2022-06

Light-weight Privacy Infrastructure - A Blockchain-based Privacy-Preservation Platform for Data Storage and Query Processing

Mireku Kwakye, Michael

UNIVERSITY OF CALGARY

Light-weight Privacy Infrastructure - A Blockchain-based Privacy-Preservation Platform for Data Storage and Query Processing

by

Michael Mireku Kwakye

A THESIS
SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE
DEGREE OF DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN COMPUTER SCIENCE

CALGARY, ALBERTA

JUNE, 2022

Abstract

Privacy-preservation policies are guidelines and recommendations formulated to protect data provider's private, sensitive data in data repositories. These policies are implemented using privacy-preservation methodologies. Previous privacy-preservation methodologies have addressed privacy in which data are permanently stored in repositories and disconnected from changing data provider privacy preferences. This becomes evident as the data moves to another data repository. The ability of data providers to flexibly update or change their privacy preferences when it is required is a known challenge. Moreover, the ability for data providers to control their existing privacy preferences due to changes in data usage continues to remain a problem.

This research proposes a Light-weight Privacy Infrastructure (LPI); which is a methodology/framework for privacy-preservation of data provider's private and sensitive data. The approach offers data providers flexibility to easily change and monitor privacy preferences on their stored data when the data usage requirements change. Additionally, the approach offers data providers control over access and usage of their private, sensitive data by data collectors and/or accessors and third-party data accessors. The research proposes to tightly couple data provider's private attribute data element to privacy preferences and data accessor data elements. The implementation presents a framework of tightly-coupled relational Database Management System (DBMS), blockchains, and genomic data store. The coupled database framework delivers a secure and query-efficient platform for management and query processing of data provider's private data.

The implementation adopts an Alberta biotechnology platform that provides commercial oncogenomic services, as a case study. The healthcare platform processes both cancer-related healthcare data and next generation sequencing (NGS) genomic data. Data privacy in healthcare data is a necessary requirement in the processing of data provider private and sensitive data across varied data repositories. The implementation provides data providers (*i.e.,* patients) and data collectors and/or accessors (for *e.g.,* physicians) the platform to efficiently manage data whiles eliminating the risks of privacy breaches and unauthorized data access.

The major contributions are: first, provide an approach to tightly couple data provider private, sensitive data with privacy preferences, and data accessor data elements into a privacy tuple. Second, provide a tightly-coupled immutable, tamper-resistant data processing platform where data providers monitor and control all forms of access to their private, sensitive data. Third, provide implementation of a privacy infrastructure where data providers have maximum flexibility to change their privacy preferences on all transactions processed on their underlying private, sensitive data without requiring the data collector. Finally, provide an implementation framework applicable to healthcare and genomic data processing that uses a biotechnology platform as a case study. The evaluation analysis from the implementation procedures offers a validation for the research based on the query processing output of privacy-aware queries on the privacy infrastructure.

Preface

This dissertation is an original work by the author. No part of this dissertation has been previously published.

Some of the research conducted for this dissertation forms part of a cross-disciplinary research collaboration with Dr. Etienne R. Mahe and Dr. Davinder Sidhu at the Department of Pathology & Laboratory Medicine, and Hematology & Hematological Malignancies, Department of Medicine, University of Calgary.

Acknowledgements

”But we have this treasure in earthen vessels, that the excellency of the power may be of God, and not of us” (2 Corinthians 4:7)

I thank God for helping me through this doctoral programme. His grace, favour, and faithfulness has continuously held me through the academic journey and generally in my entire life.

I also express my profound appreciation to my academic advisor - Dr. Kenneth Edwin Barker. Ken has been instrumental and pivotal in getting through the various stages of my doctoral programme. I thank Ken for his untiring support, careful supervision, and guidance in my overall research. I also express my sincere gratitude to Dr. Etienne R. Mahe and Dr. Davinder Sidhu at the Department of Pathology & Laboratory Medicine and Hematology & Hematological Malignancies, Department of Medicine, University of Calgary, for their support and collaboration (through Tunote Oncogenomics Inc.) in offering me the requisite information on patient genomic data processing and analysis. I acknowledge the financial support from Department of Computer Science, Tunote Oncogenomics Inc., and Mitacs Inc., for funding received during the programme. Moreover, I acknowledge the work of the dissertation supervisory committee members, Dr. Michael John Jacobson Jr. and Dr. Joel Christopher Reardon. Their immense reviews, comments, and supervision until the final draft cannot go unmentioned. Many thanks also go to the internal examiner, Dr. Behrouz Homayoun Far, and external examiner, Dr. Suprio Ray, for their insightful reviews and corrections. I acknowledge and thank Dr. Wayne Michael Eberly for serving as the neutral chair during my thesis defence.

I cannot forget my privacy and security research lab mates - Leanne Wu, Emmanuel Onu, Timothy Ohanekwu, Maryam Majedi, Khosro Salmani - and others who have been very and helpful in discussions.

Finally, I thank my wife, Florence, and kids, Afia and Akosua; who have endured all the difficult times during this programme and encouraged me to its completion. The entire family had to deal with different summer travels across the Atlantic ocean and limited time together. Florence managed difficult times of our early family life, in joggling between her extremely demanding banking job and managing home and kids early education (and not to mention helping in school homework after long days at the office). Afia and

Akosua had to cope with many nights of missing their daddy who will read late bedtime stories to them.

Moreover, I acknowledge the motivation and support from the pastorate and membership of my church, the Church of Pentecost, Calgary. Also much mention to my mum and siblings, as well as, other family and friends (especially the Boateng, Oku, Eduful, and Kyeremanten families) for their love, encouragement and caring support. Many thanks to all others who helped in varied ways.

Dedication

This dissertation is dedicated to these precious people in my life:

To my wife, Florence Naa Karley Mireku-Kwakye, and kids, Afia Gyamfuah Mireku-Kwakye and Akosua Kwakyewaa Mireku-Kwakye.

To my late dad, Martin Kwakye Addo - whose hard-work, tenacity, and motivation gave me a solid educational foundation to reach this far.

Table of Contents

List of Tables

List of Figures, Illustrations, Other Graphics

List of Symbols, Abbreviations, Nomenclature

Symbol or Abbreviation	**Definition**
LPI	Light-weight Privacy Infrastructure
NGS	Next Generation Sequencing
DBMS	Database Management System
CI	Contextual Integrity
GDPR	General Data Protection Regulation
PoC	Point-of-Contact
DNA	Deoxyribonucleic Acid
SALT	Sequential, Agreed-on, Ledgered, and Tamper-resistant
EHR	Electronic Health Records
API	Application Programming Interface
*ERC*20	Ethereum Blockchain Token (Ethereum Request for Comment)
UI	User Interface
BAM	Binary Alignment Map
SAM	Sequence Alignment Map
VCF	Variant Call Format
AWS	Amazon Web Services
HPC	High Performance Computing
UML	Unified Modelling Language
CLS	Clinical Laboratory Scientist
H&E	Hematoxylin and Eosin
PHP	PHP: Hypertext Preprocessor
HTML	Hypertext Markup Language
CGI	Common Gateway Interface
HTTP	Hypertext Transfer Protocol
SQL	Structured Query Language
DES	Data Encryption Standard
ER	Entity Relationship
DBA	Database Administrator
DDL	Data Definition Language
DML	Data Manipulation Language
JSON	JavaScript Object Notation
ASCII	American Standard Code for Information Interchange
MB	Mega Bytes
GB	Giga Bytes
SCOM	System Center Operations Manager
PAM	Privileged Access Management
KMS	Key Management Service

$3DES$	Triple Data Encryption Algorithm (3DEA)
AES	Advanced Encryption Standard
TLS	Transport Layer Security
SSL	Secure Sockets Layer
$IPsec$	Internet Protocol Security
SSH	Secure Shell
PGP	Pretty Good Privacy
B/BC	Bioinformatic and Computational Biology
IoT	Internet-of-Things
PoW	Proof-of-Work
BFT	Byzantine Fault Tolerance

Publication Credits

Published Paper(s)

1. Michael Mireku Kwakye and Ken Barker. 2016. Privacy-Preservation in the Integration and Querying of Multidimensional Data Models. In: *Proceedings of the 14th Annual Conference on Privacy, Security and Trust (PST 2016)*, December 2016, Auckland, New Zealand, pp. 255-263. https://doi.org/10.1109/PST.2016.7906971

2. Michael Mireku Kwakye. 2019. Privacy-Preservation in Data Pre-processing for Web Usage Mining. *International Journal of Information Privacy, Security and Integrity*, vol. 4, no. 2, April 3, 2020, pp.134-152. https://doi.org/10.1504/IJIPSI.2019.106605

Chapter 1

Introduction

Privacy-preservation in data repositories is a fundamental and necessary requirement in the processing of personalized, private data across varied data management areas. The concept of preserving privacy in data stores outlines methodologies and implementations that help protect data provider's private and sensitive; during data storage, management, and query processing in the data stores. Hence, privacy-preservation methods offer data collectors and/or accessors the platfrom to process (and draw data utility) from data provider's private and transaction data without breaching privacy and personal identification. In practical terms, the discussion and applications of privacy concepts has significant influences in different domains, such as, legal frameworks, healthcare management, and government data platforms, amongst others. The emergence of multiple diverse data collection points across different application domains allows the processing of large data volumes on data providers. Most especially, associated transactions from collected data provider's private, sensitive data through Point-of-Contacts user interface platforms opens up important requirements for storage and management of these data.

Consider the application domain of a healthcare system where patient healthcare records are collected and maintained for healthcare service delivery. The primary source of data is from the patient and so we classify the patient as the data provider. Different forms of data are collected from the data providers. These collected data may be hypothetically categorized as less-sensitive, medium-sensitive, and highly-sensitive. The categorization of these collected data defines an efficient specification of privacy preferences for related data of the same sensitivity level. The varied data forms enable efficient methods for storage, management, and disclosure of related patient private, sensitive data items. Moreover, this hypothetical approach enable better query processing and data retrieval of related data values for intended data accessors. It is noted that based on the information from the application domain and case study requirement analysis, we adopt these attribute data categorizations. A different application domain implementation may present the need specify related attribute data categorizations. For example, data item, such as, *health_insurance_carrier* and *consent_witness_name* could be characterized as less-sensitive data, whiles *patient_last_name* and *postal_code* could be categorized as medium-sensitive data. Data items, such as, *date_of_birth* and *personal_health_number* could be categorized as highly-sensitive data. Additionally, collected data in healthcare domain may be categorized as biographical information (*i.e.*, *first_name*, *last_name*, *date_of_birth*, and *gender*, amongst others), demographic information (*i.e.*, *home_address*, *postal_code*, *province*), patient consent information (*i.e.*, *witness_biographic_data*, *witness_address*, *dependent_data*, and *allergies*, amongst others), and health insurance information (*i.e.*, *carrier_name*, *policy_number*, and *group_number*).

The healthcare system presents different categories of data collectors (or service providers) who collect disparate data items at separate stages of healthcare delivery. Moreover, we identify data accessors who are designated as individuals (or groups) that request data access and process the collected data. In some instances, data collectors become data accessors due to their work needs and responsibilities. Data accessors may be classified as clinical physicians, nurses, laboratory analysts, and third-party accessors (such as, allied healthcare centres, courthouse, *etc.*), amongst others. To ensure efficient privacy preservation of collected private, sensitive data, different data access levels must be provisioned to these data accessors.

We consider the case study of managing general data provider biographical and demographic data, healthcare data, and next generation sequencing (NGS) genomic meta-data within the broader domain of healthcare system. In this vein, we study procedures and analyse related data within the context of molecular pathology requisition, medical consent management, and patient private data management. Moreover, we analyse measures that offer data providers flexibility in management and monitoring of data privacy preferences. We study measures that enable data accessors to acquire necessary accessibility permissions to process data provider private healthcare data.

Processing massive data volumes demands enhanced operational efficiency, performance, and optimal service delivery for both data and service providers (and third-party data accessors). Within the healthcare domain, where private data management is vital, issues of information security and trust between parties involved in service delivery must be adequately addressed. This motivates the need to secure data access and authenticate users on data management systems. Conversely, the need to protect data provider's data privacy arises with the processing of such private, sensitive data provider healthcare data, such as, cancer-related data [67]. Hence, demand for efficient data privacy merits better privacy-preservation policies and methodologies to address critical service delivery on these data processing platforms.

1.1 Motivation

Managing and protecting private information in data repositories has generated much research interest since the seminal work on *Hippocratic databases* [15]. Recent studies focus on different processing and techniques applicable to various areas, such as, information systems [96], cloud computing [95], and big data [94]. Research on *contextual integrity* (CI) and purpose-based privacy policies in data provider privacy preferences provide results that seem to suggest improved methodologies and models in data privacy management [37, 38, 40, 41, 45].

Barth *et al.* [37] propose a privacy framework based on *contextual integrity* (CI) which outlines the identification of temporal *norms*, the *value* of information, and the *purpose* of data collection, as requirements to ensure integrity for information flow. Their research identifies a logical framework for expressing and reasoning about norms of transmission of private data between data providers, data collectors and data accessors. Subsequent research on purpose-based privacy by Jafari *et al.* [41, 42, 45] argue that workflow models can be used as a framework for purpose-based privacy policies. Their approach assumes how data providers behave and reflects the *purpose* or intent in the collection, storage, processing, and disclosure of the data. They formulate and design a privacy protection methodology to support such behavioural activities. Despite the results gained from contextual integrity and purpose-based privacy, the problem of a practical implementation for protecting data provider's privacy preferences for various application domains is still a challenge. Moreover, a privacy-preservation approach for managing data provider's private, sensitive data that adopts an immutable, tamper-resistant data platform to process data provider's privacy preferences on large data volumes is yet to be devised.

The approaches studied so far do not adopt an ontology framework to model data provider preferences and privacy semantics in their data protection policies. Moreover, current privacy ontologies provide insufficient semantics to support the complex nature of data provider privacy preferences. The ontologies lack expressive language and semantics to describe data provider preferences [29, 30]. These prior approaches based on contextual integrity and purpose-based privacy are not easily applied to real-world scenarios and are not readily applied in diverse application domains, such as, healthcare and genomic meta-data management.

1.2 Problem Statement

Previous privacy-preservation methodologies address privacy for data permanently stored in repositories – sometimes called "data-at-rest." Data provider privacy preferences are rarely considered (especially, as the data moves to another data repository or third-party accessor).

We address the problem statement from two main areas. First, with the persistence of these static data in data repositories, data providers or data collectors infrequently or minimally respond to changes in data provider privacy preferences. The addition of new data records or expectation of new privacy preferences on new and/or existing data poses a major challenge in handling data provider privacy preferences [10]. This opens up a situation where data providers do not have much access and control over their privacy preferences. Zakerzadeh *et al.* [23], Iwuchukwu and Naughton [24], and LeFevre *et al.* [25] independently study data privacy management for large data sets, but their approaches do not consider dynamic contextualized data provider's privacy preferences.

Second, one critical challenge in current data privacy management approaches is the inability of data providers to flexibly update or change their existing privacy preferences when desired, or when data usage changes. As a result, the inability to track and/or control privacy violations by service providers or third-party accessors continues to remain a problem.

1.3 Research Assumptions and Limitations

Assumptions made to frame this research are:

(a) The implementation uses a private blockchain only accessible to the data providers and the data collectors/accessors (*i.e.*, service providers and/or third-party accessors, where necessary) through a system framework.

(b) Encryption protocol is used for all transmissions including queries and results between user interfaces, the blockchain platform, and the relational database repository. The encrypted data communication medium sufficiently reduces (or best case prevents) any form of attack or data loss as data moves from one system component to another.

(c) The relational database is secure from non-privacy related attacks.

(d) Communication through the Internet is secure, reliable, and appropriately encrypted.

1.4 Research Contributions

We propose a Light-weight Privacy Infrastructure (LPI) to preserve data provider privacy and protect sensitive data offered by them. This approach addresses a practical implementation to protect data provider's private, sensitive data. We classify the methodology approach as ”light-weight” to illustrate a privacy model and privacy preservation infrastructure over a relational database using an external access control authentication wrapper. A heavy-weight approach for privacy preservation needs to alter the database engine of the relational database so as to offer privacy preservation for private and sensitive data values stored in the database. This alteration process of the database engine will attempt to address privacy-aware procedures of query parsing, query evaluation, and query execution as part of overall data processing and/or data retrieval in the database engine.

The light-weight model/infrastructure builds a data processing wrapper for query processing, privacy preferences access control, authentication, and authorisation over the relational database. This access control authentication medium is implemented using a blockchain platform. The blockchain data platform is chosen because it offers an immutable, tamper-resistant platform for data storage and access. With this light-weight wrapper approach, every query processing and data retrieval activity is efficiently authenticated for each data accessor.

The Light-weight Privacy Infrastructure (LPI) offers control on the access and usage of private, sensitive data by service providers, and/or permitted third-party data accessors. The approach offers an efficient platform for managing data provider's personal data (such as, healthcare data), through controlled procedures for data access authorizations by both data provider and data collector. The privacy infrastructure delivers a better application framework and implementation to different domains. Additionally, the model platform presents an effective medium where, on one hand, data providers manage and control access to their private data. On the other hand, it offers data providers the flexibility to change privacy preferences on their private data when desired.

The technical contributions are summarized as follows:

(a) In Chapter 3, we introduce a privacy framework where data providers have ultimate access and control of their privacy preferences, and every change request is validated by both data providers and data collectors (or service providers). This is achieved through a integrated privacy tuple generated from the tight-coupling of data elements.

(b) We introduce an efficient procedure to tightly couple attribute data, data provider privacy preferences, and data accessor data elements into a privacy tuple in Chapter 3. The tightly coupled privacy tuple is transmitted through an encrypted application transmission medium and stored in an immutable data platform, such as, blockchain.

(c) In Chapter 3, we introduce an efficient coupling between relational database management system and private blockchains platform; as part of an encrypted, secured data processing platform for data accessors is presented [55, 57]. We also discuss the implementation details of the system architecture in Chapter 4. We establish that the blockchain is feasible to use as an authentication platform for all forms of data access and changes to data provider's privacy preferences.

(d) Moreover, Chapter 3 introduces a privacy model framework that uses a formulated, dynamic contextualized privacy ontology as foundational model with crafted design constructs of semantics and predicates. Additionally, in Chapter 4 we describe the ontology's key characteristics of *purpose*, context, data provider privacy preferences, user roles, contextual norms, transmission medium, and other privacy-related factors that may be relevant in a data provider's privacy policy.

(e) Finally, in Chapter 4, we discuss the implementation of the privacy model framework as applied to patient healthcare data and genomic data. We use an Alberta biotechnology company (*i.e.,* Tunote Oncogenomics) which is interested in providing Alberta cancer patients with access to commercial oncogenomic services, as a case study. This implementation validates overall research propositions. We discuss detailed implementation overview in the next Section 1.5. An evaluation analysis of implementation query processing for privacy-aware queries and the privacy cost implications are discussed in Chapter 5.

1.5 Application Domain Implementation

We implement the privacy model framework using Tunote Oncogenomics healthcare as a case study. Tunote Oncogenomics is an Alberta biotechnology company interested in providing Alberta cancer patients with access to affordable commercial oncogenomic services. Their main purpose is to provide affordable access for all Alberta cancer patients using local expertise in oncogenomics. Moreover, they offer services to assist in the planning and execution of cancer treatments, from diagnosis through every step along a patient's healthcare journey.

Tunote Oncogenomics utilizes a combination of distributed laboratory workflows and a state-of-the art data management and informatics system. Tunote adopts locally outsourced laboratory processes to develop an affordable approach to the traditionally high-overhead whole-genome sequencing workflow. To deliver on the need for peak-performance analytics, Tunote leverages local technical expertise in bioinformatics, including affiliates in some of the country's leading local research cores. To provide patients with optimal and guaranteed patient data control and ownership, Tunote is developing a hybrid blockchain-informed data storage and transaction architecture.

Tunote Oncogenomics believes that patients (*i.e.,* data providers) should have full control over their personal healthcare and genomic data. The proposition is that all Tunote-based healthcare and genomic data are generated in a direct-to-patient framework, using an advanced blockchain-based authorization and transaction control system. Furthermore, all Tunote patient clients have full and unhindered ownership and control of their healthcare and genomic data.

A Tunote Distributed Ledger (TDL) is created from the onset of healthcare delivery and administration. A client's TDL account is used to store identifiable personal data, healthcare data, demographic data, payment information, and data transaction data. Leveraging a blockchain architecture, the TDL provides an immutable record of healthcare and genomic data ownership and all associated transactions. This data management approach reduces Tunote and its partners of the liability of data privacy and security; such that data access is allowed with the express permission and acknowledgement of the patient (*i.e.,* data provider).

1.6 Research Significance

Management of large data sets has experienced varied challenges over the years with inherent data privacy and security problems; and issues relating to data ownership and control. For example, in the area of healthcare and genomic data management, data collectors from both public and private operate using centralized data transaction architectures [69]. Current system data architectures run risks of data provider privacy breaches, data loss or corruption, and unauthorized data access (and use by third-parties). This is because data are stored and managed exclusively by data collectors, and data are housed in a way that data providers have very little knowledge or control about how their data are accessed or used. Moreover, data providers are obligated to data collector (*i.e.*, service provider) data ownership privacy policies, and are exposed to having their data used without their knowledge or consent [103]. Our proposition of a privacy tuple (of tightly-coupled data elements) and coupled data platform offers an improved privacy infrastructure where data providers contribute (and agree to) privacy preferences specified on their private data. Additionally, data providers are offered much control and access to data and query processing procedures in the privacy-controlled and authenticated coupled data platform.

Recent General Data Protection Regulation (GDPR) [104] in the European Union and emerging regulations in both Canada and the United States of America require that personal and sensitive data be managed in a highly protective way. Our research offers a data platform where data providers have greater access and control to their privacy preferences over the underlying private data stored in databases. The privacy infrastructure enables data providers to verify data access and accuracy, revoke data access as required/desired, delete data when desired, and ensure availability of verifiable access audit, to ensure that systems are in regulatory compliance.

We design a tighly-coupled data platform of blockchains, relational database, and genomic data store for efficient data storage and authenticated privacy-aware query processing. Blockchain technologies provide an immutable means of recording information relating to data provider preferences that includes *"how"*, *"by whom"*, and *"when"* their data might be legitimately accessed for a given purpose [51, 53]. When data provider privacy preferences are stored on a blockchain platform, access rights can be verified against the privacy preferences; which guarantees compliance and transparency by all parties (of data provider, data collector, and data accessor). The relational database stores content data and provides better data transaction throughput which are not readily available in the blockchain. Moreover, the genomic data store - which manages large genomic data sets - provides an optimum platform for processing NGS data types. The incorporation of all these platforms provide sufficient data protection of data provider's private, sensitive data stored in relational database and/or genomic data store. Moreover, this architecture offers data providers control over authorized access to their private data in the data repositories.

The proposed privacy model presents a novel approach where each data provider data item is tightly-coupled or bound to privacy preferences and data accessor data items. This makes query processing for each data item strongly tied to the privacy preferences and data accessor. Additionally, the methodology approach employs a blockchain platform as an access control query processing medium. The privacy methodology is implemented using a case study of healthcare data. The experimental implementation and evaluation results offers relevant metrics on privacy overhead cost, query recall (*i.e.*, visibility privacy), and query precision (*i.e.*, granularity privacy) for processing queries and data retrieval on privacy-aware relational databases.

1.7 Dissertation Outline

The dissertation is organized, as follows. In Chapter 2, we discuss background studies regarding data privacy models and methodologies, privacy ontologies, purpose-based privacy models, blockchains and data privacy, and database coupling approaches of blockchains and relational databases. Chapter 3 addresses and describes the proposed privacy methodology of Light-weight Privacy Infrastructure (LPI). This chapter discusses modelling constructs of formulated contextualized privacy ontology and tight-coupling of data elements. Additionally, proposed tight-coupling approach and implementation of the blockchains and relational DBMS platforms are discussed.

In Chapter 4, we discuss system design, development, and implementation of the research methodology of light-weight privacy infrastructure. We discuss implementation procedures for privacy ontology, tight-coupling of data elements, and tight-coupling of relational database, blockchain platform, and genome data store. Chapter 5 discusses overview of the security or adversarial model on the system. We evaluate procedures on the different aspects of system implementation for the proposed research methodology. Additionally, we discuss system adversarial model, system security assessment on all system components, and privacy model attacks. We discuss a comparison to other methodology approaches and address qualitative analyses based on some defined criteria. Finally, we discuss and analyse query transaction evaluations, and query processing and response times. In Chapter 6, we conclude by summarizing research contributions, and consider vital areas for application in academia and industry. We also address some open issues and likely areas of future work.

Chapter 2

Background Research

There have been several studies into data privacy over the years, and recently in purpose-based privacy, usable privacy, and context-based privacy-preservation. The results of these prior research bring into view novel ideas, functional methodologies and models, and insights into data privacy. Other investigations study cases where data provider private and sensitive data will be efficiently managed and protected in the near future, despite sophisticated information processing and data analytics [94, 101, 102].

In this chapter, we discuss previous research studies. We highlight major contributions and address some of their drawbacks. We discuss privacy-preservation models and methodologies in Section 2.1, and address privacy ontology modelling, and contextual privacy in Section 2.2. In Section 2.3, we discuss blockchains platform and its relationship to information security and encryption. Section 2.4 discusses research work on coupling blockchains and relational databases. We discuss related research works on the application of blockchains and privacy-preservation in healthcare and genomic data in Section 2.5. In Section 2.6, we summarize research contributions on research works within the area of applying blockchains in data privacy preservation. We also examine research gaps arising from these background studies and pertinent problems that must be addressed.

2.1 Privacy-Preservation Models and Methodologies

Several data privacy-preservation approaches have been addressed and proposed by researchers in the data privacy domain. This section addresses different forms of privacy-preservation models and methodologies for data either stored in repositories or being published in the form of micro-data. First, we discuss techniques and algorithms for data protection. Second, we discuss various models and privacy policies that formalize these methodologies. Moreover, we address various approaches of privacy-preserving data publishing.

2.1.1 Privacy Models and Methodologies

Privacy preservation approaches provide useful outcomes and/or results in policies, techniques, algorithms, and useful knowledge in securing data provider's private data in data repositories. The study of these methodologies underline merits and foundations for reliable and robust data privacy.

The development of privacy-preserving anonymization methods prevents disclosure of private, sensitive information of persons in published data. One early study on privacy-preserving anonymization methods is provided by Sweeney's *k-anonymity* [1]. There has been a succession of other approaches that seek to address the weaknesses in earlier methods. These models and methodologies are presented in different anonymization techniques (such as, generalization and suppression [1, 2, 3], randomization or data perturbation [4, 5, 6, 7, 8], and data swapping [9]).

Privacy-preserving data mining methodologies are techniques and algorithms that are employed to protect private, sensitive data while drawing relevant knowledge from underlying data repositories. Aggarwal and Yu [10] provide a foundational study on privacy-preserving data mining. The authors propose an approach that does not require recent problem-specific algorithms, but maps and condenses groups of original data sets into new anonymized set. The approach uses statistical information to create anonymized data which has similar statistical characteristics. Subsequent investigations of privacy-preserving data mining discuss critical areas in handling broad data distributions. Additionally, other studies analyse the trade-off between sharing information and privacy-preservation, and generate valuable methodologies and results [11, 12, 13, 14].

2.1.2 Privacy Models and Data Management

The study of better privacy-preserving polices, and privacy-aware database models have garnered interest in the research community since the seminal work on *Hippocratic databases* [15]. Agrawal *et al.* [15] outline and address several principles for protecting data stored in database management systems. Moreover, the authors propose an architectural design framework (in terms of privacy meta-data schema, privacy policy schema, privacy authorization schema, *etc.*). This design models a privacy-aware data repository where measures for data collection, query processing, and data retention, amongst others, are outlined. Further studies by Barker *et al.* [16] seek to outline a privacy taxonomy for the collection, sharing, and disclosure of private data stored in repositories. The authors develop a privacy taxonomy to define parameters for a privacy policy while protecting the data provider's personalized information.

Barker's [18] conceptual approach to privacy protection addresses the need to identify trade-off between maximizing data utility and optimum privacy in data repositories. His study proposes a conceptual approach where data provider approves or consents to privacy policies or changes in existing policies with the service provider. Moreover, Kwakye and Barker [19] propose a privacy-preserving methodology in relation to integrating multidimensional data models. This approach is based on Barker *et al.* [16] privacy policy taxonomy and adaptation of Barker's proposed conceptual model [18]. Other studies on privacy-preserving policies and models focus on policy visualizations, and ability to represent different use cases on a number of real-world scenarios [17].

2.1.3 Privacy-Preserving Data Publishing

In the broad overview of managing and publishing data that contains private, sensitive data provider data, a number of studies have surveyed and presented reviews on efficient data publishing approaches [20, 21, 22]. Further investigations focus on large-scale data sets and these address insightful results in processing of large data volumes [23, 24, 25]. Additionally, recent investigations in cloud services have gained much interest in managing sensitive data, preserving data provider's privacy, security, and identities, and data publishing [26, 27, 28].

2.2 Privacy Ontology Modelling, Contextual and Purpose-Based Privacy Models

The need to address data privacy from different contextual viewpoints and purposes generates better ways of managing, analysing, and retrieving data provider private data. This section discusses research studies on privacy ontologies, and how these ontologies are used as foundations in analysing contextualized data provider's privacy and purpose-based privacy policies.

2.2.1 Privacy Ontologies and Context-based Ontologies

Privacy ontologies form the structure and model for most context-based privacy policy formulation. Ontologies formalize diverse preferences, unique attributes, and distinct rules for managing, processing, and modelling privacy policies. Additionally, ontologies help in instantiating contextualized privacy policies for different individuals or scenarios.

Several research efforts have investigated ontologies, and these studies define and describe how ontologies are used in modelling real-world scenarios and application domains. Additionally, these studies discuss how predicates regarding these ontologies state the purpose of concepts (*things*) found in the real-world scenario. Predicates describe behaviour of the concepts in the instance under review. The results gained from these studies have been vital in drawing insights from ontologies and their predicates, especially in the framework of controlling privacy in context-aware systems [29]. Bawany and Shaikh [30] model a data privacy ontology for various ubiquitous computing platforms. Their research work defines ontology semantic constructs (of data, producer, and consumer) and predicates for privacy preservation of data throughout its life cycle.

Other studies propose ontology modelling to protect private, sensitive data for various application domains, such as, healthcare delivery, supply chain services, and financial services, amongst others. Some studies connect domain-specific ontology modelling to associated processing platforms, such as, mobile devices, and Point-of-Contact (PoC) user interfaces [31, 32]. Accordingly, the study of context-aware and rule-based ontologies offer many results that address such unique scenarios. Moreover, the outputs of context-aware and rule-based ontologies are relevant in defining the semantics and predicates of privacy ontology modelling [34, 35, 36].

2.2.2 Contextual Integrity and Purpose-based Privacy Models

Modelling ontologies to efficiently adapt to different application domains and unique data processing situations demands incorporation of context-specific predicates. The research paradigm of contextual integrity (formalized by Nissenbaum *et al.* [37, 38]), focus on context-specific viewpoints, such as, *role*, *norms*, *value*, and *transmission principle*, among others. These viewpoints provide further insights and perspectives to modelling of privacy ontologies and frameworks. The proposition by Nissenbaum, combined with research by Krupa and Vercouter [39] and Ajam *et al.* [40], seek to outline different context-based privacy norms adopted in virtual communities and data processing paradigms. These contextual norms are used in addressing procedures for managing and controlling access to private, sensitive data in such situations.

Jafari *et al.* [41, 42, 45] investigate purpose-based privacy policies and modelling. Their studies address different insights and motivations behind *purposes* for application domains. Moreover, the authors propose a formal definition for *purpose* (being a condition of reason among inter-related actions) and its constraints. The authors develop a framework model to express and enforce such purposes in privacy policies. Their formal framework further discusses the properties of purpose and shows how purpose-based constraints can be connected to more general access control policies. Finally, the authors specifically address enforcement of purpose within the framework of purpose-based privacy using workflow-based information systems. Other contributions in using purpose lattices and application of purpose in graph modelling are outlined by Kumar and Shyamasundar [43], Tian *et al.* [44], and van Staden and Olivier [46].

2.2.3 Usable Privacy

One aspect in modelling and application of ontologies for privacy data processing is the ability to make it usable and functional for different technological platforms. This functional aspect must meet unique user-specific data processing needs. Bolchini *et al.* [47], Gerber *et al.* [48], Birge [49], and Brodie *et al.* [50] independently address different aspects of contextualizing privacy for user interfaces. Their studies outline improved interface design and enhancements that come with achieving usable privacy (with extensions to security). Moreover, their studies seek to investigate trade-offs between usability and privacy (and security) in terms of adopting efficient data provider interactions and interfaces.

2.3 Blockchains and Data Privacy

The study of blockchains to manage secured data and transactions has gained much attention recently. This section reviews studies on decentralized, distributed blockchains, private information processing on these blockchains, and their encryption approaches. Additionally, processing of privacy-related data collected from different application domains and platforms, and their relationship to blockchains are considered.

2.3.1 Blockchains Applications to Data Privacy

Blockchains (in comparison to the conventional relational database) offer a secured, tamper-resistant, and immutable framework for data processing. These platforms provide a better "database" framework for processing private, sensitive data provider's data. Since the earlier introduction of blockchains as underlying data base for Bitcoin cryptocurrency platform by Nakamoto [51], there have been varied proposed uses for blockchains. Romano and Schmid [52] discuss the adoption of the blockchain technology platform in diverse data processing systems. In their study, the authors discuss the current and potential application of blockchains for different domains. The authors discuss smart property management, where private information, such as, copyrights and patents of data owners are protected. Additionally, the research discusses the use of blockchains for smart contracts and other related fields of eScience, healthcare, and financial services. The highlight of their study address the use of blockchains to protect private data scientific outcomes (as in eScience), patient medical records and access to genetic data (as in healthcare), and financial transactions and credit information (as in financial services).

A number of studies have investigated a background framework and design of blockchains data platform. These studies outline unique benefits of offering data verifiability, identity, immutability, and decentralization [58, 59]; necessary for protecting data provider's private data and the transactions that are performed on them. Zyskind *et al.* [55] and Halpin and Piekarska [57] independently investigate aspects of blockchains security, encryption, and extensions to protecting personal data within the framework of decentralized processing. In these works, the research identifies ways in which users (data providers) can own and control their data whiles having service providers offering them personalized services [55]. The research addresses how to protect private data using blockhains as an access-control moderator with an off-blockchain storage solution. Moreover, other research works discusses a variety of blockchain primitives in the form of centralized e-cash with privacy properties that enforce privacy between senders and recipients of financial transfers. Other approaches offer addition of some data anonymization forms to existing blockchain techniques [57]. All these works address methods in protecting privacy and identities of data providers in domains where blockchains are applied.

2.3.2 Blockchain Methodologies and Healthcare Data Privacy

The use of blockchains for managing and storing healthcare data has recently gained much consideration. This approach has become necessary due to the sensitive nature of storage, management, and disclosure of healthcare data to data accessors. Moreover, healthcare data in recent times have attracted strong reforms in terms of data protection, privacy, and security [60]. Tian *et al.* [61] study an approach of medical data management using blockchains to preserve privacy. In their work, the authors adopt the methodology of storing encrypted medical data in a blockchain using a shared key; which is accessible by all authenticated parties for medical diagnosis and treatment. Similar approach of storing medical data in blockchains is addressed by Cao *et al.* [62]. The authors propose a reward mechanism for medical data management which maximizes data processing benefits of sharing medical data to a medical data producer and/or medical data miner. Their proposal outlines measures to reduce the risk of patient's private medical information getting leaked to unauthorised persons; whiles offering to achieve a reward on the shared medical data. In their approach, the research addresses ways in which unauthorised access to the patient's private information is minimally reduced. The unauthorized data access could come from different persons (within the entire medical healthcare management system) who might have access or even curious adversaries, such as, hackers.

Different authors study and propose methodologies for preserving privacy in healthcare or medical data using blockchains and encryption methods [63, 64, 65]. Huang *et al.* [63], in their recent research address the privacy needs of patients during the phase where information regarding medical data is transmitted between several entities. Additionally, the authors address issues including the unsatisfactory transaction processing capacity and implementation efficiency caused by the current blockchain-based medical system. Their research presents a blockchain-based privacy-preserving scheme, which realizes secure sharing of medical data between several entities involved such as, patients, research institutions and semi-trusted cloud servers. Similar methodology of a blockchain-based privacy preserving platform for healthcare data is proposed by Al Omar *et al.* [64]. Their study proposes a storage system to attain privacy where preservation of data providers' (*i.e.*, patients) identities and protection of their sensitive healthcare data are ensured using cryptographic functions. In their work, the authors address accountability, integrity, security, and privacy of healthcare data as critical functions that must be adhered to. Their work presents authorizations where data sharing is predominantly managed by the patient, and the encrypted data is well-protected. Additionally, in the case of system attack the stolen data makes no sense to the attackers. This claim by the study is based on the fact that only encrypted data are stored by the parties in the system. Moreover, the blockchain platform maintains data integrity on both parties in the system. Liu *et al.* [65] investigate a methodology where electronic medical records are stored securely in cloud database servers and the indexes are reserved in a tamper-proof consortium blockchain. In their approach, secure data sharing is accomplished automatically according to predefined patient access permissions through blockchain smart contracts.

In line with security and privacy implications for adopting blockchains as a data platform for healthcare or medical data, we review merits of genomic data storage using blockchains. Carlini *et al.* [66], Shabani [67] and Ma *et al.* [68] study the need to adopt blockchains technology in protecting and improving the wide availability of low-cost genomic data services. Carlini *et al.* [66] investigate the adoption of blockchain technology as the foundation of an ecosystem for acquiring and sharing of genomic data. Their approach presents the *Genesy* model of a blockchain platform, that transcribes genomic data. The project adopts the Hyperledger Fabric blockchain platform to notarize genomic data and prevent their unauthorized use; while making the data accessible for transactions between different parties.

Shabani [67] studies blockchain-based platforms as emerging solutions for technical and governance challenges associated with genomic data sharing. The author's investigation highlights the need for effective ways to enforce data access agreements and data ownership on blockchain platforms. The author identifies several blockchain-based projects and describes their basic features and core functionalities. The key outcome of the study addresses the emerging need of blockchains to provide a platform for maintaining data provider's personal genomic- and health-related data, and methodologies for data sharing and disclosure. Additionally, the study outlines blockchain's merits and capabilities, such as, participatory access control, automating access agreements and consent, and data ownership and DNA coins. Ma *et al.* [68], on the other hand, study the cross-site genomic data sharing system mechanism where a secure and transparent access control audit ensures accountability. Their research identifies several propositions for blockchain-based access audit and proposes a blockchain-based log system which is a light-weight module over existing blockchain platforms. Their adopted and designed system module supports efficient range queries and complex queries that contain multiple predicates.

As part of adopting blockchain technology as an underlying data repository for genomic data, one critical area that must be addressed is handling encrypted queries. Additionally, there is the need to consider management of query results that are transmitted in-between the blockchains and other components of the genomic data storage platform. Several studies have focused on controlling data access, encapsulation mechanisms and data communication techniques. The research results serve as foundational knowledge to determine which approach to adopt in handling encrypted queries and results [78].

2.4 Coupling Blockchains and Relational Databases

The approach of coupling blockchains to a relational database management system (DBMS) is a new problem that is directed at obtaining merits from both data platforms for secured data processing. Few studies regarding how to efficiently link blockchains and DBMSs or simulate the coupling of both platforms exist. We consider some research work that attempt this approach in this section. Other studies on approaches of query processing, scalability, and throughput on blockchain systems are reviewed, as well.

2.4.1 Blockchains and Relational Databases

The approach of coupling blockchains and relational database management systems arise because of the unique properties of blockchains and relational databases, respectively. Blockchains present a platform for an immutable, tamper-resistant, and encryption-enforced medium for distributed authentication of data transactions by consenting parties. This authentication approach is designed in a non-trusting relationship. Conversely, traditional relational databases provide an efficient platform to store and manage large data volumes and process expected high transaction throughput.

Very little research has addressed the benefits of combining blockchains and relational database management systems. Mohan [81] in his introductory tutorial on blockchains and databases discusses ongoing projects combining the merits of both database platforms. The highlights of the tutorial discuss the initial studies on different open-source projects that involve platforms, such as, Hyperledger Fabric, Enterprise Ethereum, and R3 Corda. The tutorial further discusses the architecture, components, and algorithms; and their related open issues. Meunier [82], Dinh *et al.* [83], and Seng [84] independently investigate database capabilities that a blockchain platform can exhibit, and how these capabilities are adopted in traditional DBMSs. Meunier [82] addresses blockchains in the form of distributed database management systems (DDBMS) that leverage on cryptographic functions to provide a decentralized multi-version and concurrent control mechanism. This helps to maintain consensus about the state of data stored in each distributed database. Dinh *et al.* [83] study the evaluation work of designing a private blockchain platfrom. Their research work addresses key components of using a private blockchain as a "special case" of a database platform. The work discusses four main layers to the private blockchain; namely, consensus, data model, execution layer, and application layer. The research work by Seng [84] discusses the concept of a blockchain as a database without a central data administration. The work does not present technical analysis or design architecture of a blockchain, but identifies key data security guarantees of confidentiality and reliability; as advantages of the blockchain platform in comparison to the conventional relational databases. Karafiloski and Mishev [85] discuss the adoption of blockchain for big data and propose solutions to the challenges of big data management. Their study address how these challenges are managed using a blockchain technology.

Notable research combining the merits of both blockchains and big data database platforms is addressed by McConaghy *et al.* [86]. In their studies, the authors simulate coupling distributed, decentralized blockchains and big data database management systems. Their approach embeds the properties of blockchains, such as, asset trust and autonomy, verifiability, tamper-resistance, and decentralized data processing into data processes of big data DBMS. This approach offers storage of large data volumes, efficient query processing, data scalability, and high transaction throughput.

2.4.2 Blockchains Query Processing, Data Scalability, and Transaction Throughput

With the emergence of blockchains and the need for efficient transaction data processing, Li *et al.* [87], Xu *et al.* [88], and Dinh *et al.* [89] independently investigate query semantics, transaction management, and query framework protocols of blockchains. Tai *et al.* [90] propose a novel form of blockchains transaction processing, known as SALT (Sequential, Agreed-on, Ledgered, and Tamper-resistant). This transaction processing supports symmetric and time-consensual data processing. Additionally, the data transaction processing is free from a central administrative control. Kiayias and Panagiotakos [91] analyse the trade-off advantages on transaction data processing speed and data security in blockchain protocols. In their study, the authors investigate a wide range of viewpoints regarding processing speed and security of transaction queries over a blockchain protocol. These viewpoints are demonstrated in the data storage approach in blockchains and the data protection that blockchains offer against all forms of data tampering.

An important challenge in blockchain data processing is the assessment of scalability and transaction throughput for significant volume of stored data. Ren *et al.* [92] and Xu *et al.* [93] study and analyse improved approaches for blockchain consensus mechanism to achieve high throughput during real-time transaction processing (such as, e-commerce). The authors in Ren *et al.* [92] address current issues (such as, scalability of Byzantine fault tolerance (BFT), increasing throughput of proof-of-work, leader selection, and sharding) as key provisions for successful deployment of blockhain platforms with large data volumes. Their work introduces a permissioned blockchain-based distributed ledger that consists of four layers. Moreover, their work outlines an overview of how BFT is handled, reliability guarantees of side-chain transactions, and how some rational nodes are used to optimize storage and message transmission. On the other hand, Xu *et al.* [93] study a blockchain consensus mechanism that is mainly focuses on an e-commerce platform. The case study of e-commerce is chosen because of the expected real-time mode of transactions and high-throughput data processing. The research design offers a two-layer peer blockchain network with a dynamic verification network, and a negotiated consensus algorithm. The key advantage of the consensus algorithm is addressed in the measure that it does not rely on computing power, and ensures credible block verification.

2.5 Related Research Works

We study and address some related research and developer projects on privacy-preservation of data provider's private, sensitive data. The discussion presents insights into how these methodology approaches protect data provider's private data and preserve data provider privacy and identities. Moreover, we address how these applications (arising from the research works) manage domain-specific healthcare data, and more specifically genomic data. We further identify novel propositions from these research work and discuss their contributions.

2.5.1 Medicalchain – Blockchain-based Healthcare Data Management

Medicalchain is a decentralised platform that enables secure, fast, and transparent exchange and usage of Electronic Health Records (EHR) medical data [70]. It uses blockchain technology to create a user-focused electronic health record while maintaining a single true version of the user's data. Each interaction with the medical data is auditable, transparent, and secure on Medicalchain's distributed ledger. These interactions are recorded as transactions on the distributed ledger and the patient's privacy is always protected. To ensure privacy, health records are encrypted using symmetric key cryptography. Medicalchain enables users to give conditional access to different healthcare agents, such as, doctors, hospitals, laboratories, pharmacists, and insurers to interact as required.

Medicalchain is built using a dual blockchain structure. The first blockchain controls access to health records and is built using Hyperledger Fabric. The permission-based Hyperledger Fabric architecture allows varying access levels for users to control which data accessors can view their records, how much they see, and for what length of time. The second blockchain is powered by an ERC20 token on Ethereum and underlies all the applications and services for the platform. Using blockchain technology, smart contracts, and cryptocurrency functionality, Medicalchain provides the infrastructure for digital health applications and services to be built. These applications and services are seamlessly powered by user's healthcare data.

Medicalchain's platform enables developer applications that complement and improve the user experience to be built. Medicalchain offers an API that will permit third parties to obtain and interact with EHRs with the user's permission. All endpoints available in the UI are available to developers. Users can leverage their medical data to control several applications and services. There is a marketplace that enables Medicalchain users to negotiate commercial terms with third parties for alternative uses or applications of their personal health data. For example, putting forward their data to be used in medical research.

2.5.2 Blockchain-based Genomic Data Processing Platforms

We identified some other research projects that use blockchain within the context of genomic data storage. This work focus on patient data privacy protection in relation to genomic data, with others adopting a cloud storage for the genomic data. These platforms use blockchains for some aspect of their network ecosystem in storing genomic data file references [73]. Gürsoy *et al.* [71] study and develop a novel method of storing reference-aligned reads on-chain in a private blockchain network. The authors develop tools that provide open-source blockchain-based storage and access for advanced genomic analyses, such as, variant calling [71]. Other developer projects include CrypDist [72], Zenome [74, 75], Nebula Genomics [76], and Cancer Gene Trust [77].

CrypDist uses a custom blockchain to store links to genomic data files (such as reference-aligned BAM files), which are stored in Amazon Web Services (AWS) data buckets [72]. Zenome uses Ethereum smart contracts to facilitate transactions of genomic data but appears to store the data off-chain in a distributed file storage system [73, 74, 75]. Nebula Genomics uses Ethereum smart contracts to facilitate communication between nodes and Blockstack file system; thus, providing safe data storage and access permissions. The Blockstack infrastructure stores data off-chain either on a local drive or in the cloud (such as, Digital Ocean, Amazon S3, and Dropbox) [76]. Cancer Gene Trust stores data off-chain in InterPlanetary File System and raw genomics data locally. It uses Ethereum smart contracts to store references to the data files [73, 77].

2.5.3 Blockchain-based Privacy-Preserving Methodology Approaches

Some prior research work that use blockchain platform to protect data provider's data and preserve privacy and identities of these data providers have been addressed in the literature. These approaches adopt varied methodologies in their privacy modelling and architecture. The research work of Peng *et al.* [136] and

Azaria *et al.* [137] present overview of blockchain-based data management systems. Their methodologies address the use of blockchains to establish trust between different collaborative databases, and using a blockchain to manage data access and permissions, respectively. Notable works by Daidone *et al.* [129], Fernandez *et al.* [130], and Griggs *et al.* [131] present important methodologies and outcomes with regard to the adoption of blockchains to facilitate preservation of data provider's private data. Daidone *et al.* [129] propose a blockchain-based privacy enforcement architecture where users can define how their data are collected and managed, and ascertain how these data are used without relying on a centralized manager. Their methodology adopts a blockchain to perform privacy preferences' compliance checks in a decentralized fashion on IoT devices; whiles ensuring the correctness of the process through smart contracts.

Fernandez *et al.* [130] propose a cloud-IoT architecture, called *Data Bank*, that aims at protecting users' sensitive data by allowing them to control which kind of data is transmitted by their devices. Their architecture consists of several layers of IoT objects, web and mobile applications, and a cloud layer for enforcement and access control. Griggs *et al.* [131] propose a healthcare blockchain system for secure automated remote patient monitoring. In their approach, the research outlines using blockchain-based smart contracts to facilitate secure analysis and management of medical sensors. Their architecture uses a system where the sensors communicate with a smart device that calls smart contracts, and writes records of all events on the blockchain.

Other methodology approaches that use blockchains to enforce privacy on data provider's private data are addressed by Loukil *et al.* [132], Dwivedi *et al.* [133], and Alblooshi *et al.* [135]. Loukil *et al.* [132] propose a privacy-preserving platform for IoT data sharing using a service-oriented approach based on the blockchain technology. Their methodology for privacy preservation offers a generic set of services over applications that take note of users' privacy requirements, such as data purpose, disclosure, and retention. The system modelling consists of five parties: data producer, Ethereum blockchain and smart contract, *patriot* platform, distributed database, and data consumer. The overall system architecture and design present an IoT data sharing platform, while adopting a service-oriented approach that provides several components including the privacy preference matching service and privacy policy compliance service.

Dwivedi *et al.* [133] address a decentralized privacy-preserving healthcare blockchain for IoT platforms. The research outlines use of blockchain platform to provide secure management and analysis of healthcare big data. The approach employs a framework of modified blockchain models suitable for IoT devices that rely on their distributed nature and other additional privacy and security properties (based on advanced cryptographic primitives) of the network. The proposed blockchain model eliminates the concept of ”proof-of-work” (PoW) to make it suitable for computation and data retrieval on IoT devices. Transactions in blockchain are broadcast publicly in the blockchain network and contain additional information about the sender and recipient. The overall model architecture offers IoT privacy and security using a hybrid approach. The hybrid approach combines advantages of private key, public key, blockchain and other lightweight cryptographic primitives to develop a patient-centric access control for electronic medical records.

Alblooshi *et al.* [135] address a blockchain-based ownership management for medical IoT devices. The research work presents a system architecture of Ethereum blockchain network, manufacturer smart contract node, and IoT device smart contract node. Manufacturer smart contract is used by manufacturer in order to deploy IoT device smart contract whenever IoT device is manufactured. The other smart contract gives the IoT device owner the ability to set rules and conditions for access and records modification pertaining to the medical IoT device. The medical IoT device records (from the data provider) are stored on the Ethereum blockchain network. The authors discuss and analyse the security vulnerabilities and defences from their methodology, and address some forms of attacks that can be mitigated.

We discuss three of the methodology approaches (namely; Daidone *et al.* [129], Fernandez *et al.* [130], and Griggs *et al.* [131]) into detail in Chapter 5 (see Section 5.7). These research work present prominent outcomes and technical contributions with regards to privacy preservation using blockchains, in comparison to the others. For each research work, we discuss the overall privacy model and/or infrastructure and core merits that arise from the research. Moreover, we perform a comparative analysis from these research work to our proposed methodology based on some defined criteria in Section 5.7.4. Finally, we outline the areas in which our proposed methodology offers better privacy-preservation solutions.

2.6 Research Gaps and Summary

Previous research has found some methods of protecting data provider's private and sensitive information. Some of these research are comparable to what is proposed in this research. However, a better blockchain-based approach to efficiently manage and control data access, availability, and usage of data provider's private, sensitive data stored in data repositories is still a problem. This is because of: (1) the absence of an expressive privacy model and/or ontology model, (2) lack of an adaptable privacy infrastructure and architectural platform, (3) lack of adoption of an expressive and robust private blockchain platform, and (4) minimal degree of data protection provided by data collectors (*i.e.,* service providers). Moreover, the ability to ensure data integrity and data monitoring to changes to data provider's privacy preferences for data use by data collectors/accessors (and consenting third-party accessors) continue to remain a challenge.

A formulated methodology in which there is tight-coupling of private, sensitive data, with contextualized data provider privacy preferences, and data accessors' profile has not been addressed yet. This coupling approach will offer efficient control in storage, access, sharing, and usage of private, sensitive information of data providers. Furthermore, a technology that couples blockchains and relational database management system is yet to be devised. This coupling approach will enable the storage of data provider's privacy preferences on their private, sensitive information in a completely tamper-resistant, immutable, and untrusted platform.

The next chapter discusses key contributions and implementation approaches for a light-weight privacy framework. We discuss an overview of the formal contextualized (dynamic) privacy ontology. We discuss tight-coupling of attribute data, data provider privacy preferences, and data accessor data elements. We also discuss tight-coupling of relational database, blockchain platform, and genomic data store. Moreover, we discuss the query transaction processing approach on the system. This framework will efficiently manage and control data provider privacy preferences from privacy policies that are agreed upon by both data providers and data collectors (*i.e.*, service providers).

Chapter 3

Light-weight Privacy Infrastructure

Privacy preservation models offer important methodologies in management and processing of data provider's private, sensitive data. The formulation, and subsequent implementation, of these models presents an efficient platform in the collection, storage, access, and query processing of related private data from data providers by service providers (*i.e.*, data collectors). We describe and discuss details of our proposed Light-weight Privacy Infrastructure (LPI). This model offers an effective integration of various system components to provide a platform where data provider's information and transaction data are collected, processed, and managed by data collectors.

The detailed outline of the methodology is as follows: (1) formulation of a formal contextualized and usable privacy ontology; (2) tight-coupling of data elements (attribute data, data accessor profile, and privacy preferences); (3) tight-coupling of decentralized blockchains and a relational database; (4) query processing required for proposed methodology; and (5) encryption approaches on system architecture of (blockchains, relational database, and genome data store). Each of these system components and methodologies work together to provide privacy-preservation in a privacy-aware data platform.

We discuss the detailed implementation of each aspect of the methodology in Chapter 4. We discuss the implementation modelling of the formal contextualized privacy ontology in Section 4.1. The implementation of tight-coupling of data elements is described in Sections 4.2.3 and 4.2.4. Section 4.3 discusses the modelling, design, and development of relational database, and Section 4.4 discusses the detailed design and development of permissioned blockchain platform. Finally, we discuss data platform integration and data communication between user interfaces, relational database, blockchains and genomic data store in Section 4.5.

In this chapter, we address the formulation of formal contextualized privacy ontology in Section 3.1, and discuss the overview of the proposed system methodology in Section 3.2. We discuss tight-coupling of data elements to generate data privacy tuples in Section 3.3 and describe implementation approach of the data elements coupling in Section 3.4. In Section 3.5, we describe the tight-coupling of relational database, blockchains, a genomic data store, and the implementation approach in Section 3.6. Section 3.7 discusses query processing approach on proposed research methodology, and we finally summarize the discussion in Section 3.8.

3.1 Formal Contextualized Privacy Ontology

The overall approach is to formulate a contextualized privacy ontology with unique semantics and properties for privacy preservation. This privacy ontology serves as a basis to formalize the privacy model for data processing. The formulated contextualized privacy ontology becomes the foundation for the design of all entities and predicates in the data privacy-preservation process. The privacy ontology provides a platform for design of the proposed light-weight privacy infrastructure. This is necessary as the privacy ontology offers a modelling framework for procedural activities and semantics for private, sensitive data and querying processing by data collectors/accessors (*i.e.*, service providers) and third-party data accessors.

In the formulated privacy ontology, entity classes, sub-classes, their respective attributes, and associated properties are classified to handle prevalent privacy-related scenarios in privacy-preserving data management. One important aspect that must be considered in a formal contextualized privacy ontology is the need to address semantics in the ontology. To this end, the formulated contextualized privacy ontology model details and defines language structure, semantic knowledge, and formalisms of the entities. These modelling semantics handle privacy and data protection authorizations of related *contextual privacy norms* and *perspectives* (of *when*, *who*, *why*, *where*, and *how*). Moreover, the formulated formal contextualized privacy ontology addresses sensitive data provider data usage and authorizations applicable to various data processing units in within the case study of genome application domain. The units could be identified as medical examination and diagnosis, prescription and medication administration, laboratory and specimen analysis, molecular pathology requisition, and genomic sequencing and analysis, amongst others.

An important aspect of the formulated formal contextualized privacy ontology addresses data provider privacy violations and the obligations that must be applied. With the incorporation of semantic rules in the adopted contextualized privacy ontology, the privacy model addresses different levels of related actions or measures that are activated to control accessibility and usage of sensitive data attributes. The semantic rules for the contextualized privacy ontology define procedural steps and actions that must be taken to correct any form of data provider privacy violation. Moreover, the ontology retroactively outlines future procedural actions for data protection.

In Section 4.1, we discuss the overall implementation of the formulated contextualized privacy ontology. We discuss abstraction ontology modelling (see Section 4.1.1), detailed ontology formulation and implementation overview (see Section 4.1.2), ontology maintenance update, and application domain adaptability (see Section 4.1.3) of the formal contextualized privacy ontology.

3.2 System Methodology Overview

This section discusses and summarizes the methodology for protecting data provider's private, sensitive data in relational databases using privacy preferences stored in the blockchains. We discuss the details of each implementation component and the data transmission between system components in Section 4. The overall procedure for the methodology involves the following steps: data elements collection, binding data elements into a privacy tuple, applying a hash function to the privacy tuple, storing the hashed privacy tuple on the blockchains, storing data provider data and transactional data in the relational database, and executing queries against privacy-aware stored data with reference to privacy tuples. In terms of data elements collection, the attribute data element, data provider's privacy preferences data element, and data accessor profile data element are collected from system user interfaces.

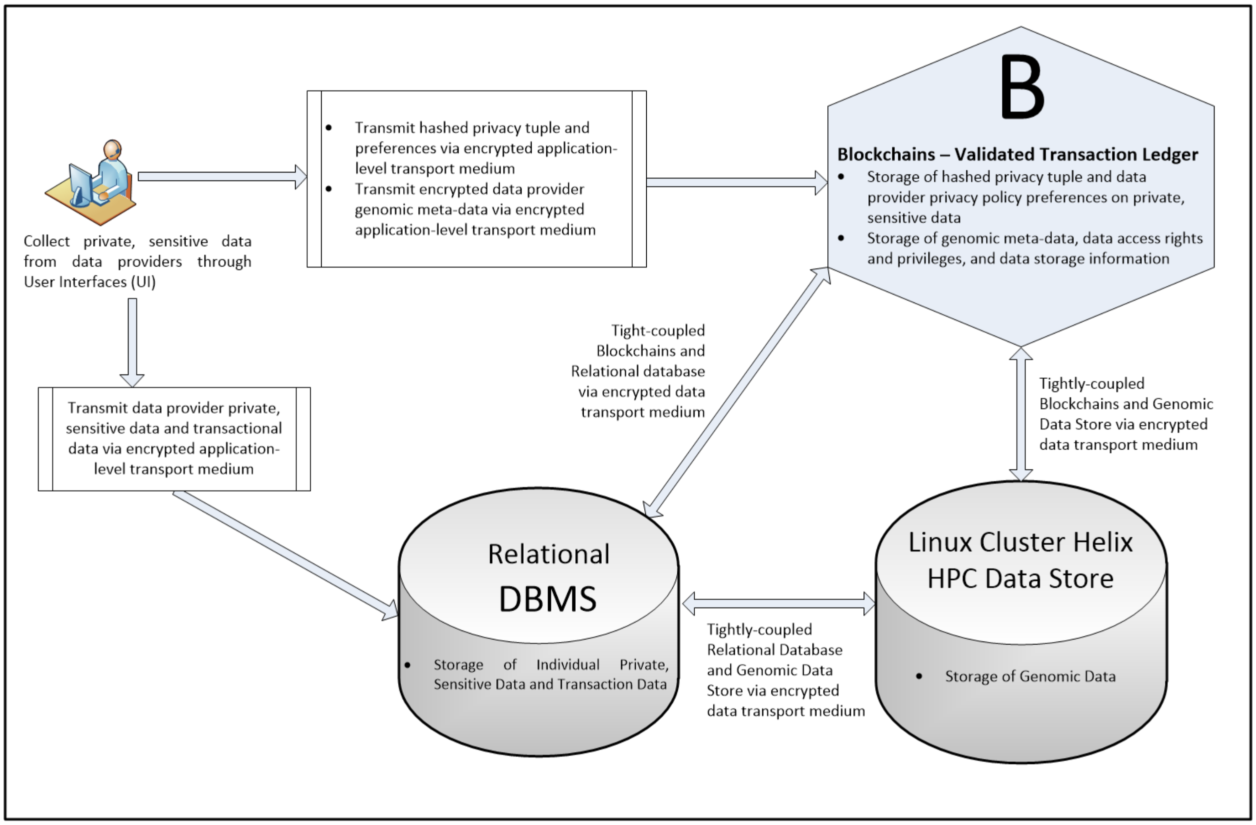
First, privacy-related personal data values are collected from the data provider through Point-of-Contact user interfaces. Devices designated as Point-of-Contact interfaces for data collection are portable mobile devices and personal computing platforms. The privacy-related data values may come from biographical, demographic, financial, and personal healthcare information data. Additionally, other application domain transaction data are collected.

The data provider agrees to the privacy policy preferences needed to access data utility services by the data collector (or service provider). Once the data elements are collected, attribute meta-data (from the private, sensitive data) is extracted and bound together with data provider's privacy preferences data (from the privacy policy) and data accessor profiling data, to form a *privacy tuple*.

The privacy tuple is hashed, saved, and permanently stored on the blockchain platform, while the collected data provider private data and transactional data are saved on the relational Database Management System (DBMS). The raw privacy tuple data value is stored in the relational database. The transmission of the raw and hashed privacy tuples to the relational database and blockchain platform, respectively, are facilitated through encrypted data communication channels that exist between the system components: relational database, blockchain platform, user interfaces, and genomic data store.

The general architecture of the blockchain platform (which involves the transaction ledgers, data streams, and data items) are configured in a way that the data items are directly accessible to every (authenticated) user who has secured access to query the blockchain node. Hence, the adoption of hashing methodology on the privacy tuples before they are stored on the blockchain provides another layer of data protection to the privacy tuples. In terms of domain specific application for genomic data processing, the genome meta-data arising from sequencing genomic data is encrypted and permanently stored on the blockchain platform, while the actual genome data is stored on a Linux cluster Helix High Performance Computing (HPC) platform.

During query processing, hashed privacy tuples are retrieved from the blockchains. These hashed privacy tuples are verified against the privacy tuples stored in the relational database (to check for data integrity). In cases where there is data inconsistency in the verification process of the hashed privacy tuples from the blockchains to the privacy tuples retrieved from the relational database, the query is aborted or otherwise the overall query retrieval process is allowed to proceed. Based on the data value composition in the privacy tuples, all privacy preferences (which are outlined in the privacy policy and) related to the attribute data to be queried are enforced. The query parsing process is subsequently allowed, and the required privacy-aware data records are then retrieved and provided to the requesting data accessor. Figure 3.1 provides a general overview of the methodology.



Collect private, sensitive data from data providers through User Interfaces (UI)
• Transmit hashed privacy tuple and preferences via encrypted application-level transport medium
• Transmit encrypted data provider genomic meta-data via encrypted application-level transport medium
B
Blockchains – Validated Transaction Ledger
• Storage of hashed privacy tuple and data provider privacy policy preferences on private, sensitive data
• Storage of genomic meta-data, data access rights and privileges, and data storage information
Transmit data provider private, sensitive data and transactional data via encrypted application-level transport medium
Tight-coupled Blockchains and Relational database via encrypted data transport medium
Tightly-coupled Blockchains and Genomic Data Store via encrypted data transport medium
Relational DBMS
• Storage of Individual Private, Sensitive Data and Transaction Data
Tightly-coupled Relational Database and Genomic Data Store via encrypted data transport medium
Linux Cluster Helix HPC Data Store
• Storage of Genomic Data

Figure 3.1: System Methodology Overview and Architecture

3.3 Tight-Coupling of Data Elements

The data coupling component of the overall approach combines three data elements (namely, attribute metadata, data provider privacy preferences data, and data accessor data) into a privacy tuple to be stored in the blockchains. In this approach, we bind all data provider's data elements for data privacy-preservation. This binding enables data retrieval or query processing on the attribute data to be dependent on the privacy preferences and data accessor profiles specified in the privacy policy and agreed upon by the data collector and/or accessor.

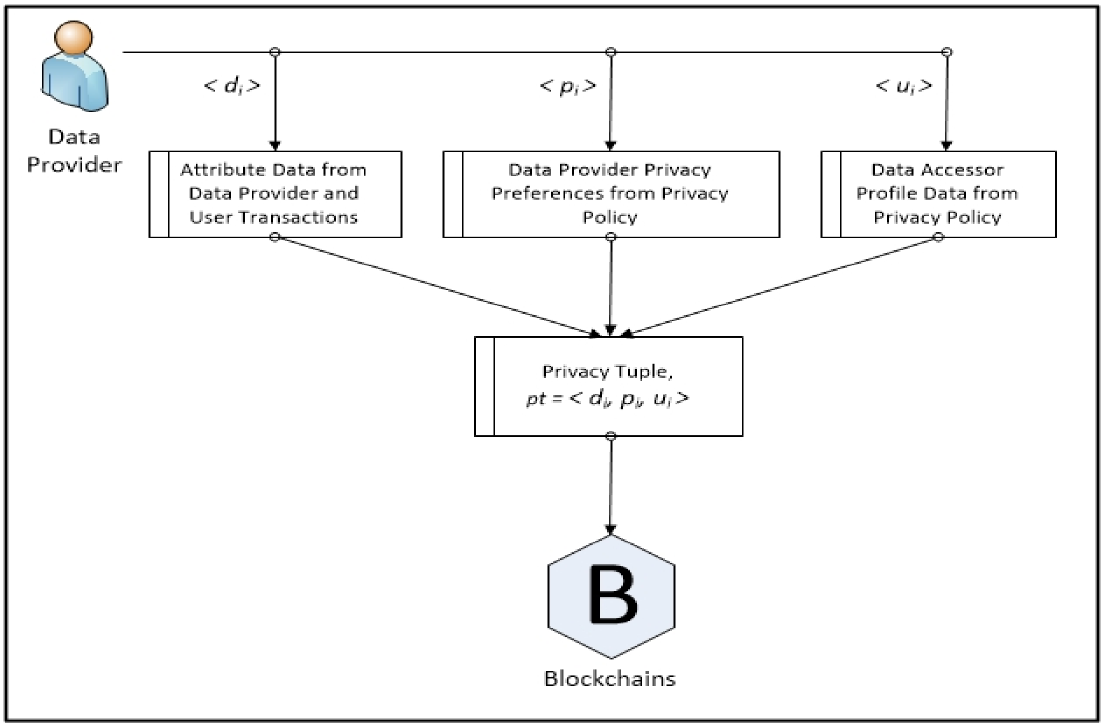

Data Provider
$< d_i >$
$< p_i >$
$< u_i >$
Attribute Data from Data Provider and User Transactions
Data Provider Privacy Preferences from Privacy Policy
Data Accessor Profile Data from Privacy Policy
Privacy Tuple, $pt = < d_i, p_i, u_i >$
B
Blockchains

Figure 3.2: Tight-coupling of Data Elements into Privacy Tuple

The *attribute data* (d_i) identifies each data provider private, sensitive attribute data collected from or provided by the data provider; as part of service account creation, personal biographical, demographic, and healthcare information, or data transactions performed by the data provider. Forms of attribute data collected are *birth date*, *phone number*, *home address*, *pathology requisition information*, *medical consent witness information*, and *health insurance information*, amongst others. In terms of the data provider privacy preferences, forms of data values collected are *purpose*, *granularity (data precision)*, *visibility (data accessibility)*, *retention duration*, *effective date*, *data information*, *third-party data accessor*, and *purpose contextual norms* (of *when*, *who*, *why*, *where*, and *how*). It will be noted that these contextual privacy preferences are outlined based on the formal contextualized privacy ontology; which models and defines the overall contextualized data provider data privacy. A *data provider privacy preferences data (p_i)* is generated based on the data values extracted from the data provider's privacy preferences in the privacy policy.

Access to the underlying data values stored in the relational database is controlled and granted to data accessors with assigned access privileges. The assignment involves setting up different access levels, in accordance with the privacy policy statements. Moreover, the assignment outlines a number of pre-defined parameters for various accesses to underlying data. The determination of the data accessor profiling is based on factors and semantics, such as, *user role* (for *e.g.*, laboratory analyst, clinical physician), *data permission level* (for *e.g.*, view data, update data), and *data sensitivity level* (for *e.g.*, Level-2, Level-4), amongst others. A *data accessor profile data (u_i)* is generated to uniquely identify and grant access to which types of attribute data the data accessors can query or retrieve information.

The three data elements (*i.e.,* attribute data, data provider's privacy preference data, and data accessor profile data) are bound together to constitute the *privacy tuple*, $pt = < d_i,\, p_i,\, u_i >$. Hence, this privacy tuple is always preserved and respected whenever an attribute data value is to be retrieved from the relational database repository. Additionally, when the attribute data value is transferred to another data repository (for example, a third-party accessor repository), the commitment to the privacy policy tuple is respected and enforced. Consequently, access and data retrieval on the attribute data value at the secondary data repository is granted. The descriptive illustration of the data elements combined into the privacy tuple is illustrated in Figure 3.2.

3.4 Implementation Approach of Data Elements Coupling

The approach to tightly couple data elements involves development of user interfaces to collect data providers' privacy preferences and for the user interfaces to serve as a medium for data providers to adjust privacy preferences.

We adopt an object-oriented approach to binding the data elements. Each data element is classified as an object. Hence, each of the objects is bound together to form the single object of a privacy tuple. With regards to the attribute data element, we adopt a model in which attribute category data comprises a number of related attribute data. For example, each of *Street*, *City*, *Province*, and *PostalCode* attribute data is grouped into a single attribute category data of *DataProviderDemographic.* We then assign a single object to the attribute category data. Hence, for an attribute category data, say *DataProviderDemographic*, we assign an *ObjectId* value.

The data provider's privacy preference data element for the attribute category data is represented by an *ObjectId*. This object has privacy preference methods that determine access, sharing, and disclosure under a particular context and/or data processing purpose. Some forms of privacy preferences are: *RetentionDurationPrivacy*, *VisibilityPrivacy*, *GranularityPrivacy*, *PurposeUsePrivacy*, and *PurposeContextPrivacy*. All these privacy preferences are instantiated with unique data values to determine accessibility to attribute data. Examples of instance values for these privacy preferences are of the form *'1-Specific'* for *GranularityPrivacy*, *'1-Single'* for *PurposeUsePrivacy*, and *'2021-03-27'* for *RetentionDurationPrivacy*.

Finally, the data accessor profile data has a unique *ObjectId* assigned to it. This *ObjectId* has methods, such as, *DataAccessorRole*, which is instantiated with data values as in *'Patient'*, *'ClinicalPhysician'*, *'LaboratoryAnalyst'*, and *'Third-PartyAccessor'*, amongst others. Other methods, such as, *SecurityClassificationLevel*, are instantiated with data values as in, *'Level-1'*, *'Level-3'*, and *'Level-4'*, amongst others; and *DataAccessorPurpose*, with instance data values as in, *'ViewData'*, *'AddViewData'* and *'AddViewUpdate'*, amongst others.

As part of the system infrastructure, an encrypted application-level transport medium is established between the user interface and the blockchain platform to aid data transmission between both components. Access to the blockchain platform is done primarily through an API wrapper protocol. Each access to the blockchain is done through authenticated username, password, host address, port, and chain (node) name. All connections for data access and query processing are completely terminated after closure of data processing activities. Moreover, all configuration settings for the blockchain platform are securely protected. The implementation of secured connections and network monitoring application tools establish efficient protection of the blockchain configuration. The encrypted data communication medium is enhanced to offer a protected, secured, and tamper-resistant medium for the transmission of the privacy tuples. We discuss the details of the system connection of the blockchain platform to the relational database and genomic data store in Section 3.6. Figure 3.3 illustrates the system architecture to generate data objects, bind them into privacy tuple, and transmit the hashed privacy tuples to the blockchain platform.

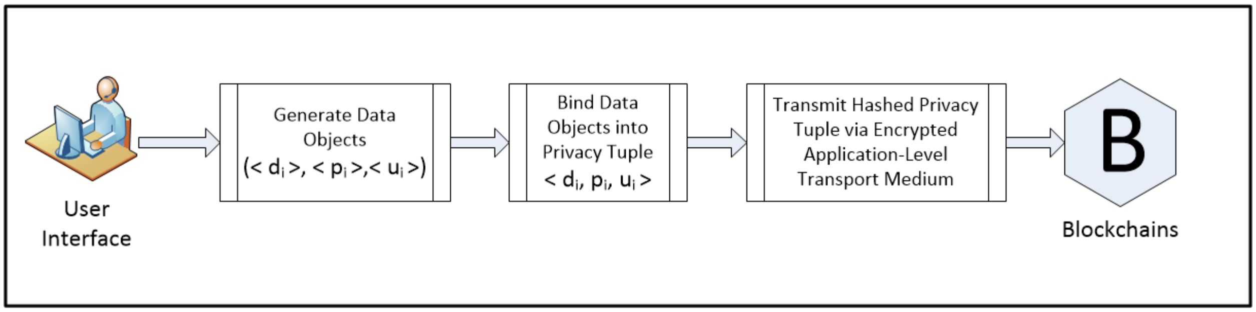

User Interface
Generate Data Objects
(< d_i >, < p_i >,< u_i >)
Bind Data Objects into Privacy Tuple
< d_i, p_i, u_i >
Transmit Hashed Privacy Tuple via Encrypted Application-Level Transport Medium
B
Blockchains

Figure 3.3: System Methodology of Generating and Storing Privacy Tuples on Blockchains

Moreover, the data provider genomic meta-data information is encrypted and transmitted to the blockchain platform. The data collector retrieves the meta-data from the genomic data that has been sequenced and variant analysed. Some of the meta-data information saved on the blockchain platform are data provider personal health number, genomic data file date, genomic data sequencing date, genomic data file format, normal (forward and reverse) genome file names, and cancer (forward and reverse) genome file names. These data are saved on the blockchain platform to offer privacy protection over access and querying of data provider genomic data.

3.5 Tight-Coupling of Blockchains Platform, Relational Database, and Genome Data Store

The data storage mechanism proposed is based on the principle of coupling different database platforms: namely, relational database, blockchain platform, and genome data store. All three data store components are hosted on a single computing machine and this enables an efficient implementation of the data store coupling. We explain data store tight-coupling to indicate that privacy-aware data and query references are stored on the blockchain platform; and these are directly related to the content data stored on the relational database and genome data store. These privacy preferences determine, authorize, and authenticate query processing and data retrieval of privacy-aware data records (from the relational database and genome data store) that need to be presented to the intended data accessor. Hence, data processing on the relational database and genome data store is always dependent (or tightly-knit) to the privacy preferences stored on the blockchains. Subsequently, this approach underlies no direct, independent query access or data retrieval from the relational database and genome data store. For example, to query and retrieve data values from the relational database, the data provider attribute data items need to reference privacy preferences stored on the blockchain for authorized data retrieval to the intended data accessor. Additionally, to run NGS data retrieval on the genome data store, the genome meta-data stored on the blockchain authenticates privacy-aware query processing on the genome data store to the intended data accessor.

The reason for this approach is to maximize the merits of managing and processing transaction data from each of the data platforms. The relational database offers a scalable, high throughput, and efficient querying engine to meet the expected high data processing needs. Conversely, the blockchain platform offers decentralized data control that is tamper-resistant, and immutable for the protection of privacy-related data values. Additionally, the decentralized control of the blockchains offers efficient change request and approval (from service providers and data providers, if necessary). The blockchains offers a platform that can be used to facilitate privacy audit procedures for data provider privacy preferences.

One key advantage for the incorporation of the blockchains in the privacy model is the ability to provide a platform for all data accessors (data and service providers) to agree on the state and value of data stored. Additionally, the blockchain platform provides an infrastructure platform to effect efficient changes to privacy-related data values. The tight-coupling approach also enables either input of data provider and service provider in the agreement of the parameters for privacy policies; thereby ensuring access consistency for all data accessors. Moreover, the blockchain platform offers efficient monitoring and control of privacy policy parameters that data and service providers have agreed upon, in a reliable data provider privacy and identity management.

The genome data store (*i.e.*, Linux cluster Helix HPC) is designed to store and manage NGS genomic data (which do have very large file sizes). There is also tight-coupling between the blockchain platform and the genome data store. Through this approach there is a safe procedure of identifying data provider genome data with related data provider biographic, demographic, and healthcare information. The overall architecture of data repository coupling presents an efficient process of coupling disparate, but related, healthcare data on data providers within a uniform system architecture.

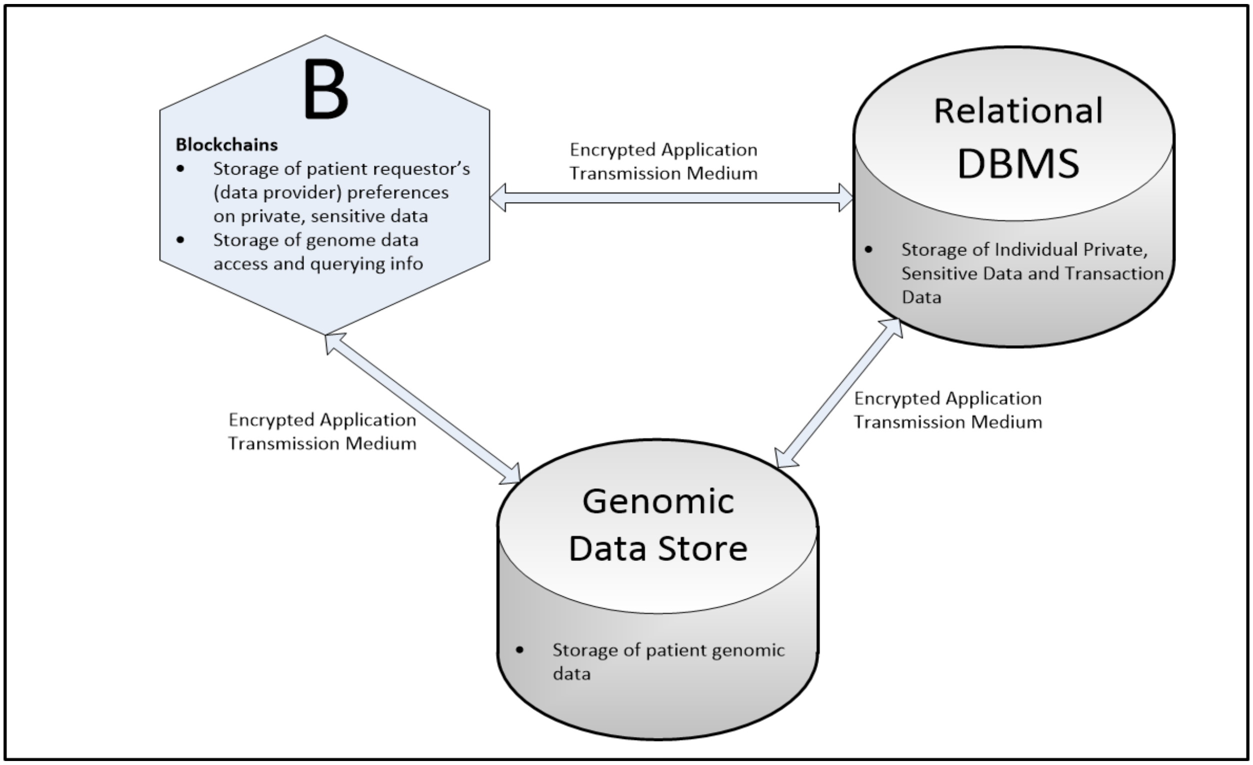

B
Blockchains
• Storage of patient requestor's (data provider) preferences on private, sensitive data
• Storage of genome data access and querying info
Encrypted Application Transmission Medium
Relational DBMS
• Storage of Individual Private, Sensitive Data and Transaction Data
Encrypted Application Transmission Medium
Encrypted Application Transmission Medium
Genomic Data Store
• Storage of patient genomic data

Figure 3.4: System Methodology and Architecture of Tight-Coupling of Blockchains, Relational Database, and Genome Data Store

The high-level methodology is, as follows: the entire set of data values (on data provider personalized information and transaction data) are collected and stored in the relational database. Moreover, an instance of the integrated privacy tuple data is stored in the relational database. Data provider genome data is collected and stored in the genome data store, whiles the genome meta-data and data access information is stored in the relational database. On the blockchains, another instance of the tightly-coupled privacy tuple and privacy preferences (which is subsequently hashed) are stored. Furthermore, data provider genome meta-data and data access information (which is subsequently encrypted) are stored in the blockchains. Figure 3.4 illustrates the proposed architecture of the relational database, blockchain, and genome data store. With this data platform integration, every data retrieval request from the relation database or genome data store is authenticated and authorized from the blockchain platform (through its related transaction ledger). This establishes the tight-coupling approach of the integrated data platform, and no other data retrieval (on the relational database or genome data store) is performed independent of the blockchain. The data and communication connection between the relational database, the blockchains, and genome data store is via an encrypted data communication protocol. Furthermore, any form of data transfer (query requests and results) between the data repositories is completed in a secured, tamper-resistant access-controlled protocol.

3.6 Implementation Approach of Tight-Coupling of Blockchains, Relational Database, and Genomic Data Store

The key components of the implementation for tight-coupling of data platforms are: a secured, completely-protected relational database, private blockchains, and genome data store. Additionally, there is a tamper-resistant and secured data transmission medium between the blockchains, relational database, and genome data stores. There are Point-of-Contact (PoC) user interfaces, and a secured data transmission medium between the user interfaces and the data platforms. The next sections describe the functional details of these components, and ability of the framework to offer an efficient and robust data processing platform for protection of data provider's private, sensitive data.

3.6.1 System Architecture of Relational Database, Blockchain, and Genome Data Store

The system architecture describes a tight-coupling of the data platforms for an integrated data processing framework. Each data platform component presents unique functionality for transaction data management and this is based on the type of data transaction. Figure 3.4 provides a description of the integrated data platform. The relational database stores and processes data provider private, sensitive data and transaction data. The blockchain stores and processes the hashed form of privacy tuple and genome data access and querying information. The genomic data platform stores and processes data provider genomic data.

We assume that native database security and protection measures that are embedded in the relational databases are configured to protect against unwarranted access and data retrieval. Recent relational databases have been enhanced with efficient protection from compromised database administrators. These preventive security controls include privileged user and multi-factor access control, database activity monitoring and blocking, consolidated auditing and reporting, and encryption key management, amongst others [100]. Additionally, we assume safe use of database objects (such as, relational tables, views, and stored procedures, *etc.*) by database administrators in a manner where object access and management are performed in a privacy-aware manner.

The data connection between the relational database, blockchains, and genome data stores is established using an encrypted transmission medium to secure the communications. This encryption protocol offers a protected application-level data transport medium for key agreement or establishment, entity authentication, confidentiality and data integrity. Thus, we adopt efficient encryption methodologies for the transmission of data accessor query requests or returned query answers (results) between the data repositories.

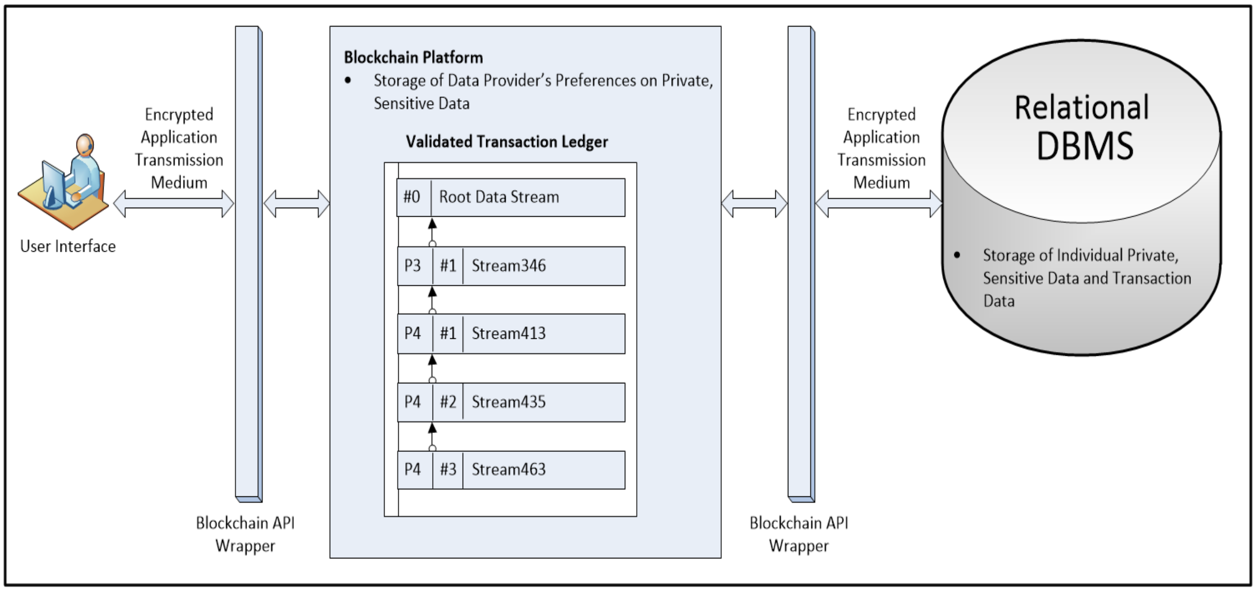

User Interface
Encrypted Application Transmission Medium
Blockchain API Wrapper
Blockchain Platform
• Storage of Data Provider's Preferences on Private, Sensitive Data
Validated Transaction Ledger
#0 Root Data Stream
P3 #1 Stream346
P4 #1 Stream413
P4 #2 Stream435
P4 #3 Stream463
Blockchain API Wrapper
Encrypted Application Transmission Medium
Relational DBMS
• Storage of Individual Private, Sensitive Data and Transaction Data

Figure 3.5: Tight-Coupling System Architecture of Blockchains and Relational Database

We adopt a permissioned (private) blockchains for storage of data provider privacy policy preferences and privacy tuple information. Figure 3.5 provides an illustration of the blockchain and its constituent validation transaction ledgers. The diagram further describes the encrypted data connection between the blockchain and user interfaces on one hand, and the relational database on the other hand. The blockchain is accessible only to the data providers and data collectors (or service providers and possibly, third-party data accessors via user interfaces). The blockchain contains a *validated transaction ledger* which stores transaction data blocks (on privacy tuples) that are independently verified, accepted, and validated by data providers and data collectors. Each transaction ledger is configured and composed of transaction data streams and each data stream incorporates transaction data blocks which contains or stores actual privacy tuple transaction data.

Typical blockchain transaction ledger consists of one or more data streams which stores transaction data. The initial configuration of the blockchains consists of automatically generating a *root* (*genesis*) data stream that does not contain transaction data blocks. Consequently, the first user-generated data stream is appended to the existing *root* data stream, and all other subsequent data streams are also appended to their preceding data stream. Processed and validated transaction data block (associated to a privacy tuple) is appended to first data stream or any other related data stream (in the list of data streams created on the chain). Thus, the privacy tuple (together with policy preferences) which are stored as transaction data blocks become valid data, and are always accessed during query processing.

An illustration of the system architecture depicting the connectivity between the blockchain platform transaction ledger (which contains data streams) and the relational database is displayed in Figure 3.5. In the diagram, there are four data streams created and appended to the root data stream. The first data stream which is appended directly to the *root* data stream stores the first data provider privacy tuple transaction data block. Subsequent (and updated) privacy tuple transaction data block for the same data provider is appended and stored in this data stream. When a new privacy tuple transaction data block for a different data provider has to be appended to the transaction ledger, a new data stream is created and appended to the second data stream. This makes up a total of three data streams contained in the validated transaction ledger. In summary any new privacy tuple transaction data block for an existing data provider is appended to the related data stream, whiles any other privacy tuple transaction data block for a new data provider creates a new data stream by appending to the most recent data stream.

Following creation of a different transaction ledger and constituent transaction data streams for privacy policy preferences and privacy tuples, we create another transaction ledger and transaction data streams to store data provider genome meta-data and data access information. An illustration of the system architecture depicting transaction ledger and transaction data streams for data provider genome meta-data is displayed in Figure 3.6. Here, four data streams are created and appended to the preceding data stream (which is the *root* data stream). The first two data streams store genome meta-data for the same data provider but with different genome analysis. Moreover, the next two data streams store different data provider genome meta-data information. In summary, every new data stream genome meta-data and query information is appended to the most recent data stream.

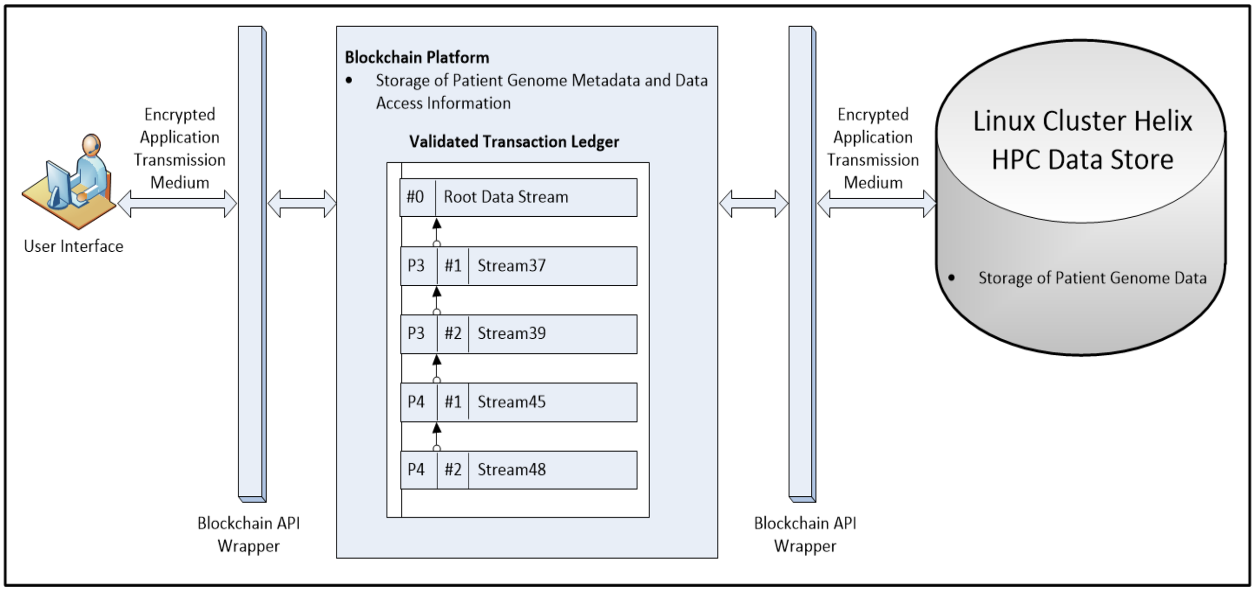

Encrypted Application Transmission Medium
User Interface
Blockchain API Wrapper
Blockchain Platform
• Storage of Patient Genome Metadata and Data Access Information
Validated Transaction Ledger
#0 Root Data Stream
P3 #1 Stream37
P3 #2 Stream39
P4 #1 Stream45
P4 #2 Stream48
Blockchain API Wrapper
Encrypted Application Transmission Medium
Linux Cluster Helix HPC Data Store
• Storage of Patient Genome Data

Figure 3.6: Tight-Coupling System Architecture of Blockchains and Genome Data Store

3.6.2 Procedural Data Operations on Blockchains Platform

The implementation procedures follow a scenario where a data provider completes and signs on to a privacy policy with a service provider (via a user interface). Additionally, data provider can request changes to the existing privacy policy preferences already in place. It is noteworthy that data provider privacy policy preference requests are received at the blockchains as data transaction tuples. The procedural aspects that discusses generation of tightly-coupled privacy policy tuple is described (recall Sections 3.3 and 3.4).

The service provider (via a user interface) accesses the privacy policy preferences and effects a signature validation or acceptance of the privacy policy signed onto by the requesting data provider. The access by the service provider to the data provider privacy preferences is routed to the blockchains. Each data provider data attribute and its associated privacy preferences is signed, validated, and accepted as a single transaction block in the blockchains.

These privacy policy preference requests are transmitted to the blockchains. At the blockchains, each privacy tuple is stored as a data item transaction block and appended onto the most recent transaction block. For instance, in the *transaction ledger* (see Figure 3.5), the privacy tuple data stream for attribute category data, "biographical information (first name, middle name, last name)" and data accessor, "laboratory analyst" is created. This data stream is appended to the *"root* transaction data stream. Consequently, the privacy tuple data stream for attribute category data, "consent witness information (last name, first name, phone number, street, city, postal code)" and data accessor, "physician" is created. The new data stream is always appended to the most recently appended data item transaction block.

The validated data item transaction blocks (respecting all the privacy policy preferences) are permanently committed on the *validated data streams transaction ledger* in the blockchains. This provides enough information for privacy policy audit and privacy assessment in the occurrence of privacy breaches or violations. This is because the information saved in the blockchain incorporates all privacy preferences on the data provider's attribute data. More importantly, each privacy tuple instance data serves to identify information on the attribute data, the data accessor's access levels, and the set of privacy preferences (regarding granularity or precision privacy, visibility or access privacy, purpose privacy, retention duration, third-party accessibility, *etc.*) This privacy tuple data item can be queried from the blockchain transaction ledger (data streams), and the query output can be used identify more privacy information about on the attribute data; such as data accessors and their privacy privileges [121][122]. To query a privacy tuple, all blockchain data stream transaction ledgers for a particular data provider are identified. Next, the unique data stream pertaining to the attribute data and data accessor is retrieved. This data stream stores all current and previous transaction data items on data provider privacy preferences; that are validated by both data provider and data collector.

Figures 3.5 and 3.6 illustrate the overall implementation approach of the tight-coupling of blockchains and relational database, and blockchains and genome data store, respectively.

3.7 Query Processing Procedures on the Proposed Methodology

The proposed methodology takes query requests to the relational database by first routing them to the blockchains platform before delivering query results. The query request is first transmitted to the privacy-aware query analyser. The privacy-aware query analyser parses the query, identifies and extracts attribute data (regarding the requested query) that needs to be accessed for data retrieval. The privacy-aware query attribute data are then transmitted to the blockchains.

At the blockchains, the data provider's data stream needed for data retrieval is identified. The blockchain platform traverses through the transaction data stream to access the most recent privacy tuple transaction data item block. Transaction data item block corresponding to privacy tuple is subsequently accessed and retrieved. This data item contains recent data provider privacy policy preference's tuple (on attribute metadata, contextualized privacy preferences data value, data accessor value, and other privacy preferences).The MultiChain blockchain API delivers retrieved data items (in its hashed form) to the privacy-aware query processer.

At the query processor, the privacy-aware query is rewritten and transformed. Data items consisting of privacy policy preferences are verified against the retrieved hashed data values from the blockchain. In cases where there is data inconsistency between both data item values, the query is denied and aborted. By data inconsistency, we identify that the hashed data values stored in the blockchain is different from data values retrieved from the relational database. Upon valid consistency check on data item values, the query processor grants access for query processing and data retrieval. The privacy preferences on each data attribute are used to rewrite or transform the privacy-aware query to conform to the current context based on data provider preferences in the privacy policy. The privacy-aware query processor uses the privacy-aware rewritten query statement to generate expected query result or data instance tuples. Consequently, data instance tuples regarding the attribute data are retrieved and presented to the data accessor through user interfaces.

In summary, the blockchain platform acts as a privacy-aware data access control and query analysing medium to the underlying relational database. Hence, whenever a data accessor wants to query or retrieve data values in the underlying relational database, access is routed to the blockchains before privacy-aware data values stored in the relational database are retrieved and presented to the data accessor. Figure 3.7 illustrates the implementation of the query processing architecture for accessing data provider private, sensitive data from the relational database repository.

The architecture presents an integrated query platform of user interface, relational database, blockchain, and genome data store. The user interface provides the medium for data accessors to post queries and display returned query results. The relational database is made up of the data store, privacy-aware query analyzer, and privacy-aware query processor. The data storage stores all private, sensitive and transactional data. The privacy-aware query analyzer is designed in the form of a database stored procedure, that accepts data parameters to allow or reject queries. The privacy-aware query processor executes the queries. The blockchain is composed of validated transaction ledgers for data provider privacy tuples and genomic data access and retrieval information. The genome data store stores data provider genomic data. We discuss the detailed implementation of query processing on the integrated data platform in Section 4.3.3.

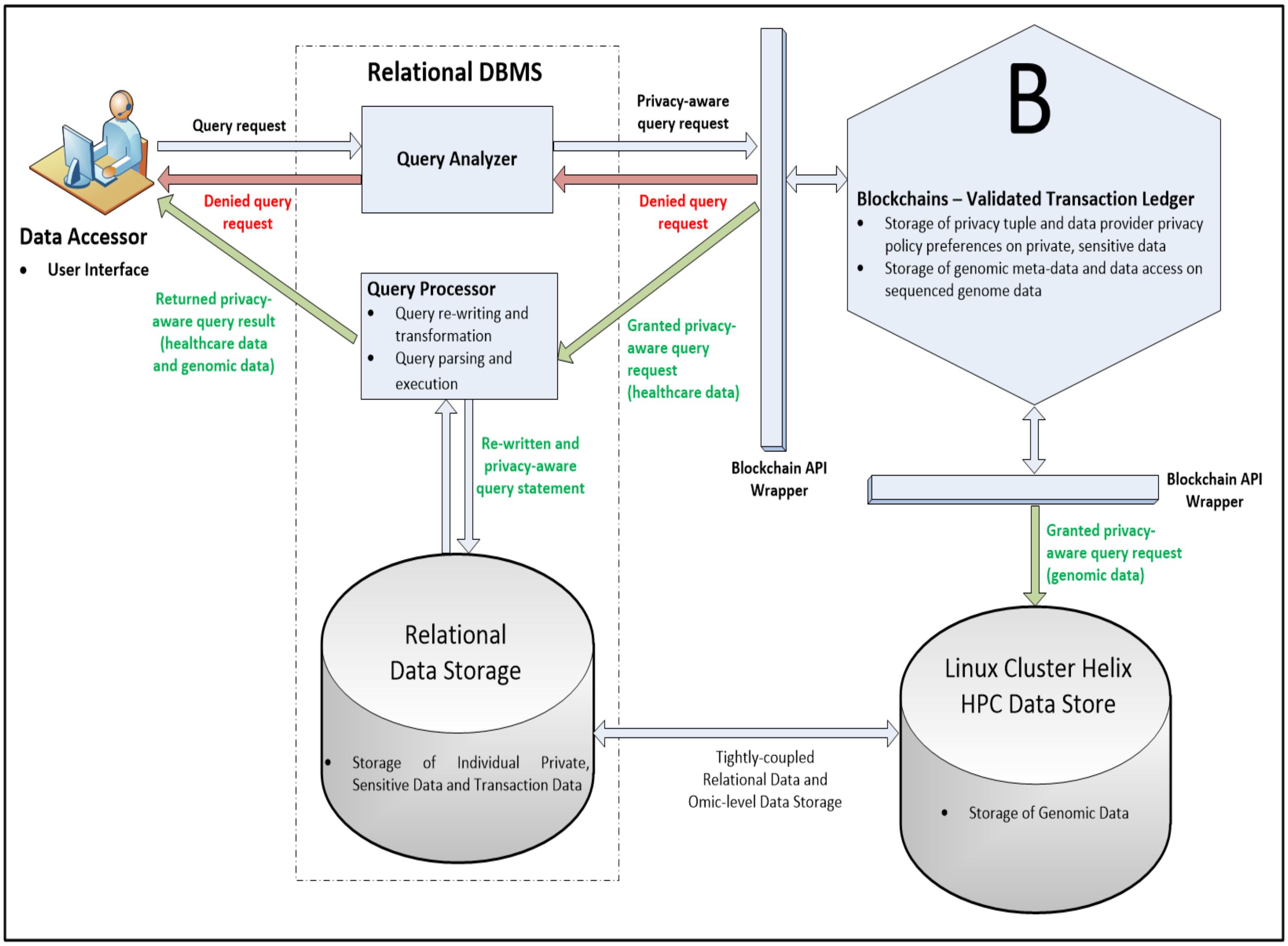

Relational DBMS
Query request
Query Analyzer
Privacy-aware query request
Denied query request
Denied query request
B
Blockchains – Validated Transaction Ledger
Storage of privacy tuple and data provider privacy policy preferences on private, sensitive data
Storage of genomic meta-data and data access on sequenced genome data
Data Accessor
User Interface
Returned privacy-aware query result (healthcare data and genomic data)
Query Processor
Query re-writing and transformation
Query parsing and execution
Granted privacy-aware query request (healthcare data)
Re-written and privacy-aware query statement
Blockchain API Wrapper
Blockchain API Wrapper
Granted privacy-aware query request (genomic data)
Relational Data Storage
Storage of Individual Private, Sensitive Data and Transaction Data
Tightly-coupled Relational Data and Omic-level Data Storage
Linux Cluster Helix HPC Data Store
Storage of Genomic Data

Figure 3.7: Query Processing Methodology on Blockchains, Relational Database, and Genomic Data Store

3.8 Summary

This chapter discussed the proposed research methodology of Light-weight Privacy Infrastructure which provides the framework platform for preserving data provider private, sensitive data. We provide our overview of formal contextualized privacy ontology. Moreover, we discuss methodology approach of tight-coupling of data elements (attribute data, data accessor, and privacy preferences), and its implementation approach. We discuss methodology and implementation approach of tight-coupling of private blockchain platform, relational database management system, and genomic data store. We discuss data transmission procedures from the user interfaces to relational database, and between relational database and blockchain platform. Finally, we discuss query processing approach and procedures for data retrieval on the proposed research methodology.

Chapter 4 discusses system design, development, and implementation of the proposed research methodology. We discuss formulation of formal contextualized privacy ontology. We discuss system modelling and design of user interfaces, as well as programming procedures to extract data values from system user interfaces. We discuss modelling, design, and development of relational database management system and permissioned (private) blockchain platform. Finally, we address data communication and encryption protocols between system components and summarise all the discussions.

Chapter 4

System Design, Development, and Implementation

The proposed research methodology of Light-weight Privacy Infrastructure (LPI) discussed in Chapter 3 details methodology overview, tight-coupling of data elements, tight-coupling of database components of relational database, blockchains, and genomic data store, integration of user interfaces to data repositories, and query processing. We describe practical methods and implementation performed based on the proposed methodology. These implementation activities and procedures lead to the desired research results.

In this chapter, we discuss formulation and modelling of formal contextual privacy ontology in Section 4.1, and describe system modelling, user interface, and the privacy preferences tuple in Section 4.2. In Section 4.3, we describe modelling, design, and development of relational database, and in Section 4.4, we discuss design and development of permissioned blockchain platform. We then discuss integration of system components of data communication and encryption protocols of user interface, relational database, blockchains and genomic data store in Section 4.5, and a summary of the overall discussion in Section 4.6.

4.1 Formal Contextualized Privacy Ontology

In this section, we discuss adoption of formal contextualized privacy ontology for the overall research methodology (recall Section3.1). We discuss design modelling and identification of entities and predicates in the formulation of the privacy ontology for healthcare (and genomic) application domain. Moreover, we discuss measures in which the privacy ontology is updated, and the adaptation of the privacy ontology to different application domains.

4.1.1 Abstraction Modelling and Adoption of Formal Contextualized Privacy Ontology

We model the privacy policy with a formal contextualized privacy ontology. A privacy ontology models or designs a real-world environment with different entities and predicates; and describes entity behaviours and relationships between the entities and their predicates. In the broader sense, the privacy model is defined based on the privacy policy. Here, key entities of an attribute data value, data provider, data collector, and data privacy policy, are identified. Moreover, specific instantiation of the privacy ontology to an application domain needs to be specified and modelled (according to the entities and their behaviour in relation to the application domain).

The research methodology adopts healthcare (and more specifically genomic) data management application domain. The privacy ontology model attained is based on the information provided by domain experts (that is, clinical physicians who collaborated with us in the overall research). We, therefore, state that the adopted privacy ontology is verifiable by domain experts. In the formulation of the privacy ontology, we identify specific entities, such as, data provider (*i.e.*, patient), data collector (*i.e.*, clinical physician, laboratory analyst), data accessors (*i.e.*, clinical physician, laboratory analyst, bioinformatic data analyst), and third-party accessors (*i.e.*, Alberta Cancer Repository, insurance analyst, and law court). With regards to the data privacy policy entity, we identify general sub-entities, such as, *GranularityPrivacy*, *VisibilityPrivacy*, *etc.* Moreover, we identify specific sub-entities and their instantiations (which are applicable to healthcare and genomic data processing and management). For example, *PurposeContextPrivacy* is instantiated as, *MolecularPathologyRequisition*, *GenomicSequencingAnalysis*, and *LaboratorySpecimenAnalysis*. We discuss the detailed privacy ontology modelling in the next section (see Section 4.1.2).

4.1.2 Formulation and Modelling of Formal Contextualized Privacy Ontology

In the formulated privacy ontology, we model and design several entities, sub-classes and their predicate classes. Each entity contributes to the overall modelling of the privacy policy regarding a particular data attribute or category of attribute data. Figure 4.1 illustrates the overall formulated privacy ontology for the proposed methodology. This privacy ontology outlines and details every aspect of the privacy policy for each attribute data, and the procedures for data storage and management. We implement the ontology modelling of formal contextualized privacy ontology using the Protégé Semantic Web Ontology [33] application tool.

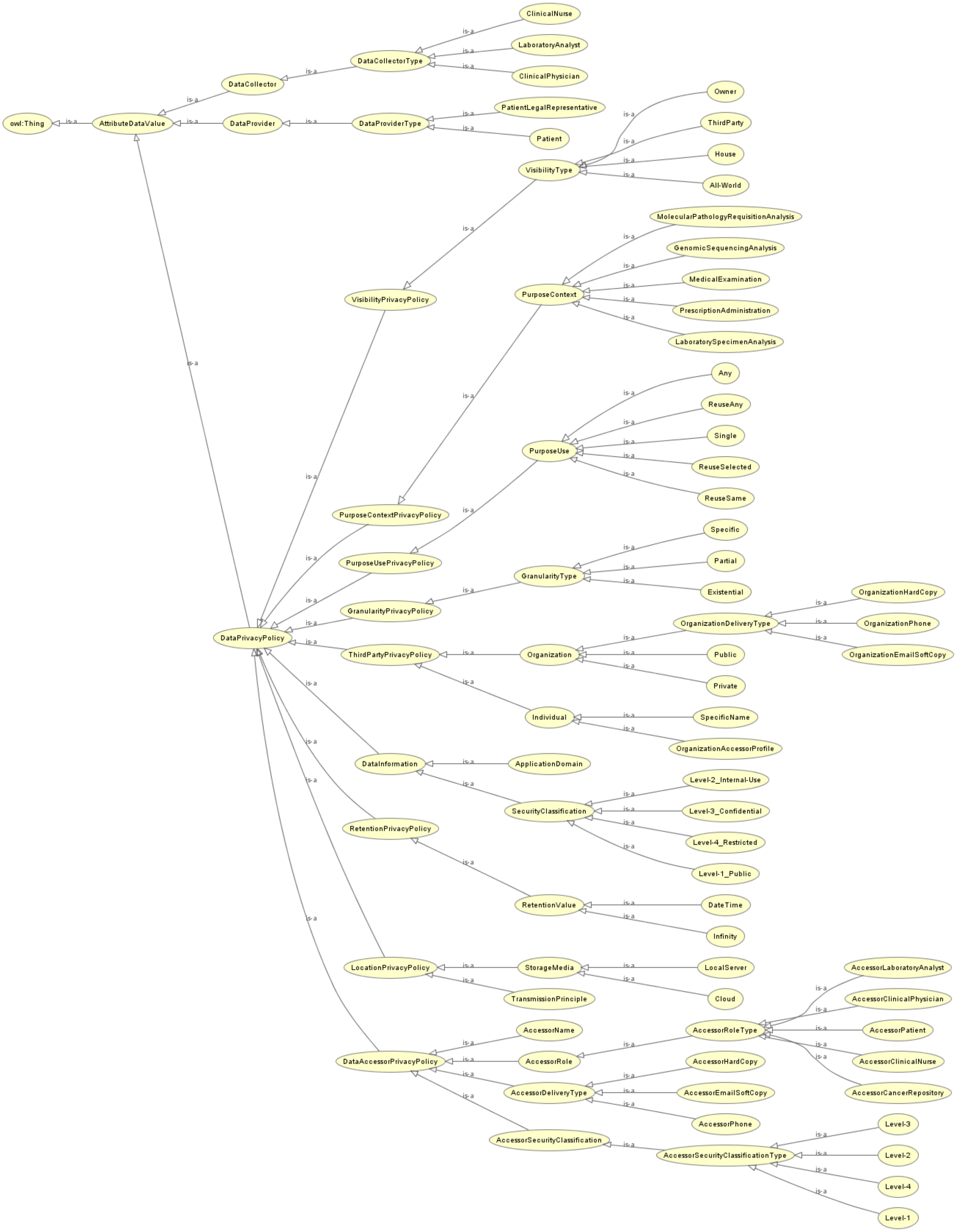


Figure 4.1: Formulated Contextualized Data Privacy Ontology Model

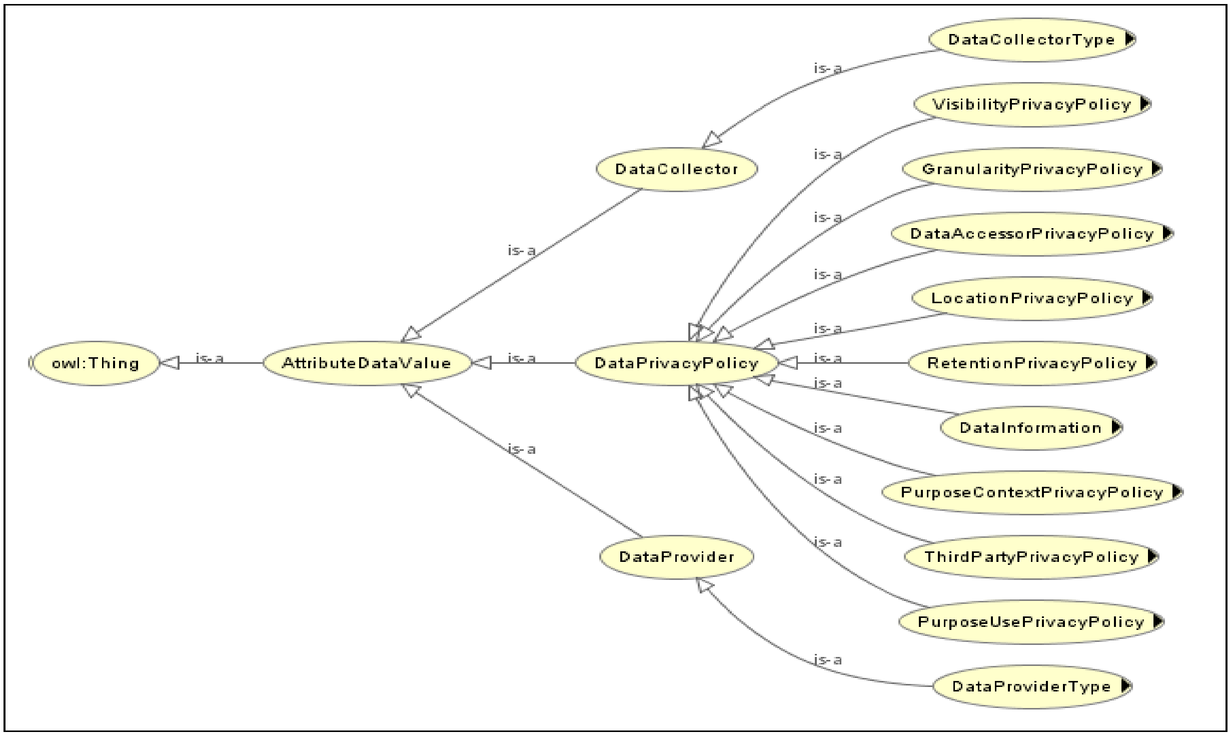


Figure 4.2: Attribute Data Privacy Ontology Model

In the formulated privacy ontology, the primary (root) entity item is the *AttributeDataValue*; as each attribute data has unique privacy policy for data storage and retrieval. The *AttributeDataValue* entity has three sub-entity classes; namely, *DataProvider*, *DataCollector*, and *DataPrivacyPolicy.* Figure 4.2 illustrates a summarized description of the formal contextualized ontology which focuses on the attribute data.

The *DataProvider* sub-entdescribes all information regarding the data provider, (that is, patient requesting healthcare service). This sub-entity has a predicate class, *DataProviderType*, which details data on the various forms of persons who serve as the primary source of data. The instantiated values for this predicate class are *Patient* and *PatientLegalRepresentative*; where *Patient* identifies with the patient who requests for medical service and offers the necessary data for service delivery. The *PatientLegalRepresentative* is a legal representative for the patient, in cases where the patient is a minor or does not have the capacity to offer requested data. This individual may be a parent, family representative, close friend or an associate.

The *DataCollector* sub-entity describes the individuals who collect and store data on data subjects (*i.e.*, patients). This sub-entity also has a predicate class as *DataCollectorType*, which has instantiated values as *ClinicalNurse*, *LaboratoryAnalyst*, and *ClinicalPhysician.*

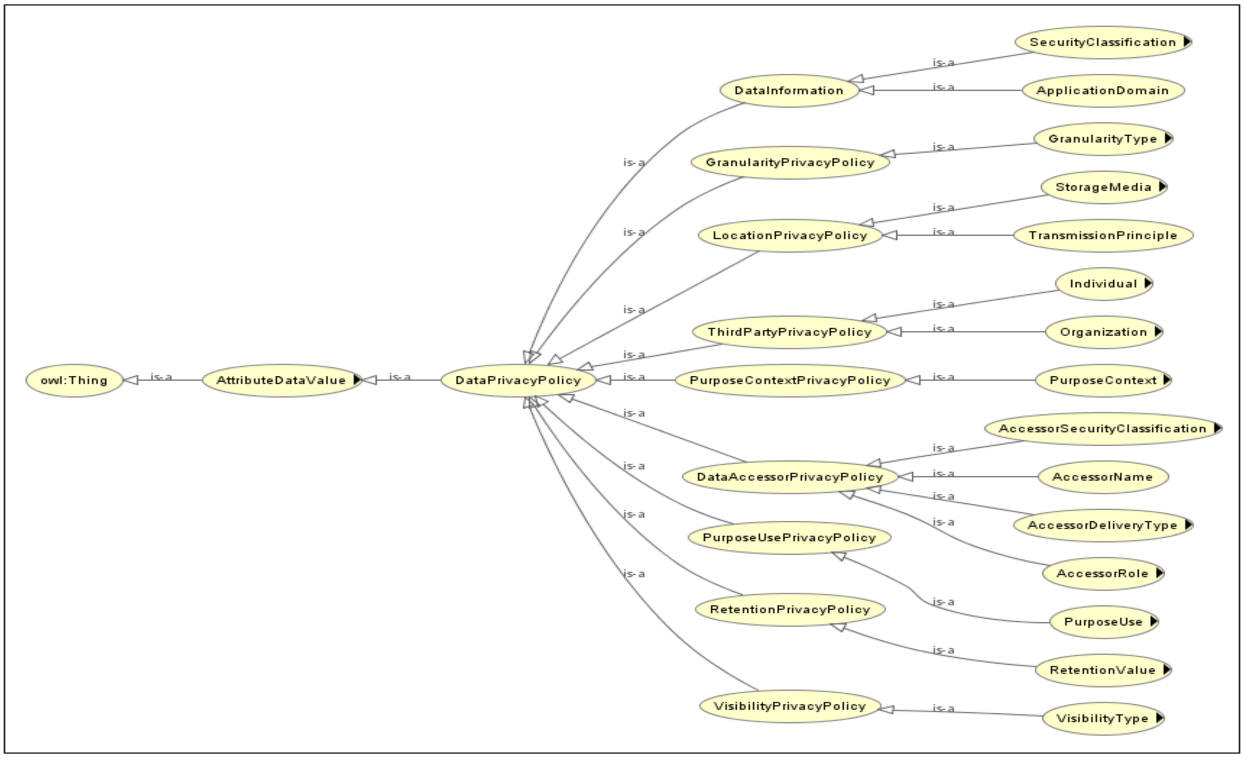


Figure 4.3: Attribute Data Privacy Preferences Ontology Model

The *DataPrivacyPolicy* sub-entity is the most expressive sub-class of the *AttributeDataValue* class. It describes all aspects of the privacy policy on each attribute data value; which has been fully authorized by the data provider (*i.e.*, patient) and validated by the data collector (*i.e.*, clinical physician). We model this entity class based on prior research work of Barker *et al.* [16] on privacy taxonomy for data privacy policies. The *DataPrivacyPolicy* class incorporates other sub-classes which are called, *PurposeUsePrivacy*, *VisibilityPrivacy*, *GranularityPrivacy*, and *RetentionPrivacy*. Each of these entities describe different aspects of the privacy policy and offer unique modelling semantics for preserving data provider's private and sensitive data. We illustrate the broad description of the *DataPrivacyPolicy* entity for attribute data privacy preferences modelling in Figure 4.3.

The *PurposeUsePrivacy* class describes data accessibility and usage of related attribute data values by data accessors. This entity class is further described by a predicate sub-entity class, *PurposeUse*, which has instantiated values as, *Any* (for data used for any purpose any multiple times), *Reuse-Any* (for data used for un-foreseeable related purposes), *Single* (for data used for one time and for single purpose), *Reuse-Selected* (for data used for primary purpose for which data), and *Reuse-Same* (for data used for same purpose and multiple times).

The *VisibilityPrivacy* class describes privacy preferences on data accessor's access permissions or use on data provided by the provider (*i.e.*, the patient). This entity class has a predicate sub-entity class, *VisibilityType*, which has instantiated values as, *Owner* (for data accessible to only the data provider), *House* (for data accessible to everyone who collects, access, and utilize the data), *Third-Party* (for data accessible to parties not covered by explicit agreement with House), and *All-World* (for data accessible to anyone with access to the data repository).

The *GranularityPrivacy* class describes privacy preferences on the data which captures precision level of retrieved data to facilitate appropriate use and for the intended purpose. The entity class has a predicate sub-entity class, *GranularityType*, which has instantiated values as, *Specific* (for a specific data item retrieval of which a query for data item returns actual data value), *Partial* (for a partial or altered and non-destructive retrieval of data values to accessor), and *Existential* (to specify information released to the accessor indicate the existence of data in repository, not actual data values).

The *RetentionPrivacy* class outlines privacy preferences on the attribute data for explicit retention period for data retrieval by data accessors. This is to ensure data collected is for particular purpose and remain current for that purpose. The *RetentionPrivacy* class has a predicate sub-entity class, *RetentionValue*, which is instantiated as *DateTime* (for an explicitly stated date and time) and *Infinity* (for an undefined or perpetual time for data retrieval).

Other sub-entity classes that uniquely express the *DataPrivacyPolicy* class are *PurposeContextPrivacy*, *ThirdPartyPrivacy*, *DataInformationPrivacy*, *LocationPrivacy*, and *DataAccessorPrivacy*. The *PurposeContextPrivacy* expresses data provider privacy preferences on the need, intended use, and/or contextualized norm of data retrieval by the data accessor. The class has a predicate sub-entity, *PurposeContext*, which has unique instantiated values for purposes of data collection, storage, disclosure, and access within the context of genomic data processing. The instantiated values are *MolecularPathologyRequisitionAnalysis* (for data collected and used for molecular pathology requisition and analysis), *GenomicSequencingAnalysis* (for data collected and used for genomic sequencing and related analysis), *MedicalExamination* (for data collected and used for patient medical examination and diagnosis), *PrescriptionAdministration* (for data collected and used for prescription and administer medication), and *LaboratorySpecimenAnalysis* (for data collected and used for laboratory and specimen analysis).

The *LocationPrivacy* class expresses information on the physical storage location of the data collected from the provider. The class describes sub-entity classes, such as, *TransmissionPrinciple* and *StorageMedia*. The *TransmissionPrinciple* sub-entity class identifies data transmission medium for data transfer. This class could have instantiated values as *Electronic-Softcopy* and *Hardcopy*. The *StorageMedia* class expresses information on the specific storage medium for data storage and management. Instantiated values for this class are *Local-Server* and *Cloud-Server.*

The *DataInformationPrivacy* class expresses the meta-data information about the attribute data value. This class is further expressed by the *ApplicationDomain* and *SecurityClassifications* sub-entity classes. The *ApplicationDomain* describes information relating to the application domain from which the intended data is processed, while the *SecurityClassification* expresses the various forms of security classification upon which data retrieval restrictions are applied. Predicate class values for the *SecurityClassification* class are instantiated as *Level-1* (for general public data retrieval), *Level-2* (for internal use within the confines of the collector's data repository), *Level-3* (for confidential use), and *Level-4* (for the highest form of data restriction; where maximum security clearance is needed for data retrieval).

The *ThirdPartyPrivacy* class provides the explicit definition of third-party persons or organizations that the data collector must share or disclose the attribute data value with. For example, it is mandated by law for the clinical physician (data collector) to share the data with the Alberta Cancer Centre (for purposes of data and information sharing). The *ThirdPartyPrivacy* class is further expressed by 2 sub-entity classes: namely, *Individual* and *Organization.* The *Individual* sub-entity class is expressed by the predicate classes of *SpecificName* and *AccessorProfile.* The *Organization* sub-entity class is expressed by predicate classes of *Private*, *Public*, and *OrganizationDeliveryType.*

The *DataAccessorPrivacy* class expresses and describes the individuals who will access the attribute data. The access profiles and privileges are uniquely outlined in this entity class. The class is expressed by sub-entity classes; namely, *AccessorName*, *AccessorRole*, *AccessorDeliveryType*, and *AccessorSecurityClassification.* The *AccessorName* class identifies with the unique name for the intended data accessor, whiles the *AccessorRole* identifies with the access role and privileges for individuals requiring such intended data access. The *AccessorRole* class is instantiated by predicate values, such as, *AccessorClinicalPhysician*, *AccessorLaboraoryAnalyst*, *AccessorPatient*, and *AccessorCancerRepository.* The *AccessorDeliveryType* is instantiated by predicate values as *AccessorHardCopy*, *AccessorEmailSoftCopy*, and *AccessorPhone.* The *AccessorSecurityClassification* is instantiated by predicate class, *Type*, which has values in the form of *Level-1* (for general public data retrieval), *Level-2* (for internal use within the confines of the collector's data repository), *Level-3* (for confidential use), and *Level-4* (for the highest form of data restriction; where maximum security clearance is needed for data retrieval).

4.1.3 Privacy Ontology Update and Application Domain Adaptability

The management of privacy policies and data provider privacy preferences involve a regular change update by both data provider and collector. This is necessary because conditions regarding data provider's private data keep changing, and this will necessitate required updates or changes in existing privacy preferences. We discuss an overview of the update and maintenance procedures that are implemented on the privacy ontology. These procedures underlie and validate the dynamic formulation of the privacy ontology to handle diverse maintenance changes and its adaptability to different application domains. These update and maintenance procedures are expressed based on two instances.

First, it essential to update and subsequently remodel the privacy model (regarding new changes in the agreed privacy policy) for the overall private data management. This could arise in the context when third-party privacy policies are updated, retention duration is updated, or new data accessor role is defined, for some of the attribute data values. For example, in the event that new data access role needs to be defined for Canadian Cancer Centre or oncogenomic data analyst. This implies that the ontology must evolve and change to align with the new atrribute data value privacy preferences. Hence, as privacy preferences change, the privacy policy ontology model also changes. For this form of ontology update, it is noted that the change does not affect the core entity and predicate structure of the ontology. Moreover, the ontology changes mainly affect the data instances aspect of the ontology and their required semantics for data retrieval.

Second, the privacy ontology evolves to support different application domain implementations. These expected maintenance changes may affect some structural aspects of the overall ontology design. For instance, the privacy ontology may be applied to a dental healthcare application domain; and this will require some changes in the formulation of entity classes and their predicates. Thus, some predicates could be deprecated while new predicates are modeled to instantiate query processing and data retrieval within the context of current application domain.

In summary, we define the formulation of the privacy ontology as a generic (but dynamic) foundation for the privacy model in the overall privacy infrastructure methodology. Moreover, the dynamic form of the privacy ontology modelling allows it to be verifiable by domain experts to reflect the entire privacy-aware data processing needs for an application domain.

4.2 User Interface Modelling and Development for Privacy Tuple

The overall user interface system architecture involves modelling and design of system internal component activities and interaction with the environment. This modelling approach includes the design of user interfaces to collect and display data to the actors of the system. As part of data collection, the data provider offers data privacy policy preferences which are validated by the data collector (or service provider). Additionally, the data provider produces private, sensitive transactional data. Other related data are collected by data collectors alongside the privacy preferences data. These data privacy policy preferences and private, sensitive data are transferred from user interfaces and stored in the relational database for data processing and report generation. Furthermore, a hashed form of the privacy preferences and privacy tuple is saved on the blockchain platform.

As part of system design and implementation, we address the idea that the research implementation is a preliminary work that displays reports on individuals (*i.e.*, data providers, data collectors, consent witness), laboratory requisition, and medical consent processing. Moreover, the implementation work does not allow relational joins or other complex queries. We discuss further on these elaborate forms of query processing as part of future work in Chapter 6 (see Section 6.4).

We now describe system modelling approaches for user interfaces and outline procedures of collecting data privacy preferences to generate privacy tuple. Moreover, we describe the transmission of the data privacy preferences and privacy tuple to the blockchain platform.

4.2.1 User Interface System Modelling – Uses Cases, Activities, and Actor(s)

We present system modelling for the entire user interface system implementation. A system modelling process develops conceptual models of the system, with each model presenting a different view or perspective of the system. This system modelling explains the system from different perspectives, such as: external context or environment, interactions between the system components and the environment, organizational structure of the data processed by the system, and dynamic behaviour of the system. It is important to note that the system modelling and development are based on the outcome of the requirements engineering procedures that were conducted from interactions with domain experts (*i.e.*, clinical physicians).

Complete User Interface System Modelling

The overall user interface system modelling is presented with a Unified Modelling Language (UML) use case diagrams. Use case diagrams show the interactions between a system and its environment. In this model, there a several use cases which represent unique sub-system operations and modelling. Each use case provides a coherent unit of functionality provided by the system. Each use case has actors that indicate a role played by the outside objects. The actors for the system application domain case study are individuals, such as, patient, clinical physician, laboratory analyst, third-party data accessor, or blockchain platform. In terms of sub-system operations, a use case activity may have only one actor (for example, patient actor for *“Register Patient Record”* use case) or several actors (for example, patient, clinical physician, and third-party data accessor actors for *“Generate Data Provider healthcare Report”* use case).

A use case activity may have an extension to another use case using *Extends* functionality; where this extended use case activity presents a background activity needed for the primary use case to act. For example, *"Complete Medical Consent"* use case activity extends *"Generate Pathology Requisition"* use case activity. A use case activity may incorporate an instance where one use case activity will also "use" or "include" the behaviour specified in another use case activity. For example, the *"Update Patient Record"* use case activity will always need the behavioural existence of the *"Register Patient Record"* use case activity. Finally, the system modelling is incorporated with a boundary, represented by a rectangle, which limits within which all use case activities can be performed. Figure 4.4 illustrates the system modelling use case diagram for the overall system design.

Each sub-system modelling has one or more use case activities, their associated use case activity extensions, and one or more actors that interact with use cases. We present a summary of each use case functionality for the user interfaces design.

Register and/or Update Patient Data

This use case involves activities that capture patient basic bio-information, demographic information, as well as, some personal healthcare information, such as, personal healthcare number. The entire use case has two activities; namely, *"Register Patient Record"* and *"Update Patient Record."* The *"Register Patient Record"* activity saves initial patient information into the relational database, while the *"Update Patient Record"* activity saves modification changes on already saved patient information into the relational database. The only actor responsible for this activity is the patient who has sole system privilege to login, and subsequently update his/her information.

Register and/or Update Physician Data

This use case outlines activities for saving and updating the physician information. Here, different types of physicians are identified; namely, clinical physician, primary care physician, secondary care physician, and dentist. The use case has 2 activities; namely, *"Register Physician Record"* and *"Update Physician Record."* The *"Register Physician Record"* activity saves initial physician information into the relational database, while the *"Update Physician Record"* activity saves modification changes on already saved physician information into the relational database. The actor that performs the use case activity is a clinical physician.

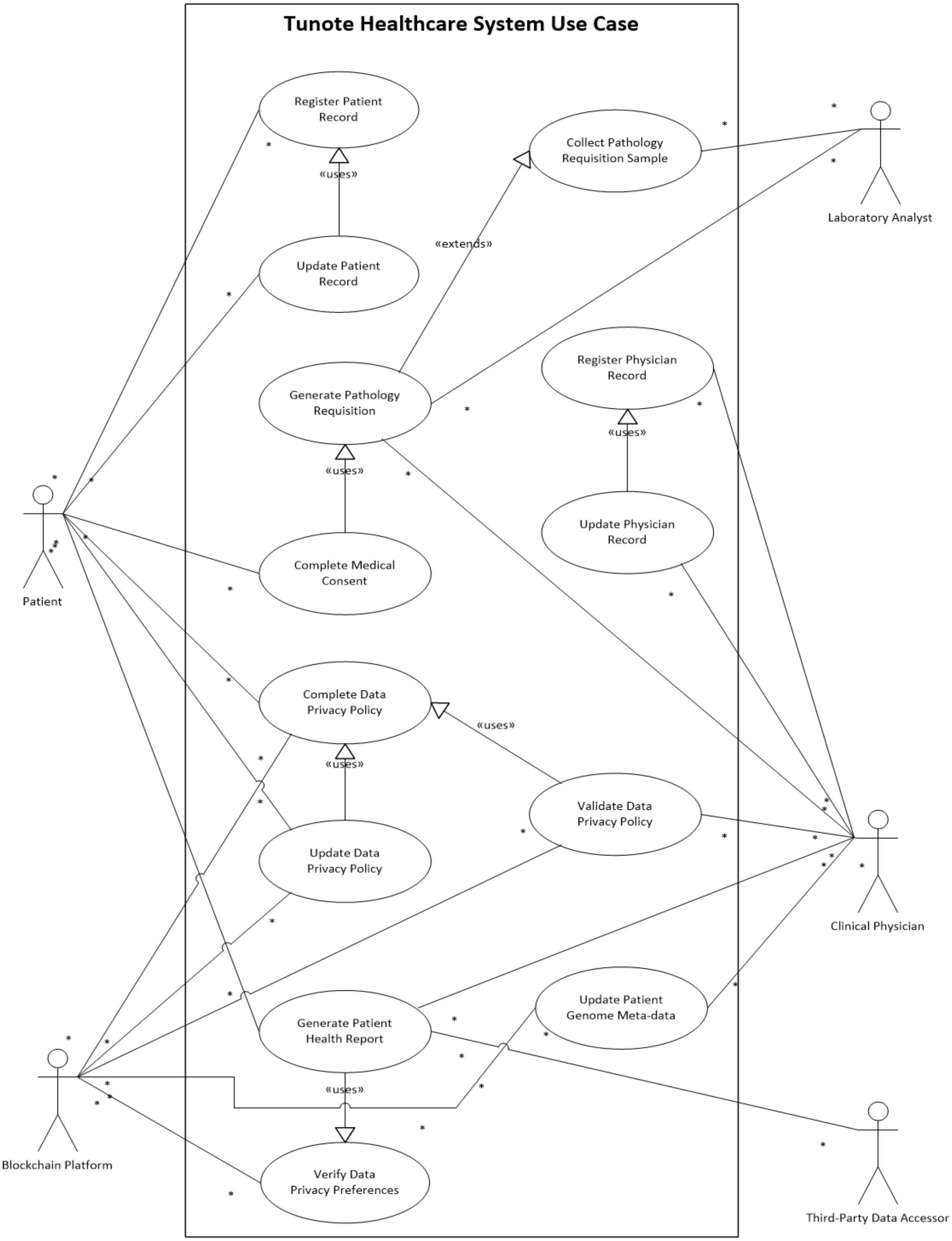

Tunote Healthcare System Use Case
Register Patient Record
«uses»
Update Patient Record
Collect Pathology Requisition Sample
«extends»
Laboratory Analyst
Generate Pathology Requisition
«uses»
Complete Medical Consent
Register Physician Record
«uses»
Update Physician Record
Patient
Complete Data Privacy Policy
«uses»
«uses»
Validate Data Privacy Policy
Update Data Privacy Policy
Clinical Physician
Update Patient Genome Meta-data
Generate Patient Health Report
«uses»
Verify Data Privacy Preferences
Blockchain Platform
Third-Party Data Accessor
*

Figure 4.4: System Use Case Modelling (of Actors and Activities)

Generate Pathology Requisition

The *"Generate Pathology Requisition"* use case saves into the relational database data pertaining to molecular pathology requisition for a patient. Different physician (that is, clinical, primary care, and secondary) information is captured to make the data provider healthcare information available to them, where necessary, for further or future analyses. Pathology information regarding CLS site, H&E, specimen data, and history or diagnosis data are saved as part of the requisition. All other cancer markers information needed for genomic analysis are saved. The use case activity requires and extends the performance of *"Collect Pathology Requisition Sample"* activity. This use case requires *"Laboratory Analyst"* and *"Clinical Physician"* actors to perform the activity.

Complete Medical Consent

The system modelling that captures data on completion and signing of medical consent is *"Complete Medical Consent"* use case activity. This use case modelling involves saving of patient and legal representative data, relationship type, and transmission medium of consent undertaking. Other details of consent witness and health insurance information are also saved into relational database. Moreover, information regarding dependent information and other medications and allergies, as well as consent date and confirmation sign are saved into the relational database. The use case activity requires and extends performance of a prior activity of *"Generate Pathology Requisition"*. The only actor for this use case activity is the patient.

Complete and/or Update Data Privacy Policy

This use case system modelling describes the task of signing off data privacy policy on attribute data. The use case has two activities namely, *"Complete Data Privacy Policy"* and *"Update Data Privacy Policy."* The *"Complete Data Privacy Policy"* activity saves initial data privacy policy for a data attribute into the relational database, whiles *"Update Data Privacy Policy"* activity saves modification changes on already saved data privacy policy on an attribute data into the relational database. Each of these use case activities requires and extends performance of a prior use case activity of *"Complete Medical Consent"*.

Each of the use case activities is performed by *"Patient"* and *"Blockchain Platform"* actors. The "Patient" actor saves primary privacy policy data into the relational database. The *"Blockchain Platform"* serves as an external data platform that saves a hashed form of the privacy policy data that is already saved in the relational database.

Validate Data Privacy Policy

The *"Validate Data Privacy Policy"* use case system modelling involves the confirmation procedures performed by the clinical physician (service provider and data collector) to authorize initial data privacy policy signed by the patient (data provider). The use case activity has two main functionalities. The first functionality involves specific data retrieval of pre-saved data privacy preferences. The second functionality involves either making changes on the pre-saved data preferences, and/or making final data save commitments of the data privacy policy preferences into the relational database by the clinical physician. The *"Validate Data Privacy Policy"* use case activity requires or extends a prior use case of *"Complete Data Privacy Policy"* activity for initial one-time data privacy policy validation. Additionally, this use case requires or extends the prior use case of *"Update Privacy Policy"* activity for a change modification of data privacy policy validation.

The use case activity is performed by *"Clinical Physician"* and *"Blockchain Platform"* actors. The *"Clinical Physician"* actor saves primary privacy policy data into the relational database. The *"Blockchain Platform"* serves as an external data platform that saves a hashed form of the privacy policy data that is already saved in the relational database.

Update Data Provider Genome Meta-data

The *"Update Data Provider Genome Meta-data"* use case system modelling involves saving encrypted patient genome meta-data into the relational database and blockchain platform. The use case activity is performed by *"Clinical Physician"* and *"Blockchain Platform"* actors. The *"Clinical Physician"* actor saves encrypted patient genome meta-data into the relational database. The *"Blockchain Platform"* serves as an external data platform that saves a copy of encrypted data that is already saved in the relational database.

Generate Data Provider Health Report

This use case system modelling generates intended data report to data accessors. The use case activity involves selecting data provider details, attribute data details, and data accessor details. Additionally, the privacy context for data retrieval is selected for authorized data retrieval. For each activity, a prior use case activity of *"Verify Data Privacy Preferences"* must be performed. This required and extended use case activity authorize data retrieval by retrieving hashed data privacy preferences (saved on the *"Blockchain Platform"*) for the attribute data and comparing with privacy preferences saved in the database.

Upon validate authorization of the privacy preferences, the details are used to generate correct data values for a data provider healthcare report. The use case activity is performed by different data accessors in the form of *"Patient"*, *"Clinical Physician"* and *"Third-Party Data Accessor"* actors.

4.2.2 Programming of User Interfaces, Web Data Processing, and Relational Database

The user interfaces are programmed using the PHP (PHP: Hypertext Preprocessor) [105] web development application tool. We adopt PHP (version 7.3.21) for system development and implementation. PHP is a general-purpose server-side scripting language embedded in HTML (Hypertext Markup Language) and suited to web development. It is used to manage dynamic content, databases, and session tracking. Figure 4.5 illustrates the system architecture's server-side programming.

PHP code is usually processed on a web server by a PHP interpreter implemented as a module, a daemon or as a Common Gateway Interface (CGI) executable. We adopt an Apache HTTP (Hypertext Transfer Protocol) web server (version 2.4.46) [106], which is configured to use PHP's CGI executable to process all requests to PHP files. On the web server, the result of interpreted and executed PHP code (which may be any type of data, such as, generated HTML) forms the whole or part of an HTTP response. The HTML response is then transmitted to an integrated database, file management system, email system, or HTML browser interface.

With the functionality of PHP being an object-oriented and procedural language, the operability to create dynamic content and transfer data to interact with databases is easily achieved. The integration is done with several popular databases, including MySQL, PostgreSQL, Oracle, and Microsoft SQL Server. In our system implementation, we employ MySQL relational database management system (DBMS) alongside (HTML client-side) user interfaces and (web server-side) PHP scripting language.

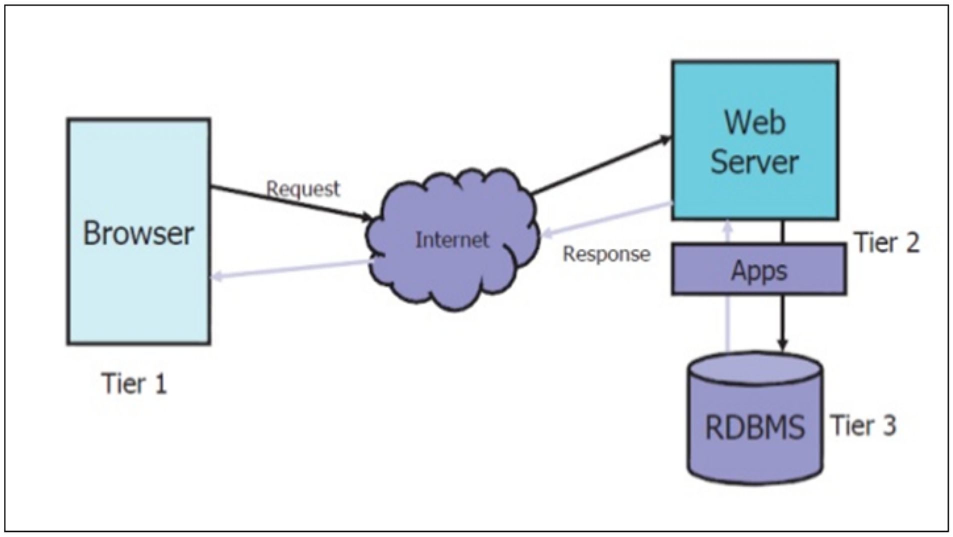


Figure 4.5: System Architecture of Server-Side Programming

The database integration with the web server is facilitated by native PHP functions, such as, *mysqli*, *mysql_connect*, and *mysql_query*. Through this integration, we are able to add, delete, and modify data elements within the database through PHP scripts. The adoption of MySQL relational database (version 8.0.21) [107] provides the platform for data processing on related information on data provider biographical and demographic information, physician data, and data provider genome metadata. Additionally, processed data transactions regarding molecular pathology requisition, data provider medical consent, and (validated) data privacy policy preferences are transmitted to the MySQL relational database.

In terms of data storage, management, and query processing, native data objects in MySQL relational database are adopted. These are databases, tables, and stored procedures. Three different databases are created in the relational database integration. These are *user_admin*, *tunote_ppdb*, and *tunote_ngs_genomics*. The *user_admin* database serves to process and store data on system user login details and access privileges. This database contains two relational tables. The *tunote_ppdb* database processes and stores data on data provider privacy policy preferences, such as, attribute category data, data accessor profiles, granularity privacy, purpose use privacy, purpose context privacy, and visibility privacy. The database contains 16 relational tables. The *tunote_ngs_genomics* database processes and stores data on content data relating to data provider information, physician information, pathology requisition details, and medical consent details. The database contains 9 relational tables. The query processing functionality of the system is executed using a stored procedure (which acts as a privacy-aware API over the database) to process and execute queries. The stored procedure generates a privacy-aware SQL query statement and results to the data accessor.

4.2.3 Extraction of Attribute Data, Data Accessor, and Privacy Preferences Data Elements for Privacy Tuples

The user interfaces provide a platform to collecting data values from the data provider (*i.e.*, patient) and other system users (such as, clinical physician, laboratory analyst). Most of the data values collected are classified as data provider private sensitive data. For an efficient application of privacy policy on attribute data, we categorize each of the attribute data elements and their associated data values. These are: *data provider consent witness*, *data provider health insurance*, *data provider demographics*, *data provider biographical information* (medium-sensitive), *data provider biographical information* (highly-sensitive), *data provider healthcare record*, *data provider pathology requisition*, and *data provider genome meta-data*. For example, *data provider demographics* attribute category data comprises *street*, *city*, *province*, and *postal code*. Additionally, *data provider health insurance* attribute data comprises *health insurance carrier*, *health policy number*, and *health group number*.

Each attribute category data value is assigned an *ObjectId* which serves to uniquely identify the attribute category data tuple and all constituent attribute data values needed for data retrieval. For each data value that is collected and permanently saved in the relational database, related attribute data elements are saved in the database. Privacy-preservation procedures are applied and enforced on the attribute data elements for storage, management, sharing and disclosure of their related data values to intended data accessors. Hence, a privacy-aware set of data values are presented to data accessors during query processing and data retrieval.

In terms of data accessors, different types of accessors are identified. These are: *clinical physician*, *patient*, *laboratory analyst*, and third-party *allied health*. Each data accessor data value is uniquely assigned an *ObjectId*. Each *ObjectId* serves as a tuple identifier for associated data accessor data values of *DataSecurityClassificationLevel*, *DataAccessorPurpose*, and *DataAccessorDeliveryType*. For example, the clinical physician is assigned an *ObjectId* as '1', *DataSecurityClassificationLevel* as 'Level-4', *DataAccessorPurpose* as 'Add-View-Update', and *DataAccessorDeliveryType* as 'OnlineSoftcopy'.

Data privacy policy preferences are extracted from the user interfaces and saved in the relational database. On the user interface, a data provider makes a selection of different data privacy preferences for the combination of each category of attribute data and each data accessor. As a result, each selection of a privacy policy preference is assigned an *ObjectId*, that is instantiated as a data value. The composition of the object data values of attribute category data, privacy preferences, and data accessor then constitutes a data privacy tuple for the data provider pertaining to a particular attribute category data. The data values are saved and permanently stored in the relational database.

4.2.4 Tight-Coupling of Data Elements to Generate Privacy Tuple

The data collector writes the privacy policy in agreement with the data provider on the terms of data collection, data access, and third-party data sharing, *etc.* For each privacy policy item, the attribute data, the data accessor profile, and the privacy preferences are each uniquely identified by *ObjectId* data values. A privacy tuple is subsequently generated from these three data values by coupling the data values.

The privacy tuple on each attribute data value is generated by combining data values from attribute data, data accessor, and privacy preferences. The privacy tuple is assigned a unique object identifier value; that is *ObjectId* value. The *ObjectId* value of the privacy tuple, together with the constituent data values, are saved, and permanently stored in the relational database. For purposes of time-stamping privacy tuple, the current date is saved alongside the privacy tuple in the relational database.

4.2.5 Transmission of Privacy Tuple to Blockchain Platform

Storage of privacy tuples and privacy preferences occurs in two different storage media. One form of storage is performed in the relational database. The storage of the actual privacy tuples in the relational database offers to control and authenticate access to data provider private and transactional data stored in the relational database by data accessors. The second storage occurs on the blockchains. The blockchains offers a data platform for storage and management of the hashed privacy tuples and privacy preferences that ensure data validation and stable data provenance. Through this approach, the blockchain platform offers an immutable medium where hashed data provider privacy tuple and privacy preference data values are preserved and protected from direct public access and against unconsented changes. Additionally, hashed privacy tuples stored on the blockchain offer an approach so that any unauthorized modifications can be detected (thereby providing data integrity assurance).

Data values generated from data provider privacy preferences and privacy tuple are combined into single data value. The resultant data value is hashed, transmitted through an encrypted communication medium, and saved on the blockchain platform. In this approach, the adoption of hashing methodology on the privacy data values offers an additional data security layer to the privacy data values so that data provider privacy preferences are managed in an efficient and most-secured manner. The hashing process employs *Argon2i* algorithm. The adoption of this algorithm offers a better and secure mechanism as opposed to a simple hash function. Moreover, the hashing algorithm presents an efficient hashing procedure and offers an effective defence against possible adversarial attacks, such as, offline dictionary attacks (in which an adversary computes a set of hashes of likely privacy tuples and looks for a match in the blockchain). The hashed privacy tuples stored in the blockchains are verified against the privacy tuples stored in the relational database during query retrieval process. The generated hash value contains all information essential to verify the hash. Hence, verification of hash value does not require separate storage for algorithm information.

4.2.6 Transmission of Genome Meta-data to Blockchain Platform

Data providers (*i.e.*, patient) genome meta-data values are extracted and bound together into a single tuple. This tuple comprises of meta-data, such as, data provider healthcare details, pathology requisition details, genome vcf standard version, variant caller name, variant caller version, variant call date, variant caller regions, annotation software name, annotation software version, annotation date, annotation reference, annotation reference version, and annotation fields, *etc.* The meta-data tuple value is encrypted before it is stored in the blockchains. We employ native PHP encryption libraries (*i.e.*, *Libsodium*) [116, 117] for the encryption procedure of the meta-data tuple. This encryption offers an additional data security to the private, sensitive genome meta-data of data providers during their storage in the blockchains. We discuss the details of genome meta-data encryption using *Libsodium* encryption in Section 4.4.4.

The encrypted meta-data tuple is transmitted to the blockchain platform. The transmission is facilitated by an encrypted data communication medium between the user interface and blockchains. For robust and efficient data transmission encryption, we adopt TLS (Transport Layer Security) protocol - with TLS_AES_256_GCM_SHA384 cipher suite - to ensure secure data communications for all forms of data packets transmission between system components.

4.3 Modelling, Design and Development of Relational Database

The relational database is an integral component of the overall system design and implementation. The database stores and manages all related data provider data and transaction processing data by both data provider and data collector (*i.e.*, service provider). We develop different databases to manage different forms of data, as part of system modelling and design.

In this section, we describe modelling and design of databases, relational entities, and their associated relationships. Moreover, we describe various data objects that are useful in the operations of the relational database component, and discuss database security, access control, and user permissions and privileges.

4.3.1 Relational Entities and Relationships Modelling, Database Development

We adopt conventional Entity Relationship Model to design the relational database. In the database development, we adopt MySQL relational DBMS [107]. Three different databases are developed. These are *user_admin*, *tunote_ppdb*, and *tunote_ngs_genomics*. The *user_admin* database serves to process and store data on system user login details and access privileges. This database contains two relational tables: *user_login* and *user_audit_trail*.

The *tunote_ngs_genomics* database processes and stores content data relating to data provider (*i.e.*, patient) information, physician information, pathology requisition details, and medical consent and privacy details. The database contains five relational tables. The tables include: *genome*, *patient*, *physician*, *requisition*, and *consent*. Moreover, the entity relationships are represented by *patient_physician*, *physician_requisition*, *patient_requisition*, and *patient_requisition_consent*. Figure 4.6 illustrates the relational model for data provider healthcare information and genome meta-data.

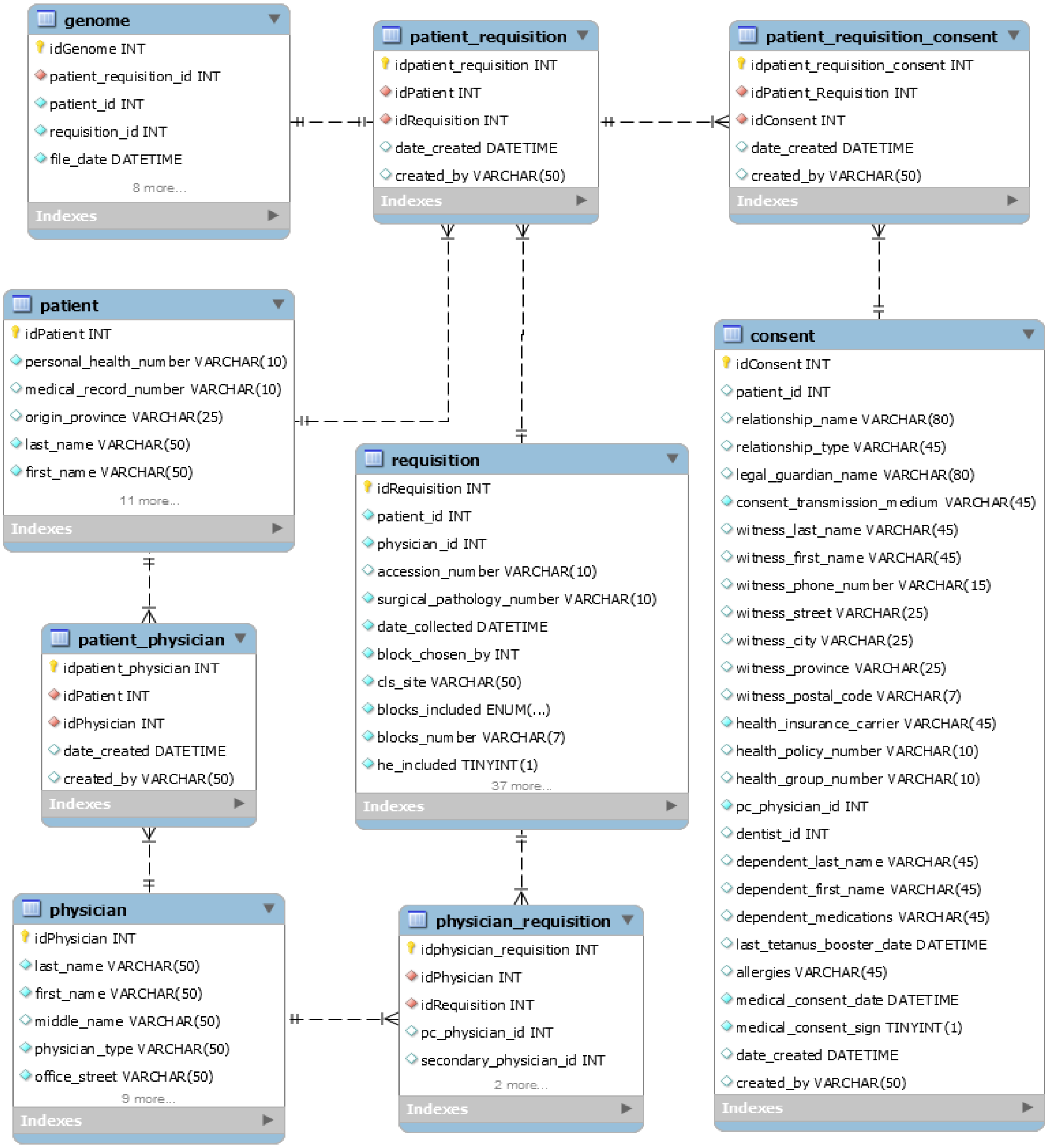

genome
idGenome INT
patient_requisition_id INT
patient_id INT
requisition_id INT
file_date DATETIME
8 more...
Indexes
patient_requisition
idpatient_requisition INT
idPatient INT
idRequisition INT
date_created DATETIME
created_by VARCHAR(50)
Indexes
patient_requisition_consent
idpatient_requisition_consent INT
idPatient_Requisition INT
idConsent INT
date_created DATETIME
created_by VARCHAR(50)
Indexes
patient
idPatient INT
personal_health_number VARCHAR(10)
medical_record_number VARCHAR(10)
origin_province VARCHAR(25)
last_name VARCHAR(50)
first_name VARCHAR(50)
11 more...
Indexes
consent
idConsent INT
patient_id INT
relationship_name VARCHAR(80)
relationship_type VARCHAR(45)
legal_guardian_name VARCHAR(80)
consent_transmission_medium VARCHAR(45)
witness_last_name VARCHAR(45)
witness_first_name VARCHAR(45)
witness_phone_number VARCHAR(15)
witness_street VARCHAR(25)
witness_city VARCHAR(25)
witness_province VARCHAR(25)
witness_postal_code VARCHAR(7)
health_insurance_carrier VARCHAR(45)
health_policy_number VARCHAR(10)
health_group_number VARCHAR(10)
pc_physician_id INT
dentist_id INT
dependent_last_name VARCHAR(45)
dependent_first_name VARCHAR(45)
dependent_medications VARCHAR(45)
last_tetanus_booster_date DATETIME
allergies VARCHAR(45)
medical_consent_date DATETIME
medical_consent_sign TINYINT(1)
date_created DATETIME
created_by VARCHAR(50)
Indexes
requisition
idRequisition INT
patient_id INT
physician_id INT
accession_number VARCHAR(10)
surgical_pathology_number VARCHAR(10)
date_collected DATETIME
block_chosen_by INT
cls_site VARCHAR(50)
blocks_included ENUM(...)
blocks_number VARCHAR(7)
he_included TINYINT(1)
37 more...
Indexes
patient_physician
idpatient_physician INT
idPatient INT
idPhysician INT
date_created DATETIME
created_by VARCHAR(50)
Indexes
physician
idPhysician INT
last_name VARCHAR(50)
first_name VARCHAR(50)
middle_name VARCHAR(50)
physician_type VARCHAR(50)
office_street VARCHAR(50)
9 more...
Indexes
physician_requisition
idphysician_requisition INT
idPhysician INT
idRequisition INT
pc_physician_id INT
secondary_physician_id INT
2 more...
Indexes

Figure 4.6: Data Provider Healthcare Information Relational Model - *Tunote_ngs_genomics* Database

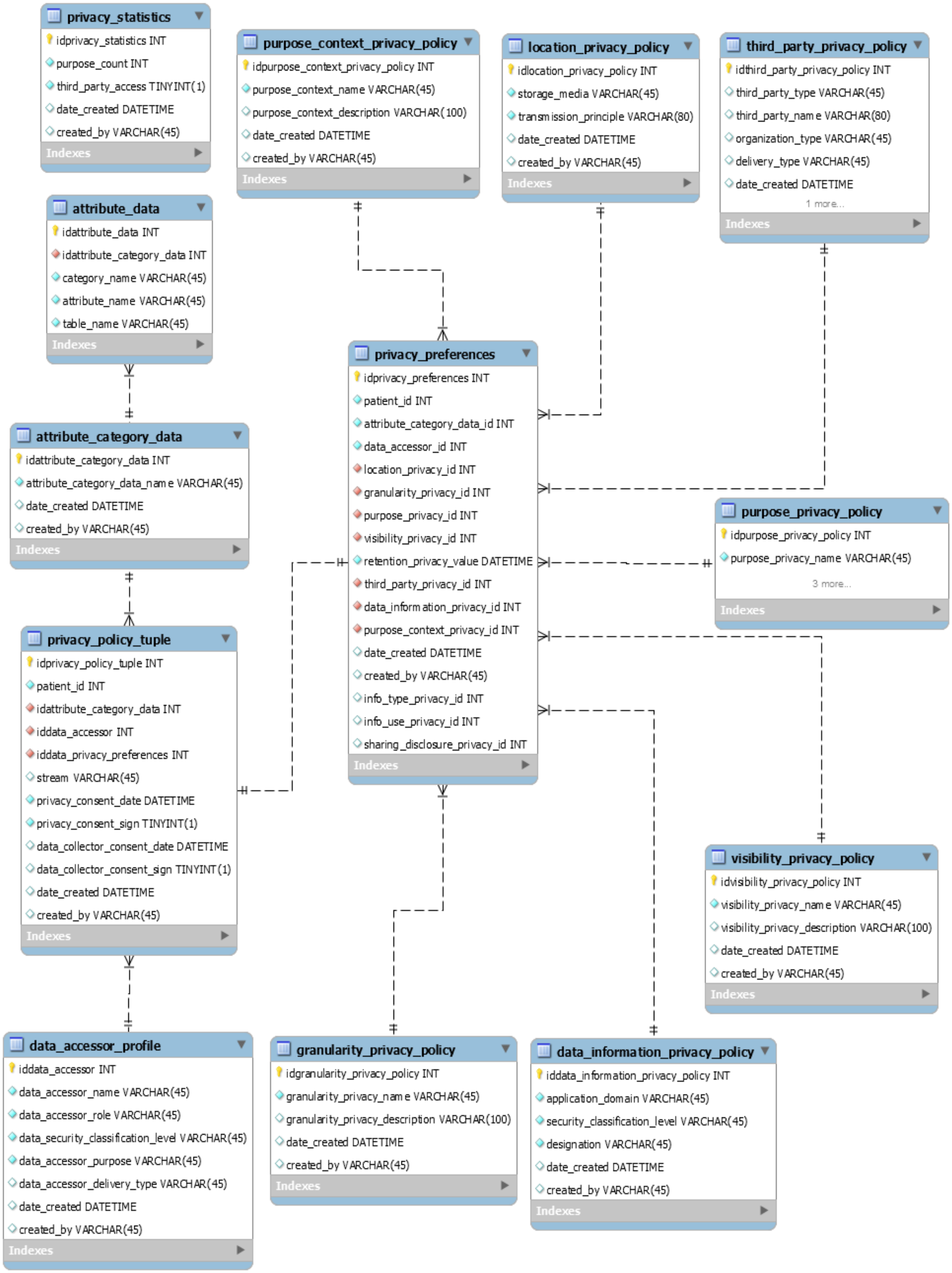

privacy_statistics
idprivacy_statistics INT
purpose_count INT
third_party_access TINYINT(1)
date_created DATETIME
created_by VARCHAR(45)
Indexes
purpose_context_privacy_policy
idpurpose_context_privacy_policy INT
purpose_context_name VARCHAR(45)
purpose_context_description VARCHAR(100)
date_created DATETIME
created_by VARCHAR(45)
Indexes
location_privacy_policy
idlocation_privacy_policy INT
storage_media VARCHAR(45)
transmission_principle VARCHAR(80)
date_created DATETIME
created_by VARCHAR(45)
Indexes
third_party_privacy_policy
idthird_party_privacy_policy INT
third_party_type VARCHAR(45)
third_party_name VARCHAR(80)
organization_type VARCHAR(45)
delivery_type VARCHAR(45)
date_created DATETIME
1 more...
Indexes
attribute_data
idattribute_data INT
idattribute_category_data INT
category_name VARCHAR(45)
attribute_name VARCHAR(45)
table_name VARCHAR(45)
Indexes
privacy_preferences
idprivacy_preferences INT
patient_id INT
attribute_category_data_id INT
data_accessor_id INT
location_privacy_id INT
granularity_privacy_id INT
purpose_privacy_id INT
visibility_privacy_id INT
retention_privacy_value DATETIME
third_party_privacy_id INT
data_information_privacy_id INT
purpose_context_privacy_id INT
date_created DATETIME
created_by VARCHAR(45)
info_type_privacy_id INT
info_use_privacy_id INT
sharing_disclosure_privacy_id INT
Indexes
attribute_category_data
idattribute_category_data INT
attribute_category_data_name VARCHAR(45)
date_created DATETIME
created_by VARCHAR(45)
Indexes
purpose_privacy_policy
idpurpose_privacy_policy INT
purpose_privacy_name VARCHAR(45)
3 more...
Indexes
privacy_policy_tuple
idprivacy_policy_tuple INT
patient_id INT
idattribute_category_data INT
iddata_accessor INT
iddata_privacy_preferences INT
stream VARCHAR(45)
privacy_consent_date DATETIME
privacy_consent_sign TINYINT(1)
data_collector_consent_date DATETIME
data_collector_consent_sign TINYINT(1)
date_created DATETIME
created_by VARCHAR(45)
Indexes
visibility_privacy_policy
idvisibility_privacy_policy INT
visibility_privacy_name VARCHAR(45)
visibility_privacy_description VARCHAR(100)
date_created DATETIME
created_by VARCHAR(45)
Indexes
data_accessor_profile
iddata_accessor INT
data_accessor_name VARCHAR(45)
data_accessor_role VARCHAR(45)
data_security_classification_level VARCHAR(45)
data_accessor_purpose VARCHAR(45)
data_accessor_delivery_type VARCHAR(45)
date_created DATETIME
created_by VARCHAR(45)
Indexes
granularity_privacy_policy
idgranularity_privacy_policy INT
granularity_privacy_name VARCHAR(45)
granularity_privacy_description VARCHAR(100)
date_created DATETIME
created_by VARCHAR(45)
Indexes
data_information_privacy_policy
iddata_information_privacy_policy INT
application_domain VARCHAR(45)
security_classification_level VARCHAR(45)
designation VARCHAR(45)
date_created DATETIME
created_by VARCHAR(45)
Indexes

Figure 4.7: Data Provider Data Privacy Relational Model - *Tunote_ppdb* Database

The *tunote_ppdb* database processes and stores data on data provider (*i.e.*, patient) data privacy policy preferences, such as, attribute category data, data accessor profiles, and granularity privacy. The database contains 16 relational tables: *privacy_statistics*, *attribute_data*, *attribute_category_data*, *privacy_policy_tuple*, *data_accessor_profile*, *privacy_preferences*, *purpose_context_privacy*, *location_privacy*, *third_party_privacy*, *purpose_use_privacy*, *visibility_privacy*, *granularity_privacy*, and *data_information_privacy.* Figure 4.7 illustrates the relational model on data provider privacy policy preferences on healthcare data and genome meta-data.

In the relational model for data privacy preferences, two key relational tables are required: namely, *privacy_preferences* and *privacy_policy_tuple.* The *privacy_preferences* table stores data on all data provider data privacy preferences for each attribute data. This relational table has foreign key relationships with related tables for privacy preferences, such as, *purpose_context_privacy*, *granularity_privacy*, and *visibility_privacy.* On the other hand, the *privacy_policy_tuple* table stores data on each privacy tuple for every related attribute data, data accessor, and privacy preference. The relational table has foreign key relationships with *attribute_category_data*, *data_accessor_profile*, and *privacy_preferences* tables.

4.3.2 Development and Programming of Database Objects

Programming the database involves identifying all components necessary to define the logical data structure and architecture of the relational database. We identify and discuss the following data objects in the database development: tables, indexes, and stored procedures. Relational tables and indexes are created to hold data, and these are for database data manipulation. Moreover, stored procedures consist of a logical definition that help in creating and modifying other data objects.

Different relational tables are created, and for each table, we identify a unique column to function as a *primary key* field. The creation of a *primary key* automatically creates index for the table data object. The index helps in storage and management of tuple data in the tables. Additionally, the index helps facilitate data retrieval and query processing. For example, in relational table *Patient*, a unique column field *idPatient* is assigned as a primary key field. Additionally, *idPatient_idx* is created as an index.

A stored procedure is created as part of programming the relational database. This stored procedure serves as a privacy-aware API for query analyses, parsing, and transformation. Additionally, the privacy-aware API helps to execute queries and present privacy-aware query results to the data accessor. In this way, the privacy-aware stored procedure offers functionalities of data validation and access-control mechanisms.

4.3.3 Privacy-Aware Query Processing

We describe the procedures for privacy-aware query processing in this section. We process queries using database stored procedures. Our adoption of stored procedures for privacy-aware query processing is based on the merits of: reusable query code, efficient query response where executable code is automatically cached, increase in data retrieval scalability by isolating application processing on the server, and allows users to manipulate data through secured access privileges.

Within the stored procedure, the privacy-aware API query analyzer and privacy-aware API query processing are implemented (recall Section 3.7 and Figure 3.7). The data logic for privacy-aware query processing on data provider privacy preferences are programmed in the stored procedure. The stored procedure accepts parameters in the form of data provider privacy preferences object identifier values. The output from the stored procedure is a set of privacy-aware data tuples based on the input parameters.

We define a stored procedure (*i.e.*, *GetPrivacyAwareUserData*) which accepts parameter data values, process query statements, and outputs privacy-aware data tuples. The complete definition of the stored procedure is outlined in a Github repository (see https://github.com/mikemireku/PSecTunote). The input parameters are: *data_provider_id*, *attribute_category_data_id*, *data_accessor_id*, *data_info_privacy_id*, *privacy_retention_date*, *granularity_privacy_id*, *visibility_privacy_id*, *purpose_use_id*, and *purpose_context_id*. To process privacy-aware queries, we execute the following statement:

CALL tunote_ngs_genomics.GetPrivacyAwareUserData (*data_provider_id*, *attribute_category_data_id*, *data_accessor_id*, *data_info_privacy_id*, *privacy_retention_date*, *granularity_privacy_id*, *visibility_privacy_id*, *purpose_use_id*, *purpose_context_id*);

A typical instance data query processing is executed as:

CALL tunote_ngs_genomics.GetPrivacyAwareUserData (4, 1, 3, 3, ’2021-09-01’, 2, 2, 2, 1);

where the object identifier data values (4, 1, 3, 3, ’2021-09-01’, 2, 2, 2, 1) represent the parameters in their respective order as defined in the stored procedure.

4.3.4 Database Security, Access Controls, and User Permissions

We enforce security over the relational database which delivers effective database access, use, and administration. The creation of access control and user permissions allow access to private data to authorised persons (*i.e.*, database users) while restricting access to unauthorised ones. This process involves two components of authentication and authorisation. Authentication verifies the identity of a person who is accessing your database, whiles authorisation determines whether a user should be allowed to access the data or execute transactions.

The database administrator is the only permitted user role (*i.e.*, root) to access and manage the database in our relational database implementation. Database access control permissions and privileges are categorised into *account limits*, *administrative roles*, and *schema privileges. Account limits* specifies parameters, such as, *Maximum Queries* (number of queries the account can execute within one hour), *Maximum Updates* (number of updates the account can execute within one hour), *Maximum Connections* (number of times the account can connect to the server per hour), and *Concurrent Connections* (number of simultaneous connections to the server the account can have).

The *Administrative roles* access control specifies parameters, such as, *DBA* (grants the rights to perform all tasks), *MaintenanceAdmin* (grants rights needed to maintain the server), *SecurityAdmin* (rights to manage logins and grant and revoke server and database level permissions), *BackupAdmin* (minimal rights needed to backup any database), *DBManager* (grants full rights on all databases), and *DBDesigner* (rights to create and reverse engineer any database schema).

The *Schema privileges* access control specifies *Object Rights* (*i.e.*, select, insert, update, delete, execute, show view), *DDL Rights* (*i.e.*, create, alter, references, index, create routine, create view, drop, trigger), and *Other Rights* (*i.e.*, grant option, create temporary tables, lock tables) on the database schema.

4.4 Design and Development of Permissioned (Private) Blockchain Platform

We adopt a blockchain platform to offer data access authentication and authorize query processing on data provider private, sensitive data and transaction data to data accessors. In this approach, the user interface generates and issues data service requests to the blockchain platform, and subsequently receive data responses from the blockchain platform to the user interface. The procedures of data privacy transactions are performed through a blockchain API wrapper; which serves as an intermediary between the blockchain platform, encrypted data communication channel, user interfaces, and relational database.

In this section, we discuss the adoption of Multichain blockchain platform, design and development of data stream transaction ledgers, generation and confirmation of privacy policy chain blocks on data stream transaction ledgers, and generation and confirmation of genome metadata on data stream transaction ledgers.

4.4.1 Adoption of MultiChain Blockchain Platform

We adopt the MultiChain (version 2.1.2) [108] blockchain platform, which is one of the most prominent open-source platforms for private blockchain development. The blockchain platform is designed and based on a fork of the Bitcoin Core platform [109], the official client for Bitcoin network. MultiChain is an off-the-shelf platform for creation and deployment of private blockchains, either within or between organizations [110]. MultiChain is designed to offer functionalities which: (a) ensures that the blockchain's activity is only visible to chosen participants, (b) introduces controls over which transactions are permitted, and (c) enables mining to take place securely without proof-of-work (PoW) and its associated costs.

Creation of Data Streams and Data Items

The primary configuration of MultiChain blockchain transaction ledgers involves the design and creation of "data streams". The data stream feature allows users to create multiple key-value, time-series, or identity "databases" that can be used for data-sharing, time-stamping, and encrypted archiving [111]. Data streams are append-only, on-chain lists of data where the *key-value* retrieval capability makes store and query functionality extremely easy. Each data stream is an ordered list of items, with the following characteristics: one or more *publishers* (string data type) - who digitally signed the data item; and one or more *key-value* pairs (string data type) - which is set between 0-256 ASCII characters excluding whitespace and single/double quotes - to allow efficient retrieval. Other information about data streams are storage of related *data* (hex string) in JSON (JavaScript Object Notation), text or binary format - which can be on-chain (embedded in the transaction) or off-chain (represented by a hash in the transaction). Moreover, some other information are meta-data (about the transaction and block corresponding to the data item) in the form of transaction ID *txid* (string data type), *block-hash* (hex string), *block-time* (integer data type), and confirmations (integer data type) [111].

Our assessment of data streams identifies the creation of closed data streams. These data streams are restricted to changes by the host node address. General data stream configuration specifies some permissions set on the data streams when they are created. The *create* permission code syntax is used to create data streams by a special transaction output, which must only be signed with the host node address. In the blockchain setup for the privacy infrastructure, there is only one node address set up for transaction management. Hence, no other external node participates in creating data streams. The node address that creates data streams automatically has *admin*, *activate*, and *write* permissions for the streams.

Data Publishing, Subscription, and Retrieval

MultiChain data streams make it possible for a blockchain to implement some basic data management features of a general-purpose database. Storing data on the blockchain involves the process of *publishing* data. The data published in every stream is stored by all nodes in the network. Each data stream on a MultiChain blockchain consists of a list of data items. When data needs to be queried or ”streamed”, it can be retrieved by searches using the *key-value* pairs. Publishing a stream item to a data stream constitutes a transaction. When a node subscribes to a data stream, it indexes the stream items in different ways to enable fast retrieval, and the index entry points to the transaction ID.

The core value of data streams is in indexing and data retrieval. The node address *subscribes* (with a granted permission to retrieve transaction data) to the created data stream. After data stream subscription, the node can retrieve data items in a specified order and retrieve data items with a particular *key.* Our configuration of the blockchain API wrapper use *start* and *count* parameters; that allows subsections of long lists to be efficiently retrieved (like a LIMIT clause in SQL). Moreover, the adopted API wrapper offers functionalities to: (a) retrieve the first or last data item of a given stream entry, (b) retrieve a full data item history for a stream entry, and (c) retrieve information about multiple data item entries, including the first and last data items.

Transaction Access Privileges, Permissions Management, and Mining

In the design architecture of MultiChain, all privileges are granted and revoked using network transactions containing special meta-data. Transactions submitted to the network are time stamped via Linux timestamp. When a transaction happens, it is held in the memory pool. The miner of the first “genesis” block automatically receives all privileges, including administrator rights to manage the privileges of other users [70]. The mining process does not follow the traditional Proof-of-Work procedures adopted by public blockchains; where there is requirement of a difficult mathematical computation to create a new block. The mining and signing of validated transactions are completed by the primary node and all other connected nodes. The miner of a valid block must be on the permitted node list after applying all privilege changes defined within that block’s transactions. This mining process offers many advantages, such as: faster rate of transaction confirmations, and eliminates the risks of some miners refusing to confirm transactions for various reasons.

It will be noted that our implementation only uses only one node to process transaction data blocks; and which is only accessible via the API wrapper protocol. After transaction mining is complete, the transaction is added to a block. Each block has a maximum transaction size, (*i.e.*, after a block reaches its maximum size or the time to create a block reaches its limit), and the block is sealed and appended to the chain. This means a data stream in MultiChain can span multiple blocks based on the time of the transaction (*i.e.*, time of the publishing the data to the blockchain).

The architecture and implementation of MultiChain addresses keys and permission management based on identity and security using public key cryptography. User nodes randomly generate their own private keys and never reveal them to other participants. Each private key has a mathematically related public address which represents an identity. Once keys are sent to a public address, those asset funds can only be used using the corresponding private key to "sign" a new transaction. In this regard, access to a private key is equivalent to ownership of any asset funds which it protects [70]. This type of cryptography enables any message to be signed by a user node to prove that they own the private key corresponding to a particular address. MultiChain uses this property to restrict blockchain access to a list of permitted users, by expanding the "handshaking" process that occurs when two blockchain nodes connect. A description of the key management is as follows:

(a) Each node presents its identity as a public address on the permitted list.

(b) Each node verifies that the other's address is on its own version of the permitted list.

(c) Each node sends a challenge message to the other party.

(d) Each node sends back a signature of the challenge message, proving their ownership of the private key corresponding to the public address they presented.

4.4.2 Design of Transaction Ledgers and Data Streams for Privacy Tuples

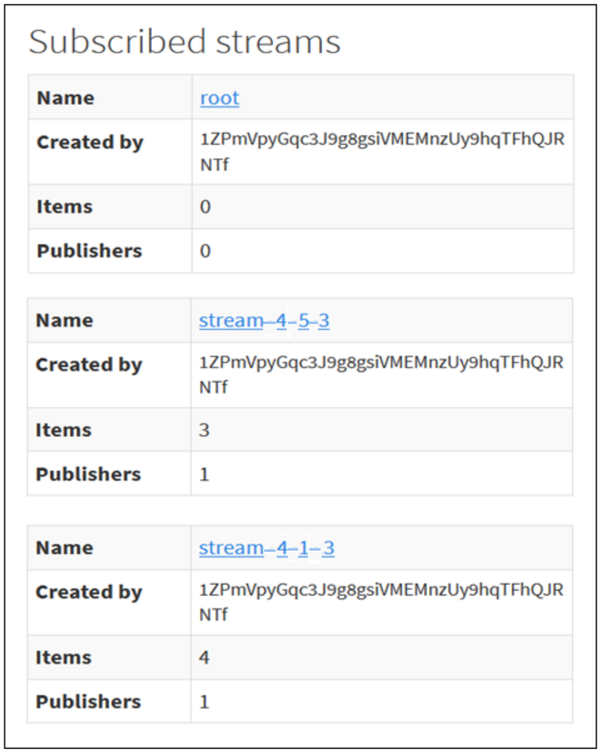
Privacy tuples created from the data privacy policy transaction represent privacy preferences validated by both data provider (*i.e.*, patient) and data collector or service provider (*i.e.*, clinical physician). These tuples denote unique attribute data, privacy preferences, and data accessor data elements for a particular data provider. Data streams are created on the MultiChain blockchain for each privacy tuple. The initial configuration of the blockchains consists of appending a *root (genesis)* data stream that does not contain transaction data block. Consequently, the first data stream is appended to the existing *root* data stream, and all other subsequent data streams are also appended to the preceding data stream. A validated transaction data block (associated to a privacy tuple) is appended to first data stream or any other related data stream (in the list of data streams created on the chain). Thus, privacy tuples stored as transaction blocks become valid data items and are always accessed during query processing.

The identification of data streams on the validated transaction ledger is based on a designated data provider. To this end, each data provider is associated with a unique data stream for privacy tuple data storage on the MultiChain blockchain platform. Moreover, each data provider is related to some attribute data, that is designated as private. Since there are many private attribute data, there will be different data streams for the same data provider, but on different attribute data. On another hand, each private attribute data for a particular data provider will have different privacy policy preferences for different data accessors. Based on these three data identifiers (that is, data provider, attribute data, and data accessor), we create data streams on the transaction ledger with respect to three parameters for efficient data retrieval from the blockchain platform.

The implementation approach for designing and creating data streams is as follows. We adopt a data representation format of: *Stream-X-Y-Z.* The X component represents a data value for supposed data provider, Y represents a data value for supposed attribute data, and Z represents a data value for the data accessor. This approach offers a unique data stream creation and identification on MultiChain blockchain platform. Furthermore, the methodology offers an efficient privacy tuple data storage for each data provider, the associated attribute data, and the data accessor. In terms of data retrieval and query processing on MultiChain blockchain platform, we reference the parameters using object identification data values to uniquely identify data streams created on the blockchain platform.

With regards to practical implementation creating data streams on the MultiChain blockchain platform, we use MultiChain API syntax: *create (”stream”, stream_name, True)*; to create a data stream from the API. The parameter *“create”* indicates the method invoked to execute processing on the blockchain platform. The *“stream”* parameter indicates the type of operation, whether data stream or wallet asset. The *“stream_name”* indicates the specific name assigned to the data stream. *“True”* is a value for *“rescan”* parameter that indicates the node will index all items from when the assets and/or streams are created. The *“create”* method can be incorporated with another parameter, such as, *“restrict”*; which can take values of *offchain*, *onchain*, *write*, or *read.*

Figure 4.8 illustrates a set of subscribed data streams, containing the *“root”* stream (which is automatically created) and user-created streams of *“stream-4-5-3”* and *“stream-4-1-3”*. It is noticed from the diagram that all three data streams are created by the same node address. Data streams *“stream-4-5-3”* and *“stream-4-1-3”* are designated to the same data provider with an object identifier value, *4*. On the other hand, each of the streams are associated with different attribute data with an object identifier value of *5* and *1*, respectively. Finally, each data stream is related to the same data accessor with an object identifier value of *3.* Moreover, *“stream-4-5-3”* stores three data items and has one node address data publisher. Likewise, *“stream-4-1-3”* stores four data items and has one node address data publisher.



Subscribed streams

Name	root
Created by	1ZPmVpyGqc3J9g8gsiVMEMnzUy9hqTFhQJRNTf
Items	0
Publishers	0

Name	stream-4-5-3
Created by	1ZPmVpyGqc3J9g8gsiVMEMnzUy9hqTFhQJRNTf
Items	3
Publishers	1

Name	stream-4-1-3
Created by	1ZPmVpyGqc3J9g8gsiVMEMnzUy9hqTFhQJRNTf
Items	4
Publishers	1

Figure 4.8: MultiChain Blockchain (Subscribed) Data Streams

4.4.3 Publish and Confirmation of Privacy Tuple Data Items on Data Streams

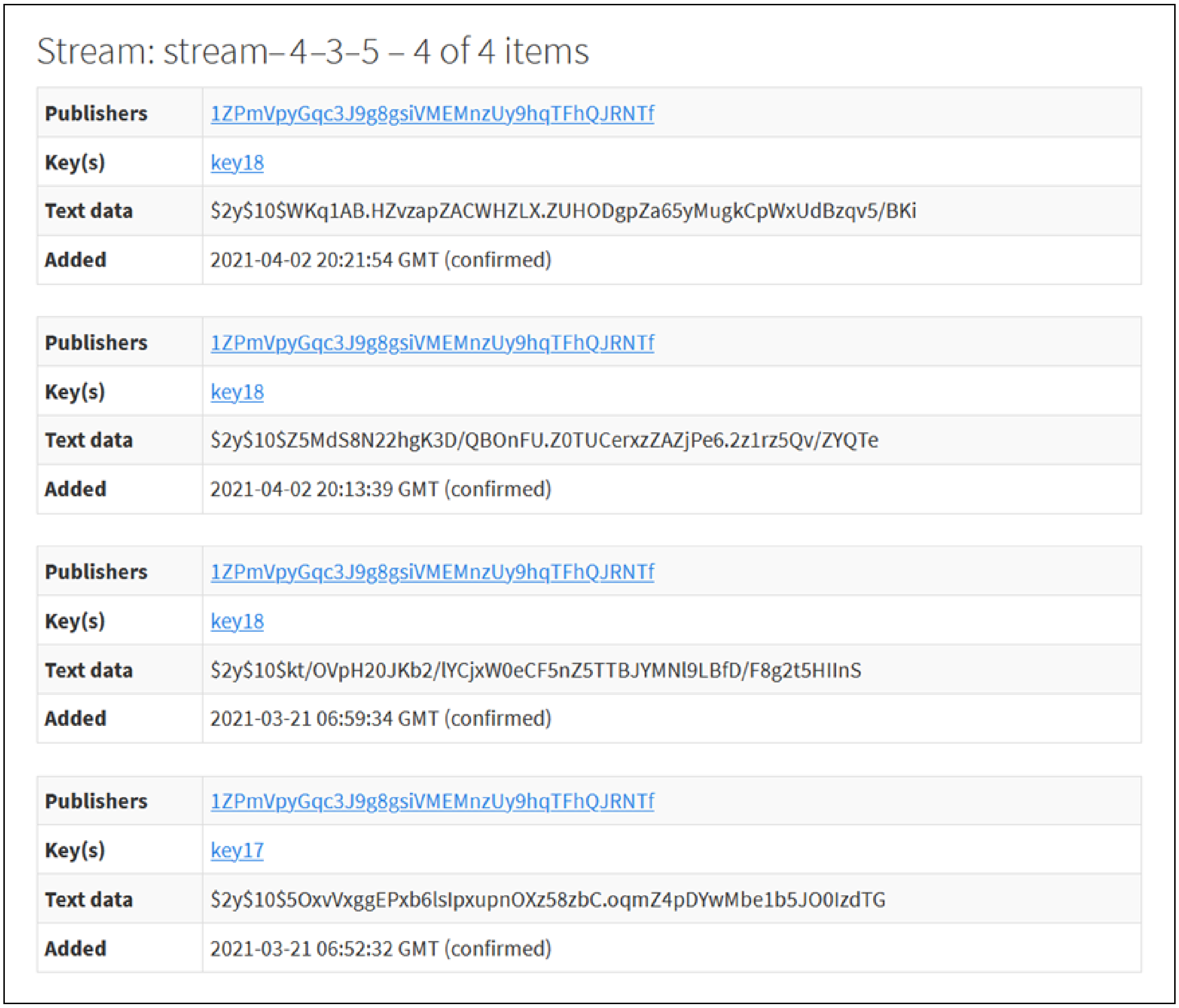
Publishing data is the methodology for saving and storing data value items on the MultiChain blockchain. To publish a data value item requires initial creation of a data stream on which the data value item can be stored. Hence, the process of publishing data value items involves identifying a data stream, specifying a key value, and specifying the data value item. The unique feature of specifying a key value helps to efficiently sign the data value item on the blockchain platform. The key value also helps in data retrieval for the data value item.

When privacy tuple data items are generated into the data streams, there is a need to permanently persist them in the data stream. The activity ensures that every ”created” privacy tuple data item is permanently stored in the blockchain platform. Moreover, storing a privacy tuple data item guarantees its availability for data retrieval and querying when needed. Publishing a privacy tuple data item requires the identification of a particular data stream (which is already created and related to a unique data provider, attribute data, and data accessor). Additionally, we identify the unique privacy tuple data item for data publishing. One important requirement for data item publishing is the identification of a data token or key. In the architecture of MultiChain blockchain with regards to data publishing, the key token helps to uniquely save and effectively store the data items in the data streams. Moreover, the key value helps in efficient data retrieval and querying processing on the data items. Hence, three parameters needed to publish a data item in the data stream are: data stream, data item, and key token.

As part of implementation, we adopt a hashing algorithm to transform privacy tuple data items before finally saving them on the blockchains platform. We implement hashing on the data items based on the fact that data items are stored in the data streams in their raw form. This means that data items are openly exposed to anyone who accesses and queries data streams and their constituent data items. We implement native hash algorithm offered by PHP Interpreter for web development. We use *bcrypt* algorithm (in the form of *Argon2i*) to transform the privacy tuple and privacy preferences data. The resultant hashed value contains the hash algorithm, hash cost and salt. Therefore, all information that is needed to verify the hash is included in the resultant hash value. Hence, during hash value retrieval from the blockchains, and subsequent verification for query processing, the function to verify the hash does not need separate storage for the salt or algorithm information.

For practical implementation on the MultiChain blockchain platform, we use MultiChain API syntax: *publish (stream_name, stream_key, stream_data)*; to publish or store a data item in the data stream. The parameter *"publish"* indicates the method invoked to execute processing on the blockchain platform. The *"stream_name"* indicates the specific name data stream into which data is stored, and *"stream_key"* parameter indicates the data value of key that is used to sign the storage of data item on the data stream. The *"stream_data"* indicates the actual data value item saved on the data stream.

Figure 4.9 illustrates the set of data value items for *"stream-4-3-5"*. Each storage item outlines data publisher, key value, actual data value item (which is a text format), and storage date (which timestamps the date and time for save the data value item). The data publisher references the blockchain node address into which the data stream and data item are created and stored. The key token is used to digitally sign and save the data item into the data stream. The added time designation is the timestamp for the calendar date and time at which the data item is saved into the blockchain platform. It is observed (from the timestamp) in the illustration that ordering of the data value items is in an ascending order from the bottom. Hence, the latest data value item is displayed at the top. We notice that all the data storage streams are published by the same node address. Moreover, the first data value item is signed with a key value of *"key17"* whereas all subsequent data value items are signed with *"key18"*. It is also noted that all the data value items are "confirmed" by the subscribing nodes to the data stream. This indicates the data value item has been accepted as a valid data from each subscribing node.



Stream: stream–4–3–5 – 4 of 4 items

Publishers	1ZPmVpyGqc3J9g8gsiVMEMnzUy9hqTFhQJRNTf
Key(s)	key18
Text data	$2y$10$WKq1AB.HZvzapZACWHZLX.ZUHODgpZa65yMugkCpWxUdBzqv5/BKi
Added	2021-04-02 20:21:54 GMT (confirmed)

Publishers	1ZPmVpyGqc3J9g8gsiVMEMnzUy9hqTFhQJRNTf
Key(s)	key18
Text data	$2y$10$Z5MdS8N22hgK3D/QBOnFU.Z0TUCerxzZAZjPe6.2z1rz5Qv/ZYQTe
Added	2021-04-02 20:13:39 GMT (confirmed)

Publishers	1ZPmVpyGqc3J9g8gsiVMEMnzUy9hqTFhQJRNTf
Key(s)	key18
Text data	$2y$10$kt/OVpH20JKb2/lYCjxW0eCF5nZ5TTBJYMNl9LBfD/F8g2t5HIInS
Added	2021-03-21 06:59:34 GMT (confirmed)

Publishers	1ZPmVpyGqc3J9g8gsiVMEMnzUy9hqTFhQJRNTf
Key(s)	key17
Text data	$2y$10$5OxvVxggEPxb6lsIpxupnOXz58zbC.oqmZ4pDYwMbe1b5JO0IzdTG
Added	2021-03-21 06:52:32 GMT (confirmed)

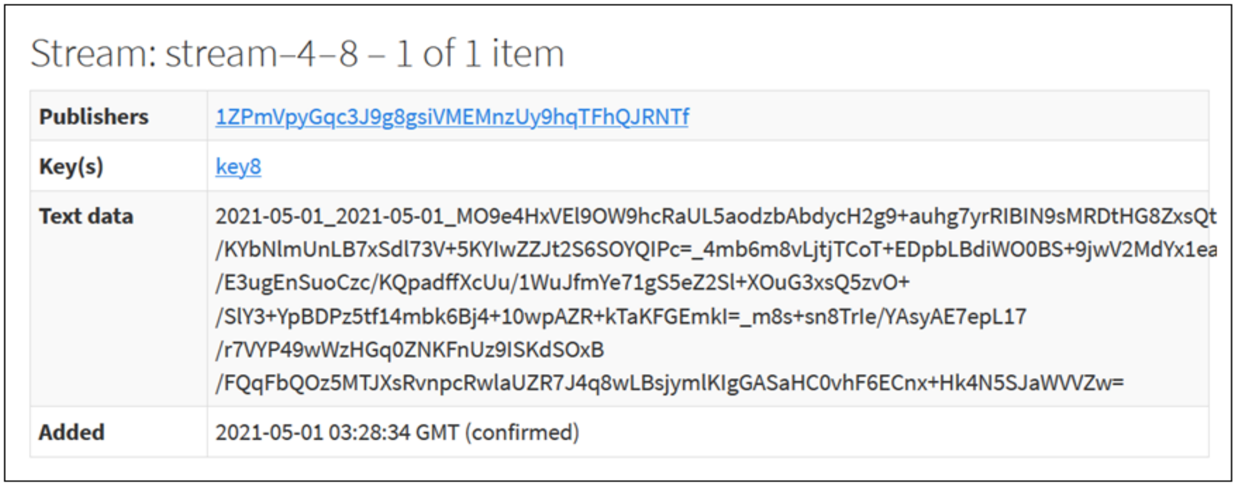
Figure 4.9: MultiChain Blockchain Data Stream Items – Data Provider Privacy Tuple

4.4.4 Encryption, Publish, and Confirmation of Genome Meta-data on Data Streams

Data provider genome meta-data are also independently saved and stored (that is, published) into data stream in the blockchain platform. On one hand, creating a unique data stream for each data provider genome meta-data ensures that every genome data requisition is efficiently identified in the blockhain platform and effectively accessed in a privacy-aware approach. On the other hand, we create a data tuple data item which comprises all genome meta-data values. These data tuple data items store relevant genome information on the data provider in the blockchain platform.

As part of providing a privacy-aware data access medium on the blockchain platform, we encrypt the genome meta-data data tuples before saving them on the blockchain platform. A unique (encryption-based) data stream is created for each data provider genome meta-data. We adopted a PHP string encryption with the *Libsodium* library. This library is a modern, easy-to-use software collection for encryption, decryption, and signatures. The encryption schemes implemented in *Libsodium* are fast, and require a unique (nonce, key) tuple for every plaintext. The encryption methodology employs generation of random 32 bytes *nonce* and *secret key*; where each nonce (or key) represents 256 bits and none of them are predictable by observing other bits from that nonce (or key) or from previous nonce (or keys). This encryption methodology is considered efficient against side-channel or server-side attacks [116, 117]. Our implementation with *Libsodium* encryption adopts different algorithms for different purposes of: encryption/data integrity, AES key agreement, and source/entity authentication. For encryption/data integrity, the encryption library employs XSalsa20 to encrypt stream cipher and AES-GCM-256 is used for key agreement and for message-authentication code (MAC) in one operation.

After genome meta-data encryption, the ciphertext output (encoded as hex value) is saved in the relational database. Moreover, the encrypted genome meta-data data item is published onto related data provider genome meta-data data stream. Each data provider genome meta-data is identified with a unique validated transaction ledger and a data stream in the blockchain platform. Because each data provider (*i.e.*, patient) will make several medical pathology requests, and subsequent storage of different genome data on these requests, we store genome meta-data for each of these genome data. We assign a data value to each pathology requisition related to the genome meta-data. In terms of saving and storing genome meta-data, we identify the data provider and designate a unique object identifier value. Additionally, we assign a unique object identifier value to the pathology requisition. Based on these two object data identifiers (that is, data provider and genome meta-data), we create data streams on the transaction ledger with respect to two parameters for efficient data retrieval from the blockchain platform.



Stream: stream–4–8 – 1 of 1 item

Publishers	1ZPmVpyGqc3J9g8gsiVMEMnzUy9hqTFhQJRNTf
Key(s)	key8
Text data	2021-05-01_2021-05-01_MO9e4HxVEl9OW9hcRaUL5aodzbAbdycH2g9+auhg7yrRIBIN9sMRDtHG8ZxsQt /KYbNlmUnLB7xSdl73V+5KYIwZZJt2S6SOYQIPc=_4mb6m8vLjtjTCoT+EDpbLBdiWO0BS+9jwV2MdYx1ea /E3ugEnSuoCzc/KQpadffXcUu/1WuJfmYe71gS5eZ2Sl+XOuG3xsQ5zvO+ /SlY3+YpBDPz5tf14mbk6Bj4+10wpAZR+kTaKFGEmkI=_m8s+sn8TrIe/YAsyAE7epL17 /r7VYP49wWzHGq0ZNKFnUz9ISKdSOxB /FQqFbQOz5MTJXsRvnpcRwlaUZR7J4q8wLBsjymlKIgGASaHC0vhF6ECnx+Hk4N5SJaWVZw=
Added	2021-05-01 03:28:34 GMT (confirmed)

Figure 4.10: MultiChain Blockchain Data Stream Item – Data Provider Genome Meta-data

The implementation approach for designing and creating genome meta-data data streams on the MultiChain blockchain platform is as follows. We adopt a data representation format of: *Stream-A-B*; where *A* represents a data value for supposed data provider, and *B* represents a data value for supposed molecular pathology requisition for the data provider (*i.e.,* patient). This methodology offers a unique data stream creation and identification on the MultiChain blockchain platform. Furthermore, the methodology offers an efficient data storage for each data provider genome meta-data.

With regards to practical implementation on the MultiChain blockchain platform, we use MultiChain API syntax: *create (”stream”, stream_name, True)*; to create a data stream from the API. The parameter *“create”* indicates the method invoked to execute processing on the blockchain platform. The *“stream”* parameter indicates the type of operation, whether data stream or wallet asset. The *“stream_name”* indicates the specific name assigned to the data stream. *“True”* is a value for *“rescan”* parameter that indicates the node will index all items from when the assets and/or streams are created. The *“create”* method can be incorporated with another parameter, such as, *“restrict”*; which can take values of *offchain*, *onchain*, *write*, or *read*.

Figure 4.10 illustrates the published data value item for *“stream-4-8”*. The data stream stores genome meta-data for data provider with *ObjectId* value, *4*, and pathology requisition *ObjectId* value, *8*. Moreover, in terms of actual storage representation, this data stream outlines data publisher, key value, actual data value item (which is a text format), and storage date (which timestamps the date and time for save the data value item). The data publisher references the blockchain node address into which the data stream and data item are created and stored. The key token is used to digitally sign and save the data item into the data stream. The added time designation is the timestamp for the calendar date and time at which the data item is saved into the blockchain platform. This data value item is signed and permanently saved in the blockchain with a key token value of *"key8"* and with a timestamp value.

4.4.5 Privacy Tuple Data Retrieval on MultiChain Blockchains

Once data has been successfully saved on the MultiChain blockchain, query processing and data retrieval can be performed. Processing data retrieval from MultiChain blockchain involves two main procedures: subscription to the data stream by the node address, and listing of data value items by the subscribed node.

A blockchain node's subscription to a data stream offers access to the data items saved and stored in the data stream. Data streams and their constituent data items are closed to data access by their default settings. Subscription to a node's data stream grants permissions of *read*, *write*, and *execute*. These permissions opens up the data stream to the node addresses to process data items. In the architecture of the MultiChain blockchain, we activate data stream subscription to allow utmost data access for data retrieval and query processing.

For a node to subscribe to a data stream, we use MultiChain API syntax: *subscribe (stream_name, True)*. The parameter *"subscribe"* indicates subscription method invoked to execute data access by the node. The *"stream_name"* indicates the specific name assigned to the data stream for node data access. *"True"* is a value for *"rescan"* parameter that indicates the node will re-index all items from when the assets and/or streams are created, as well as those in other subscribed entities.

The MultiChain blockchain API that enables data retrieval from a node's data stream is the *liststreamitems*. Listing the stream items displays rows of data items stored in a data stream and presents them to the node in either ascending or descending order (based on timestamp on when the data item was saved).

For a subscribed node to retrieve data items, we use MultiChain API syntax: *liststreamitems (stream_name, verbose, countItems, -startItems, local-ordering)*. The parameter *"liststreamitems"* indicates the process of retrieving data value items saved on the data stream. The *"verbose"* parameter is set to *"True"* for additional information about each data value item's transaction. The *"countItems"* and *"startItems"* parameters to retrieve part of the list only, with negative *"startItems"* values (like the default) indicating the most recent items. The *"local-ordering"* parameter is set to *"True"* to order data value items by what time each is first accepted by this node, rather than their order in the chain.

To retrieve data values stored in a data stream, we implement an approach that identifies data streams created on the blockchain platform. Hence, for data retrieval on data stream, say *"stream-4-5-3"*, we specify the same data stream name in the procedures of data subscription and listing data stream items. We execute the following syntax: *liststreamitems(Stream-4-5-3, True).* This syntax retrieves all privacy tuple data items saved in the data stream, *Stream-4-5-3*, for the data provider with object identifier value, *4*, and attribute data with object identfier value, *5*, and data accessor with object identifier value, *3*.

4.4.6 MultiChain Blockchain Query Processing, Data Scalability, and Transaction Throughput

Multichain blockchain offers a platform for efficient query processing, robust scalability, and high query transaction throughput. Regarding the size of data values per transactions, the current limit is 8 MB [112]. The default value is 4 MB. Moreover, there is the option of chaining multiple transactions together. To make changes to transaction data size, one must set the *max-std-tx-size* and *max-std-op-return-size* parameter values, high enough in the blockchain configuration parameters to support desired transaction data sizes.

The number of data streams created for the blockchain platform is essential. This could impact on the data transactions to store in the blockchain. The architecture of MultiChain offers no limit to the creation of data streams, apart from available disk space needed to maintain the data streams [113]. Data streams are efficiently indexed using on-disk indexes that can expand to millions of items or beyond, if the blockchain parameters are set correctly.

Data transaction scalability for blockchain data management is vital for efficient data storage, management, and query processing. For each chain created on the blockchain platform, the maximum number of transactions in a data stream block depends on the maximum size of a data stream block. This block size can be configured to be up to 1 GB in size. The smallest conceivable transaction (with a real signature) is about 200 bytes in size. Hence, theoretically, up to five million transactions can be processed in a data stream block [114]. This projection of transactions depends on the hardware capabilities and large system requirements to process such large data stream blocks.

In terms of transaction throughput, the number of transactions that can be processed per second may be independent of how they are grouped into data item blocks in the data streams. MultiChain blockchain offers an estimate of 1000 tx/sec (transactions per second) under ideal conditions on a mid-range server. Average server conditions (depending on operating system, network setup, server performance, and the size of data stream items) processes 600 tx/sec [114]. Generally, MultiChain blockchain processes transactions faster on Linux operating system than on Windows operating system because of the Linux kernel configuration.

4.5 Data Communication and Encryption Protocols between System Components

The key system components for the proposed methodology, which we discussed in the prior sub-sections, are user interfaces, relational database, blockchain platform, and genomic data store. Each of these components offers different functionality in their operation. The unified integration of the components delivers a seamless platform for collection, processing, and storage of data provider's private, sensitive data. Moreover, this platform offers an efficient approach for privacy preservation in the operational data repositories.

In this section, we discuss data communication processes (for transaction data) that is transmitted between user interfaces, relational database management system, and MultiChain blockchain platform. Additionally, we discuss encryption protocols that are established in the data communication medium between the system components.

4.5.1 Data Communication between System Components

We establish and implement secured data communication channels between user interfaces, relational database management system, and blockchain platform. Here, we assume that the data communication network and routes connecting one component to another is adequately protected from threats. Our assumption on the adoption of a secured data communication medium is based on the premise that robust and efficient data communication encryption (see Section 4.5.2) is employed in the integration of the system components. The encryption algorithm provide confidentiality of the data message's contents, verifies the data message's origin, and provides proof that a data message's contents have not changed since it was sent. Hence, the encrypted data communication medium sufficiently reduces (or best case prevents) any form of attack, data loss or data leakage as data moves from one system component to another [123, 124, 125]. For instance, when data provider's private data is collected from the user interface, the transaction data is processed. The processed data is further transmitted from the interface through communication channels to the relational database. The function of the data communication channel is to offer an immutable transport medium where complete and precise data value collected from the interface is permanently persisted at the relational database; and there is no data loss or data modification.

Different forms of data communication security are implemented to achieve the desired secure data transmission from one system component to another. We adopt several application tools to provide data communication security over the entire system setup. These application tools are offered as part of hosting the application on the corporate platform. The adoption of these application tools appeals to using best practices to secure the system. To this end, we specify the adoption of WhatsUpGold network monitoring, System Center Operations Manager (SCOM), Vulnerability Assessment Scan, FortiClient virus updates and scan, Privileged Access Management (PAM), Key Management Service (KMS) license manager, and windows firewall to offer protection over the system infrastructure. It is noted that these application tools are not exhaustive and other tools can be adopted to offer similar best practices for system protection.

4.5.2 Encryption Protocols for Data Communication between System Components

We follow best practices of adopting secured application and transport protocols to provide data security over the communication channels between system components. For robust and efficient data encryption, we adopt TLS (Transport Layer Security) protocol - with TLS_AES_256_GCM_SHA384 cipher suite - to ensure secure data communications for all forms of data packets transmission between system components. The cipher suite helps to offer end-to-end security services including entity authentication, end-to-end encryption and integrity protection above the network layer and transport layer, respectively. The adoption of AES is based on the standard protection that the block cipher offers among other block cipher considerations [123, 124, 125, 127].

4.6 Summary

This chapter presented overall system design, development, and implementation for the proposed research methodology. We first describe formulation and modelling of formal contextualized privacy ontology. We discuss that privacy ontology offers system modelling for provision privacy preservation platform, and processing of related data provider's private, sensitive data. Moreover, we discuss system modelling and design of user interfaces. The user interfaces served as platforms to collect data from system users. We describe system interaction modelling of use cases, activities, and actor(s) for data transaction processing.

We discuss programming procedures behind the software system user interfaces leading to the extraction of related data values of attribute data, data accessor profiles, and privacy preferences of data providers. We explain the methodology of tight-coupling of data values to generate privacy tuple, and discuss the transmission of privacy preferences and privacy tuple to the blockchain platform. We discuss modelling, design, and development relational database management system and permissioned (private) blockchain data platforms. We discuss the security architectures on these data platforms, and describe the implementation procedures for privacy-aware query processing on the light-weight privacy infrastructure. Finally, we discuss data communication and encryption protocols between system components.

Chapter 5 discusses the security or adversarial model for the system. We discuss the software security assessment and data repositories design principles adopted for the privacy model. We discuss privacy model attacks and their preventive measures. Additionally, we discuss query recall and precision criteria for query processing evaluation analysis. We then summarise by addressing query processing rates on privacy-aware and native (non-private) queries for all forms of queries executed on the system implementation.

Chapter 5

System Security Assessment, Evaluation, and Results Analysis

In this chapter, we discuss evaluation procedures on aspects of the system implementation for the methodology. Our analysis verifies and determines the completeness of the implemented privacy infrastructure. We discuss security assessments for related system components. We discuss data repositories design principles and assess their use as system components for the light-weight privacy infrastructure. Moreover, we discuss evaluation of query processing on the system architecture. We compare and analyze query response time for both privacy-aware and native (non-private) queries, and evaluate the privacy overhead cost in query response time for privacy-aware queries.

We discuss the components of the security or adversarial model for the overall security infrastructure in Section 5.1. Moreover, we examine the details of system security assessment in Section 5.2 and assess the suitability of data repositories adopted for the privacy model system architecture in Section 5.3. We discuss privacy model attacks and their respective preventive measures in Section 5.4. Section 5.5 discusses query evaluation of recall and precision for granularity privacy preferences. Section 5.6 analyses query processing and response time. We also analyse the average query response time for the queries and their privacy overhead query cost. Section 5.7 discusses other methodology approaches in relation to our proposed methodology. We evaluate the methodology contributions and perform comparative technical analyses based on some defined criteria. Section 5.8 summarizes the overall discussions on software security assessment, system evaluation, and query results analysis.

5.1 Security / Adversarial Model

We present the overview of the security model based on the system design discussed in Chapter 4. The security model addresses the design of secure schemes or protocols as applicable in the software and system architecture. Figures 5.1 and 5.2 illustrate the security model design and components that describe the assumptions, adversary, goals, and capabilities. We adapt this security model from the adversary model components discussed and illustrated in Do *et al.* [128].

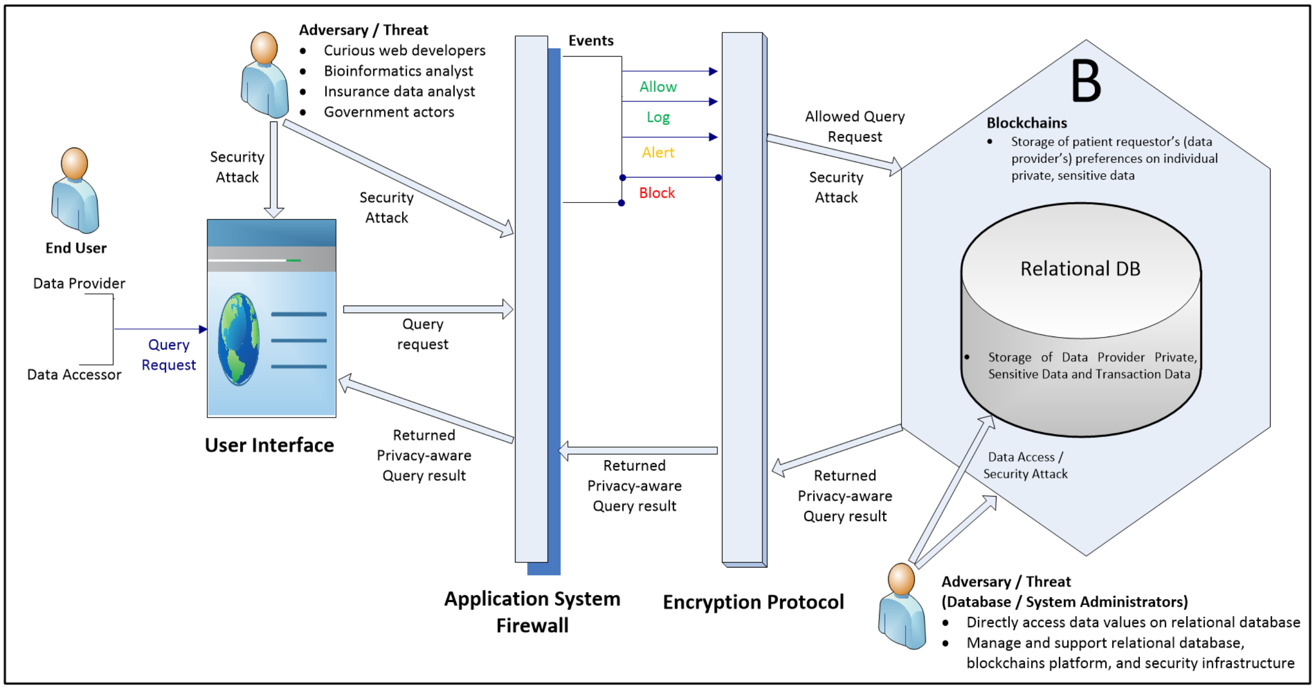

Adversary / Threat
Curious web developers
Bioinformatics analyst
Insurance data analyst
Government actors
Security Attack
Security Attack
End User
Data Provider
Data Accessor
Query Request
User Interface
Query request
Returned Privacy-aware Query result
Events
Allow
Log
Alert
Block
Returned Privacy-aware Query result
Application System Firewall
Encryption Protocol
Allowed Query Request
Security Attack
Returned Privacy-aware Query result
B
Blockchains
Storage of patient requestor's (data provider's) preferences on individual private, sensitive data
Relational DB
Storage of Data Provider Private, Sensitive Data and Transaction Data
Data Access / Security Attack
Adversary / Threat (Database / System Administrators)
Directly access data values on relational database
Manage and support relational database, blockchains platform, and security infrastructure

Figure 5.1: Adversarial Model System Design

5.1.1 Adversarial Model Components

From the illustrations, we describe the following components of the adversary model: assumptions, adversaries, goals, and assumed capabilities.

Assumptions

The assumptions framed for the adversarial model is based on the environment and resources available. In terms of the environment, the primary platforms and devices are: the user interfaces, encrypted local network, relational database, and blockchains platform. For example, the database administrator has direct access to the relational database platform and probably the blockchains platform, whiles systems administrator has access to the network and encrypted protocol infrastructure. Some resources available to the adversaries are: access privileges to the user interface, relational database, blockchains platform, knowledge of tight-coupling of relational database, blockchain, and genomic data store.

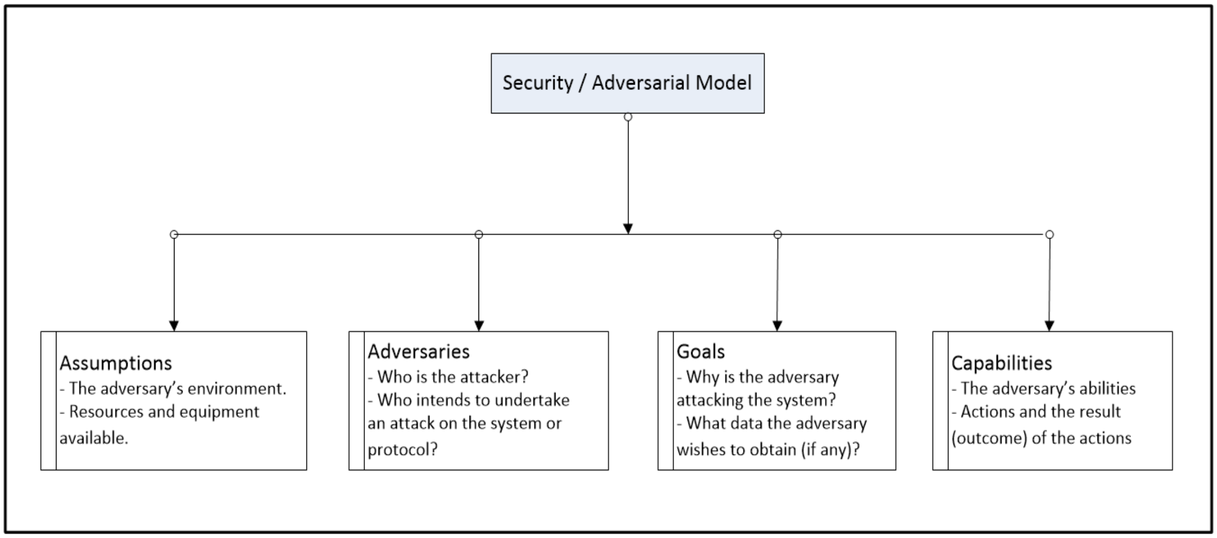

Security / Adversarial Model
Assumptions
- The adversary's environment.
- Resources and equipment available.
Adversaries
- Who is the attacker?
- Who intends to undertake an attack on the system or protocol?
Goals
- Why is the adversary attacking the system?
- What data the adversary wishes to obtain (if any)?
Capabilities
- The adversary's abilities
- Actions and the result (outcome) of the actions

Figure 5.2: Adversarial Model Components

Adversaries

An adversary to the system is described as an individual or a system (whether authorised or unauthorised) that tries to access data from either the relational database or blockchains platform, or both. Some end users who serve as system threats are the database administrator and system administrator. These users have direct access to the relational database and blockchain platform. They may be categorised as either active or passive threats. Some passive adversaries are curious authorised data providers who try to access unauthorised data through different means, and curious web application developers. Active adversaries are represented in the form of dishonest bioinformatics (genetic) data scientists or analysts, health insurance data analysts, and government actors who may need further (unauthorised) private, sensitive information of data providers.

Adversary Goals

The primary goal of the adversary is to have the ability to access data values in the relational database without querying the blockchain platform; which serves as an access control authentication medium. Another goal is to execute queries on the relational database (through the blockchains) to identify unauthorised data and make inferences to the data provider. Additionally, the adversary may try to make changes to the privacy tuple generated for a data provider (by either changing the data values for data provider, privacy preferences, or data accessor).

More importantly, the adversary may try to modify existing privacy policy preferences entries, for example, granularity privacy, visibility privacy, retention duration, *etc.* The adversary may have access to read data provider healthcare information and genomic meta-data (from the relational database and blockchains platform) and make data inferences or perform data linkage to other unauthorised data items. Secondly, the adversary may try to access the transaction ledgers and data streams on the blockchain platform. The adversary may try to query hashed privacy preferences data values (regarding data provider information such as healthcare, demographic, requisitions, and insurance coverage) stored on the blockchain platform.

Assumed Capabilities

In terms of the security infrastructure, the systems administrator or network engineer (or other active adversary) may have *listen* capability that allows her to track or eavesdrop into data packets flowing through the application network medium.

On the other hand, the database administrator may have the access to read (or query) data items from the relational database or blockchain platform. In some other instances, the database administrator may modify data values regarding the data provider, privacy tuples, privacy policy preferences, healthcare consent information, and genomic meta-data. Subsequently, the database administrator may have access to the blockchains platform to read data items stored in the data streams and the transaction ledgers.

Other assumed capabilities may be represented in the form where insurance professionals and/or government actors may have access to read data provider insurance and healthcare related data items from the relational database or blockchain platform for data provider (*i.e.*, patient) insurance coverage and liabilities assessment.

5.1.2 Privacy Infrastructure Security Assessment

We discuss the privacy infrastructure systems security overview in terms of the defences it offers for data provider's private data and overall data processing for data accessors. We discuss these assessment based on the following: confidentiality, integrity, and availability.

Confidentiality

We address the proposed methodology to offer security assurance of *confidentiality* based on the adopted privacy infrastructure. We evaluate the system to protect sensitive, private information from unauthorized access using defined data provider privacy preferences saved on the blockchain platfrom. Through this privacy methodology, each data accessor is allowed data retrieval only on the data values defined in the privacy preferences. Moreover, the composition of the *privacy tuple* authorizes data access to each defined accessor based on privacy preferences. Beside these privacy modelling constructs, we envisage that every other unauthorized data access is denied.

Integrity

In terms of *integrity*, our proposed methodology offers a framework where private, sensitive data and the preferences defined on them are protected from deletion or modification from unauthorized individuals. We assess the privacy infrastructure to ensure that changes to privacy preferences (regarding data provider's private, sensitive data) stored in the blockchain platform is completed in "consensus" or agreement by both data provider and data collector. The agreement is completed through a privacy preferences validation process performed in the blockchain transaction ledgers; using the user interfaces as a connection platform to the blockchains. Moreover, the privacy infrastructure protects data provider's private data (stored in the relational database) from changes, and these changes are allowed with the data provider's consent. Regarding query processing, we determine that query statements are protected from changes due to external attacks. We adopt a form of parameterized queries to retrieve supposed data values to intended data accessors. This query processing approach is discussed further in SQL injection query attacks (see Section 5.4.2).

Availability

We address the security assurance of *availability* based on the tight-coupling of the data storage and processing platforms. The configuration of network communications channel, user interfaces, and web server components deliver effective platform for the data they protect, and ensure data values are available to data accessors when needed. The architecture and configuration of the blockchain nodes offer functional transaction ledgers and data streams that ensure controlled data throughput to the API and the relational database. The relational database is configured with stable query analyzers and processors to parse, analyse and execute query statements, and to deliver data values to data accessors via the user interfaces. We discuss into detail the ability of the system components to ensure security *availability* in Section 5.2.

5.2 System Security Assessment

As part of system implementation, we develop a software platform to realize the privacy model and/or infrastructure. The details of system modelling and user interface design are discussed in Chapter 4 (recall Section 4.2). System security assessment addresses security architecture of an entire system's implementation and various measures to prevent attacks. This software security assessment generally outlines vulnerabilities and proposed defences.

We identify four main system components for software security assessment and evaluation. These are: application transmission medium (for network data communications), client-side user interface, web server code, and data repositories. Our assessment establishes building a verifiable software system for the light-weight privacy infrastructure. Furthermore, we discuss the security assessment, and design principles and assessment on the data repositories (of relational database and blockchains platform). In this section, we discuss measures for secure and reliable operations in the user interface and web server code of the system implementation.

5.2.1 Network Communications Component: Application Data Transmission Assessment

Application data transmission medium is the communication pipeline between system components of user interface, relational database, and blockchain platform. We assess the application transmission medium by ensuring that communication is effectively secured and safe from any form of external tampering. We assess the security infrastructure for network communications in various ways:

How adversaries could attack

An adversary could try to access the data communication channel to tamper with or change data provider's privacy preferences data or actual private, sensitive data and transaction data during data transmission through the channel. Moreover, an adversary may try to capture access credentials of authenticated users that are transmitted over the data channels. Finally, an adversary could try to break down the data transmission channel.

What adversaries try to achieve

An adversary's aim is to make changes to important data provider's private information. For example, an adversary could change the health information so as to increase the monetary claims for an insurance coverage. Secondly, when an adversary tries to access the data communication channel, the adversary's aim is to capture relevant data for malicious purposes. For example, an adversary can use the data provider's login credentials and biographic information, such as, name, birth date, and other relevant data to impersonate the data provider and make fraudulent health insurance claims on altered false information. Moreover, when an adversary breaks down the transmission channel, the aim is take hold and prevent data movement from one system component to another.

How attack is prevented

These threats are adequately prevented based on secured data access and transmission channel. We employ best practices to secure the system by adopting standard application tools. We ensure that there is a protective firewall to monitor incoming and outgoing network traffic based on defined security rules. We adopt WhatsUpGold application tool which delivers network availability and performance monitoring. This tool provides complete visibility into the status and performance of applications, network devices and servers. Moreover, we adopt FortiClient application to protect the server against malware by leveraging real-time scanning and detecting vulnerabilities. The application tool helps to centrally manage, monitor, patch, quarantine, dynamically categorize and provide deep real-time endpoint visibility for the server hosting the system components. It is noted that these application tools are not exhaustive and other tools can be adopted to offer similar best practices for system protection.

An encryption protocol is adopted to provide data security over the communication channels between system components. The encryption protocol helps prevent these aforementioned attacks, such as, passive attacks (*i.e.*, data snooping, eavesdropping, *etc.*), and active attacks (*i.e.*, brute-force, linear crypt-analysis, *etc.*). For robust and efficient data encryption, we adopt TLS (Transport Layer Security) protocol - with TLS_AES_256_GCM_SHA384 cipher suite - to ensure secure data communications for all forms of data packets transmission between system components. The cipher suite helps to offer end-to-end security services including entity authentication, end-to-end encryption and data integrity protection above the network layer and transport layer respectively.

5.2.2 User Interface Component: Design Principles and Assessment

The user interface is the primary medium through which the data provider inputs private data into the relational database or blockchains. Additionally, the user interfaces provide the medium through which data records retrieved from the relational database are displayed to the data accessors (or possible third-party accessors). We discuss ways in which an adversary could attack the system (through the user interfaces), the aim of the adversary attack, and how these attacks are prevented.

How adversaries could attack

An adversary could try to inject false data pertaining to a data provider into the data fields of the user interface. This develops into a form of HTML script injection. Additionally, the adversary could use a web application to send malicious code, generally in the form of a browser side script. Moreover, the adversary may try to inject some malicious code (or any form of a malware) into the website address link.

An attacker can use cross site scripting (XSS) to send a malicious script to an unsuspecting user. The end user's browser has no way to know that the script should not be trusted, and will execute the script. Because end user's browser thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site.

What adversaries try to achieve

When false data are added to the data fields on the user interface, the adversary aims to alter the correct data input (in the user interface data fields) from the data provider before the data is submitted over to the web server and/or relational database across the communication channel. Secondly, when a malicious code is injected into the website address link, the adversary aims to send a malware or other harmful codes (in the form of tags or script codes) to tamper with the data transmitted over the channel. Moreover, when an adversary sends a malicious script to the unsuspecting user, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser. These scripts can even try to rewrite the content of the HTML page. For most of the attacks, the main purpose of the malicious code is to successfully travel through the transmission channel and get to the relational database or blockchains and effect changes in the stored data.

How attack is prevented

We assess efficient security at the client-side user interface by proposing measures to prevent cross-site script attacks, HTML script injection, and cross-site request forgery. Each user interface is designed with labels (to display vital information to the user) and data input fields (to collect data from the user or data provider). We develop user interfaces in which many of the data fields are drop-down lists. These drop-down lists are populated with read-only data for user selection. Some other data fields that require specific user inputs are equipped with validation features to authenticate the kind of data provided by the user.

We implement validation on the input data collected from the user interfaces by checking for character length, type (for *e.g.*, input data consisting of alphanumeric characters), format (for *e.g.*, data field rejects any tags or characters, such as "<script>" tags), and data range. These HTML development mechanisms ensure robust scripts that deliver secure user interfaces for data collection from users and display of relevant data values and information to intended data accessors.

With regards to protecting the application website link from adversary attacks, we adopt an approach where the website link is protected with an encrypted Transport Layer Security (TLS)-enhanced communication protocol. This offers secured host authentication by delivering privacy protection and data exchange integrity while private, sensitive data is in transit. Moreover, the web hosting of the user interfaces is secured with efficient encryption protocols to present protection against "man-in-the-middle" attacks, *etc.* The provision of bidirectional encryption of communications between client-side user interface and web server code offers protection against eavesdropping and tampering attacks.

5.2.3 Web Server Component: Codebase Security Assessment

The web server offers to serve as an intermediary platform to process and execute code on every data generated on HTML user interface. These data are further transmitted for onward storage into the relational database or blockchains. The web server usually accepts requests through HTTPS, and can also accept and store resources sent from the web browser user agent. We discuss ways in which an adversary could attack the web server, what the adversaries aim to achieve, and how these adversary attacks are prevented.

How adversaries could attack

An attacker may try to access the data connection between the web server and relational database or blockchain platform and add some malicious query code to run on these data platforms.

An adversary may cause a denial-of-service (DOS) attack by sending numerous service request packets; which ends up causing a service request overload. This affects the service delivery operation of the web server and may cause a buffer overflow attack or HTTP GET Request Flooding. Additionally, the attacker may try to exploit a programming error in the web application causing a DOS attack.

The attacker could determine that input data fields are not validated properly, and can add data values or SQL strings to maliciously craft a query which is processed by the web browser and subsequently executed by the web server. This could lead to a form of SQL injection attack. The attacker may store malicious or unrelated data in the database.

What adversaries try to achieve

When the attacker accesses the data connection to run some malicious code in the relational database or blockchain platform, the attacker aims at using the query code to alter some data stored in the data platform.

In terms of buffer overflow attack or HTTP GET Request Flooding, the attacker hopes block all access from authenticated users. This results in denying overall web service of uploading and downloading data from the source relational database or blockchain platform. This activity could lead to some forms of session attacks in terms of data connectivity from the web server to the user interfaces or data platforms.

When the attacker adds SQL strings into the input data fields, the aim is to change the actual program code query. Through this process, malicious or unrelated data is stored in the database. As a result, when the website is requested, it will show irrelevant data on the website, thus displaying a defaced client-side user interface website. Secondly, the added SQL string introduces a wrong query code which could be executed by the relational database query processor. Moreover, the added data in the input data fields could result in buffer overflow, integer overflow, or format string attacks.

How attack is prevented

We determine the effectiveness of server-side code in preventing attacks by adopting secure coding for the software application operations. For code that runs behind the user interfaces, we establish safe and secure connection to the relational database server with an authenticated administrator access. With regards to the blockchain platform, we establish a secure connection through the API; where we specify the host address, port, username, and password. We also identify the chain name as part of defining complete connection parameters for the blockchain. After each successful connection to the web server and subsequent database operational activities, we activate a permanent closure to the connection for the database server or blockchain platform to avoid unauthorized access and data leakages. These processes help prevent any form of data connection attacks by the adversary.

We prevent every form of session attacks, as follows. For every operation on the web server, we initiate user sessions which controls all activities from the client-side user interfaces to the web server (as well as data transfers to relational database server and blockchain platform). Some of the session activities include, but not limited to: *LoginFrequency*, *LocationFrequency*, *SessionElapsedTime*, *SessionOutput*, *SessionIO*, *SessionPages*, *SessionLastActivity*, and *LocationFails*. All server session information are destroyed after successful connection closure from the relational database server and blockchain platform.

We assess web server code to prevent buffer overflow, integer overflow, format string attacks, and return-oriented programming attacks. With regards to buffer and integer overflows, we declare and initialize data variables or identifiers before using them. Moreover, we determine other measures for appropriate identifier data sizes, release of the memory buffer after identifier usage, ensure proper integer identifier conversions, and ensure opened unsigned integer operations. We prevent format string attacks by properly validating user input and specifying formatted strings as part of the program, and not using the data string.

For SQL injection attacks, we adopt parameterised queries in the programming code to prevent attackers from running malicious code in the relational database. We discuss into detail how SQL injection attacks are prevented in Section 5.4.2.

5.3 Data Repository Design Principles and Security Assessment

We discuss the measures established for each data repository based on the security or adversarial model discussed. We examine varied configurations for data storage and management in the relational database and blockchain platform. Additionally, we discuss design approaches for data consistency, comprehensive data scalability, and high data transaction throughput. This section discusses our assessment of the data repositories architecture to offer an efficient tight-coupling for light-weight privacy infrastructure.

5.3.1 Relational Database Design and Security Assessment

The relational database is assessed to determine effectiveness of data access, data storage and management, and query processing. We discuss how adversaries could attack the data platform, the aim of the adversary attacks, and measures designed to prevent the intended attacks.

How adversaries could attack

An adversary may use malicious means to access the relational database by impersonating the login credentials of the system administrator. Secondly, the adversary may use unauthorized login credentials execute query statements to retrieve data values.

What adversaries try to achieve

The adversary aims to alter or make changes to data values stored in the relational database that pertains to data provider's privacy preferences, personal data or transaction data. Moreover, the adversary anticipates to retrieve unauthorized data provider private, sensitive data values and/or make inferences to stored private data values in the relational database by executing SQL query statements. Finally, the adversary expects to alter or make changes to data objects (that is, the relational tables, indexes, stored procedures) definition that are configured for effective operation of the relational database.

How attack is prevented

In relation to the security model discussed in Section 5.1, we ensure access to the relational database is enabled through secured access credentials. The access credentials of each user is authenticated through secured username and password. The primary access to the relational database is by the database administrator, data architect, or any system engineer. All other access to the relational database is through secondary means of using the user interfaces. The user interfaces (and related data communication channels) are equipped with robust and secured encryption protocols to control unauthorised access to the relational database. Through this approach, we ensure that access threats (such as query processing, data values tampering) are substantially reduced or prevented.

The entire design and development of the relational database offer a robust platform for management of data provider's private, sensitive data, non-private data, and other transactional data. We build strong relational constraints on the database and data objects, such as, relational tables, indexes and stored procedures. We model and develop domain integrity constraints where we enforce restrictions on the attribute data values in a relational table. All these domain constraints ensure that the right type of data value, as well as the expected data size for data values are stored in the attribute fields. Any data value that violates the restrictions is flagged and prevented from getting stored in the schema database.

We also develop and establish entity integrity constraint where every relational table has a unique attribute that stores distinct data values. The development of entity constraint ensures data consistency, data completeness, and uniqueness in the storage of entity instance data. We establish and enforce referential integrity constraint to ensure that storage and management of tuple data values that require prior instance of another tuple data is processed in a safe and complete procedure. Any violation of the constraint prevents new data insertion and update or subsequent modification of existing data values. We develop referential integrity constraint cardinality to control the number of tuple occurrences in a relational table. All these constraints help to maintain data consistency and prevent unauthorised data modifications by adversaries.

There is real-time monitoring of the query statements executed on the relational database to monitor regular queries programmed and identify all unfamiliar query statements. We adopt tools embedded in the relational database management system to facilitate overall monitoring of server performance, views of input-output hotspots, SQL statements, network, and data engine. These tools are *performance dashboard*, *explain plan*, and *query statistics*. Additionally, we use these tools to implement real-time monitoring of the transaction logs for data insertion, data modification and updates. Through these procedures we ensure data consistency on all data provider's privacy preferences data and actual private personal data stored in the relational database.

5.3.2 Blockchain Platform Design and Security Assessment

The MultiChain blockchain platform stores and manages all data provider's privacy preferences on private, sensitive data, and genomic meta-data information. We discuss various ways in which adversaries can attack the blockchain platform, the objective of the attackers, and how the attacks can be prevented.

How adversaries could attack

An adversary could deny data communication services from the blockchain platform to the other system components. Secondly, an adversary could to use the chain (node) name, host address, and port credentials to access the transaction ledgers and data streams on the blockchain platform; so as to query the data values stored in the data streams.

What adversaries try to achieve

When the adversary tries to deny data communication service from the blockchain platform, the aim is to alter or delete the configuration settings on the blockchain. Secondly, the main aim of the adversary in querying the data values (regarding privacy preferences on data provider private, sensitive data) stored in the blockchain is to obtain unauthorised privacy preferences information about a data provider. In this approach, the adversary aims to retrieve, make changes to the existing privacy preferences, or delete the existing privacy preferences data values stored in the blockchain platform.

How attack is prevented

Based on the security model discussed in Section 5.1, we establish secured access to the blockchain platform. Access to the blockchain platform is done primarily through an API wrapper protocol. Each access to the blockchain is through authenticated username, password, host address, port, and chain (node) name. All connections for data access and query processing are completely terminated after closure of data processing activities. Moreover, all configuration settings for the blockchain platform are securely protected in the server platform, and monitored by the WhatsUpGold application tool (which delivers network availability and performance monitoring for all network devices and servers). The implementation of secured connections and network monitoring application tools establish efficient protection of the blockchain configuration against adversary access (or attacks).

The primary configuration of MultiChain blockchain transaction ledgers involve the creation of data streams. The data streams store hashed data value items in the blockchain. The application of hashing algorithm on the stored data offers another layer of data security to the data values. This security layer prevents any goal of data access or tampering by adversaries.

The design and development of transaction ledgers and data streams deliver optimum data management, data retrieval, data scalability, and transaction throughput. Each data stream is an ordered list of items, with the following characteristics: one or more *publishers* who digitally signed that item, and one or more *keys* between 0 and 256 bytes in length, to allow efficient retrieval. Information about data item's transaction and block, include *txid* (transaction id), *blockhash*, and *blocktime*. The data streams created are identified as closed data streams. The closed feature restrict data access and data item changes to be effected primarily by the host node address. Data streams are created by a special transaction output, which must only be signed with the host node address with *create* permission privilege. The node address that creates data streams automatically has *admin*, *activate*, and *write* permissions for the streams. In the blockchain setup for the light-weight privacy infrastructure, there is only one (host) node address set up for transaction management. Hence, no other external node participates in creating data streams. This feature further restricts adversaries to gain access (and query hashed data items stored) in the blockchain platform.

5.4 Privacy Model and Related Attacks

Privacy attacks expose different forms of vulnerabilities that affect a privacy model. A privacy attack devises methodologies and mechanisms to disclose data provider's private, sensitive information to unauthorized data accessors; which may be through authorized or unauthorized means. Sensitive data, such as, healthcare records, biographical identity information, demographic information, or data provider consent information may be retrieved from the underlying data store by adversaries. When these adversary attacks are successfully executed, then the privacy model becomes undesirable.

The description and analyses of privacy attacks address likely vulnerabilities that may be exploited in the overall architecture and design of the privacy model, which is discussed extensively in Chapter 4 (recall Section 4.1). We consider attack forms that may be posed to the overall privacy model (and ultimately access unauthorized private data stored in the data repositories). We identify privacy model provisions to overcome these potential compromises to private data disclosure. Our assessment of standard (and best) practices implementation adopted in both system infrastructure (recall Section 4.2, Section 4.3, and Section 4.4) and security infrastructure (recall Section 5.2) address these provisions against the related privacy attacks. We identify system implementation best practices (in the design of user interfaces and query preparation and processing) to address compromises to unauthorized private data disclosure. To this end, we discuss and demonstrate inference attacks (see Section 5.4.1) and SQL injection attacks (see Section 5.4.2) by using instance examples.

5.4.1 Inference Attacks

Inference attacks identify data relationship between attribute data that allows an attacker to determine whether data about a data provider is stored in the database. This form of attack enables the attacker to suggest or make deductions on other data values from known data values. In our privacy model, we adopt a data accessor-centric form of data retrieval, where every data tuple retrieved is based on tight-coupling of the attribute data, privacy preferences, and data accessor profile. Hence, the methodology approach offers to minimally reduce different forms data retrieval inferences. We demonstrate and analyse inference attacks using the instance example below.

Query Description

Suppose we want to retrieve data values on data provider pathology requisition information. The attribute data elements from which the query transaction is performed, include: *Date Collected*, *Specimen Type*, *Tissue Fluid Source*, and *History Diagnosis*. The query statements for privacy-aware and native query syntaxes are outlined in the Appendix (see Section 7.1).

Query Analysis and Discussions

Analysing the query, we identify a form of data decoupling for unique data categories. There are different attribute data categories that are specified for different data accessor sensitivity or classification levels. Hence, we categorise the data elements (from the query) different from highly-sensitive data elements on cancer biomarkers (such as, mutation analysis, translocation analysis, epigenetics, *etc.*) for the pathology requisition. Moreover, data retrieval for the query above does not reveal or infer data values in the other attribute category data (such as, cancer biomarker). As a result, for each data accessor we determine the attribute data category that is allowable for data access, and all other data are deemed inaccessible. Through this mechanism, data retrieval inferences (or inference attacks) are appreciably reduced, as data report (and their underlying query statement) are analysed through privacy-aware (query analyser and processor) API for private data preservation.

5.4.2 SQL Injection Attacks

SQL injection is a technique where malicious users can inject SQL commands into a database SQL statement via web page input. One approach of SQL injection query attack is the introduction of SQL statement that unknowingly runs on your database, instead of regular user input data like a valid data value for a username. Other forms involve the use of dynamic query statements where user input is string concatenated or appended to SQL statement to build queries that run “on-the-fly”. We adopt prepared query statements with parameterized queries to prevent SQL injection query attacks on the privacy model and relational database from the user interface. This approach requires binding of query statements with parameters entered by the user as its values. We demonstrate measures adopted to prevent query attacks using the example below.

Query Description

Suppose we want to retrieve data values on data provider information. The attribute data elements from which the query transaction is performed, include: *personal health number*, *last name*, *first name*, *birth date*, and *gender*, *etc.* The PHP web server-side program code that illustrates adopted approach to prevent SQL injection attacks on the query statements is outlined in the Appendix (see Section 7.2).

Query Analysis and Discussions

In the PHP web server-side code snippet, we describe the procedure for query processing. The query syntax is presented as a string and transformed into a parameterised query (denoted by a question mark sign). The attribute on which a parameter value is expected is *“patient_id”*. The query statement is “prepared” together with the database connectivity for the parameterised query. The parameterised query is then bound with parameter value before the query is finally executed. With this approach, all forms of dynamic queries are eliminated, and attackers are unable to secure unauthorised access to data provider’s private data.

On the other hand, we use stored procedures (which are programmed to enforce data providers’ privacy preferences) to run SQL queries for report generation. Generally, stored procedures offer another framework approach to prevent manipulation of unrelated database tables. Stored procedures check the data type and data size of input parameters and prevent supposed input data that violates data field type designation.

5.5 Query Processing Evaluation and Analysis

We discuss the evaluation of our methodology by analysing queries processed. Firstly, our analysis focuses on query recall and precision evaluation, where we discuss two key components of the data provider’s privacy preferences. These are: visibility privacy and granularity privacy. Secondly, we discuss the rate of query processing and query response times for different forms of predominant expected queries. Finally, we discuss privacy overhead cost and analyse their impact on overall query processing.

5.5.1 Experimental Implementation, Framework Setup, and Query Processing

To evaluate the system implementation discussed in Chapter 4, we perform query processing on the privacy-aware model infrastructure. The data storage, management, and query processing is processed on a single computing machine with eight logical processor(s), 3.6 GHz processing speed, and 16 GB of RAM. Moreover, the single computing machine has 900 GB of total disk storage capacity. This single computing machine hosts all three main components of the system; namely, blockchain, relational database, and genomic data store. The single site hosting for all three system components help to minimize the latency in data transmission from one component to another.

We implement our methodology using data sets for patient (genomic) healthcare service delivery. We work with two different data sets: healthcare data and genomic data. For experimental implementation, all data sets are free from personally-identifiable data provider (*i.e.*, patient) information. The healthcare data contains data provider information in the form of healthcare, biographical and demographic, physician, medical consent, and pathology laboratory requisition. We use an experimental testing data set that consists of 1,526 records for the *patient* relational table. Other relational tables have the following record numbers: *physician* relational table - 5 tuples, *requisition* relational table - 10 tuples, *consent* relational table - 6 tuples, and *genome meta-data* relational table - 26 tuples. An overview of the detailed data description for the healthcare data set is discussed in Chapter 4 (recall Section 4.3.1).

In terms of the genome data, we use the genome meta-data and de-identifiable genome data (of both forward and reverse reading for normal and cancerous genomes). The meta-data consists of the following information: vcf standard version, variant caller name, variant caller version, variant call date, variant caller regions, annotation software name, annotation software version, annotation date, annotation reference, annotation reference version, and annotation fields, amongst others. The genome data stored on the genomic data platform consists of the following data sizes: normal genome forward reading - 18,026,662 KB, normal genome reverse reading - 18,647,746 KB, first cancer genome forward reading - 18,474,818 KB, first cancer genome reverse reading - 19,188,434 KB, second cancer genome forward reading - 17,760,848 KB, second cancer genome reverse reading - 17,883,908 KB, and human genome reference of 3,173,742 KB. We use *lymphoma cases* data set with total file size of 4,815,248 KB.

We record query response times for four different query categories. We discuss the detailed description and analyses of each formulated query in Section 5.6 - where we discuss statistical metrics and privacy overhead cost for processed queries. Our analysis approach is primarily based on comparing query processing for native (or non-privacy aware) queries in relation to privacy-aware queries. Each (native or privacy-aware) query is run in isolation on the single computing machine and independent from other resource processing interferences. This query processing approach offers a unique method of accessing each query (parsing and execution) and determines optimum response time for processed queries.

We undertake the evaluation as follows: For each formulated query type, we run both native and privacy-aware queries. We first randomly run a native (or non-privacy aware) query independently on the relational database, without applying data provider privacy preferences stored on the blockchains platform. We then randomly run a privacy-aware query on the same formulated query type with the query processor and query execution plan allowing an access control parsing on the blockchains platform. It will be noted that query processing for both native and privacy-aware queries are performed in a non-concurrent manner.

Based on the query processing evaluation, for each formulated query and for each form of either native or privacy-aware query, we record the query response times for 20 random executions of the formulated query. We then compute averages for each native or privacy-aware query form. The average query response time for each native and privacy-aware query on formulated query type is computed and the privacy overhead cost is calculated. As a result, we effectively compare the query results and establish the effectiveness and efficiency of the privacy-aware query medium.

5.5.2 Evaluation and Analysis of Query Recall and Precision

We evaluate the outcome of the query processing experiments performed based on a set of criteria from information retrieval proposed by Junker *et al.* [115]. We perform a gap analysis on their study and adapt an approach and criteria for evaluating data value correctness and query processing rate. We propose standard measures of *recall* and *precision* as adequate measures to evaluate data retrieval on queries performed on the proposed system.

Query Recall Evaluation

Recall is the fraction of the relevant data values that are successfully retrieved. It computes the number of tuples that are retrieved by the privacy-aware API divided by the number of tuples retrieved by native query syntax. It is therefore a definition of the ratio of the number of relevant tuples returned to the total number of relevant tuples that should have been returned from the relational database. For recall, an evaluation of 100% is trivially attained and verified. The verification is based on the assertion that the formulated query executed through the privacy-aware API presents the exact set of tuple values as the queries that were run through native query syntax. In some instances, recall is 0% because of privacy policy preferences set on visibility (or access) privacy which prevents data retrieval for certain data accessors. In such cases, there is no visibility (or access) privacy to the underlying data due to the data provider's privacy preferences.

Query Precision Evaluation

Precision is the fraction of retrieved data values that are relevant to the query. It computes the correct number of tuples retrieved through privacy-aware API divided by the number of tuples retrieved through native query syntax. Thus, precision measures the ratio of correct number of relevant tuple values returned to the total number of tuple values for a given query. Precision evaluation is very important, as it evaluates the number of correct results divided by the number of all returned results based on a formulated query through the privacy-aware API. Based on the privacy policy modelling from the formulated privacy ontology (recall Section 4.1), we identify three main parameter types for granularity (or precision) privacy. These are: *'Specific'*, *'Partial'*, and *'Existential'*. Each of these parameters are set for an attribute data based on the sensitivity level of the attribute data and the data accessor.

We attain precision rates of 100% for privacy-aware data values which has the granularity privacy value set to *'Specific'*. By the granularity privacy of *'Specific'*, we mean that the exact data values stored in the relational database are returned as query results based on the formulated query.

For privacy policy preferences where granularity privacy is set to *'Partial'*, we mean that a proportion of data value characters stored in the relational database is returned as query results. We attain variable precision rates for different privacy-aware attribute data due to indeterminate number of data characters stored in the relational database for these attribute data. This is because in the implementation of granularity privacy, we set granularity privacy policy preference to retrieve only the first five data characters for *'Partial'* granularity. To evaluate *Partial* granularity privacy, we formulate a query for data retrieval on the privacy-aware API. The formulated query retrieves data provider demographic attribute data (such as, *street*, *city*, *province*, *postal code*, and *phone number*). Our analysis is based on data retrieval on a single data tuple. The number of actual instance data characters retrieved using native (non-private) query syntax are 17, 7, 2, 7, and 12 for the attribute data: *street*, *city*, *province*, *postal code*, and *phone number*, respectively. Hence, a precision level of 29.41%, 71.43%, 100%, 71.43%, and 41.67% are returned for the identified attribute data based on the preset granularity privacy of first five data characters for *'Partial'* granularity. It will be noted that for the same set of attribute data, these granularity (precision) rates may vary for different data tuples and their related number of instance data characters.

Finally, for privacy policy preferences where granularity privacy is set to *'Existential'*, we attain average precision rates of 0%. This is because an existential granularity privacy only indicates a default value of *"Yes"* to signify that the actual data value is present in the underlying database, without revealing the actual instance data values. It will be noted that setting the granularity privacy preference to *'Existential'* is determined by the high-level sensitivity of the attribute data for the intended data accessor. This is because of the expectation where the actual data value is not retrieved and presented to the data accessor, but a data value that indicates the presence of the actual data value stored in the relational database.

In summary, the form of granularity privacy that is set for each attribute data (and their intended data accessor) in the system implementation largely determines the precision level of data tuples and query retrieval. As a result, highly sensitive attribute data are assigned granularity privacy of *'Existential'* whiles low sensitive attribute data will have a granularity privacy of *'Specific'*. For moderately sensitive attribute data, their granularity privacy are set to *'Partial'*.

5.6 Query Processing and Response Time Analysis

We analyze query processing rate on the privacy infrastructure system implementation to evaluate the effectiveness of running data queries. We identify four categories of queries (on data provider demographic, healthcare information, consent witness information, and genome meta-data information) and discuss the results output from running these queries. Each query presents unique set of attribute data elements, as well as, different set of data provider privacy preferences. We run random independent queries for both privacy-aware and native queries for each query type.

Our presentation of query evaluation results is based on comparing each privacy-aware query alongside the corresponding native queries. The evaluation of each query category type assesses data provider privacy preferences specified in the data privacy policy; namely, granularity privacy, visibility privacy, purpose use, and purpose context privacy, amongst others. For each query category type (of both native and privacy-aware), we determine statistical metrics of *highest*, *lowest*, and *average* query response time for all random independent queries. Moreover, we determine the privacy overhead cost (in terms of query response time) on the privacy-aware queries in relation to the native queries. This offers a random distribution of the query response time for each execution run. The inferences on these statistical metrics offer a deductive confidence interval of valid data values for query response times on privacy-aware and native queries. Based on these statistical inferences, outlier query response data values are rejected and a better random distribution of the data values is attained. Furthermore, the statistical inferences offer the merit to retroactively produce query response data values based on the implementation framework setup.

We discuss and analyse the query processing rate for each query category type in the next sections. Moreover, we determine the statistical metrics (of *highest*, *lowest*, and *average*) query response time and evaluate the privacy overhead cost. Finally, we discuss a summary of average query response time and privacy overhead cost for all processed queries. The gives an evaluation overview of privacy-aware queries in comparison to native queries processed.

5.6.1 Query 1: Data Provider Demographic Information

Suppose we want to retrieve data values on data provider demographic information. The attribute data elements from which the query transaction is performed, include: *Street Name*, *City*, *Province*, *Postal Code*, *Original Province*, and *Phone Number*, *etc.* The result of query processing rate for data provider demographic information is illustrated in Figure 5.3. The privacy-aware queries and native queries are designated with blue and gray colouring, respectively. We define the query syntaxes for both privacy-aware and native (non-private) queries in the Appendix (see Section 7.3.1).

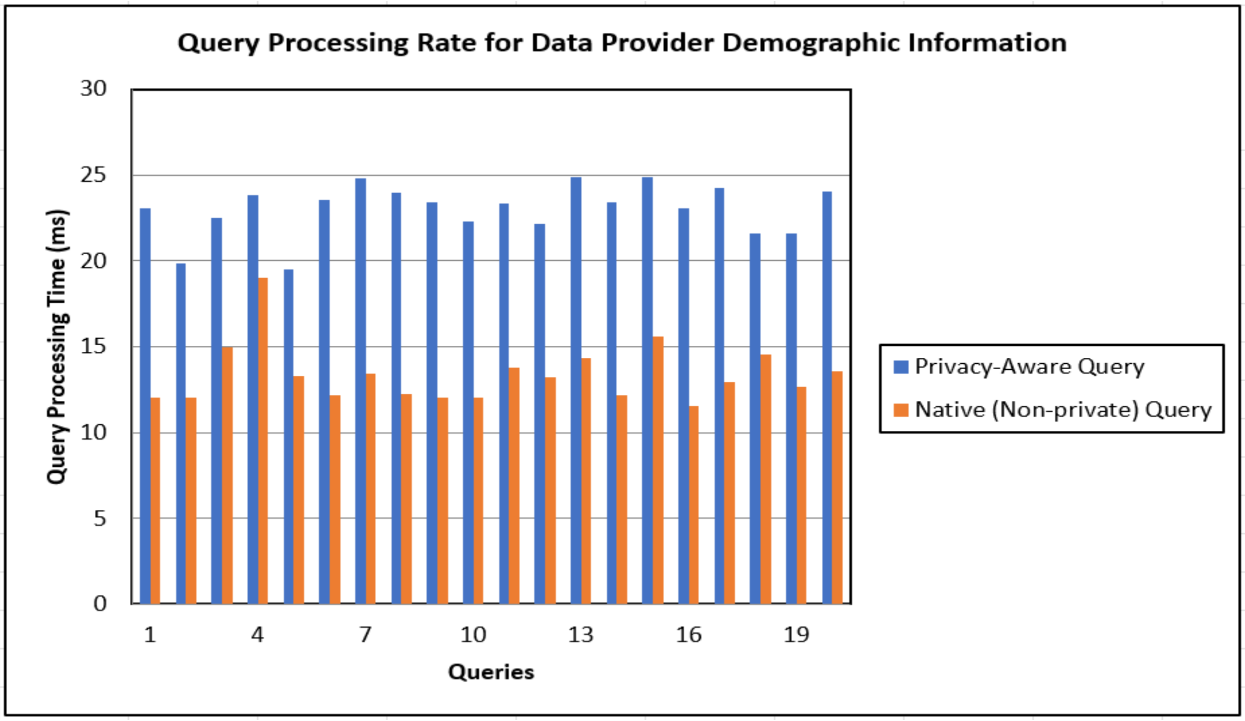


Figure 5.3: Query Processing Rate for Data Provider Demographic Information

Results Analysis and Discussion

In Figure 5.3, we observe an increase in the query processing time for the privacy-aware queries in comparison to native queries. This is expected because of the added activity of privacy-aware query parsing, processing, and execution which leads to an increase in query data retrieval load or query cost. From the query results, we realize the highest and lowest query processing time for native queries are 19.01 ms (milli seconds) and 11.52 ms (milli seconds), respectively. Moreover, an average query processing time of 13.37 ms is attained. In terms of privacy-aware query processing, we attain highest and lowest query processing time of 24.86 ms and 19.48 ms, respectively. We also record an average privacy-aware query processing time of 22.99 ms.

We observe privacy overhead cost of 41.85% for privacy-aware query processing. This is indicative of the varied number of demographic attribute data and the instance data stored on these attribute data. More importantly, we analyze the privacy preference parameters of data information privacy with instance data representation of: *'Level-4: Restricted'*, granularity privacy with instance data representation of: *'Specific: Specific data item is accessed. A query for data item returns actual data value'*, visibility privacy with instance data representation of: *'Third-Party Allied Health Access: Data accessible to parties not covered by explicit agreement with House'*, and purpose privacy with instance data representation of: *'Reuse-Same: Data used for same purpose and multiple times'.* This analysis by the privacy-aware query analyzer increases the query overhead cost for the overall query processing. Based on the query response times attained from both privacy-aware and native query processing, we determine that query processing on the light-weight privacy model architecture is efficient, and there is safe, cost-effective data retrieval on the overall architecture.

5.6.2 Query 2: Data Provider Healthcare Information

Suppose we want to retrieve data values on data provider healthcare information. The attribute data elements from which the query transaction is performed, include: *Personal Health Number*, *Medical Record Number*, *Chart Number*, *Personal Care Physician Name*, and *Dentist Physician Name*, *etc.* The result of query processing rate for data provider healthcare information is displayed in Figure 5.4. The privacy-aware queries and native queries are designated with blue and orange colouring, respectively. We define the query syntaxes for both privacy-aware and native (non-private) queries in the Appendix (see Section 7.3.2).

Results Analysis and Discussion

Similar to *Query 1*, *Query 2* also shows an increase in the query processing time for the privacy-aware queries in comparison to the native queries. In terms of native (non-private) query processing, we attain highest and lowest query processing time of 18.38 ms and 9.74 ms, respectively. We also record an average query processing time of 13.38 ms. Regarding privacy-aware query processing, we note that the highest and lowest query processing time are 23.97 ms and 19.03 ms, respectively. Moreover, an average privacy-aware query processing time of 21.67 ms is attained.

In terms of privacy overhead query cost, we observe 38.26% for privacy-aware query processing in relation to native (non-private) query processing. It will be noted that data provider healthcare information processed here involve unique set of privacy-aware attribute data, which are not many in comparison to demographic data in *Query 1*. Moreover, the privacy overhead query cost is attributed to data provider privacy preference parameters of data information privacy with instance data representation of: *'Level-3:*

Confidential', granularity privacy with instance data representation of: *'Partial: Partial or altered and non-destructive data values returned to accessor'*, visibility privacy with instance representation value of: *'Third-Party Allied Health Access: Data accessible to parties not covered by explicit agreement with House'*, and purpose privacy with instance data representation of: *'Reuse-Selected: Data used for primary purpose for which data'*. Finally, we note that query response times attained from both privacy-aware and native query processing determine that light-weight privacy model architecture offers an efficient platform for processing data provider private information.

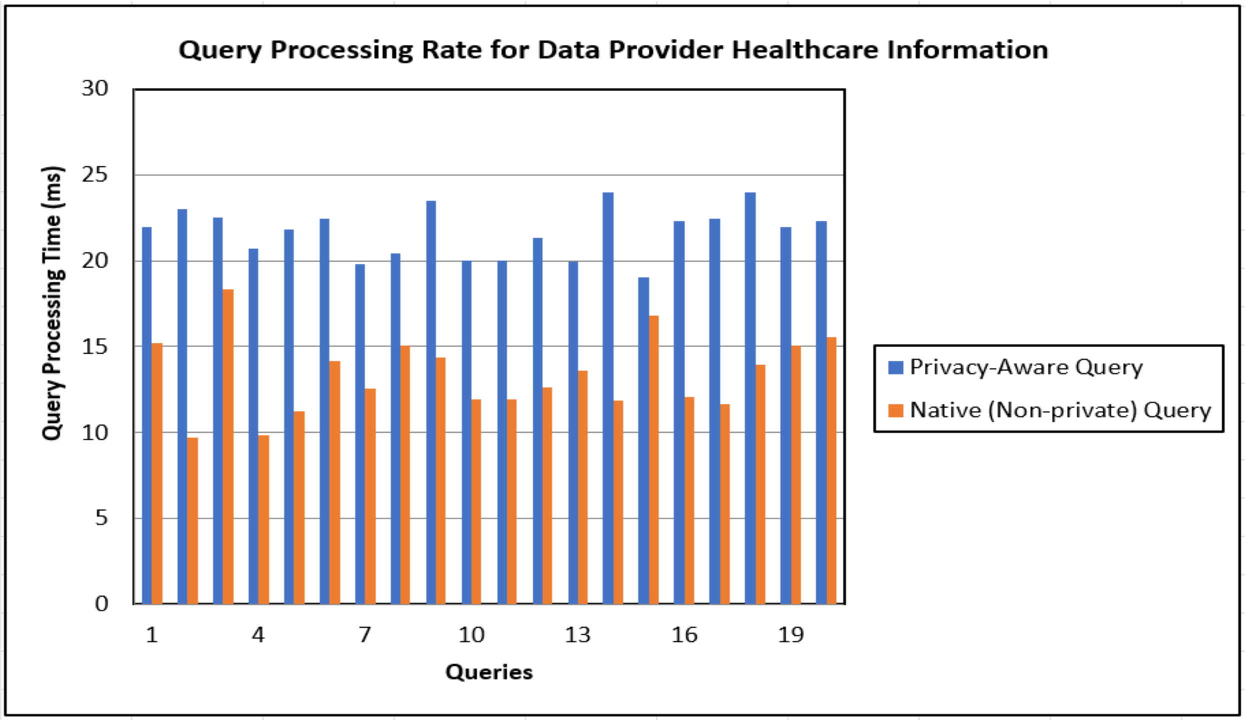


Figure 5.4: Query Processing Rate for Data Provider Healthcare Information

5.6.3 Query 3: Data Provider Consent Witness Information

Suppose we want to retrieve data values on data provider health information. The attribute data elements from which the query transaction is performed, include: *Witness Last Name*, *Witness First Name*, *Witness Phone Number*, *Witness Street*, *Witness City*, *Witness Province*, and *Witness Postal Code*, *etc.* The result of query processing rate for data provider consent witness information is displayed in Figure 5.5. The privacy-aware queries and native queries are designated with orange and green colouring, respectively. We define the query syntaxes for both privacy-aware and native (non-private) queries in the Appendix (see Section 7.3.3).

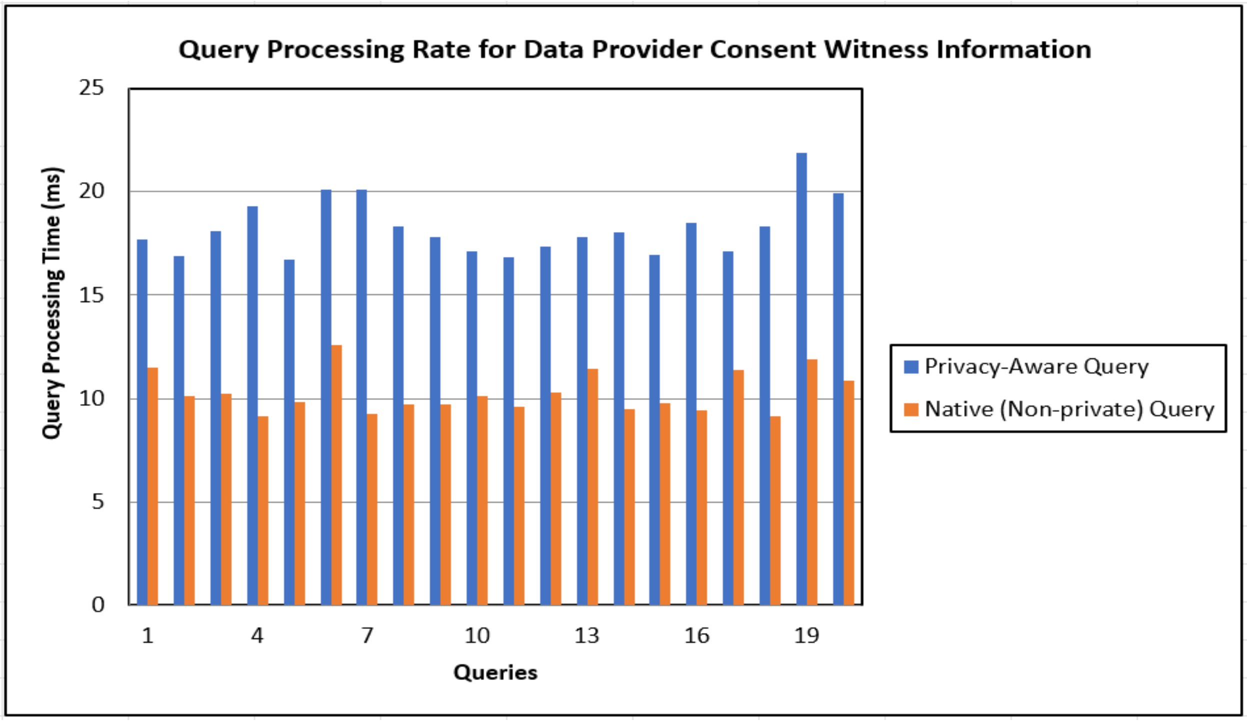


Figure 5.5: Query Processing Rate for Data Provider Consent Witness Information

Results Analysis and Discussion

We observe that query processing rate is in the same range of query response time as the preceding queries. Similar to preceding queries (*i.e.*, *Query 1* and *Query 2*), there is an increase in query processing time on privacy-aware queries in comparison to native (non-private) queries. From the query results, we note that the highest and lowest query processing time for native queries are 12.57 ms (milli seconds) and 9.15 ms (milli seconds), respectively. Moreover, an average query processing time of 10.28 ms is attained. In terms of privacy-aware query processing, we attain the highest and lowest query processing time of 21.85 ms and 16.71 ms, respectively. We also record an average query processing time of 18.24 ms.

We observe privacy overhead cost of 43.65% for privacy-aware query processing. This is attributed to data retrieval that involves greater number of attribute data elements in comparison to queries *Query 1* and *Query 2*. Moreover, the privacy-aware query parsing and analyzes involving these attribute data contribute to the increase in privacy query overhead cost. An analysis on the data provider privacy preferences indicate parameters of data information privacy with instance data representation of: *'Level-2: Internal Use'*, granularity privacy with instance data representation of: *'Existential: Information released to indicate the existence of data in repository, not actual data values'*, visibility privacy with instance data representation of: *'House: Data accessible to everyone who collects, access, and utilize the data'*, and purpose privacy with instance data representation of: *'Reuse-Selected: Data used for primary purpose for which data'*. We determine that based on the query response times attained from both privacy-aware and native query processing, the light-weight privacy infrastructure offers an efficient platform for data provider privacy-aware query processing.

5.6.4 Query 4: Data Provider Genome Meta-data Information

Suppose we want to retrieve data values on data provider genome meta-data information. The attribute data elements from which the query transaction is performed, include: *VCF Standard Version*, *Variant Caller Name*, *Variant Caller Version*, *Variant Call Date*, *Filters Applied*, *Variant Caller Regions*, *Annotation Software Name*, *Annotation Software Version*, *Annotation Date*, *Annotation Reference*, *Annotation Reference Version*, and *Annotation Fields*, *etc.*. We note that there are 46 attribute data covering different meta-data on genome files (that is, *fastq*, *bam*, and *vcf*). The result of query processing rate for data provider genome meta-data information is displayed in Figure 5.6. The privacy-aware queries and direct queries are designated with red and purple colouring, respectively. We define the query syntaxes for both privacy-aware and native (non-private) queries in the Appendix (see Section 7.3.4).

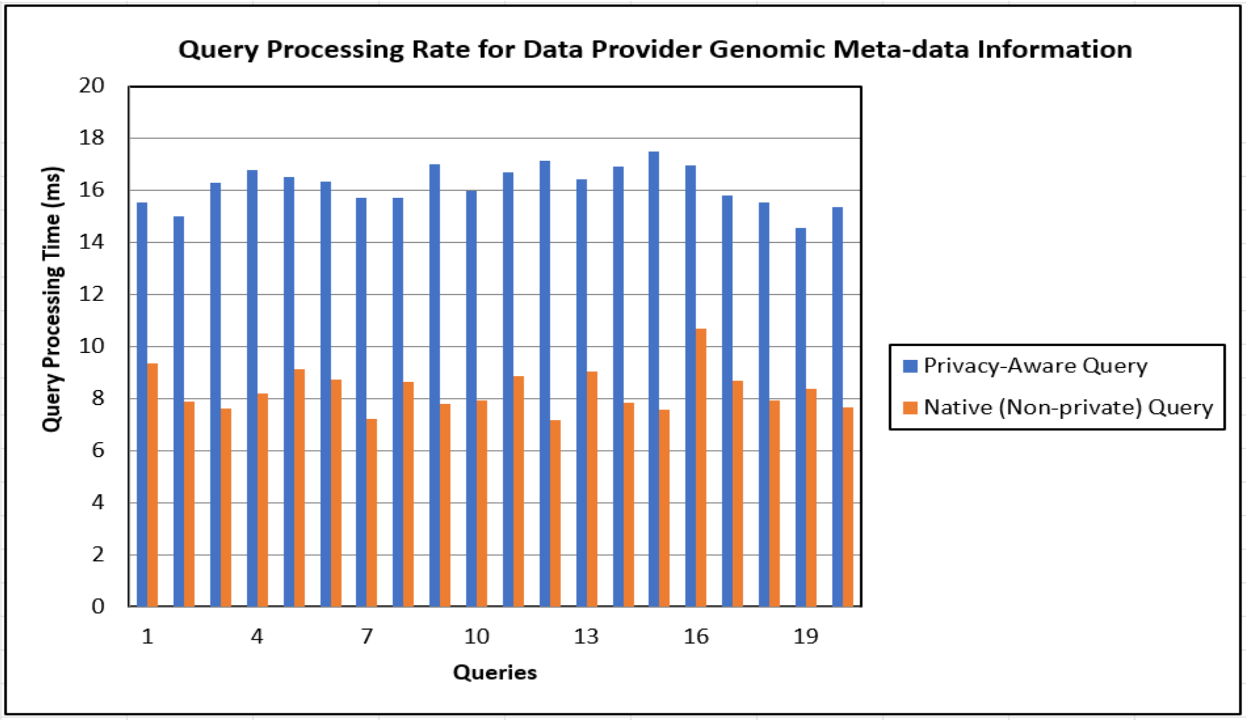


Figure 5.6: Query Processing Rate for Data Provider Genome Meta-data Information

Results Analysis and Discussion

Similar to previous queries, there is an increase in the query processing time for the privacy-aware queries in relation to native queries. From the query results, we note that the highest and lowest query processing time for native (non-private) queries are 10.70 ms (milli seconds) and 7.19 ms (milli seconds), respectively. Moreover, an average query processing time of 8.32 ms is attained. In terms of privacy-aware query processing, we attain highest and lowest query processing time of 17.49 ms and 14.58 ms, respectively. We also record an average query processing time of 16.19 ms.

Compared to the previous three queries, this query shows the highest privacy overhead query cost. We note 48.61% privacy overhead cost for privacy-aware query processing. This is mainly based on the higher number of attribute data on which privacy-aware data retrieval is performed. Additionally, privacy-aware query parsing and processing also contribute to this overhead cost. We analyzed the data provider privacy preferences and noted the following parameters for privacy-aware data retrieval: data information privacy with instance data representation of: *'Level-4: Restricted'*, granularity privacy with instance data representation of: *'Specific: Specific data item is accessed. A query for data item returns actual data value'*, visibility privacy with instance data representation of: *'House: Data accessible to everyone who collects, access, and utilize the data'*, and purpose privacy with instance data representation of: *'Reuse-Selected: Data used for primary purpose for which data'*. We determine that the query response times attained from both privacy-aware and native (non-private) query processing indicate an effective platform for query processing on the light-weight privacy model architecture.

5.6.5 Average Query Processing and Privacy Overhead Cost Analysis

We compute average query processing rate for all four queries and analyse the query response time for each query form. We note that there is a considerable increase in the query response time for privacy-aware queries in relation to direct queries. This is mainly attributed to the privacy overhead query cost from privacy-aware query parsing, analyzes, and execution.

Table 5.1 illustrates the average query response time and their respective privacy overhead cost. Here, we note that queries, *Query 1* (Data Provider Demographic Information), *Query 2* (Data Provider Health Information), *Query 3* (Data Provider Consent Witness Information), and *Query 4* (Data Provider Genome Meta-data Information) deliver privacy overhead query response time of 9.62 ms, 8.29 ms, 7.96 ms, and 7.87 ms, respectively. Moreover, an average privacy overhead cost of 8.44 ms (milli seconds) is realized for all queries processed. These privacy overhead query response times translate into 41.85%, 38.26%, 43.65%, and 48.61% for queries: *Query 1*, *Query 2*, *Query 3*, and *Query 4*, respectively. We note an average of 43.09%

privacy overhead query cost for all queries processed. We illustrate the average query processing time and privacy overhead cost for all queries in Figure 5.7.

Table 5.1: Summary of Average Query Response Time and Privacy Overhead Cost

Queries	**Average Query Response Time and Overhead Cost (ms)**		
	Native (Non-private) Query	**Privacy-Aware Query**	**Privacy Overhead Cost**
Query 1: Data Provider Demographic Information	13.37	22.99	9.62 (41.85%)
Query 2: Data Provider Health Information	13.38	21.67	8.29 (38.26%)
Query 3: Data Provider Consent Witness Information	10.28	18.24	7.96 (43.65%)
Query 4: Data Provider Genome Meta-data Information	8.32	16.19	7.87 (48.61%)
All Queries	11.34	19.77	8.44 (43.09%)

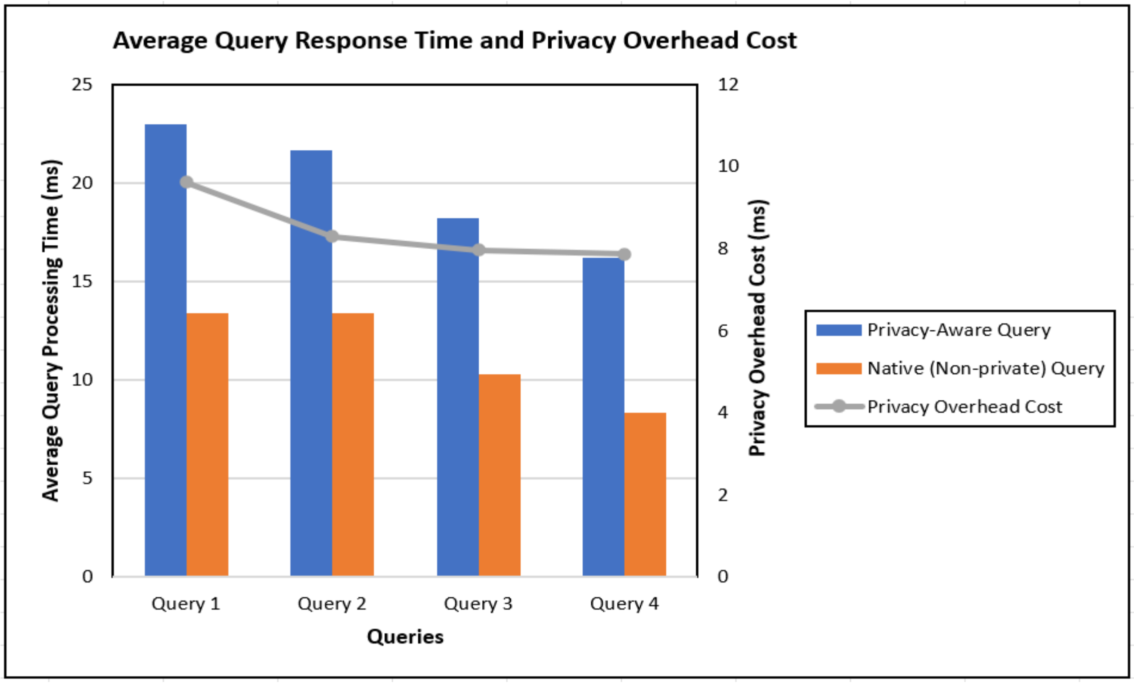


Figure 5.7: Average Query Processing Time and Privacy Overhead Cost

In general, we note that the average privacy overhead query cost is relatively very low. This indicates an effective privacy-aware query processing platform from the light-weight privacy infrastructure, and infers an optimum run of privacy-aware queries in comparison to native (non-private) queries. A review of the attained privacy overhead cost indicates that the light-weight privacy model system architecture and infrastructure, and resultant privacy overhead cost does not excessively or negatively impact on overall query transaction processing. Hence, we establish that the light-weight privacy infrastructure offers an efficient, reliable, and robust framework for privacy-aware query processing on data provider data (and related genome meta-data).

5.7 Comparison to Other Methodology Approaches

There have been some recent significant studies in the area of using blockchain platforms to enforce privacy on data providers' private and sensitive data. These approaches present important contributions with regards to various data protection measures against unauthorized data access to data provider's private data. For each research work, we discuss the overall privacy model and/or infrastructure and core merits that arise from the research. In comparison to our methodology, our approach addresses the privacy-preservation problem from important concepts of data element coupling, privacy-preservation modelling using privacy ontologies, and data store coupling. We comparatively explain how our methodology approach offers efficient and better measures for data provider's private data protection and identity preservation.

5.7.1 Daidone *et al.* [129]: Blockchain-based Privacy Enforcement in IoT domain

Daidone *et al.* [129] propose a blockchain-based privacy enforcement architecture where users can define how their data are collected and managed, and ascertain how these data are used without relying on a centralized manager. Their methodology adopts a blockchain to perform privacy preferences' compliance checks in a decentralized fashion on IoT devices; whiles ensuring the correctness of the process through smart contracts. The overall system is based on a privacy model designed for IoT systems and which has been outlined in Sagirlar *et al.* [134]. This privacy model supports standard privacy-related elements (*e.g.*, purpose). Moreover, the privacy model supports set of features to allow data owners to limit how and which data can be retrieved during IoT analytic processes. The privacy model allows the automatic generation of privacy preferences for newly derived data (*e.g.,* information resulting from data fusion). The research considers the privacy model to focus mainly on the traditional privacy preference components (*e.g.*, purpose, retention time).

The system architecture identifies the registration of both data owner (*i.e.*, user) and consumer devices on the blockchain. The smart environment at the data owner's device and consumer servers constitute the IoT network. The IoT network senses data from the user environment and sends these data to the consumer servers. For each of the data owner's IoT network and consumer IoT network, there are attached gateway blockchain nodes. Data collected by the gateway nodes are complemented with metadata that encodes the data owner's privacy preferences. Consumers register their privacy policies into the blockchain. For each IoT device, manufacturers specify system-defined privacy preferences. On the other hand, the data owner is free to add further restrictions to system-defined privacy preferences; by specifying owner-defined privacy preferences.

The system architecture relies on smart contracts for the enforcement of privacy preferences. The gateway nodes (which contains the smart contracts) verifies whether the consumer policy satisfies the constraints specified by the data owner in his/her privacy preferences. The proposed architecture leverages on a permissioned blockchain that is configured such that only given stakeholders are authorized to join the privacy compliance check. The system adopts Hyperledger Fabric blockchain platform. The Hyperledger Fabric creates peer-to-peer links between two or more nodes to exchange off-chain information, and keeping on-chain an evidence of the data exchange. Sensitive data are processed only within the data owner's IoT blockchain node. The remaining nodes only see the hashed data stored on the blockchain as a result of the smart contract execution. All communications take place via an encryption layer, established through Transport Layer Security (TLS).

5.7.2 Fernandez *et al.* [130]: Privacy-Preserving Architecture for Cloud-IoT Platform

Fernandez *et al.* [130] propose a cloud-IoT architecture, called *Data Bank*, that protects users' sensitive data by allowing them to control which kind of data is transmitted by their devices. The system architecture provides users (*i.e.,* data owners) with mechanisms to specify data-collection policies at device level and data-sharing policies at cloud level. Both cloud and local data repositories to allow users to keep their private data safe. Before data is transferred to the cloud repository, it will be temporarily stored in the local *Data Pocket*. The *Data Pocket*, which is under the user's control, consists of memory (*e.g.,* disk storage) and a microprocessor to keep pre-defined data collection policy and filter users' data before uploading to the cloud repository.

The system architecture contains a privacy-utility mechanism. This mechanism aims at finding the right balance between benefits gained and privacy lost when data is provided to external services. It recommends services to users based on users' pre-defined privacy criteria. Users can view and customise their privacy policy via the interface provided. The system offers enforcement of access control policies to restrict access to users' data by third parties.

Their system architecture consists of a four-tier architecture for cloud-IoT platforms: *application*, *cloud layer*, *data pocket*, and *sensors*. The application layer is the topmost layer and consists of a *user interface*. This specifies how users should interact with the *Data Bank*. The application layer could be deployed on internet-connected devices, via a website and/or mobile application. Moreover, there is a *data-sharing interface/manager*, which is considered as the application program interface (API) and controls the data visible to the services. The cloud layer contains five main components: (1) *access control enforcement module*, which receives requests from services and checks whether the service is authorised to access the requested data according to the policy; (2) *auditing module* keeps a log of all transactions that occur in the Data Bank; for example, granted access to data, denied access to data, data transfer transactions, *etc*; (3) *repository* is located in the cloud, and stores all the users' data in the form specified by the data collection policy (for example, data might be filtered/processed before it is stored in the cloud repository); (4) *privacy-utility mechanism* suggests services to the user, taking into account the specified preferences/privacy settings and the benefits provided by the services, trading data for benefits; (5) *Privacy policy* is a reference schema which is suggested by the privacy-utility mechanism and customised by the user. It is a specification for the access control enforcement module to check whether a service is authorised to access the requested data and for the privacy-utility mechanism to allocate tokens to services.

The *data pocket* layer runs in a local computing device (*e.g.*, the IoT hub) and deals with the data-collection policies. The purpose of the user interface in this layer is to capture the user's privacy preferences. The *sensors* layer contains IoT devices connected through the internet or local networks. The physical devices connect to the *Data Bank* via embedded drivers. The objects in this layer are not allowed to communicate to each other. In summary, the system provides a privacy preservation architecture where the user has control of the data collection and data sharing policies enforced, and the user can update the policies at any time.

5.7.3 Griggs *et al.* [131]: Healthcare Blockchain System using Smart Contracts for Secure Automated Remote Patient Monitoring

Griggs *et al.* [131] propose a healthcare blockchain system for secure automated remote patient monitoring. In their approach, the research outlines using blockchain-based smart contracts to facilitate secure analysis and management of medical sensors. The overall system architecture uses a private blockchain based on Ethereum protocol. The system architecture approach involves using sensors to communicate with a smart device that calls smart contracts, and writes records of all events on the blockchain.

The system activity flow is described as follows. A data provider (*i.e.,* patient) is equipped with various medical devices, such as blood pressure monitor, and is remotely monitored by a service provider (*i.e.,* doctor). The raw data is sent to a master *"smart device,"* typically a smartphone or tablet, for aggregation and formatting by the application. Once complete, the formatted information is sent to the relevant smart contract. In the Ethereum blockchain protocol, the source for information fed to the smart contracts is known as the *"Oracle"*. The *Oracle* in the smart device (master device) communicates directly to the smart contracts. The smart contract will then evaluate the provided data and issue alerts to both the patient and healthcare provider, as well as automated treatment instructions for the actuator nodes, if desired. Through this approach, no confidential medical information is stored on the blockchain or in the smart contracts because government privacy compliance specifications. The actual data transaction measurements are forwarded to a designated Electronic Healthcare Records (EHR) storage database. Moreover, the trace of new transactions are added to the blockchain platform.

Hence, blockchain transactions are linked to the EHR in order to provide authentication of the data in the data provider's medical history as a comprehensive record of healthcare. This authentication helps prevent and detect alterations of a data provider's EHR, whether it be on purpose or accidental. The system has a private and consortium-led blockchain which offers the functionality that only authorized viewers can read the blocks and only designated nodes can execute smart contracts and verify new blocks. In the consortium style of blockchain management, a set of pre-approved members operate the nodes in the blockchain, and a valid block must contain signatures from a minimum number of members. The smart contracts are modular and customizable for each data provider and their devices. The structure is tiered; having all master devices calling the same initial smart contract. This will in turn call the relevant sub-contract for the specific data provider (that is, patient's) device and pass it the input data and custom threshold values for privacy-aware data retrieval.

5.7.4 Methodology Evaluations and Comparison

We access each of these methodology approaches and compare with our proposed methodology. We identify some criteria for qualitative comparative analysis between the different methodology approaches. These criteria present different aspects of assessing the effectiveness of each methodology, and how the methodology delivers an efficient privacy preserving architecture for data provider private data. We outline the criteria as follows: data processing platform, blockchain platform technology, privacy model, privacy methodology and infrastructure, and data and query processing approach. Based on each criterion, we discuss the technical contributions and address the shortcomings of the methodology approaches. Finally, we highlight some ways in which our proposed methodology offers a better privacy preservation solution to data provider's private, sensitive data.

Data Processing Platform

With regards to data processing platform, our proposed methodology processes data primarily on a relational database and genomic data store. The methodology approach presented in Griggs *et al.* [131] states an Electronic Healthcare Record (EHR) database. The literature does not clearly specify the data processing or management model. Conversely, Daidone *et al.* [129] and Fernandez *et al.* [130] adopt an IoT device platform. It will be noted that data storage and organization in an IoT platform is completely different from relational database platform. These different data processing platforms bring into focus the comparison for data processing efficiency in between our proposed methodology to the other methodologies. As a result, we are unable to clearly specify which data platform offers an effective data storage and processing platform.

Blockchain Platform Technology

In terms of the adopted blockchain platform, Fernandez *et al.* [130] methodology does not employ any form of blockchain platform; as it uses a cloud repository for privacy management. The methodology in Daidone *et al.* [129] and Griggs *et al.* [131] adopt Hyperledger Fabric and Ethereum blockchain platforms, respectively. Ethereum blockchain platform is a decentralized, open-source blockchain with smart contract functionality. Moreover, the platform uses Proof-of-Work algorithm for the mining procedures. The public nature of Ethereum blockchain, coupled with Proof-of-Work mining algorithm, creates an overhead of processing power, energy requirements, additional configurations for private data management. This makes it an undesirable adoption for an effective privacy-preserving architecture. The methodology in Griggs *et al.* [131] adopts separate, private chain using Ethereum's protocol. This enables the privacy infrastructure access to experiment outside the set parameters of the Ethereum blockchain and eliminates the need to spend *Ether* cryptocurrency.

Hyperledger Fabric uses a number of different setup configurations and parameters. This blockchain platform does not use Proof-of-Work for the mining procedures. Hyperledger offers "plug-and-play" ability to integrate components such as consensus algorithm and membership services. Moreover, it offers smart contracts; which are hosted using a form of data container technology. Our proposed methodology adopts Multichain blockchain. From the literature and platform documentation, Multichain blockchain is an off-the-shelf platform for creation and deployment of private blockchains. It is one of the robust and effective blockchain platform designed to offer functionalities of dynamic private blockchain activity. Additionally, the blockchain platform offers a mining procedure to securely process a consensus agreement without Proof-of-Work (PoW) and its associated overhead computational costs.

In summary, the Hyperledger Fabric delivers a private platform, but the overhead setup configurations and parameter settings require much more specialized knowledge for its adoption and deployment. This offers a better motivation for our adoption of Multichain blockchain as an efficient approach for user-friendly API connectivity, data/asset management, and application programming.

Privacy Model

A privacy model for privacy-preserving system infrastructure presents a framework and data flow design for private data management. Hence, a privacy model offers an efficient design construct to manage, protect data provider's private data, and provide the foundation for the privacy infrastructure. The methodologies presented and discussed in Fernandez *et al.* [130] and Griggs *et al.* [131] do not clearly specify any form of privacy model. Fernandez *et al.* [130] implement a form of a privacy design which adopts a cloud layer of different components (of privacy utility mechanism and privacy policy reference) for privacy management. Griggs *et al.* [131], on the other hand, use a smart contract framework embedded in the blockchain network to enforce a privacy model.

The methodology in Daidone *et al.* [129] adopts privacy model defined in Sagirlar *et al.* [134]. This privacy model is designed specifically for IoT devices. The privacy model supports standard privacy-related variables (*e.g., purpose* and *retention time*). Moreover, the privacy model supports set of features to allow data owners to limit how and which data can be retrieved during IoT analytic processes. The privacy model allows automatic generation of privacy preferences for newly derived data (*e.g.,* information resulting from data fusion). Our proposed methodology adopts a formal (dynamic) contextualized privacy ontology that applies to different application domains and platforms. The contextualized privacy ontology offers the foundation for the design of all entities and predicates in the data privacy-preservation process. Additionally, it offers a modelling framework for procedural activities and semantics for private, sensitive data and querying processing by data providers, data collectors/accessors (*i.e.*, service providers) and third-party data accessors. The privacy ontology defines constructs for *attribute data*, *data provider*, *data collector*, and *data privacy policy.* The *data privacy policy* construct expresses *purpose*, *granularity*, *visibility*, *retention duration*, *location*, and *third-party* privacy semantics.

In summary, the privacy model adopted in our proposed methodology offers a better modelling and design construct for data provider's private data. This is because of its dynamic adaptability and application to different data processing domains and device platforms.

Privacy Methodology & Infrastructure

Relevant and useful outcome for a privacy-preservation approach is dependent on the underlying privacy methodology and infrastructure. A privacy methodology outlines design principles, procedures, and adopted technology platforms to protect data provider's privacy and identity. Griggs *et al.* [131] adopt a system architecture that involves coupling a master (smart) device, Electronic Healthcare Record (EHR) database, and an Ethereum protocol (that consists of private and consortium-led blockchain platforms). In their approach, data provider private medical data and transactions are stored in the EHR database, and a smart contract embedded in the blockchain stores, processes, and authenticates data provider privacy preferences. From the evaluation, we observe that the literature in Griggs *et al.* [131] does not specify procedures and activities for private data protection. There is no information on how the smart contract is customized for each patient and their devices. Moreover, the methodology does not discuss how the smart contract generates and processes privacy preferences and their linkage to data provider private data.

Fernandez *et al.* [130] propose privacy methodology/infrastructure that consists a four-tier platform of: application, cloud layer, data pocket, and sensors. The system architecture involves cloud and local data repositories and data provider IoT devices. In their approach, the privacy infrastructure contains a *Data Bank* that facilitates both cloud and local data repositories to allow users to keep their private data safe. Before data is transferred to the cloud repository, it is temporarily stored in the local *Data Pocket*. The *Data Pocket*, which is under the user's control, consists of memory (*e.g.*, disk storage) and a microprocessor to keep the pre-defined data collection policy, and filter users' data before uploading to the cloud repository. The *Data Bank*, which contains a privacy-utility mechanism, controls the right balance between benefits gained and privacy lost when data is provided to external services. Users can view and customise their privacy policy via the interface provided. Our evaluation indicates that the *Data Bank* enforces access control policies to authenticate access to users' data. This component is housed in the cloud layer and its protection from attacks is based on adopted encryption protocols and security infrastructure. This highlights a major drawback as the encryption protocols could be susceptible to different attack techniques and vulnerabilities.

The privacy methodology and infrastructure addressed by Daidone *et al.* [129] present an architecture that involves user IoT smart devices, blockchain platform, and consumer (*i.e.*, service provider) system servers. The methodology adopts a blockchain to perform privacy preferences' compliance checks in a decentralized fashion on IoT devices; whiles ensuring the correctness of the process through smart contracts. The system architecture identifies the registration of both data owner and consumer devices on the blockchain. The smart environment at the data owner's device and consumer servers constitute the IoT network. For each of the data owner's IoT network and consumer IoT network, there are attached gateway blockchain nodes. Data collected by the gateway nodes are complemented with metadata that encodes the data owner's privacy preferences. Consumers register their privacy policies into the blockchain. For each IoT device, manufacturers specify system-defined privacy preferences. On the other hand, the data owner is free to add further restrictions to system-defined privacy preferences; by specifying owner-defined privacy preferences. The system architecture relies on smart contracts for the enforcement of privacy preferences. The gateway nodes (which contains the smart contracts) verifies whether the consumer policy satisfies the constraints specified by the data owner in his/her privacy preferences. The use of a Hyperledger Fabric blockchain platform offers an important contribution in which the privacy preferences are well secured, and the smart contract efficiently authenticates data retrieval. One drawback from the methodology is that the literature does not offer information for a conceptual privacy methodology for data provider attribute data protection.

Our proposed methodology tightly couples data elements (of attribute data, privacy preferences data, and data accessor data) to generate a *privacy tuple.* Moreover, the methodology tightly couples different data repositories of relational database and genomic data store with blockchain platform. Based on the unique database architectures, each data repository stores different data values. The database coupling architecture (which is primarily controlled by the blockchain platform for private data query authentication and data retrieval authorization) offers a robust platform for storage of data provider private data and privacy preferences.

In summary, we observe that none of the previous approaches distinctively specify a conceptual privacy methodology or model for attribute data protection and data accessor query processing. Most of the approaches focus primarily on the system architecture framework and the technological platform of either using IoT devices, or smart contracts coupled together with blockchains, and/or cloud computing networks.

Data & Query Processing Approach

The ability to process data and query request on a privacy system infrastructure determines the efficiency in data processing; whiles delivering data protection in the underlying data provider's private data. The methodology approach addressed in Daidone *et al.* [129] specifies an evaluation based on: (1) the data throughput (that is, the amount of data that the blockchain can manage in a time unit); (2) the space overhead implied by the additional information (metadata) that our solution requires to insert in the original IoT device streams; and (3) the time spent in privacy preference enforcement and data release. The evaluation results indicate that data throughput is largely dependent on the time and continuous rate of data transmission from an IoT device system. Additionally, there is an overhead cost in the data processing of private data transmitted from the IoT device and the privacy compliance checks. The overhead cost increases with large data streams and increase in consumers.

In Fernandez *et al.* [130], the authors do not discuss an evaluation of the privacy infrastructure. From the literature, the research work indicates a conceptual overview, and discusses guidelines for implementation using class diagrams; without actual system implementation. Hence, we are unable to access and evaluate the effectiveness of the overall privacy methodology and infrastructure. The privacy infrastructure addressed in Griggs *et al.* [131] do not discuss evaluation and assessment of data retrieval and query processing. Though the research work specifies implementation, the literature does not outline evaluation metrics and privacy overhead cost in adopting a blockchain platform.

Methodology Evaluation Analysis

We determine that the research work in Daidone *et al.* [129] present relevant propositions and recommendations for devising an efficient privacy methodology to protect data provider's private data. Moreover, the work outlines expressive and practical metrics for privacy preservation in the context of IoT domains. A closely-related research work in Azaria *et al.* [137] addresses adoption of blockchain to manage data access and permissions on medical records. The system methodology enables patient-initiated data exchange between different medical parties, and there are specific authorizations and restrictions, such as, an expiration date for viewership rights. Moreover, medical records are stored locally in separate service provider and patient databases whiles copies of authorization data are stored on each node in the blockchain network.

Conversely, our proposed methodology approach identifies data and query processing based on the privacy preferences compliance, authentication, and authorization from the blockchain platform. Our work addresses query recall (visibility privacy), query precision (granularity privacy), query response time, and privacy overhead cost of privacy-aware queries over corresponding native (non-private) queries. Evaluating and assessing the effectiveness of all methodology approaches, we can deduce that based on the uniqueness of each privacy model, privacy methodology and infrastructure, and the type of adopted blockchain platform, it is impractical to compare the performance of data and query processing of our methodology approach in relation to the other approaches.

We present a comparative analysis and evaluation of our proposed methodology in relation to other approaches in Table 5.2. This tabular analysis summarizes the discussions regarding methodology approaches addressed in the literature, and outlines the key contributions from each of the approaches.

Table 5.2: Qualitative Analysis of Proposed Methodology and Other Approaches

Methodology Approach & Criteria	**(1) Proposed Methodology**	**(2) Daidone *et al.* [129]**	**(3) Fernandez *et al.* [130]**	**(4) Griggs *et al.* [131]**
(1) Data Storage Platform	Relational database	IoT device platform	Cloud-IoT platform	Electronic Healthcare Records (EHR) storage database
(2) Blockchain Platform Technology	Multichain (permissioned or private blockchain platform accessible by data providers and data collectors/accessors)	Hyperledger Fabric (permissioned or private blockchain platform accessible by data owner and consumer devices)	No defined blockchain platform. Uses a cloud repository for *privacy utility mechanism*	Ethereum protocol (private and consortium-led blockchain platform)
(3) Privacy Model	Adopts a formal (dynamic) contextualized privacy ontology that applies to different application domains and platforms	Adopts privacy model defined in Sagirlar *et al.* [134] (that handles *purpose* and *retention time* privacy), and is designed for IoT devices	No clearly defined privacy model. Adopts a cloud layer of different components (of privacy utility mechanism and privacy policy reference) for privacy management	No clearly defined privacy model. Uses a smart contract embedded in a blockchain network node to enforce a privacy model
(4) Privacy Methodology & Infrastructure	1. Tightly couple data elements (attribute data, privacy preferences data, and data accessor data) to generate a *privacy tuple* 2. Tightly couple data stores (relational database and genomic data store) with blockchain platform	1. The architecture involves user IoT smart devices, blockchain platform, and consumer (*i.e.*, service provider) systems servers 2. Each of the blockchain nodes are embedded with smart contracts 3. A gateway node processes data and enforces privacy preservation	1. System architecture involves cloud and local data repositories and data provider IoT devices. 2. System architecture consists of four-tier platform: application, cloud layer, data pocket, and sensors. 3. Architecture controls user data to cloud services	1. System architecture involves coupling of master (smart) device, EHR database, and Ethereum protocol (private and consortium-led blockchain platform) 2. Smart contract is embedded in the private blockchain platform
(5) Data/Query Processing Approach	Privacy preferences are stored in the blockchain platform. The blockchain platform serves as an access control and query authentication medium. All queries are processed through the blockchains to ensure privacy preservation	Privacy preferences are stored in the data owner blockchain gateway node. The gateway node processes and authenticates data values before data moves to consumer	*Data pocket* repository keeps predefined data collection policy and filters users' data before uploading to the cloud repository. Privacy-utility mechanism determines data retrieval based on users' pre-defined privacy criteria	Smart contract stored in the Ethereum blockchain contains user privacy preferences. Smart contract and blockchain control access and prevent changes to users' data stored in the EHR database.

In summary, a deductive and prudent recommendation for a preferred choice of privacy methodology from among other methodology approaches discussed could be made. Based on the analysis of data storage platform, blockchain technology, privacy model, privacy methodology and infrastructure, and data or query processing approach, the methodology approach by Daidone *et al.* [129] can be chosen as a preferred methodology approach. This is based on the detailed examination of the unique and expressive adopted privacy model; which handles peculiar privacy metrics, such as, purpose, retention privacy, *etc.* Moreover, the methodology approach adopts permissioned Hyperledger Fabric blockchain platform which is of an open-source architecture and can be configured uniquely for different application domains. Finally, there is a better and efficient approach (as compared to the other methodology approaches) in the storage and management of privacy preferences regarding data provider's private, sensitive data.

5.8 Summary

This chapter discusses software security assessment, data repositories design principles and assessment, query processing evaluation, and query response results analysis on proposed light-weight privacy infrastructure. We address software system security assessment procedures for an effective system design. The procedures focus on application data transmission medium, client-side user interface, and web server code assessments. Moreover, we discuss data repository evaluation on the effectiveness of the coupled data platform to offer efficient data processing, storage, and management. We address inference and SQL injection privacy model attacks and discuss their preventive measures from the architecture of the light-weight privacy infrastructure.

Finally, we discuss overall query processing on the privacy infrastructure. We evaluate query effectiveness in terms of query recall (visibility privacy) and query precision (granularity privacy). We discuss query processing and response time for four query categories, and analyze the average privacy overhead cost of privacy-aware queries over corresponding native (non-private) queries. We discuss other methodology approaches and evaluate qualitative comparison of our proposed methodology to these related methodology approaches, based on some defined criteria. We highlight key technical contributions and discuss areas where our proposed methodology offers better a privacy-preservation infrastructure.

Chapter 6 concludes this dissertation. We summarize our research propositions, and outline major contributions from this research. We discuss some key research significance and applications in line with managing genomic data. We address possible open issues areas and future work.

Chapter 6

Conclusion

This dissertation addresses the need for a practical approach to protect data provider's private and sensitive healthcare data using immutable, tamper-resistant data platforms. We discuss different data provider attribute data categories and address the need for efficient privacy-preservation methods on these data. We also discuss research methodology and implementation methods to provide a framework platform of light-weight privacy infrastructure. Moreover, we discuss the security model for the light-weight privacy infrastructure, and analyse evaluations and query results of privacy-aware query processing. We address merits and key outcomes from the proposed methodology framework.

In this chapter, we review discussions on providing an efficient platform of light-weight privacy infrastructure in Section 6.1. We discuss overall contributions arising from this research in Section 6.2, and address likely areas of research applications in Section 6.3. In Section 6.4, we address some areas of open issues and future work based on which the research may be pursued further.

6.1 Research Overview

Data privacy-preservation is a necessary requirement in the processing of data provider's personalized, private data across varied data stored in repositories. Much more important is the need to preserve privacy in data provider demographic, biographical, healthcare data (and genomic meta-data within the case study application domain of genomic healthcare). Privacy-preservation policies provide guidelines and recommendations to protect data provider's private, sensitive data in data repositories. Additionally, the formulation and implementation of methodologies offer efficient platforms to preserve privacy in the private data.

We propose a light-weight privacy infrastructure. This model offers an effective integration of various system and data repository components to provide a data platform where data provider's private information and transaction data on private data are proficiently collected, processed, and managed by data collectors. The overall approach for the proposed research is to formulate a contextualized privacy ontology with unique semantics and properties for privacy preservation. We implement tight-coupling of three data elements (namely, attribute meta-data, data provider privacy preferences data, and data accessor data) into a privacy tuple to be stored in the blockchains. Moreover, we implement tight-coupling of different data platforms of relational database, blockchains, and genome data store. The key reason for this approach is to maximize the merits from all data management platforms. We discuss that the methodology approach and the evaluation results attained validate a better approach for managing data provider's private, sensitive data. Our evaluation and result analysis establishes a robust infrastructural implementation for the light-weight privacy infrastructure. This privacy model offers a secured, immutable, and efficient privacy-aware query processing platform for data provider private data.

6.2 Research Contributions

We discuss the main research contributions arising from the dissertation in line with the light-weight privacy infrastructure discussed in Chapter 3, and the implementation procedures described in Chapter 4. Moreover, the evaluation and results analysis discussed in Chapter 5 establishes these research contributions outlined in Chapter 1 (recall Section 1.4). We summarize the technical contributions as follows:

6.2.1 Data Provider Access and Control of Privacy Policy Preferences

In Chapter 3, we formulate a privacy framework where data providers have ultimate access and control of their privacy preferences. The system methodology overview of the privacy framework is discussed in Section 3.2. This is based on tight-coupling of data provider's attribute data, privacy preferences, and data accessor data elements. We discuss this tight-coupling approach in Section 3.3 and its implementation in Section 3.4. Additionally, this privacy framework offers the platform where data providers can flexibly update or change their privacy preferences, and every change request is validated by both data providers and data collectors (or service providers). The privacy model framework platform delivers a permanent and immutable record of all data provider privacy preferences, which is easily referenced and offers a necessary input platform for privacy audit. Hence, this framework offers opportunities to review all forms privacy violations and presents an essential tool for privacy obligations.

6.2.2 Formulated Contextualized Privacy Ontology

In Chapter 3, Section 3.1 proposes a privacy model framework that uses a contextualized privacy ontology as foundational model; with crafted design constructs of semantics and predicates. We discuss the detailed implementation of the privacy ontology in Chapter 4 (recall Section 4.1). This formulated privacy ontology outlines key characteristics of *purpose*, context, data provider privacy preferences, user roles, contextual norms, transmission medium, and other privacy-related factors per data provider's privacy policy. Moreover, the formulated privacy ontology presents different entity classes, sub-classes, their respective attributes, and associated properties to handle prevalent privacy-related scenarios in privacy-preserving data management.

6.2.3 Tight-coupling of Data Elements

We propose and implement an efficient procedure to tightly couple attribute data, data provider privacy preferences, and data accessor profile data elements into a privacy tuple. We discuss the data element tight-coupling in Section 3.3 and its implementation approach in Section 3.4. Moreover, we discuss the detailed implementation in Chapter 4; where Section 4.2.3 discusses the extracting the data elements from user interfaces, and Section 4.2.4 discusses the coupling procedures of the data elements to generate the privacy tuple. The attribute data identifies each data provider private, sensitive attribute data value collected from or provided by the data provider. These data are derived from personal biographical, demographic, and health information, or transactions on healthcare data. The data provider privacy preferences are derived from privacy policy taxonomy items in the form of purpose use, granularity (data precision), visibility (data accessibility), retention duration, effective date, data information, third-party data accessor, and purpose contextual norms.

The data accessor profile data are derived from different access levels and permissions in line with privacy policy statements. These permissions are used to uniquely identify different data access user roles on which types of attribute data accessors can query or retrieve information. The data element binding enables data retrieval or query processing to be dependent on the privacy preferences and data accessor profiles specified in the privacy policy. The tightly-coupled privacy tuple is transmitted through an encrypted application transmission medium and stored in the blockchain platform.

6.2.4 Tight-coupling of Relational Database, Blockchains, and Genome Data Store

We implement an efficient data platform coupling that uses relational database, private blockchains, and genomic data store to deliver an encrypted, secured data processing platform for data accessors. In Chapter 3, Section 3.5 discusses the methodology for data platform coupling, whiles Section 3.6 discusses the implementation approach. The relational database offers a scalable, high throughput, and efficient querying engine to meet the expected high data processing needs. Conversely, the blockchain platform offers decentralized data control that is tamper-resistant, and immutable for the protection of privacy-related data values. The blockchains offers efficient change request and approval (from service providers and data providers, where necessary).

The genome data store (*i.e.*, Linux cluster Helix HPC) provides a platform to store and manage NGS genomic data (which do have very large file sizes). Additionally, the genome data store has specialized mode of transaction and query processing for genomic data. The overall architecture of data repository coupling presents an efficient process of coupling disparate, related, healthcare data on data providers (*i.e.*, patients) within a uniform system architecture. The detailed implementation of the data platforms coupling is discussed in Chapter 4. Section 4.3 discusses modelling, design, and development of the relational database, whiles Section 4.4 discusses design and development of permissioned (private) blockchain platform. Moreover, we discuss the implementation for privacy-aware query processing in Section 4.3.3.

6.2.5 Research Application for Healthcare Data

We implement the light-weight privacy infrastructure framework using healthcare data as a case study. This implementation involves building software user interfaces to connect to a relational database and blockchain platform. We adopt secured data communication channel for data transfer between the system components of relational database, blockchains, and genomic data store. We use an Alberta biotechnology company (*i.e.,* Tunote Oncogenomics) which is interested in providing Alberta cancer patients with access to commercial oncogenomic services, as the case study. In Chapter 4, Section 4.2 discusses user interface system modelling and development for generation of privacy tuple. Our implementation establishes the concept of developing the light-weight privacy infrastructure using a verifiable software and database platform. Moreover, this implementation validates overall research propositions. In Chapter 5, we discuss evaluation analysis of the implementation, where Section 5.3 assesses the suitability of the data repositories. Section 5.6 discusses query processing for privacy-aware queries and the privacy overhead cost in relation to native (non-private) queries.

6.3 Research Applications

The dissertation research project delivers a system prototype for managing patient healthcare data and genomic data. The case study for the research project is Tunote Oncogenomics Inc. We discuss the private healthcare and genomic data management procedures of Tunote Oncogenomics in Chapter 1 (recall Section 1.5). This is a local Alberta company specialized in genome sequencing and analysis for cancer-related diseases.

Our proposed system aims to offer healthcare end-users access to practice-informing data, with patient-centred data access assurances. The proposed system provides a platform that offers a made-in-Alberta solution, utilizing resources readily available in Alberta. Moreover, the system prototype provides potential secondary benefit of driving state-of-the-art bioinformatic and computational biology centres (B/CB) best-practices in Alberta. The potential advancements in patient-centred security that the system prototype suggests will serve to highlight the prospects in Alberta's B/CB resources. It is expected that with further developments to fully operational system, patients will willingly share genomic and other healthcare data with bioinformatic and computational biology centres. This will stem from the knowledge that state-of-the-art security measures and prospective data usage consent management have been improved and duly established from the system architecture.

Through further system interaction improvements in the design, implementation and evaluation measurement of this system prototype, this research offers an excellent opportunity to survey best-practices in B/CB in Alberta. Given the importance of data and its availability to the practice of B/CB, our research has the potential to inform database structure and utilization standards.

6.4 Open Issues and Future Work

We envision some areas of open issues and future work, as part of the proposed methodology of a light-weight privacy infrastructure. In terms of some open issues, we address the choice of private blockchain adopted for the privacy model architecture. It must be noted that different (private) blockchain platforms are incorporated with distinct and peculiar functionalities for data storage and retrieval. The functionalities deliver merits that tend to offer improved performance or otherwise negatively affect performance with relational database coupling. For example, in our system architecture we adopt Multichain blockchain platform. This blockchain is designed with a data stream transaction ledger functionality, that functions like relational database schema and tables. This functionality offers efficient storage structure and indexing of transaction data items in the data streams. Moreover, there is a better data retrieval and query processing on the blockchain.

Another open issue is the form of system administration on the private blockchain. It is expected that the blockchain is properly secured and protected from unauthorized access. The only medium of transaction processing is through the API connection from the application Point-of-Contact user interfaces. Any other interaction aside the user interfaces compromises the purpose of the blockchain platform in the overall system architecture of the light-weight privacy infrastructure.

There are some areas of future work that this research may be pursued further. We outline some of them briefly.

First, we anticipate the execution of complex relational *join* queries on data provider biographic, demographic, and healthcare data. Relational *join* queries present elaborate data retrieval from relational databases. Thus, processing such queries will improve the query processing experience on the tightly-coupled data repository platforms.

Second, we envision an approach of storing some relevant parts of the genomic data on the blockchain data platform. This, we believe will enhance the greater possibility of protecting patient (*i.e.*, data provider) privacy on the genomic data. Additionally, this will offer a better authentication approach of data access and query processing on the genomic data.

Finally, we expect to have a better mechanism or mode of comparatively analysing our proposed methodology with other methodology approaches in terms of query performance. We propose to implement their privacy models and methodologies, and adopt similar data storage platform to enable running related queries in comparison to our methodology. As a result, we are able to practically analyse and compare the metrics of data retrieval and query processing rate among the privacy methodologies.

Bibliography

[1] Latanya Sweeney. 2002. k-Anonymity: A Model for Protecting Privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, iss. 5, October 2002, pp. 557-570. Doi: https://doi.org/10.1142/S0218488502001648

[2] Ashwin Machanavajjhala, Johannes Gehrke, Daniel Kifer, and Muthuramakrishnan Venkitasubramaniam. 2006. l-Diversity: Privacy Beyond k-Anonymity. In: *Proceedings of the 22nd International Conference on Data Engineering (ICDE 2006)*, April 2006, Atlanta, Georgia, USA, pp. 24. Doi: https://doi.org/10.1109/ICDE.2006.1

[3] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. 2007. t-Closeness: Privacy Beyond k-Anonymity and l-Diversity. In: *Proceedings of 23rd International Conference on Data Engineering (ICDE 2007)*, April 2007, Istanbul, Turkey, pp. 106-115. Doi: https://doi.org/10.1109/ICDE.2007.367856

[4] Cynthia Dwork. 2006. Differential Privacy. In: *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming, Part II (ICALP 2006)*, July 2006, Venice, Italy, pp. 1-12. Lecture Notes in Computer Science, vol 4052. Springer, Berlin, Heidelberg. Doi: https://doi.org/10.1007/11787006_1

[5] Ninghui Li, Min Lyu, Dong Su, and Weining Yang. 2016. Differential Privacy: From Theory to Practice. *Synthesis Lectures on Information Security, Privacy, & Trust*, Morgan & Claypool Publishers 2016, pp. 1-138. Doi: doi.org/10.2200/S00735ED1V01Y201609SPT018

[6] Jaewoo Lee and Chris Clifton. 2012. Differential Identifiability. In: *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2012)*, August 2012, Beijing, China, pp. 1041-1049. Doi: https://doi.org/10.1145/2339530.2339695

[7] Weina Wang, Lei Ying, and Junshan Zhang. 2016. On the Relation Between Identifiability, Differential Privacy, and Mutual-Information Privacy. *International Journal of IEEE Transactions on Information Theory*, vol. 62, iss. 9, September 2016, pp. 5018-5029. Doi: https://doi.org/10.1109/TIT.2016.2584610

[8] Ninghui Li, Wahbeh H. Qardaji, Dong Su, Yi Wu, and Weining Yang. 2013. Membership Privacy: A Unifying Framework for Privacy Definitions. In: *Proceedings of 2013 ACM SIGSAC Conference on Computer and Communications Security (CCS 2013)*, November 2013, Berlin, Germany, pp. 889-900. Doi: https://doi.org/10.1145/2508859.2516686

[9] Stephen E. Fienberg and Julie McIntyre. 2004. Data Swapping: Variations on a Theme by Dalenius and Reiss. In: *Proceedings of the International Workshop on Privacy in Statistical Databases (PSD 2004)*, June 2004, Barcelona, Spain, pp. 14-29. Lecture Notes in Computer Science, vol. 3050. Springer, Berlin, Heidelberg. Doi: https://doi.org/10.1007/978-3-540-25955-8_2

[10] Charu C. Aggarwal and Philip S. Yu. 2004. A Condensation Approach to Privacy Preserving Data Mining. In: *Proceedings of the 9th International Conference on Extending Database Technology (EDBT 2004)*, March 2004, Heraklion, Crete, Greece, pp. 183-199. Doi: https://doi.org/10.1007/978-3-540-24741-8_12

[11] Charu C. Aggarwal and Philip S. Yu. 2008. A General Survey of Privacy-Preserving Data Mining Models and Algorithms. In: *Aggarwal C.C., Yu P.S. (eds) Privacy-Preserving Data Mining. Advances in Database Systems*, vol. 34, Springer, Boston, MA, USA, pp. 11-52. Doi: https://doi.org/10.1007/978-0-387-70992-5_2

[12] Rakesh Agrawal and Ramakrishnan Srikant. 2000. Privacy-Preserving Data Mining. In: *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data (SIGMOD 2000)*, May 2000, Dallas, Texas, USA, pp. 439-450. Doi: https://doi.org/10.1145/342009.335438

[13] Ramakrishnan Srikant. 2002. Privacy Preserving Data Mining: Challenges and Opportunities. In: *Proceedings of the 6th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD 2002)*, May 2002, Taipei, Taiwan, pp. 13. Lecture Notes in Computer Science, vol. 2336, Springer, Berlin, Heidelberg. Doi: https://doi.org/10.1007/3-540-47887-6_2

[14] Alex Gurevich and Ehud Gudes. 2006. Recent Research on Privacy Preserving Data Mining. In: *Proceedings of the Second International Conference in Information Systems Security (ICISS 2006)*, December 2006, Kolkata, India, pp. 377-380. Lecture Notes in Computer Science, vol. 4332. Springer, Berlin, Heidelberg. Doi: https://doi.org/10.1007/11961635_32

[15] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant and Yirong Xu. 2002. Hippocratic Databases. In: *Proceedings of 28th International Conference on Very Large Data Bases (VLDB 2002)*, August 2002, Hong Kong, China, pp. 143-154. Morgan Kaufmann. [Online]. Available: www.vldb.org/conf/2002/S05P02.pdf (Accessed on December 10, 2021).

[16] Ken Barker, Mina Askari, Mishtu Banerjee, Kambiz Ghazinour, Brenan Mackas, Maryam Majedi, Sampson Pun, and Adepele Williams. 2009. A Data Privacy Taxonomy. In: *Proceedings of the 26th British National Conference on Databases (BNCOD 2009)*, July 2009, Birmingham, UK, pp. 42-54. Lecture Notes in Computer Science, vol. 5588. Springer, Berlin, Heidelberg. Doi: https://doi.org/10.1007/978-3-642-02843-4_7

[17] Kambiz Ghazinour, Maryam Majedi, and Ken Barker. 2009. A Model for Privacy Policy Visualization. In: *Proceedings of the 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC 2009)*, vol. 2, July 2009, Seattle, Washington, USA, pp. 335-340. Doi: https://doi.org/10.1109/COMPSAC.2009.156

[18] Ken Barker. 2015. Privacy Protection or Data Value: Can We Have Both? In: *Proceedings of the 4th International Conference on Big Data Analytics (BDA 2015)*, December 2015, Hyderabad, India, pp. 3-20. Lecture Notes in Computer Science, vol. 9498. Springer, Cham. Doi: https://doi.org/10.1007/978-3-319-27057-9_1

[19] Michael Mireku Kwakye and Ken Barker. 2016. Privacy-Preservation in the Integration and Querying of Multidimensional Data Models. In: *Proceedings of the 14th Annual Conference on Privacy, Security and Trust (PST 2016)*, December 2016, Auckland, New Zealand, pp. 255-263. Doi: https://doi.org/10.1109/PST.2016.7906971

[20] Benjamin C. M. Fung, Ke Wang, Rui Chen, and Philip S. Yu. 2010. Privacy-Preserving Data Publishing: A Survey of Recent Developments. *Journal of ACM Computing Surveys (CSUR)*, vol. 42, iss. 4, article no. 14, June 2010, pp. 1-53. Doi: https://doi.org/10.1145/1749603.1749605

[21] Raymond Chi-Wing Wong and Ada Wai-Chee Fu. 2010. Privacy-Preserving Data Publishing: An Overview. *Synthesis Lectures on Data Management*, Morgan & Claypool Publishers 2010, 138 pages. Doi: https://doi.org/10.2200/S00237ED1V01Y201003DTM002

[22] Bee-Chung Chen, Daniel Kifer, Kristen LeFevre, and Ashwin Machanavajjhala. 2009. Privacy-Preserving Data Publishing. *Journal of Foundations and Trends in Databases*, vol. 2, no. 1-2, pp. 1-167. Doi: https://dx.doi.org/10.1561/1900000008

[23] Hessam Zakerzadeh, Charu C. Aggarwal, and Ken Barker. 2015. Privacy-Preserving Big Data Publishing. In: *Proceedings of the 27th International Conference on Scientific and Statistical Database Management (SSDBM 2015)*, no. 26, July 2015, La Jolla, CA, USA, pp. 1-11. Doi: https://doi.org/10.1145/2791347.2791380

[24] Tochukwu Iwuchukwu and Jeffrey F. Naughton. 2007. K-Anonymization as Spatial Indexing: Toward Scalable and Incremental Anonymization. In: *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB 2007)*, September 2007, Vienna, Austria, pp. 746-757. [Online]. Available: http://www.vldb.org/conf/2007/papers/research/p746-iwuchukwu.pdf (Accessed on December 10, 2021).

[25] Kristen LeFevre, David J. DeWitt and Raghu Ramakrishnan. 2008. Workload-Aware Anonymization Techniques for Large-Scale Datasets. *Journal of ACM Transactions on Database Systems (TODS)*, vol. 33, iss. 3, no. 17, August 2008, pp. 1-47. Doi: https://doi.org/10.1145/1386118.1386123

[26] Divyakant Agrawal, Amr El Abbadi, and Shiyuan Wang. 2013. Secure and Privacy-Preserving Database Services in the Cloud. In: *Proceedings of the 29th IEEE International Conference on Data Engineering (ICDE 2013)*, April 2013, Brisbane, Queensland, Australia, pp. 1268-1271. Doi: https://doi.org/10.1109/ICDE.2013.6544921

[27] Lukas Malina and Jan Hajny. 2013. Efficient Security Solution for Privacy-Preserving Cloud Services. In: *Proceedings of the 36th International Conference on Telecommunications and Signal Processing (TSP 2013)*, July 2013, Rome, Italy, pp. 23-27. Doi: https://doi.org/10.1109/TSP.2013.6613884

[28] Divyakant Agrawal, Amr El Abbadi, and Shiyuan Wang. 2012. Secure and Privacy-Preserving Data Services in the Cloud: A Data Centric View. In: *Proceedings of the VLDB Endowment (PVLDB 2012)*, vol. 5, no. 12. August 2012, pp. 2028-2029. [Online]. Available: http://vldb.org/pvldb/vol5/p2028_divyakantagrawal_vldb2012.pdf (Accessed on December 10, 2021).

[29] Ni Zhang and Chris Todd. 2009. Developing a Privacy Ontology for Privacy Control in Context-Aware Systems. CiteSeerX – Penn State University. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.148.9492&rep=rep1&type=pdf (Accessed on December 10, 2021).

[30] Narmeen Zakaria Bawany and Zubair A. Shaikh. 2017. Data Privacy Ontology for Ubiquitous Computing. *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 8, iss. 1, 2017. Doi: http://dx.doi.org/10.14569/IJACSA.2017.080120

[31] Nigamanth Sridhar and Wenbing Zhao. 2014. An Ontology for Enforcing Security and Privacy Policies on Mobile Devices. In: *Proceedings of the 6th International Conference on Knowledge Engineering and Ontology Development (KEOD-2014)*, vol. 2, no. 37, October 2014, Rome, Italy, pp. 288-295. ISBN: 978-989-758-049-9. Doi: https://doi.org/10.5220/0005081502880295

[32] Leonardo H. Iwaya, Fausto Giunchiglia, Leonardo A. Martucci, Alethia Hume, Simone Fischer-Hübner, and Ronald Chenu-Abente. 2015. Ontology-Based Obfuscation and Anonymisation for Privacy - A Case Study on Healthcare. Privacy and Identity Management. Time for a Revolution? *Privacy and Identity 2015.* IFIP Advances in Information and Communication Technology, vol. 476. Springer, Cham, pp. 343–358. Doi: https://doi.org/10.1007/978-3-319-41763-9_23

[33] Stanford Center for Biomedical Informatics Research: Protégé Semantic Web Ontology. [Online]. Available: http://protege.stanford.edu (Accessed on June 03, 2021).

[34] Ni Zhang and Chris Todd. 2006. A Privacy Agent in Context-Aware Ubiquitous Computing Environments. In: *Proceedings of the 10th IFIP TC-6 TC-11 International Conference on Communications and Multimedia Security (CMS 2006)*, October 2006, Heraklion, Crete, Greece, pp. 196-205. In: Leitold H., Markatos E.P. (Eds) Communications and Multimedia Security. Lecture Notes in Computer Science, vol. 4237. Springer, Berlin, Heidelberg. Doi: https://doi.org/10.1007/11909033_18

[35] Yuh-Jong Hu, Hong-Yi Guo, and Guang-De Lin. 2008. Semantic Enforcement of Privacy Protection Policies via the Combination of Ontologies and Rules. In: *Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC 2008)*, June 2008, Taichung, Taiwan, pp. 400-407. Doi: https://doi.org/10.1109/SUTC.2008.59

[36] Grit Denker and David L. Martin. 2005. Using Rules to Define the Semantics of Privacy Policies. *W3C Workshop on Rule Languages for Interoperability (W3C 2005)*, April 2005, Washington, DC, USA. [Online]. Available: https://www.w3.org/2004/12/rules-ws/paper/76 (Accessed on December 10, 2021).

[37] Adam Barth, Anupam Datta, John C. Mitchell, and Helen Nissenbaum. 2006. Privacy and Contextual Integrity: Framework and Applications. In: *Proceedings of the 2006 IEEE Symposium on Security and Privacy (S&P 2006)*, May 2006, Berkeley, California, USA, pp. 184-198. Doi: https://doi.org/10.1109/SP.2006.32

[38] Helen Nissenbaum. 2010. Privacy in Context - Technology, Policy, and the Integrity of Social Life. Stanford University Press 2010, ISBN: 978-0-8047-5237-4, pp. I-XIV, 1-288.

[39] Yann Krupa and Laurent Vercouter. 2010. Contextual Integrity and Privacy Enforcing Norms for Virtual Communities. In: *Proceedings of the Multi-Agent Logics, Languages, and Organisations Federated Workshops (MALLOW 2010)*, Workshop 1: COIN, September 2010, Lyon, France. [Online]. Available: http://ceur-ws.org/Vol-627/coin_10.pdf (Accessed on December 10, 2021).

[40] Nabil Ajam, Nora Cuppens-Boulahia, and Frédéric Cuppens. 2009. Contextual Privacy Management in Extended Role Based Access Control Model. In: *Proceedings of the 4th International Workshop on Data Privacy Management and Autonomous Spontaneous Security and Second International Workshop (DPM/SETOP 2009)*, September 2009, St. Malo, France, pp. 121-135. Lecture Notes in Computer Science, vol. 5939. Springer, Berlin, Heidelberg. Doi: https://doi.org/10.1007/978-3-642-11207-2_10

[41] Mohammad Jafari, Reihaneh Safavi-Naini, Philip W. L. Fong, and Ken Barker. 2014. A Framework for Expressing and Enforcing Purpose-Based Privacy Policies. *ACM Transactions on Information Systems Security*, vol. 17, iss. 1, no. 3, August 2014, pp. 1-31. Doi: https://doi.org/10.1145/2629689

[42] Mohammad Jafari, Philip W. L. Fong, Reihaneh Safavi-Naini, Ken Barker, and Nicholas Paul Sheppard. 2011. Towards Defining Semantic Foundations for Purpose-Based Privacy Policies. In: *Proceedings of the First ACM Conference on Data and Application Security and Privacy (CODASPY 2011)*, February 2011, San Antonio, TX, USA, pp. 213-224. Doi: https://doi.org/10.1145/1943513.1943541

[43] N. V. Narendra Kumar and R. K. Shyamasundar. 2014. Realizing Purpose-Based Privacy Policies Succinctly via Information-Flow Labels. In: *Proceedings of the 2014 IEEE Fourth International Conference on Big Data and Cloud Computing (BDCloud 2014)*, December 2014, Sydney, NSW, Australia, pp. 753-760. Doi: https://doi.org/10.1109/BDCloud.2014.89

[44] Yuan Tian, Biao Song, and Eui-nam Huh. 2009. A Purpose-Based Privacy-Aware System using Privacy Data Graph. In: *Proceedings of the 7th International Conference on Advances in Mobile Computing and Multimedia (MoMM 2009)*, December 2009, Kuala Lumpur, Malaysia, pp. 412-416. Doi: https://doi.org/10.1145/1821748.1821827

[45] Mohammad Jafari, Reihaneh Safavi-Naini, and Nicholas Paul Sheppard. 2009. Enforcing Purpose of Use via Workflows. In: *Proceedings of the 8th ACM Workshop on Privacy in the Electronic Society (WPES 2009)*, November 2009, Chicago, Illinois, USA, pp. 113-116. Doi: https://doi.org/10.1145/1655188.1655206

[46] Wynand van Staden and Martin S. Olivier. 2007. Using Purpose Lattices to Facilitate Customisation of Privacy Agreements. In: *Proceedings of the 4th International Conference on Trust, Privacy and Security in Digital Business (TrustBus 2007)*, September 2007, Regensburg, Germany, pp. 201-209. Lecture Notes in Computer Science, vol. 4657. Springer, Berlin, Heidelberg. Doi: https://doi.org/10.1007/978-3-540-74409-2_22

[47] Davide Bolchini, Qingfeng He, Annie I. Antón, and William H. Stufflebeam. 2004. 'I Need It Now': Improving Website Usability by Contextualizing Privacy Policies. In: *Proceedings of the 4th International Conference on Web Engineering (ICWE 2004)*, July 2004, Munich, Germany, pp. 31-44. Lecture Notes in Computer Science, vol. 3140. Springer, Berlin, Heidelberg. Doi: https://doi.org/10.1007/978-3-540-27834-4_5

[48] Paul Gerber, Melanie Volkamer, and Karen Renaud. 2015. Usability versus Privacy instead of Usable Privacy: Google's Balancing Act between Usability and Privacy. *Journal on ACM SIGCAS Computers and Society*, vol. 45, iss. 1, February 2015, pp. 16-21. Doi: https://doi.org/10.1145/2738210.2738214

[49] Colin Birge. 2009. Enhancing Research into Usable Privacy and Security. In: *Proceedings of the 27th Annual International Conference on Design of Communication (SIGDOC 2009)*, October 2009, Bloomington, Indiana, USA, pp. 221-226. Doi: doi.org/10.1145/1621995.1622039

[50] Carolyn Brodie, Clare-Marie Karat, John Karat, and Jinjuan Feng. 2005. Usable Security and Privacy: A Case Study of Developing Privacy Management Tools. In: *Proceedings of the 1st Symposium on Usable Privacy and Security (SOUPS 2005)*, July 2005, Pittsburgh, Pennsylvania, USA, pp. 35-43. Doi: https://doi.org/10.1145/1073001.1073005

[51] Satoshi Nakamoto. 2008. Bitcoin: A Peer-to-Peer Electronic Cash System. [Online]. Available: http://bitcoin.org/bitcoin.pdf (Accessed on December 10, 2021).

[52] Diego Romano and Giovanni Schmid. 2017. Beyond Bitcoin - Part I: A Critical Look at Blockchain-Based Systems. *Journal of IACR Cryptology ePrint Archive*, vol. 2015, September 22, 2017, pp. 1164. Doi: https://doi.org/10.3390/cryptography1020015

[53] Jan Hendrik Witte. 2016. The Blockchain: A Gentle Four Page Introduction. *Journal of Computing Research Repository (CoRR abs/1612.06244)*, December 2016. [Online]. Available: https://arxiv.org/pdf/1612.06244.pdf (Accessed on December 10, 2021).

[54] Cécile Pierrot and Benjamin Wesolowski. 2016. Malleability of the Blockchain's Entropy. *Journal of IACR Cryptology ePrint Archive*, vol. 2016, pp. 370, April 2016. [Online]. Available: https://eprint.iacr.org/2016/370 (Accessed on December 10, 2021).

[55] Guy Zyskind, Oz Nathan, and Alex Pentland. 2015. Decentralizing Privacy: Using Blockchain to Protect Personal Data. In: *Proceedings of the 2015 IEEE Symposium on Security and Privacy Workshops (SPW 2015)*, May 2015, San Jose, CA, USA, pp. 180-184. Doi: https://doi.org/10.1109/SPW.2015.27

[56] Idrissa Sarr, Hubert Naacke, and Ibrahima Gueye. 2015. Blockchain-Based Model for Social Transactions Processing. In: *Proceedings of 4th International Conference on Data Management Technologies and Applications (DATA 2015)*, July 2015, Colmar, Alsace, France, pp. 309-315. ISBN 978-989-758-103-8. Doi: https://doi.org/10.5220/0005519503090315

[57] Harry Halpin and Marta Piekarska. 2017. Introduction to Security and Privacy on the Blockchain. In: *Proceedings of the 2017 IEEE European Symposium on Security and Privacy Workshops (EuroS&P Workshops 2017)*, April 2017, Paris, France, pp. 1-3. Doi: https://doi.org/10.1109/EuroSPW.2017.43

[58] Hasventhran Baskaran, Salman Yussof, and Fiza Abdul Rahim. 2019. A Survey on Privacy Concerns in Blockchain Applications and Current Blockchain Solutions to Preserve Data Privacy. In: *Proceedings of the First International Conference on Advances in Cyber Security (ACeS 2019)*, July 30 - August 1, 2019, Penang, Malaysia, pp. 3-17.

[59] Leiyong Guo, Hui Xie, and Yu Li. 2020. Data Encryption Based Blockchain and Privacy Preserving Mechanisms Towards Big Data. *Journal of Visual Communication and Image Representation*, vol. 70, pp. 102741.

[60] Phan The Duy, Do Thi Thu Hien, and Van-Hau Pham. 2020. A Survey on Blockchain-based Applications for Reforming Data Protection, Privacy and Security. *Journal of Computing Research Repository (CoRR abs/2009.00530)*, September 2020.

[61] Haibo Tian, Jiejie He, and Yong Ding. 2019. Medical Data Management on Blockchain with Privacy. *Journal of Medical Systems*, vol. 43, iss. 2, pp. 26:1-26:6.

[62] Xudong Cao, Huifen Xu, Yuntao Ma, Bin Xu, and Jin Qi. 2019. Research on a Blockchain-Based Medical Data Management Model. In: *Proceedings of the 8th International Conference on Health Information Science (HIS 2019)*, October 18-20, 2019, Xi'an, China, pp. 35-44.

[63] Haiping Huang, Peng Zhu, Fu Xiao, Xiang Sun, and Qinglong Huang. 2020. A Blockchain-based Scheme for Privacy-Preserving and Secure Sharing of Medical Data. *Computers & Security*, vol. 99, pp. 102010.

[64] Abdullah Al Omar, Mohammad Shahriar Rahman, Anirban Basu, and Shinsaku Kiyomoto. 2017. MediBchain: A Blockchain Based Privacy Preserving Platform for Healthcare Data. In: *Proceedings of International Workshop on Security, Privacy, and Anonymity in Computation, Communication, and Storage (SpaCCS 2017).* In: Proceedings. Lecture Notes in Computer Science 10658, December 12-15, 2017, Guangzhou, China, pp. 534-543.

[65] Jingwei Liu, Xiaolu Li, Lin Ye, Hongli Zhang, Xiaojiang Du, Mohsen Guizani. 2018. BPDS: A Blockchain Based Privacy-Preserving Data Sharing for Electronic Medical Records. In: *Proceedings of the IEEE Global Communications Conference (GLOBECOM 2018)*, December 9-13, 2018, Abu Dhabi, United Arab Emirates, pp. 1-6.

[66] Federico Carlini, Roberto Carlini, Stefano Dalla Palma, Remo Pareschi, Federico Zappone, and Daniele Albanese. 2020. The Genesy Model for a Blockchain-based Fair Ecosystem of Genomic Data. In: *Proceedings of the 7th International Conference on Software Defined Systems (SDS 2020)*, April 20-23, 2020, Paris, France, pp. 183-189.

[67] Mahsa Shabani. 2019. Blockchain-Based Platforms for Genomic Data Sharing: A De-centralized Approach in Response to the Governance Problems? *Journal of the American Medical Informatics Association*, vol. 26, iss. 1, pp. 76-80.

[68] Shuaicheng Ma, Yang Cao, Li Xiong. 2019. Efficient Logging and Querying for Blockchain-Based Cross-Site Genomic Dataset Access Audit. *Computing Research Repository (CoRR abs/1907.07303)*, July 2019.

[69] Gaurav Kaul, Zeeshan Ali Shah, and Mohamed Abouelhoda. 2017. A High Performance Storage Appliance for Genomic Data. In: *Rojas I., Ortuño F. (eds) Bioinformatics and Biomedical Engineering (IWBBIO 2017)*. Lecture Notes in Computer Science, vol 10209. Springer, Cham. Doi: https://doi.org/10.1007/978-3-319-56154-7_43

[70] Medicalchain. 2018. Medicalchain Whitepaper 2.1. [Online]. Available: https://medicalchain.com/Medicalchain-Whitepaper-EN.pdf (Accessed on December 10, 2021).

[71] Gamze Gürsoy, Charlotte M Brannon, Sarah Wagner, and Mark Gerstein. 2020. Storing and Analyzing a Genome on a Blockchain. bioRxiv 2020.03.03.975334. [Online]. Available: Doi: https://doi.org/10.1101/2020.03.03.975334

[72] Mehmet Furkan Şahin, Oğuz Demir, Turan Kaan Elgin, Onur Uygur, and Gizem Çaylak. 2017. CrypDist. Github Repository. [Online]. Available: https://github.com/CrypDist. (Accessed on December 10, 2021).

[73] Halil Ibrahim Ozercan, Atalay Mert Ileri, Erman Ayday and Can Alkan. 2018. Realizing the Potential of Blockchain Technologies in Genomics. Genome Research 28 :1255–1263. [Online]. Available: https://doi.org/ 10.1101/gr.207464.116

[74] Zenome.io. 2017. Zenome Platform. Github Repository. [Online]. Available: https://github.com/zenomeplatform (Accessed on December 10, 2021).

[75] Zenome. 2017. Zenome Platform. [Online]. Available: https://zenome.io/about/ (Accessed on December 10, 2021).

[76] Dennis Grishin, Kamal Obbad, Preston Estep, Mirza Cifric, Yining Zhao, and George Church. 2018. Nebula Genomics: Blockchain-enabled Genomic Data Sharing and Analysis Platform. Harvard Molecular Technologies. [Online]. Available: http://arep.med.harvard.edu/pdf/Grishin_Church_v4.52_2018.pdf (Accessed on December 10, 2021).

[77] Cancer Gene Trust. 2018. Cancer Gene Trust: Decentralized Distributed Database of Genomic and Clinical Data. Github repository. [Online]. Available: https://github.com/cancergenetrust (Accessed on December 10, 2021).

[78] Gerome Miklau and Dan Suciu. 2003. Controlling Access to Published Data Using Cryptography. In: *Proceedings of 29th International Conference on Very Large Data Bases (VLDB 2003)*, September 2003, Berlin, Germany, pp. 898-909. [Online]. Available: http://www.vldb.org/conf/2003/papers/S27P01.pdf (Accessed on December 10, 2021).

[79] Bhushan Kapoor, Pramod Pandya, and Joseph S. Sherif. 2011. Cryptography: A Security Pillar of Privacy, Integrity and Authenticity of Data Communication. *Journal of Kybernetes*, vol. 40, iss. 9/10, pp. 1422-1439. Doi: https://doi.org/10.1108/03684921111169468

[80] Ariel Hamlin, Nabil Schear, Emily Shen, Mayank Varia, Sophia Yakoubov, and Arkady Yerukhimovich. 2016. Cryptography for Big Data Security. *Journal of IACR Cryptology ePrint Archive*, vol. 2016, pp. 12. [Online]. Available: https://eprint.iacr.org/2016/012.pdf (Accessed on December 10, 2021).

[81] C. Mohan. 2017. Blockchains and Databases. In: *Proceedings of the VLDB Endowment (PVLDB)*, vol. 10, no. 12, August 2017, pp. 2000-2001. [Online]. Available: http://www.vldb.org/pvldb/vol10/p2000-mohan.pdf (Accessed on December 10, 2021).

[82] Sebastien Meunier. 2016. Blockchain Technology - A Very Special Kind of Distributed Database, Medium Article, December 2016. [Online]. Available: https://medium.com/sbmeunier/blockchain-technology-a-very-special-kind-of-distributed-database-e63d00781118#.way70psdu (Accessed on December 10, 2021).

[83] Tien Tuan Anh Dinh, Ji Wang, Gang Chen, Rui Liu, Beng Chin Ooi, and Kian-Lee Tan. 2017. BLOCKBENCH: A Framework for Analyzing Private Blockchains. In: *Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD Conference 2017)*, May 2017, Chicago, IL, USA, pp. 1085-1100. Doi: https://doi.org/10.1145/3035918.3064033

[84] Tan Boon Seng. 2017. Blockchain - A Database with a Twist. (April 25, 2017). [Online]. Available: https://ssrn.com/abstract=2958565 or http://dx.doi.org/10.2139/ssrn.2958565 (Accessed on December 10, 2021).

[85] Elena Karafiloski and Anastas Mishev. 2017. Blockchain Solutions for Big Data Challenges: A Literature Review. In: *Proceedings of the 17th International Conference on Smart Technologies (EUROCON 2017)*, July 2017, Ohrid, Macedonia, pp. 763-768. Doi: https://doi.org/10.1109/EUROCON.2017.8011213

[86] Trent McConaghy, Rodolphe Marques, Andreas Muller, Dimitri De Jonghe, T. Troy McConaghy, Greg McMullen, Ryan Henderson, Sylvain Bellemare, and Alberto Granzotto. 2016. BigchainDB: A Scalable Blockchain Database. White paper. ascribe GmbH, Berlin, Germany, June 8, 2016. [Online]. Available: https://www.bigchaindb.com/whitepaper/bigchaindb-whitepaper.pdf (Accessed on December 10, 2021).

[87] Yang Li, Kai Zheng, Ying Yan, Qi Liu, and Xiaofang Zhou. 2017. EtherQL: A Query Layer for Blockchain System. In: *Proceedings of the 22nd International Conference on Database Systems for Advanced Applications (DASFAA 2017) Part II*, March 2017, Suzhou, China, pp. 556-567. Lecture Notes in Computer Science, vol. 10178. Springer, Cham. Doi: https://doi.org/10.1007/978-3-319-55699-4_34

[88] Yuqin Xu, Shangli Zhao, Lanju Kong, Yongqing Zheng, Shidong Zhang, and Qingzhong Li. 2017. ECBC: A High Performance Educational Certificate Blockchain with Efficient Query. In: *Proceedings of the 14th International Colloquium on Theoretical Aspects of Computing (ICTAC 2017)*, October 2017, Hanoi, Vietnam, pp. 288-304. Lecture Notes in Computer Science, vol. 10580. Springer, Cham. Doi: https://doi.org/10.1007/978-3-319-67729-3_17

[89] Tien Tuan Anh Dinh, Rui Liu, Meihui Zhang, Gang Chen, Beng Chin Ooi, and Ji Wang. 2017. Untangling Blockchain: A Data Processing View of Blockchain Systems. *Computing Research Repository (CoRR abs/1708.05665)*, August 2017. [Online]. Available: http://arxiv.org/abs/1708.05665. (Accessed on December 10, 2021).

[90] Stefan Tai, Jacob Eberhardt and Markus Klems. 2017. Not ACID, Not BASE, but SALT - A Transaction Processing Perspective on Blockchains. In: *Proceedings of the 7th International Conference on Cloud Computing and Services Science (CLOSER 2017)*, vol. 1, pp. 755-764. ISBN 978-989-758-243-1. Doi: https://doi.org/10.5220/0006408207550764

[91] Aggelos Kiayias and Giorgos Panagiotakos. 2015. Speed-Security Tradeoffs in Blockchain Protocols. *Journal of IACR Cryptology ePrint Archive*, vol. 2015, pp. 1019. [Online]. Available: https://eprint.iacr.org/2015/1019.pdf (Accessed on December 10, 2021).

[92] Zhijie Ren, Kelong Cong, Johan Pouwelse, and Zekeriya Erkin. 2017. Implicit Consensus: Blockchain with Unbounded Throughput. *Journal of Computing Research Repository (CoRR abs/1705.11046)*, May 2017. [Online]. Available: https://arxiv.org/pdf/1705.11046.pdf (Accessed on December 10, 2021).

[93] Yuqin Xu, Qingzhong Li, Xingpin Min, Lizhen Cui, Zongshui Xiao, and Lanju Kong. 2016. E-commerce Blockchain Consensus Mechanism for Supporting High-Throughput and Real-Time Transaction. In: *Proceedings of the 12th International Conference on Collaborate Computing: Networking, Applications and Worksharing (CollaborateCom 2016)*, November 2016, Beijing, China, pp. 490-496. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 201. Springer, Cham. Doi: https://doi.org/10.1007/978-3-319-59288-6_46

[94] Shui Yu. 2016. Big Privacy: Challenges and Opportunities of Privacy Study in the Age of Big Data. *Journal of IEEE Access*, vol. 4, June 2016, pp. 2751-2763. Doi: https://doi.org/10.1109/ACCESS.2016.2577036

[95] Ali Gholami and Erwin Laure. 2016. Security and Privacy of Sensitive Data in Cloud Computing: A Survey of Recent Developments. *Journal of CoRR abs/1601.01498*, January 2016. [Online]. Available: https://arxiv.org/ftp/arxiv/papers/1601/1601.01498.pdf. (Accessed on December 10, 2021).

[96] Javier Parra-Arnau, David Rebollo-Monedero and Jordi Forné. 2015. Privacy-Enhancing Technologies and Metrics in Personalized Information Systems. *Advanced Research in Data Privacy*, pp. 423-442. Studies in Computational Intelligence, vol. 567. Springer, Cham. Doi: https://doi.org/10.1007/978-3-319-09885-2_23

[97] Murat Kantarcioglu and Chris Clifton. 2004. Privacy-Preserving Distributed Mining of Association Rules on Horizontally Partitioned Data. *Journal of IEEE Transaction on Knowledge and Data Engineering*, vol. 16, no. 9, September 2004, pp. 1026-1037. Doi: https://doi.org/10.1109/TKDE.2004.45

[98] Jaideep Vaidya and Chris Clifton. 2002. Privacy Preserving Association Rule Mining in Vertically Partitioned Data. In: *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2002)*, July 2002, Edmonton, Alberta, Canada, pp. 639-644. Doi: https://doi.org/10.1145/775047.775142

[99] Madhuri N. Kumbhar and Reena Kharat. 2012. Privacy Preserving Mining of Association Rules on Horizontally and Vertically Partitioned Data: A Review Paper. In: *Proceedings of the 12th International Conference on Hybrid Intelligent Systems (HIS 2012)*, December 2012, Pune, India, pp. 231-235. Doi: https://doi.org/10.1109/HIS.2012.6421339

[100] Oracle. 2015. Oracle Database Security. White paper report. May 2015. [Online]. Available: https://www.oracle.com/database/security/index.html (Accessed on December 10, 2021).

[101] Elisa Bertino. 2015. Big Data - Security and Privacy. In: *Proceedings of the 2015 IEEE International Congress on Big Data (BigData Congress 2015)*, July 2015, New York City, NY, USA, pp. 757-761. Doi: https://doi.org/10.1109/BigDataCongress.2015.126

[102] Elisa Bertino and Elena Ferrari. 2018. Big Data Security and Privacy. A Comprehensive Guide Through the Italian Database Research Over the Last 25 Years. *Studies in Big Data*, vol. 31. Springer, Cham, pp. 425-439. Doi: https://doi.org/10.1007/978-3-319-61893-7_25

[103] Jawwad A. Shamsi and Muhammad Ali Khojaye. 2018. Understanding Privacy Violations in Big Data Systems. *Journal of IT Professional*, vol. 20, iss. 3, pp. 73-81.

[104] General Data Protection Regulation. 2016. Official Journal of the European Union. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679 (Accessed on December 10, 2021).

[105] PHP: Hypertext Preprocesser. [Online]. Available: https://www.php.net/. (Accessed on December 10, 2021).

[106] Apache HTTP Server Project. [Online]. Available: https://httpd.apache.org (Accessed on December 10, 2021).

[107] MySQL Database Management System. [Online]. Available: https://www.mysql.com/ (Accessed: December 10, 2021).

[108] MultiChain Blockchain Platform. [Online]. Available: https://www.multichain.com/ (Accessed on December 10, 2021).

[109] Bitcoin Blockchain Platform. 2021. [Online]. Available: https://bitcoin.org (Accessed on December 10, 2021).

[110] Gideon Greenspan. 2015. MultiChain Private Blockchain - White Paper. [Online]. Available: https://www.multichain.com/download/MultiChain-White-Paper.pdf (Accessed on December 10, 2021).

[111] Gideon Greenspan. 2016. Introducing MultiChain Streams. [Online]. Available: https://www.multichain.com/blog/2016/09/introducing-multichain-streams/ (Accessed on December 10, 2021).

[112] MultiChain Blockchain. Transaction Data Size Limit. [Online]. Available: https://www.multichain.com/qa/604/what-is-the-size-limit-data-that-can-be-stored-in-transaction?show=607#a607 (Accessed on December 10, 2021).

[113] MultiChain Blockchain. Data Stream Limit. [Online]. Available: https://www.multichain.com/qa/9976/limit-of-streams?show=9982#a9982 (Accessed on December 10, 2021).

[114] MultiChain Blockchain. Maximum Data Stream Block Transactions. [Online]. Available: https://www.multichain.com/qa/9879/how-many-maximum-transactions-are-there-in-block-multichain?show=9883#a9883 (Accessed on December 10, 2021).

[115] Markus Junker, Andreas Dengel, and Rainer Hoch. 1999. On the Evaluation of Document Analysis Components by Recall, Precision, and Accuracy. In: *Proceedings of the 5th International Conference on Document Analysis and Recognition (ICDAR 1999)*, September 20-22, 1999, Bangalore, India, pp. 713-716.

[116] Libsodium Documentation. [Online]. Available: https://libsodium.gitbook.io/doc/ (Accessed on December 10, 2021).

[117] PHP Encryption with Libsodium. [Online]. Available: https://www.zend.com/blog/libsodium-and-php-encrypt (Accessed on December 10, 2021).

[118] FortiClient: Fortinet Fabric Agent for Visibility, Control, and ZTNA. [Online]. Available: https://www.fortinet.com/products/endpoint-security/forticlient (Accessed on December 10, 2021).

[119] FortiWeb: Web Application Firewall (WAF) & API Protection. [Online]. Available: https://www.fortinet.com/products/web-application-firewall/fortiweb (Accessed on December 10, 2021).

[120] Privilege Access Management. [Online]. Available: https://docs.microsoft.com/en-us/microsoft-identity-manager/pam/privileged-identity-management-for-active-directory-domain-services (Accessed on December 10, 2021).

[121] Shrestha Ajay Kumar, Vassileva Julita, and Deters Ralph. 2020. A Blockchain Platform for User Data Sharing Ensuring User Control and Incentives. *Frontiers in Blockchain*, vol. 3, 22 October 2020, pp. 48. Doi: 10.3389/fbloc.2020.497985. [Online]. Available: https://www.frontiersin.org/article/10.3389/fbloc.2020.497985

[122] Hao Dai *et. al.* 2018. TrialChain: A Blockchain-Based Platform to Validate Data Integrity in Large, Biomedical Research Studies. *Journal of Computing Research Repository (CoRR abs/1807.03662)*, July 2018. [Online]. Available: https://arxiv.org/ftp/arxiv/papers/1807/1807.03662.pdf (Accessed on December 10, 2021).

[123] Shiraz Saptarshi Ghosh, Harshwardhan Parmar, Prasham Shah, and Krishna Samdani. 2018. A Comprehensive Analysis Between Popular Symmetric Encryption Algorithms. In: *Proceedings of the 1st International Conference on Data Science & Analytics (IEEE Punecon 2018)*, 30 November-2 December 2018, Pune, India, pp. 1-7. Doi: https://doi.org/10.1109/PUNECON.2018.8745324

[124] Sourabh Chandra, Smita Paira, Sk Safikul Alam, and Goutam Sanyal. 2014. A Comparative Survey of Symmetric and Asymmetric Key Cryptography. In: *Proceedings of the 2014 International Conference on Electronics, Communication and Computational Engineering (ICECCE 2014)*, 17-18 November 2014, Hosur, India, pp. 83-93. Doi: https://doi.org/10.1109/ICECCE.2014.7086640

[125] Sourabh Chandra, Siddhartha Bhattacharyya, Smita Paira, and Sk Safikul Alam. 2014. A Study and Analysis on Symmetric Cryptography. In: *Proceedings of the 2014 International Conference on Science Engineering and Management Research (ICSEMR 2014)*, 27-29 November 2014, Chennai, India, pp. 1-8. Doi: https://doi.org/10.1109/ICSEMR.2014.7043664

[126] O. Garcia-Morchon, S. Kumar. 2013. Security Consideration in the IP-based Internet of Things, CoRE, Internet-draft, 2013. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-garcia-core-security-06

[127] Kim Thuat Nguyena, Maryline Laurentb, Nouha Oualha. 2015. Survey on Secure Communication Protocols for the Internet of Things. *Ad Hoc Networks*, vol. 32, 2015, pp. 17-31. Doi: http://dx.doi.org/10.1016/j.adhoc.2015.01.006

[128] Quang Do, Ben Martini, and Kim-Kwang Raymond Choo. 2019. The Role of the Adversary Model in Applied Security Research. *Computers & Security*, vol. 81, 2019, pp. 156-181, ISSN 0167-4048, Doi: https://doi.org/10.1016/j.cose.2018.12.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404818306369 (Accessed on December 10, 2021)

[129] Federico Daidone, Barbara Carminati and Elena Ferrari. 2021. Blockchain-based Privacy Enforcement in the IoT domain. *IEEE Transactions on Dependable and Secure Computing*, pp. 1-1. Doi: https://doi.org/10.1109/TDSC.2021.3110181

[130] Maribel Fernandez, Jenjira Jaimunk and Bhavani Thuraisingham. 2019. Privacy-Preserving Architecture for Cloud-IoT Platforms. In: *Proceedings of the 2019 IEEE International Conference on Web Services (ICWS)*, 8-13 July 2019, Milan, Italy, pp. 11-19. Doi: https://doi.org/10.1109/ICWS.2019.00015

[131] Kristen N. Griggs, Olya Ossipova, Christopher P. Kohlios, Alessandro N. Baccarini, Emily A. Howson, and Thaier Hayajneh. 2018. Healthcare Blockchain System Using Smart Contracts for Secure Automated Remote Patient Monitoring. *Journal of Medical Systems*, vol. 42, no.7, pp. 130:1-130:7. Doi: https://doi.org/10.1007/s10916-018-0982-x

[132] Faiza Loukil, Chirine Ghedira Guegan, and Aïcha-Nabila Benharkat. 2020. PATRIoT: A Data Sharing Platform for IoT Using a Service-Oriented Approach Based on Blockchain. In: *Proceedings of the 18th International Conference on Service-Oriented Computing (ICSOC 2020)*, 14-17 December 2020, Dubai, United Arab Emirates, pp. 121-129.

[133] Ashutosh Dhar Dwivedi, Gautam Srivastava, Shalini Dhar, and Rajani Singh. 2019. A Decentralized Privacy-Preserving Healthcare Blockchain for IoT. *Sensors Journal*, vol 19, no. 2. pp. 326.

[134] Gokhan Sagirlar, Barbara Carminati, and Elena Ferrari. 2018. Decentralizing privacy enforcement for Internet of Things smart objects. *Computer Networks*, vol. 143, pp. 112-125.

[135] M. Alblooshi, K. Salah and Y. Alhammadi. 2018. Blockchain-based Ownership Management for Medical IoT (MIoT) Devices. In: *Proceedings of the 2018 International Conference on Innovations in Information Technology (IIT 2018)*, 18-19 November 2018, Al Ain, United Arab Emirates, pp. 151-156. Doi: https://doi.org/10.1109/INNOVATIONS.2018.8606032

[136] Yanqing Peng, Min Du, Feifei Li, Raymond Cheng, and Dawn Song. 2020. FalconDB: Blockchain-based Collaborative Database. In: *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD '20)*. Association for Computing Machinery, New York, NY, USA, pp. 637–652. Doi: https://doi.org/10.1145/3318464.3380594

[137] Asaph Azaria, Ariel Ekblaw, Thiago Vieira, and Andrew Lippman. 2016. MedRec: Using Blockchain for Medical Data Access and Permission Management. In: *Proceedings of the 2nd International Conference on Open and Big Data (OBD)*, 22-24 August 2016, Vienna, Austria, pp. 25-30, doi: https://doi.org/10.1109/OBD.2016.11

Chapter 7

Appendix: Program Code, Query Definition and Syntax

7.1 Inference Attacks

7.1.1 Embedded Query Statements

Privacy-Aware Query Syntax:

```
CALL tunote_ngs_genomics.GetPrivacyAwareUserData (4, 7, 6, 4, ’2021-12-31’, 1, 3, 2);
```

Native (Non-privacy) Query Syntax:

```
SELECT Date_collected, Specimen_type, Tissue_fluid_source, History_diagnosis
FROM tunote_ngs_genomics.requisition WHERE PatientId = 4 and RequisitionId = 5;
```

7.2 SQL Injection Query Attacks

PHP Web Server-side Code

```
<?php
// Load database connection parameters
require_once ”config/config-data.php”;
// Check for active login and ”last activity” session; if false, then redirect to login page
require_once ”config/login-session-activity.php”;
```

```
$param_patient_id = ””;

// Process form data when form is submitted
if ($_SERVER[”REQUEST_METHOD”] == ”POST”) {
    $param_patient_id = TRIM($_POST[”patient_id”]);
    $sql = ”SELECT personal_health_number, last_name, first_name, date_of_birth, gender FROM patient
    WHERE patient_id = ?”;
    If ($stmt = mysqli_prepare ($connection_data, $sql)) {
        mysqli_stmt_bind_param ($stmt, ”s”, $param_patient_id);
        mysqli_stmt_execute ($stmt);
    }
    // Close query statement
    mysqli_stmt_close($stmt);
}
?>
```

7.3 Evaluation Analysis Query Processing

7.3.1 Data Provider Demographic Information

Embedded Query Statements

Privacy-Aware Query Syntax:

```
CALL tunote_ngs_genomics.GetPrivacyAwareUserData (4, 3, 5, 4, ’2021-07-31’, 1, 3, 2);
```

Native (Non-privacy) Query Syntax:

```
SELECT Street, City, Province, Postal_Code, Origin_Province, Phone_Number
FROM tunote_ngs_genomics.patient WHERE idPatient = 4;
```

7.3.2 Data Provider Healthcare Information

Embedded Query Statements

Privacy-Aware Query Syntax:

```
CALL tunote_ngs_genomics.GetPrivacyAwareUserData (4, 6, 3, 3, ’2021-09-30’, 2, 3, 3);
```

Native (Non-privacy) Query Syntax:

SELECT Personal_Health_Number, Medical_Record_Number, Chart_Number, Personal_Care_Physician_Name, Dentist_Physician_Name FROM tunote_ngs_genomics.patient WHERE idPatient = 4;

7.3.3 Data Provider Consent Witness Information

Embedded Query Statements

Privacy-Aware Query Syntax:

CALL tunote_ngs_genomics.GetPrivacyAwareUserData (4, 1, 3, 3, ’2021-09-01’, 3, 2, 2);

Native (Non-privacy) Query Syntax:

SELECT Witness_Last_Name, Witness_First_Name, Witness_Phone_Number, Witness_Street, Witness_City, Witness_Province, Witness_Postal_Code FROM tunote_ngs_genomics.consent WHERE patient_id = 4;

7.3.4 Data Provider Genome Meta-data Information

Embedded Query Statements

Privacy-Aware Query Syntax:

CALL tunote_ngs_genomics.GetPrivacyAwareUserData (4, 8, 1, 4, ’2021-12-01’, 1, 2, 2);

Native (Non-privacy) Query Syntax:

SELECT VCF_Standard_Version, Variant_Caller_Name, Variant_Caller_Version, Variant_Call_Date, Filters_Applied, Variant_Caller_Regions, Annotation_Software_Name, Annotation_Software_Version, Annotation_Date, Annotation_Reference, Annotation_Reference_Version, Annotation_Fields FROM tunote_ngs_genomics.genome WHERE patient_id = 4 AND requisition_id = 8;