Zhang, Rui Xue, Ling Liu, and Lijuan Zheng. Oblivious multi-keyword search for secure cloud storage service. In *2017 IEEE International Conference on Web Services (ICWS)*, pages 269–276, 2017.
- [ZZ11] Bo Zhang and Fangguo Zhang. An efficient public key encryption with conjunctive-subset keywords search. *Journal of Network and Computer Applications*, 34(1):262–267, 2011.
- [ZZD18] Yinghui Zhang, Dong Zheng, and Robert H. Deng. Security and privacy in smart health: Efficient policy-hiding attribute-based access control. *IEEE Internet of Things Journal*, 5(3):2130–2145, 2018.

A

Appendix

A.1 List of Abbreviations

ABE attribute based encryption

AES-GCM AES with Galois/Counter Mode

ASE asymmetric searchable encryption

bLSAG Back Linkable Spontaneous Anonymous Group

CA collective authority

CP-ABE ciphertext-policy attribute-based encryption

cSHAKE customizable Secure Hash Algorithm **KECCAK**

CSP cloud service provider

CSV Comma-Separated Values

DAG directed acyclic graph

DFN ‘Deutsches Forschungsnetz’

DHDN distributed health data network

DO data owner

DP data provider

DU data user

ECC elliptic curve cryptography

ECDH Elliptic Curve Diffie-Hellman

ECDLP Elliptic Curve Discrete Logarithm Problem

ECDSA Elliptic Curve Digital Signature Algorithm

EdDSA Edwards-curve Digital Signature Algorithm

EHR electronic health record

ESE efficiently searchable encryption

FHIR Fast Health Interoperability Resources

FI file injection

GDPR General Data Protection Regulation

HCUP Healthcare Cost Utilization Project

HDH Hash Diffie-Hellman

HIPAA Health Insurance Portability and Accountability Act

HITECH Healthcare Interoperability Testing and Conformance Harmonization

HVE hidden vector encryption

IBE identity based encryption

ICD International Classification of Diseases

ICD-10 International Classification of Diseases, Tenth Revision

ICF International Classification of Functioning, Disability and Health

ICHI International Classification of Health Interventions

IoT Internet of things

ISP internet service provider

IT information technology

KDF key derivation function

KG keyword guessing

LSAG Linkable Spontaneous Anonymous Group

LSS linear secret-sharing

MPC multi-party computation

NIS Nationwide Inpatient Sample

OPE order-preserving encryption

OTP one time password

P2P	peer-to-peer
PEKS	public key encryption with keyword search
PHI	protected health information
PKI	public key infrastructure
PoP	point of presence
PPT	Privacy-Preserving Technologies
QR code	quick response code
RN	research network
ROM	random oracle model
RSA	Rivest-Shamir-Adleman
SCF	secure channel free
SCF-PEKS	secure channel free PEKS
SDN	Software Defined Networking
SE	searchable encryption
SQL	Structured Query Language
TLS	Transport Layer Security
TS	test server
UUID	universally unique identifier
WHO-FIC	WHO Family of International Classifications

A.2 Network Messages

This part of the appendix shall show how data messages may be structured and serve as an overview of data required by corresponding protocols. These examples align with the prototype implementation but are slightly modified to increase readability.

```
const KAPPA : usize = 32;

type HEntropy = [u8; KAPPA];
type DLogProof<G> = ([G], [G]);
type Trapdoor = (G1, G2);
type EncryptedIndex = (G1, HEntropy);
type BucketIndex = [u8; 8];
```

Message A.1: Type definitions of primitives for selected cryptographic schemes. The security parameter κ is chosen for a hash output size of 256 bits, corresponding to the targeted 128-bit security level [BDP⁺18].

```
struct Query
{
    metadata : Metadata,
    ephemeral : Trapdoor,
    procedures : [Trapdoor],
    diagnoses : [Trapdoor],
    extra : [Trapdoor],
    buckets : [Option<BucketIndex>],
}

struct Metadata
{
    public_key : G1,
    timestamp : u64,
    network: Uuid,
    data_network: Uuid,
    data_provider: [Uuid],
}
```

Message A.2: Full definition of a query featuring an extra structure for metadata including the minimum number of required fields.

```
enum DataField {
    AGE,
    FEMALE,
    I10_DX { index: u8 },
    I10_PR { index: u8 },
    ...
}
```

Message A.3: Part of the Nationwide Inpatient Sample (NIS) data field description featuring fields that are actively used by our implemented queries.

```

struct Request
{
    public_key : G1,
    metadata : Metadata,
    ephemeral : G1,
    trapdoors : [G2],
}

struct Response
{
    ephemeral : G1,
    signature : EdDsaSignature,
    trapdoors : [G2],
    shuffle : G2,
    proof : DLogProof<G2>,
}

```

Message A.4: Message definition for the **MAKE QUERY** protocol between data owner (DO) and collective authority (CA) members.

```

struct Payload
{
    query: Query,
    key_chain: [G1],
    sig_chain: [EdDsaSignature],
    signature: EdDsaSignature,
}

```

Message A.5: Message definition for publishing to a bulletin board using the **PUBLISH QUERY** protocol.

```

struct Request
{
    query_id: Uuid,
    timeout: Timestamp,
    procedures : [G2],
    diagnoses : [G2],
    extra : [G2],
    buckets : [Option<BucketIndex>],
}

struct Response
{
    /* empty */
}

```

Message A.6: Message definition for the **FORWARD QUERY** protocol between data provider (DP) and test servers (TSs).

```

struct Request
{
    known_query: Option<Uuid>,
}

struct Response
{
    new_queries: [Uuid]
}

```

Message A.7: Message definition for the **DISCOVER QUERIES** protocol between data owner (DO) and data providers (DPs).

```

struct Request
{
    queries: [RequestEntry]
}

struct RequestEntry
{
    stealth: StealthAddress,
    signature: EdDsaSignature,
}

struct Response
{
    queries: [ResponseEntry]
}

struct ResponseEntry
{
    ephemeral: G1,
    fields: [DataField],
}

```

Message A.8: Message definition for the **ACQUIRE QUERIES** protocol between data owner (DO) and data providers (DPs).

```

struct Payload
{
    stealth: StealthAddress,
    consent: AesCiphertext<ConsentPayload>,
    signature: EdDsaSignature,
}

type ConsentPayload = HashMap<DataField, Option<G1>>;

```

Message A.9: Message definition for publishing to a bulletin board using the **PUBLISH CONSENT** protocol.

```

struct Request
{
    query: Uuid,
    result: ElGamalCiphertext<u64>,
}

struct Response
{
    /* empty */
}

```

Message A.10: Message definition for the **FORWARD RESULT** protocol between test server (TS) and data providers (DPs).

```

struct Payload
{
    result: ElGamalCiphertext<u64>,
    signature: BlsagSignature,
}

```

Message A.11: Message definition for publishing to a bulletin board using the **PUBLISH RESULT** protocol.

```
struct Request
{
    query: Uuid,
}

struct Response
{
    result: ElGamalCiphertext<u64>,
    nonce: G1,
    forward: G1,
    proof: DLogProof<G1>,
}
```

Message A.12: Message definition for the **RECEIVE RESULT** protocol between data user (DU) and collective authority (CA) members.

A.3 PEKS Algorithms

PEKS (PK_R, w)

$r \leftarrow \$ \mathbb{Z}_p^*$
 $t = e(\text{H}_1(w), r.\text{PK}_R)$
return $(r.B, \text{H}_2(t))$

Trapdoor (sk_R, w)

return $\text{H}_1(w)^{\text{sk}_R}$

Test ($\mathcal{I}, \mathcal{T}$)

$(A, B) = \mathcal{I}$
return $\text{H}_2(e(\mathcal{T}, A)) \stackrel{?}{=} B$

Algorithm A.1: Cryptographic operations of the first public key encryption with keyword search (PEKS) scheme published by Boneh et al. [BCOP04].

PEKS ($\text{sk}_O, \text{PK}_R, \text{PK}_S, w$)

$r \leftarrow \$ \mathbb{Z}_p^*$
 $U = r.\text{PK}_{R,2}$
 $\rho = \text{H}_1(\text{sk}_O.\text{PK}_{R,1})$
 $V = \text{H}_3(r.e(\text{PK}_S, Q_1 + \text{H}_2(w \parallel \rho).Q_2))$
return (U, V)

dTrapdoor ($\text{sk}_R, \text{PK}_O, \text{PK}_S, w$)

$\rho' = \text{H}_1(\text{sk}_{R,1}.\text{PK}_O)$
 $T_{w'} = z.(Q_1 + \text{H}_2(w' \parallel \rho').Q_2)$
return $T_{w'}$

dTest ($\text{sk}_S, \mathcal{I}, \mathcal{T}$)

return $\text{H}_3(\text{sk}_S.e(U, T_{w'})) \stackrel{?}{=} V$

Algorithm A.2: Cryptographic operations of the Lu et al. [LLZ19] public key encryption with keyword search (PEKS) scheme.

PEKS ($\text{PK}_R, \text{PK}_S, w$)

$r \leftarrow \$ \mathbb{Z}_p^*$
 $A = r.\text{PK}_{R,1}$
 $B = \text{H}_2(e(\text{PK}_{S,1}, r.\text{H}_1(w)))$
return (A, B)

dTrapdoor ($\text{sk}_R, \text{PK}_S, w$)

$r' \leftarrow \$ \mathbb{Z}_p^*$
 $T_1 = r'.B$
 $T_2 = \text{sk}_R^{-1}.\text{H}_1(w) \text{H}(r'.\text{PK}_{S,1})$
return (T_1, T_2)

dTest ($\text{sk}_S, \mathcal{I}, \mathcal{T}$)

$T = T_2 - \text{H}(\text{sk}_S.T_1)$
return $B \stackrel{?}{=} \text{H}_2(e(A, \text{sk}_S.T))$

Algorithm A.3: Cryptographic operations of the Rhee et al. [RPSL10] public key encryption with keyword search (PEKS) scheme.

PEKS ($\text{PK}_R, \text{PK}_S, w$)	dTrapdoor ($\text{sk}_R, \text{PK}_S, w$)	dTest ($\text{PK}_R, \text{sk}_S, \mathcal{I}, \mathcal{T}$)
$r, s \leftarrow \mathbb{Z}_p^*$	$r' \leftarrow \mathbb{Z}_p^*$	$h' = H_2(C_1, C_2)$
$C_1 = r.\text{PK}_R$	$T_1 = r'.g$	if $e(C_1, h'.u + s.v + d)$
$T = r.e(\text{PK}_S, w.g_1 + h)$	$T_2 = \text{sk}_R^{-1} \cdot (w.g_1 + h)$	$\neq e(\text{PK}_R, C_3)$
$C_2 = H_1(T)$	$+ H(r'.\text{PK}_S)$	return $\perp$
$h' = H_2(C_1, C_2)$	return (T_1, T_2)	fi
$C_3 = r.(h'.u + s.v + d)$		$T = T_2 - H(\text{sk}_S.T_1)$
return (s, C_1, C_2, C_3)		return $C_2 \stackrel{?}{=} H_1(e(C_1, \text{sk}_S.T))$

Algorithm A.4: Cryptographic operations of the Guo et al. [GY15] public key encryption with keyword search (PEKS) scheme.

A.4 University Hospitals Data

Name	Cases	DP Threads	TS Threads	DO Threads
Universitätsklinikum Heidelberg	1,207,763	2	5	1
Charite Universitätsmedizin Berlin	863,216	2	4	1
Universitätsklinikum Erlangen	627,532	1	3	1
Universitätsklinikum Freiburg	623,978	1	3	1
Universitätsklinikum Würzburg	611,054	1	3	1
Universitätsklinikum Münster	572,446	1	3	1
Universitätsklinikum Frankfurt	569,475	1	3	1
Universitätsklinikum Jena	529,385	1	2	1
Medizinische Hochschule Hannover	474,026	1	2	1
Universitätsklinikum Carl Gustav Carus Dresden	472,879	1	2	1
Universitätsklinikum Hamburg-Eppendorf	465,829	1	2	1
Klinikum der LMU München	464,342	1	2	1
Universitätsklinikum und Medizinische Fakultät Tübingen	460,875	1	2	1
Universitätsklinikum Köln	436,821	1	2	1
Universitätsklinikum Schleswig-Holstein-Campus Kiel	430,871	1	2	1
Universitätsklinikum Schleswig-Holstein-Campus Lübeck	430,871	1	2	1
Universitätsklinikum Düsseldorf	422,777	1	2	1
Universitätsklinikum Gießen und Marburg - Standort Marburg	415,375	1	2	1
Universitätsklinikum Bonn	402,183	1	2	1
Klinikum Mannheim Universitätsklinikum	379,051	1	2	1
Total	10,860,749	22	50	20