UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Computer Science Curriculum

Karl Hannes Veskus

Privacy-Preserving Data Synthesis Using Trusted Execution Environments

Master's Thesis (30 ECTS)

Supervisor(s): Liina Kamm, PhD
Sven Laur, PhD

Tartu 2022

Privacy-preserving data synthesis using trusted execution environments

Abstract:
Data synthesis is the process of generating new synthetic data from existing data. Often companies do not have the the in-house competence to synthesize data themselves, and are willing to outsource the process. However, synthesis requires access to the original data. Sharing data with a third party can be complex, especially so if it contains sensitive information or is considered as personal data by regulations such as the GDPR.

The goal of this thesis is to develop a proof-of-concept privacy-preserving data synthesis service showing that it is possible to use trusted execution environments to perform data synthesis in a privacy-preserving manner. Such a service would enable outsourcing the data synthesis process to an untrusted remote server by ensuring that both the original and synthesized data are fully hidden from the untrusted server host throughout the lifecycle of the service.

A prototype of the service was developed in the scope of an ongoing proof-of-concept project. To achieve the required security goals the service prototype uses trusted execution environment technologies, specifically the Sharemind HI development platform, which is in turn based on the Intel SGX platform. The developed service shows that synthesizing data in a privacy-preserving manner is indeed feasible if trusted execution environments are used. However, future work is needed to optimize the service to allow larger input and output files, and to support additional data synthesis methods.

Keywords:
Data synthesis, trusted execution environments, privacy-preserving technologies.

CERCS: P170 Computer science, numerical analysis, systems, control.

Privaatsust säilitav andmesüntees usaldatavas täitmiskeskkonnas

Lühikokkuvõte:
Andmete sünteesimine on olemasolevate andmete põhjal uute sünteetiliste andmete loomine. Paljudel organisatsioonidel ei ole kompetentsi ise andmeid sünteesida ning nad on valmis seda teenusena ostma. Andmesüntees vajab aga juurdepääsu algandmetele. Andmete jagamine kolmanda osapoolega võib olla raske, eriti kui tegu on isikuandmetega või muul viisil privaatsete andmetega.

Magistritöö eesmärk on luua privaatsust säilitav andmesünteesi teenuse protüüp, mis tõestab, et usaldatavaid täitmiskeskkondi on võimalik kasutada andmesünteesiks. Teenust, mis tagab, et nii algsed kui ka sünteesitud andmed on kogu teenuse elutsükli jooksul serveri haldaja eest täielikult peidetud, on võimalik juurutada ebausaldusväärsetesse serveritesse, kaasa arvatud pilvteenusetarnija poolt pakutavatesse serveritesse.

Magistritöö käigus valmis töötav teenuse protoüüp, mis turvanõuete tagamiseks kasutab Sharemind HI arendusplatvormi, mis omakorda põhineb Intel SGX usaldatava täitmiskeskkonna tehnoloogial. Valminud prototüüp on tõestuseks, et usaldatavaid täitmiskeskkondi on tõepoolest võimalik kasutada privaatsust-säilitava andmesünteesi teenuse loomiseks. Edasisteks ülesanneteks jääb teenuse jõudluse optimeerimine, mis võimaldaks kasutada suuremaid andmehulki sisendina ning ka sünteesisida rohkem andmeid, ja teenusele rohkemate sünteesimeetodite lisamine.

Võtmesõnad:
Andmesüntees, usaldatavad täitmiskeskkonnad, privatsust säilitavad tehnoloogiad.

CERCS: P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria).

Acknowledgments

The prototype, on which this thesis is based on, is developed in cooperation with Cybernetica AS[1], in the scope of an ongoing proof-of-concept (PoC) project called "Data protection Aware syNthesis of test databases using secure Computing tEchnology" or "DANCE". The project is funded by the PoC grant EAG189. The author of this thesis works in Cybernetica AS as a member of the team responsible for the "DANCE" project. Writing the thesis was funded by STACC OÜ.

The author would like to thank the project team in Cybernetica for providing valuable feedback and support throughout the project, as well as the supervisors, Liina Kamm and Sven Laur, for their constructive criticism and support in writing the thesis.

[1] https://cyber.ee/

Contents

1 Introduction **9**

2 Preliminaries **11**
2.1 Data protection 11
2.2 Information security 12
2.3 Privacy-enhancing technologies 13
2.4 Data synthesis 13
2.4.1 Synthesis methods 14
2.4.2 The Gaussian mixture modeling method 16
2.5 Intel® SGX 17
2.5.1 Enclaves 18
2.5.2 Attestation 21
2.5.3 Data sealing 24
2.5.4 Security 24
2.6 Sharemind HI 26
2.6.1 Access controls and dataflow configuration. 26
2.6.2 Architecture 28
2.6.3 Security 29

3 Privacy-preserving data synthesis service **31**
3.1 Use cases and motivation 31
3.2 Design of the service 33
3.2.1 Stakeholders and roles 33
3.2.2 Components 33
3.2.3 Access rights and the DFC 36
3.2.4 Service lifecycle 36
3.2.5 User interface 37

4 Service Architecture **41**
4.1 Server-side architecture 41
4.1.1 Data synthesis library 42
4.1.2 Model task enclave 42
4.1.3 Synthesis task enclave 43
4.1.4 Adding support for the Rust language 43
4.2 Client-side architecture 45
4.2.1 CSV processing library 45
4.2.2 User interface 45
4.2.3 Public metadata 46

5 Security of the service **48**

6 Service performance and benchmarks **51**

7 Conclusion **55**

References **56**

A BPMN **59**

Acronyms

API application programming interface

BPMN Business Process Model and Notation

CA certification authority

CCPA California Consumer Privacy Act

CPU central processing unit

CSV comma-separated values

DANCE Data protection Aware syNthesis of test databases using secure Computing tEchnology

DFC dataflow configuration file

DRAM dynamic random access memory

EM expectation-maximization algorithm

EPID Enhanced Privacy ID

FFI foreign function interface

GAN generative adversarial network

GDPR General Data Protection Regulation

GMM Gaussian mixture modeling

HIPAA Health Insurance Portability and Accountability Act

IAS Intel Attestation Service

LE Launch Enclave

MAC message authentication code

MEE Memory Encryption Engine

MPC secure multi-party computation

PET privacy-enhancing technology

PoC proof-of-concept

PRM Processor Reserved Memory

QE Quoting Enclave

SGX Software Guard Extentions

TEE trusted execution environment

TTP trusted third party

UI user interface

1 Introduction

Data synthesis is the process of generating new synthetic data from existing data. Recent communications by European data protection authorities indicate that synthetic data is not considered to be personal data in terms of the General Data Protection Regulation (GDPR) [1], even when generated from personal data. This would allow to use synthetic data instead of personal data for testing of information systems, accelerating the development of many data-driven services including artificial intelligence and machine learning. Synthesizing data can be a complex process, which many organisations are willing to buy as a service as they lack the competence to implement it themselves. While several commercial solutions for outsourcing data synthesis already exist on the market, all of them have to process the original data to do so, which needs a lawful basis and trust from the data owner. The requirements set by data protection regulations for sharing personal data to third parties, can often be a blocking factor when considering outsourcing. The question arises whether it is possible to use privacy-preserving technologies to perform data synthesis in a cloud environment in a way that protects the original data from the service provider. Such a service would enable outsourcing the synthesis process in compliance with privacy regulations with minimal effort from the data owner.

Trusted execution environments (TEE) are a class of privacy-enhancing technologies that enable running trusted code in untrusted computation environments. As such they are the perfect tool to implement a privacy sensitive service where the server host is not trusted. Furthermore, some implementations of TEEs, like the Intel SGX platform [2], provide means to hide data from the host machine altogether by encrypting the working memory and implementing access controls on the processor chip level.

The goal of the thesis is to verify whether it is possible to create a privacy-preserving data synthesis service by utilizing the security guarantees provided by trusted execution environments. This is done by developing a working prototype of the service, built on existing secure computing platforms. The technology used in the service prototype to ensure privacy of the original data is Sharemind HI. Sharemind HI is a commercial product offered by Cybernetica AS, that acts as a platform for developing privacy-preserving data analysis applications. Sharemind HI is in turn based on the Intel SGX trusted execution environment platform.

The service prototype, on which this thesis is based on, is developed as a team effort. The author of this thesis worked as a member of the development team, contributing in three general fields: developing missing features for the Sharemind HI platform, designing and implementing the user interface, and implementing the logic, in form of a web application, that joins all the various parts of the solution into a coherent working service. Specifically, the author's contributions were as follows:

- added a feature to Sharemind HI, that enables a user to add public metadata to uploaded files;

- implemented critical input-output functionality to a prototype feature of Sharemind HI, which enables writing solution-specific enclave code in the Rust programming language;
- implemented the solution-specific workflow that is run inside the enclaves;
- designed and implemented the web based user interface for the service prototype;
- implemented the functionality behind the user interface that joins various parts of the service together.

The thesis is separated into 7 sections, including the introduction and conclusion. Section 2 gives an overview of the preliminary knowledge needed to discuss the results and the technologies used in the service prototype. Sections 3, 4, and 5 describe the implementation of the service prototype. Specifically, Section 3 outlines the overall motivations and design of the service, Section 4 describes the architecture details, and Section 5 provides a brief security analysis. The thesis is concluded with benchmark results of the prototype in Section 6, which prove that synthesizing data in a privacy-preserving manner is indeed feasible.

2 Preliminaries

This section gives an overview of the general background knowledge needed to understand the results of thesis as well as more detailed explanations of the technologies used in the proof-of-concept (PoC) privacy-preserving data synthesis service developed as a part of this thesis. Section 2.1 goes over the current legal situation in the privacy landscape, providing motivations for protecting data at all. While most regions have their own specific regulations regarding data protection, the thesis focuses on the European Union and the General Data Protection Regulation (GDPR). Section 2.2 describes what it means for data to be protected and secure in a practical sense, and Section 2.3 provides a brief overview of the various technologies available for protecting data. Data synthesis and the specific method used in the service prototype are described in Section 2.4. The specific technologies used to implement the service prototype are described in more detail in Sections 2.5 and 2.6.

The readers of this thesis are expected to have a basic understanding of general computer science and software development topics. Additionally, the readers will benefit from having familiarity with cryptography and information security, and can refer to *Introduction to Modern Cryptography* by Katz and Lindell [3] or any other textbook on cryptography for further reading.

2.1 Data protection

On the 25th of May 2018, the General Data Protection Regulation (GDPR) [1] started to apply, bringing a significant change to the data protection landscape in the EU. The GDPR aims to protect the personal data of European citizens by regulating how their data can be used. While similar regulations also exist elsewhere in the world, for example the California Consumer Privacy Act (CCPA) [4] and Health Insurance Portability and Accountability Act (HIPAA) [5] in the US, this thesis will limit its scope to only GDPR.

Personal data is defined as any information relating to a data subject, that is an identified or identifiable natural person. Personal data can be anything from a person's medical records to their IP address. GDPR defines anonymous data as information which does not relate to an identified or identifiable natural person, or personal data that is rendered anonymous in such a manner that the data subject is not or is no longer identifiable. It is important to note that the requirements set by the GDPR only apply to personal data but not anonymous data [6].

GDPR introduces many principles that have to be followed in order to handle personal data, including having a lawful basis for processing, and ensuring the integrity and confidentiality of personal data during processing. These obligations cannot be easily generalized and have to be considered on a case-by-case basis. The level of controls needed to fully comply with the regulation depends on many aspects, including the type of data being processed, the reason for processing, and the method of processing

itself [1].

Article 24 of the regulation states: "... the controller shall implement appropriate technical and organisational measures to ensure and to be able to demonstrate that processing is performed in accordance with this Regulation." [1]. Furthermore, Article 25 requires that processing has to implement "data protection by design and by default" [1]. In essence, these and other Articles of the GDPR require that adequate protection measures be put in place from the design phase to ensure that personal data is sufficiently protected. Such protection measures may include both technical (e.g. encryption) and organisational means (e.g. data processing contracts between involved parties). Privacy-enhancing technologies (PETs) can also function as protection measures for the purposes of the GDPR.

Determining whether or not a given dataset contains identifiable data, and which are the appropriate measures to be implemented by design and by default, has been left open to interpretation by GDPR. Recital 26 of GDPR states: "To determine whether a natural person is identifiable, account should be taken of all the means reasonably likely to be used, such as singling out, either by the controller or by another person to identify the natural person directly or indirectly. To ascertain whether means are reasonably likely to be used to identify the natural person, account should be taken of all objective factors, such as the costs of and the amount of time required for identification, taking into consideration the available technology at the time of the processing and technological developments." Deciding which means are "reasonably likely to be used" has been left as the responsibility of the data controller, who has to consult and confirm their decision with the local data protection regulators. Disagreements between data controllers and regulators on the matter of identifiability and sufficient measures are common, and are solved by courts on a case-by-case basis.

2.2 Information security

In order to protect data from a would-be attacker, it is important to first have a formalization of what it means for data to be protected. A general notion of security is often used to talk about protecting data, however a rigorous definition is rarely given. Katz and Lindell [3] split the security definition into a security guarantee and a threat model. A security guarantee describes what attacks are prevented by a given system, while a threat model describes what the attacker is capable of doing.

For example a security guarantee of an encryption scheme could be that the resulting ciphertext does not leak any information about the original plaintext, that the attacker does not already have. The threat model of an encryption scheme is usually described in the form of a plausible attack vector. For example in a ciphertext-only attack the adversary is only capable of seeing the ciphertext(s) and has to extract information from it about the corresponding original plaintext(s), whereas in a chosen-ciphertext attack the adversary can first decrypt arbitrary ciphertexts of their choosing and has to then

extract information from a different ciphertext. Whereas in the first attack the adversary must initiate the extraction with no information other than the ciphertext, in the second attack the adversary can first study an arbitrary number of plain- and ciphertext pairs before having to extract any information, giving them significantly more information and making the attack more powerful.

In the more general setting of information systems, it is important to specify what pieces of data are protected, who are they protected from, and which measures are used to do so. The threat models are often use-case specific and come directly from the specific data and actors involved in the system, while the security guarantees are chosen and influence which measures can be used.

2.3 Privacy-enhancing technologies

Privacy-enhancing technologies (PETs) are any technological solutions that make the data less identifiable. The solutions can include any combination of anonymization techniques, encryption schemes, secure computing methods, or data synthesis. The common goal of PETs is to reduce the processing of personal data to a minimal level, while still retaining all functionality of a system [7].

The Privacy Preserving Techniques Task Team of the UN considers five major PETs in their handbook: secure multi-party computation (MPC), homomorphic encryption, trusted execution environments (TEEs), differential privacy, and zero knowledge proofs [8]. Each of the listed technologies are suitable for different forms of processing and offer different security guarantees. For example, while secure multi-party computation is considered to be information theoretically secure, its potential use-cases are limited by it requiring multiple non-colluding parties and incurring significant communication overhead and latency. Differential privacy on the other hand deals with output privacy, and hence cannot be used if the input needs to be private.

Among these five PETs, TEEs are mentioned to be the most scalable and performant. Furthermore, implementing a solution based on TEEs requires minimal organizational measures. The only requirement is having access to specialized hardware, such as Intel SGX-enabled processors.

2.4 Data synthesis

Data synthesis is the process of generating new realistic looking data, based on real data. Synthetic data is created from real datasets or some knowledge of the shape and structure of the real data, using statistical or machine learning methods. The main goal of data synthesis is for synthetic data to have all the same statistical properties as real data.

From one side, synthetic data does not include any real data and, as such, it cannot be considered personal data. Hence it is also not subject to the numerous legal frameworks pertaining to personal data, such as the General Data Protection Regulation (GDPR) [1],

the California Consumer Privacy Act (CCPA) [4], or the Health Insurance Portability and Accountability Act (HIPAA) [5], making sharing and using synthetic data significantly easier than using real data [9]. It is important to note however that synthetic data is not automatically anonymous simply by the merit of being synthetic data. In order to assert whether or not a specific synthetic dataset is considered to be personal data, a thorough analysis has to be done to confirm whether any single real person is identifiable from it.

From the analytics side, the recent work on machine learning and artificial intelligence systems have allowed to create increasingly better methods for generating synthetic data [9, 10], allowing more complex data models to be synthesized as well as increasing the utility of the synthesized data, meaning it emulates the real world more accurately and retains more of the underlying connections in the real data. This directly leads to a trade-off between the utility and privacy of synthetic data. Synthetic data with high utility is more similar to real data, and hence reveals more about the underlying personal data. Conversly, in order to best hide the personal data and provide the highest levels of privacy, the more generalized the synthetic dataset has to become, losing in utility.

In addition to sharing the data with external parties without being subject to privacy regulations, synthesized data can be used for a number of use-cases, each with their own requirements for data utility. For example stress-testing a system might not need the data to have any similarity to actual real world data, there only needs to be a lot of it, hence one could use synthetic data with low utility. On the other end of the utility spectrum, synthetic data with very high utility can be used to train machine learning models [10] where a large amount of data is needed, but cannot be used due to restricted access, or lack of resources or time [9].

As the data synthesis methods rely heavily on the structure of the real data, the common approach is for the data owner to either build the data synthesis models and generate the synthetic data themselves or pass the real data to a trusted third party (TTP), who then analyses, cleans, and processes the data to create a synthesis model. The first case requires the data owner to have the skillset needed to synthesize data, which severely limits the number of potential users of the approach. The second case would incur payments to the TTP for the services, and require a significant organisational overhead, such as contracts between the data owner and the TTP, analysing if the TTP is compliant with the relevant legal frameworks (e.g., GDPR), and notifying the data subjects of the new external data processors [9].

2.4.1 Synthesis methods

As for any subject related to data analysis, there are many different approaches that allow to achieve similar results. Which method of synthesis to use depends on the shape and the distribution of real data, what level of privacy and/or utility is required of the resulting synthesized data, and what levels of computational or organizational complexity are practically achievable. For the simplest use-cases with the need for the lowest data utility,

like software testing, simpler models can be used. The most basic method is to generate the synthesized data randomly based on a few simple rules. When slightly higher utility is needed, the data points can be sampled from distributions similar to the original data. This approach already needs analysing the data to determine which distributions best describe it or using prior knowledge of the field and subject matter to come up with rough estimates. For the highest utility requirements the most complex methods need to be used, from older methods like Gaussian mixture modeling (GMM) [11], to areas of current research [12] like generative adversarial networks (GANs) [13].

The goal of the proof-of-concept project is to only synthesize tabular data, and not image or sound. Initially, both the GMM and GAN methods were implemented for use in the PoC privacy-preserving data synthesis service, however exploratory tests on tabular datasets showed that the GMM method resulted in more accurate models than the GAN method. Furthermore, training the models took significantly less time using GMM. As such, it was decided to shift focus to only implementing the GMM method for the service prototype.

Since only the GMM method was fully integrated to the service, a brief overview of the method is provided. For further reading on the topic an interested reader can refer to the book *Finite Mixture Distributions* by Everitt and Hand [14].

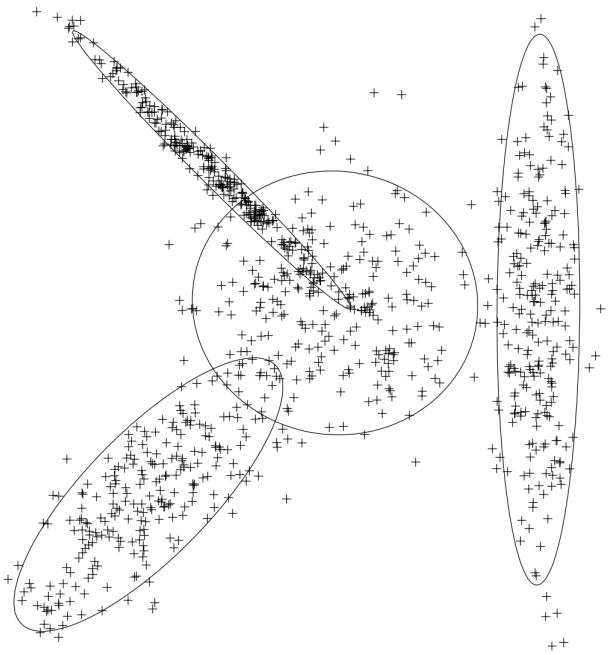

Figure 1. Example of a mixture of four Gaussian distributions fitted to two dimensional data [15].

2.4.2 The Gaussian mixture modeling method

Mixture models are a class of models in statistics, that describe data by combining multiple distributions, each of which estimate some subset of the data, into a single model. Gaussian mixture modeling (GMM) is a special case of mixture models, that only combines Gaussian distributions. A graphical representation of a GMM is shown on Figure 1. The following description of the GMM technique follows the textbook *Numerical Recipes* by Press *et al.* [15].

It is clear that if we have a mixture of distributions describing the original data, generating synthetic data from the mixture only requires sampling from it. Hence, the problem becomes finding the distributions, which, when combined, best describe the original data. This is the classic problem of unsupervised learning and is solved by maximizing the probability that the original dataset came from the mixture of distributions. The maximization is commonly done using the expectation-maximization algorithm (EM).

Let there exist a dataset $\mathbf{X}$ in the form of a $N \times M$ matrix and let the number K of Gaussian distributions used in the model be fixed. Fitting the model to the dataset is formalized by maximizing the likelihood, represented by Equation (1), where $p(\vec{\mathbf{x}}_n)$ is the probability of a given M-dimensional data point $\vec{\mathbf{x}}_n$ (a row in $\mathbf{X}$) originating from the mixture.

$$L(\mathbf{X}) = \prod_{n}^{N} p(\vec{\mathbf{x}}_n). \tag{1}$$

$$p(\vec{\mathbf{x}}_n) = \sum_{k}^{K} \mathcal{N}(\vec{\mathbf{x}}_n, \vec{\mu}_k, \Sigma_k) P(k) \tag{2}$$

$$\mathcal{N}(\vec{\mathbf{x}}_n, \vec{\mu}_k, \Sigma_k) = \frac{\exp\left(-\frac{1}{2}(\vec{\mathbf{x}}_n - \vec{\mu}_k)^T \Sigma_k^{-1} (\vec{\mathbf{x}}_n - \vec{\mu}_k)\right)}{\sqrt{(2\pi)^M |\Sigma_k|}} \tag{3}$$

The probability $p(\vec{\mathbf{x}}_n)$ in Equation (2), consists of the sum over K multivariate Gaussian density functions $\mathcal{N}(\vec{\mathbf{x}}_n, \vec{\mu}_k, \Sigma_k)$ and their corresponding mixture weights $P(k)$. Each mixture weight $P(k)$ is the fraction of rows from the dataset $\mathbf{X}$ present in the k-th distribution $\mathcal{N}(\vec{\mathbf{x}}_n, \vec{\mu}_k, \Sigma_k)$. The shape of the k-th distribution $\mathcal{N}(\vec{\mathbf{x}}_n, \vec{\mu}_k, \Sigma_k)$, as shown on Equation (3), is determined by the variables $\vec{\mu}_k$ and Σ_k, where $\vec{\mu}_k$ is the mean of the distribution, in the form of an M-dimensional vector, and Σ_k is the $M \times M$ covariance matrix for the distribution. These variables are the only ones that can be changed to maximize the likelihood L, as all other variables are fixed.

In order to estimate the variables $\vec{\mu}_k$ and Σ_k, that best fit the model to the data, the expectation-maximization algorithm (EM) is used. As the name suggests, the EM algorithm consists of an expectation step or E-step, and a maximization step or M-step. The E-step consists of calculating the probabilities p_{nk}, as shown on Equation (4), given some values of $\vec{\mu}_k$, Σ_k, and $P(k)$. The values p_{nk} represent the probability that a point

$\vec{\mathbf{x}}_n$ was sampled from the k'th distribution. These probabilities are used in the M-step, to calculate the next estimations for $\vec{\mu}_k$, Σ_k, and $P(k)$, as shown on Equations (5), (6), and (7), respectively.

$$p_{nk} = \frac{\mathcal{N}(\vec{\mathbf{x}}_n, \vec{\mu}_k, \Sigma_k)P(k)}{P(\vec{\mathbf{x}}_n)} \tag{4}$$

$$\mu_k = \frac{\sum_n^N p_{nk}\vec{\mathbf{x}}_n}{\sum_n^N p_{nk}} \tag{5}$$

$$\Sigma_k = \frac{\sum_n^N p_{nk}(\vec{\mathbf{x}}_n - \vec{\mu}_k) \otimes (\vec{\mathbf{x}}_n - \vec{\mu}_k)}{\sum_n^N p_{nk}} \tag{6}$$

$$P(k) = \frac{\sum_n^N p_{nk}}{N} \tag{7}$$

Thus the EM algorithm first needs an initial guess for the values of $\vec{\mu}_k$, Σ_k, and $P(k)$, and can then run the E-step to first obtain the probabilities p_{nk} and then the M-step to estimate the new values of $\vec{\mu}_k$, Σ_k, and $P(k)$. The steps can be repeated until the likelihood L converges under some set threshold, or until a desired number of repetitions is reached.

In practice the initial values for $\vec{\mu}_k$, Σ_k, and $P(k)$ can be fixed and only the dataset itself and the number of distributions used in mixture have to be provided by the user in order to fit the model. While the dataset is usually fixed, the number of distributions used can be changed and heavily influences the accuracy of the model. Using too few distributions will result in a overly general and inaccurate model, while using too many will quickly lead to overfitting.

Furthermore, it is apparent that GMM method can only handle datasets that are the shape of a two-dimensional matrix containing only numeric values, and as such it is important, from an implementation viewpoint, that the dataset does not contain any non-numeric values, like text fields. This is in general true for most machine learning methods and can be circumvented with preprocessing the data. Preprocessing replaces unusable data types, like text fields and null values, with meaningful numeric values, by using methods like one-hot encoding and imputation. Preprocessing can also involve transforming the data to improve the performance of the models, mostly through normalizing and standardizing the data. Which methods to use heavily depends on the model being trained and the data being used.

2.5 Intel® SGX

Intel's Software Guard Extentions (SGX) [2] is a collection of processor (CPU) instructions in Intel architectures that allows to create and manage secure containers, called

enclaves, in untrusted environments. The SGX instructions can provide protected memory areas controlled by hardware enforced access policies, isolating them from the rest of the environment, including the operating system hosting the enclave and any hardware peripherals.

To provide integrity and confirm that an enclave was correctly deployed, Intel provides an attestation mechanism. During the creation of an enclave, the trusted hardware generates a cryptographic report, which includes information about how the enclave was created, the properties it has, and the system it is running on. A remote party can then verify this report by using the Intel Attestation Service (IAS) [16].

The enclaves can also store data outside of the protected memory using data sealing. Data sealing is achieved by encrypting the data inside the enclave such that only that enclave on the current platform can ever decrypt the data. This is done by deriving a special encryption key from the immutable properties of the enclave and the underlying system [16].

All of the information regarding SGX in this section comes from Costan and Devadas' article *Intel SGX Explained* [17], unless explicitly cited otherwise. The article provides an excellent and thorough overview of the Intel SGX platform, as well as an extensive overview of background knowledge on computer architecture and security needed to understand the details. The reader is heavily encouraged to refer to the article for any additional details on the topics presented in this section.

2.5.1 Enclaves

The central component of SGX is the enclave. An enclave contains all of the code and data needed to perform security-sensitive computations. For example an enclave can contain the code for decrypting a database table, aggregating the table entries based on some complex relations and then encrypting it again.

Access protection. All code and data of the enclave is kept in the Processor Reserved Memory (PRM) section of dynamic random access memory (DRAM), which can only be accessed by the processor. Access to the PRM is restricted by two separate means, one for protecting against physical attacks and the other against software attacks. Unauthorized access through physical means, such as directly accessing the DRAM chips, is restricted by the Memory Encryption Engine (MEE)[2] [18] added to SGX-capable processors, which encrypts the content of the DRAM. Similarly, all communication outside the trusted processor, such as on the bus wires connecting the processor and DRAM chip, is kept encrypted. While an attacker can use mechanical means to read the physical bits on the DRAM chips or in the wires during communication, the encryption makes it impossible

for an attacker to interpret any of the acquired data.

Software access to the PRM is restricted by memory controllers integrated to the CPU. The controllers run memory access checks and forbid all non-enclave software from accessing the PRM. Hence the enclave can only be accessed through the limited set of instructions provided by the SGX-enabled CPU.

States. The SGX instruction set provides the means to create, manage, and destroy the enclaves. There are four major states in which the enclave can exist: non-existing, uninitialized, initialized, and running. Transitioning between the states happens only through the SGX instructions, notably the `ECREATE`, `EINIT`, `EENTER`, `EEXIT`, and `EREMOVE` instructions. The whole lifecycle of the enclave is illustrated on Figure 2.

During the creation of the enclave with the `ECREATE` instruction, the state moves from non-existing to uninitialized. The CPU allocates memory in the PRM, sets up and verifies all the needed background information about the enclave, like the enclave's size and location in memory, as well as creates a measurement of the enclave used in attestation. Before initializing the enclave, it is also possible to load code and data from the untrusted environment into the enclave with the `EADD` and `EEXTEND` instructions, the latter of which also updates the enclave's measurement.

To initialize the enclave with `EINIT`, a special pre-made Launch Enclave (LE) has to first be used to create an initialization token. The initialization token is used in the initialization process to derive a launch key and move the enclave to the initialized state.

In essence, the LE mainly exists as a licensing mechanism, as it is provided by Intel, signed with a special hard-coded Intel key, and checks if the enclave creators are licensees[3]. This mechanism ensures that only the enclaves, which are created by developers that have been verified by Intel, can be initialized, minimizing the risk of running an enclave, which containing malware.

From the initialized state it is possible for the host to run the enclave code using the `EENTER` instruction, moving the enclave to the running state. Similarly, when the enclave has finished, it will return control to the host using the `EEXIT` instruction, moving the enclave back to the initialized state.

Finally the enclave can be moved from the initialized state back to an non-existing state with the `EREMOVE` instruction. This first makes sure that the enclave's code is not running and then frees up memory resources, completely destroying the enclave.

Identity. For a user to be sure that it is communicating with or running a specific enclave, there has to first exist a mechanism for distinguishing the enclaves. This directly

[2]In newer processor models, starting from the 3rd Generation Intel Xeon Scalable Processors [19], the MEE has been replaced by the Total Memory Encryption technology [20].

[3]With certain newer hardware combinations it is also possible for enclave developers to create and sign the launch enclave themselves [21].

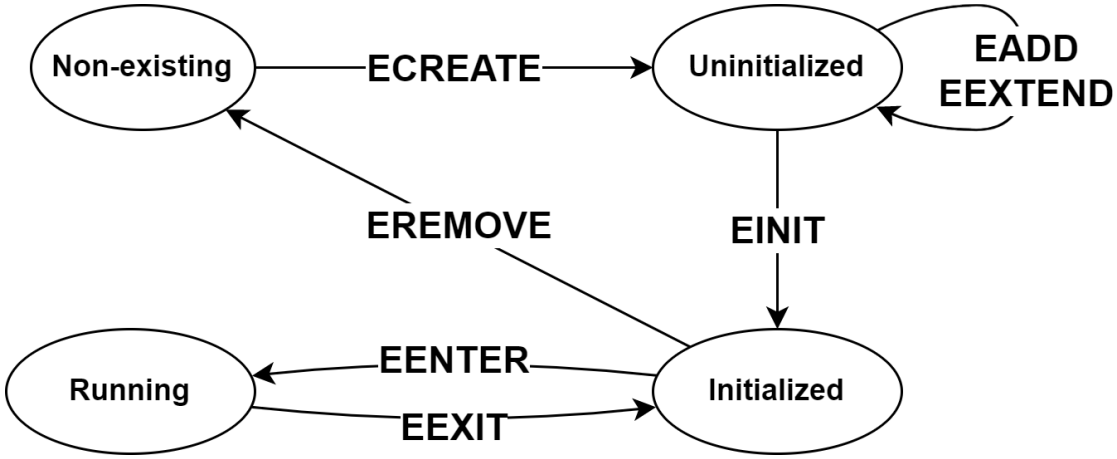

Figure 2. Lifecycle of an SGX enclave.

leads to a requirement of there existing some identifying factor for each enclave. In order to distinguish between enclaves, perform attestation, and derive cryptographic keys, the trusted hardware computes a measurement of each enclave's content. As the enclave is created from some known and specific series of inputs to the creation instructions `ECREATE`, `EADD`, and `EEXTEND`, which cannot be changed after initializing the enclave, the measurement of the enclave is defined to be the 256-bit SHA-2 hash over those inputs. The `EINIT` instruction completes the hashing process and seals the hash, so tt could no longer be updated by the `ECREATE`, `EADD`, or `EEXTEND` instructions. The completed hash is set as the measurement of the enclave, and denoted as `MRENCLAVE`.

It is important to note, however, that the measurements of two instances of the same enclave will be identical and, as such, a second mechanism is needed to distinguish between two enclaves that are running the same code on the same machine. For this the enclaves require a certificate issued by the enclave's author. The certificate is of a specific format called a Signature Structure (`SIGSTRUCT`) and contains metadata about the enclave, namely a hash of the enclave author's public key, called `MRSIGNER`, the enclave's measurement and its version number. The enclave author's private key is used to sign the `SIGSTRUCT`. To prevent the creation of uncertified enclaves, the `EINIT` instruction checks the validity of the certificate and fails to create the enclave when the certificate is invalid [16].

Key derivation. The SGX instruction set provides the `EGETKEY` instruction for deriving symmetric keys that can be used by the enclaves. The instruction can be configured to generate keys of different types, each intended for use in a different process and using different derivation materials. For example, there exists a key type used for encryption in the data sealing process and a key type for computing a message authentication code (MAC) [3] in the attestation process. In general, the derivation material uses secrets

embedded in the trusted hardware, and hence the symmetric keys can only be derived by the trusted SGX-enabled processor chip. Furthermore, the derivation material includes information about both the untrusted system and the enclave it will be used by, ensuring that the key is only valid for a specific enclave on the given system. However, it is also possible to configure the `EGETKEY` instruction to use the hash of the enclave author's public key, `MRSIGNER`, instead of the measurement of the enclave, `MRENCLAVE`, as a part of the derivation material. This allows all enclaves on the system, that are created by the same author, to derive a shared key, allowing, for example, to easily share secrets between different enclaves running on the same machine.

2.5.2 Attestation

As the main problem that SGX is attempting to solve, is running trusted code on a remote untrusted host machine, it needs a mechanism for verifying that the code and data have not been modified. The problem is solved by using software attestation, a cryptographic protocol for proving that an enclave with the specified measurements was created and is running on trusted hardware. Additionally, the attestation process allows the enclave and the remote trusted party to produce a shared key, which can be later used to create a secure communication channel between the two, allowing a remote party to transport secrets to and from the enclave.

The trust chain of the software attestation mechanism is described by Figure 3. The figure indicates that a verifier only needs to trust Intel in order to trust the attestation process as a whole, as the whole process relies on Intel's root key, which is embedded in the trusted hardware.

Intel provides two separate attestation methods, one for performing attestation between two enclaves running on the same system, and one for performing attestation between a user and an enclave initialized on a remote untrusted host machine.

Local attestation. To prove its identity to a target enclave, the enclave being attested uses the `EREPORT` instruction, which prompts the trusted CPU to generate an attestation report. The report consists of the enclave's measurement and certificate-based identity, a message provided by the enclave, and a message authentication code (MAC) [3] tag over the contents of the report. The reported MAC tag is computed by the trusted SGX processor using a special report key. The report key is derived using the measurement of the target enclave and a secret embedded in the processor chip. The target enclave can use the `EGETKEY` instruction to derive the same key and use it to compute the MAC tag over the report contents. If the computed MAC tag and the reported MAC tag match, the target enclave can be sure that the enclave being attested is legitimate and can be trusted. The report key can not be derived by anyone except the target enclave and the trusted CPU, as they would not have access to the secret embedded in the processor, ensuring the correctness of the local attestation process.

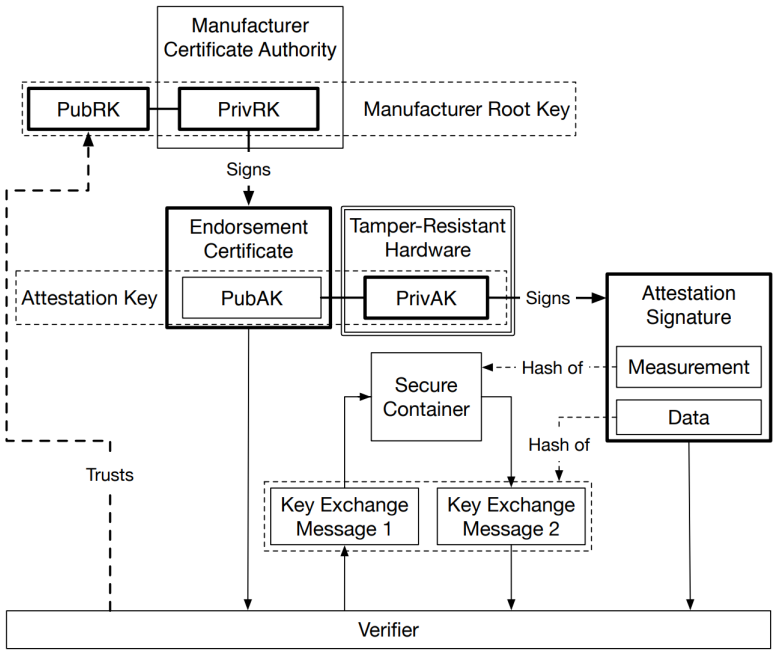

Figure 3. The trust chain of attestation [17].

Remote attestation. Remote attestation is used to verify to a remote client that the enclave is running trusted code, and to establish an authenticated communication channel between the enclave and the client. A few additional components are introduced for the process.

First a new Intel provided enclave, the Quoting Enclave (QE), is used to verify the local attestation reports coming from other enclaves, convert them into a quote and sign it using a special device-specific key, the Intel Enhanced Privacy ID (EPID) key. Secondly, the EPID signature over the quote can be verified by the Intel Attestation Service[4](IAS) [16].

EPID is a group signature scheme [16] where each signer, with their own private key, is assigned into a group based on their processor type [22]. The verifier has a single corresponding public key for the entire group, providing privacy to the signers, while still enabling cryptographic signing.

The remote attestation process consists of 7 distinct steps involving the enclave being attested, the Quoting Enclave (QE), the untrusted application orchestrating the communication, the Intel Attestation Service (IAS), and the relying party or challenger. The process is illustrated on Figure 4. The process can also be thought of as a slightly modified sigma protocol [23] between the application as the prover and the challenger

[4]It is also possible for an enclave developer to verify the quotes themselves by implementing and deploying their own attestation service [21].

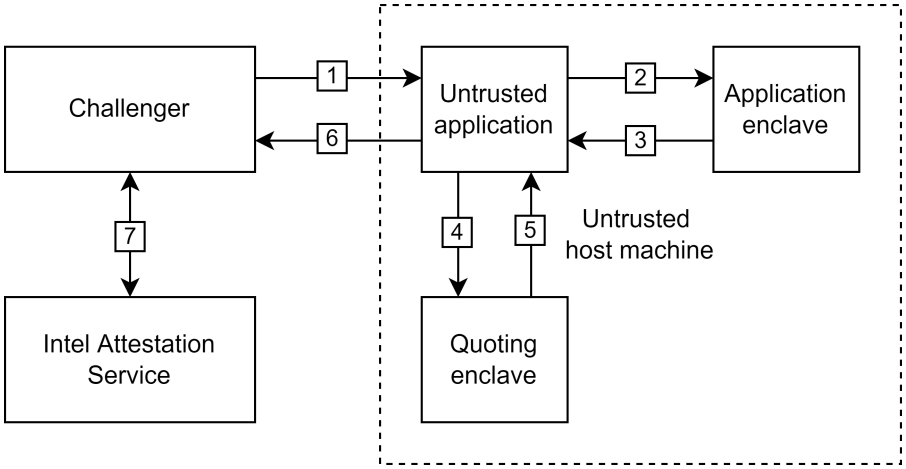

Figure 4. Example of the remote attestation process.

as the verifier, where only the commitment step has been changed from a commitment message to initializing the enclave on SGX-enabled hardware.

1. The attestation process is initiated by requesting to create an authenticated channel between the enclave and the challenger. The challenger sends the first protocol message, consisting of a challenge and a nonce.

2. The application, running in the untrusted operating system, requests an attestation report from the enclave and passes on the challenger's nonce.

3. The enclave generates the report, using the `EREPORT` instruction, and a manifest, consisting of the user data section from the report and optionally the nonce along with a public key for the challenger to create the authenticated channel. Both the report and manifest are returned to the untrusted application.

4. The application forwards the report to the QE, which verifies the report using local attestation, converts it into a quote, and signs it with the EPID key.

5. The QE returns the signed quote to the application.

6. The application returns the quote from QE and the manifest from the enclave to the challenger.

7. The challenger sends the quote to IAS, who verifies the EPID signature over the quote and returns the result of the verification.

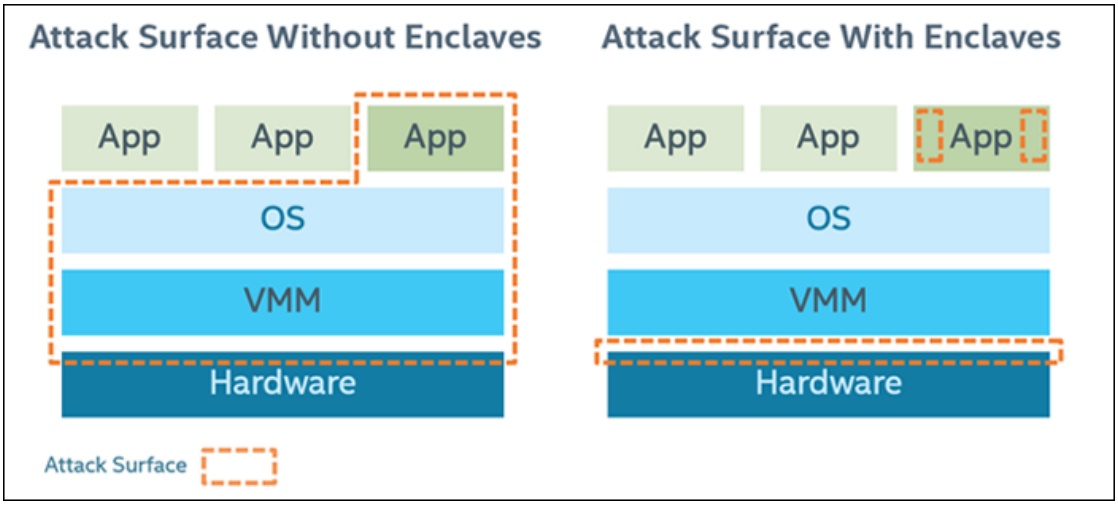

Figure 5. Attack surface minimisation [22].

2.5.3 Data sealing

In order to preserve data between multiple lifetimes of a single enclave or to share data between multiple enclaves on the same system, a data sealing procedure is used. In essence, data sealing is storing encrypted data on the file system of the untrusted machine. The key used to encrypt the data before storing depends on who needs to access the data later, and can be configured by the `EGETKEY` instruction. For example, it is possible to seal data based on the measurement of an enclave, so that only the current and future versions of that enclave, running on the same machine, can decrypt the data, or based on the signature of the enclave author, so that all enclaves of the author on the same platform can decrypt the data.

It is important to note that sealing data using the latter method goes somewhat against the goal of isolating the enclaves and minimizing the attack surface. If any of the enclaves that can access the shared data are compromised by a malicious party, all of the shared data is leaked. However, the compromised enclave is still isolated and will not gain access to any other data, limiting the damage to only the shared sections.

2.5.4 Security

The strongest security guarantees offered by SGX come from minimizing the attack surface in an application, as illustrated by Figure 5. When implementing a security sensitive application without using any PETs, a separate application running on the same machine may be compromised and give an attacker operating system level access to the machine, allowing them to attack the sensitive application with ease. In contrast, isolating any security-sensitive parts of an application into enclaves circumvents the

operating system entirely, hence the only possible attacks involve either the processor chip directly or the boundary points between the enclave and the application.

While securing the boundary points on the application side is the responsibility of the application developer, everything involving the hardware and the SGX instruction set is handled by Intel. As the specifics of Intel's processor chips and the implementation of SGX instructions are not publicly disclosed, a full comprehensive analysis of all possible vulnerabilities cannot be done outside of Intel. While significant efforts have been made in reverse engineering the chips and instructions, for example by Costan and Devadas [17], any security guarantees ultimately still rely on trusting Intel and assuming there are no backdoors or hidden critical flaws [24].

An enclave is fully isolated from other hardware by means of both physical and cryptographic protections, namely by using the Memory Encryption Engine (MEE), removing the need to trust any peripherals except for the Intel CPU. Similarly, access checks done by the CPU forbid unauthorized software access to the enclave, including access by the operating system and other priviledge software. This isolation is exactly what enables SGX to shorten the trust chain to just Intel, while normal applications need to trust all of the hardware and software present in the computer.

Even when assuming that Intel is trustworthy, the reverse engineering done by researchers has revealed several vulnerabilities. Such vulnerabilities are, however, expected from a system this large and are largely mitigated by continued updates from Intel. While most of the more serious vulnerabilities have been patched by Intel soon after they have been discovered, the complexity and lack of public documentation regarding SGX give reason to suspect other attacks still exist and are not yet mitigated.

The threat model of SGX aims to protect the data inside the enclave from a malicious host system. Combining the Memory Encryption Engine (MEE), hardware level access checks, and attestation, the Intel SGX platform manages to provide confidentiality, integrity, and freshness guarantees for the data inside the enclaves against other software, like the operating system, running on the host machine. However, while protections against direct attacks, using either physical or software based approaches, towards the enclaves are sufficient, the design of SGX is not intended to protect against side-channel attacks or attacks targeting the processor chip. As such most of the vulnerabilities that have been found utilize some forms of side-channel attacks. While it is close to impossible to eliminate all possible side-channel attacks, there are methods to minimize and mitigate such vulnerabilities, like keeping the SGX deployment up to date, enforcing additional access controls, and following strict secure programming practices. Furthermore, most vulnerabilities have only been utilized by proof-of-concept attacks carried out in controlled laboratory settings. As no such attacks have been noted in practical settings, there is hope that exploiting these vulnerabilities requires very specific conditions and specialized equipment, making them unviable in practice. Some examples of proof-of-concept attacks, which have utilized the side-channel vulnerabilities, are

the Nemesis attack [25], which leverages timing side-channels and relies on being able to externally interrupt the code running in the enclave, and the Plundervolt attack [26], which leverages voltage based side-channels. A thorough overview of the vulnerabilities in SGX applications and potential ways to mitigate them is provided in a research report by Randmets [24].

2.6 Sharemind HI

Sharemind HI is a platform for developing privacy-preserving data analysis applications, where confidential data is protected throughout its lifecycle in the service. Data is encrypted by the data owner prior to sending it to the Sharemind HI service and will remain protected through cryptographic means throughout the analysis. The host of the service will only ever have access to encrypted data and no means to remove the protections applied by Sharemind HI. The security guarantees rely on a trusted execution environment (TEE) technology, that provides secure containers for the confidential parts of the application, isolating them from the rest of the untrusted environment using trusted hardware. Sharemind HI uses Intel® Software Guard Extentions (SGX) as its TEE technology to implement the privacy-preserving data processing [27].

Sharemind HI combines the hardware-based security guarantees given by SGX with additional organizational measures and access controls. The goal is to simplify the implementation and deployment of the solution-specific logic involving multiple stakeholders, while retaining the security and privacy guarantees. It achieves this by abstracting away key management and cryptographic measures, leaving the developer to implement the business logic and access controls.

Sharemind HI operates as a client-server service, based on tasks running as SGX enclaves. The client is a user-side application tasked with calling operations on the server, encrypting data, and performing remote attestation. The server acts as a host for the SGX-enabled processor and is tasked with managing the authentication and authorization of users, managing and storing encryption keys of the data in a secure manner, orchestrating the internal and external data transfers, scheduling the running of the tasks, and keeping a log of all operations performed on the server [28].

2.6.1 Access controls and dataflow configuration.

A solution deployed on Sharemind HI contains multiple interconnected components and parties with different rights and responsibilities. For computations with confidential data, there can exist a number of tasks with different inputs and outputs. The storing and grouping of data with similar access requirements is handled by data topics. The topics can also be thought of as protection domains. The stakeholders involved in the solution can be assigned various roles, each with separate responsibilities and access rights. All rules regarding the tasks, topics, access rights, and stakeholder roles are described in a

dataflow configuration file (DFC). The DFC contains a list of all task enclaves involved in the solution along with their measurements, the certificates of all the stakeholders along with their roles, and all topics along with a list of users and tasks that are allowed to input or output data from a topic [28].

There are three main interactive roles for a stakeholder [28].

- The input provider who can upload data into topics.
- The output consumer who can download data from topics.
- The runner who can trigger the task enclave code to be run.

Additionally, there are three roles for providing increased security.

- The auditor who can have access to the audit logs and is responsible for validating critical code components before deployment, issuing the application fingerprint, and performing system audits to check if the deployment and operation conform to the specification.
- The coordinator who is responsible for coordinating any activities related to setup and deployment.
- The enforcer who can approve the DFC and is responsible for checking that the specified DFC holds the security objectives that the parties want to achieve. An agreement from all the enforcers is required before any data can be collected or any task enclaves can be run.

Figure 6 illustrates the information flow defined by a possible DFC. The figure includes two distinct sequential tasks, a number of input providers, and two output consumers, one of whom simultaneously has a runner role. The input providers encrypt their confidential data and upload it to Topic A. The runner can start the analysis by triggering the code in Task A to run. When run, Task A may download and decrypt any data uploaded to Topic A, run computations on the decrypted data inside the enclave to create some output, encrypt the output and upload it to Topic B. Similarly, the runner can start Task B, which in turn may download and decrypt any data in Topic B and upload its encrypted output to Topic C. Both consumers can at any point download and decrypt any data in Topic C. Note that the confidential data is encrypted by the input providers prior to uploading and is only ever decrypted inside the task enclaves while performing the analysis. The outputs of the tasks are similarly encrypted inside the enclaves and can only be decrypted by the intended recipients. This means all data is encrypted while in transit or at rest, and only usable by the original data owners or inside the SGX enclaves. As none of the stakeholders can access the data in Topics A and B, they never see the original confidential data uploaded by the providers, nor the intermediate results output by Task A.

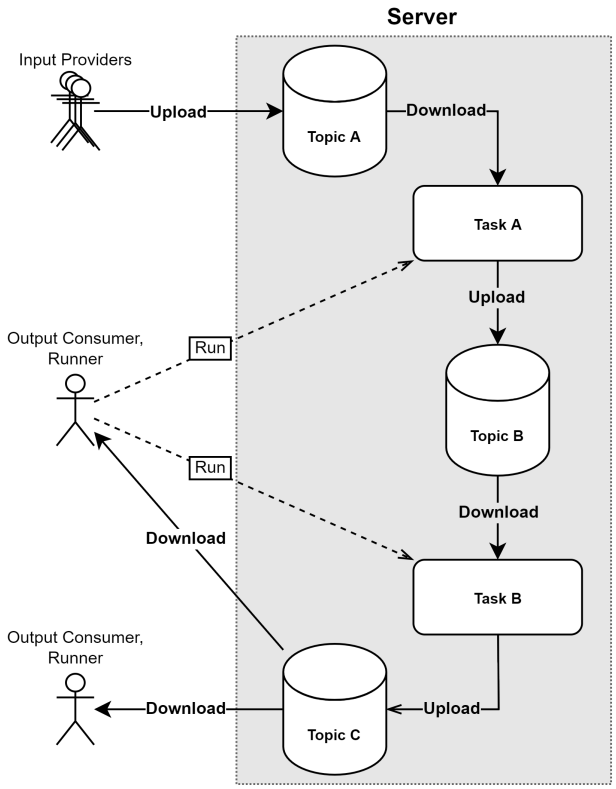

Figure 6. A dataflow configuration graph example.

2.6.2 Architecture

Sharemind HI consists of two distinct components, the client and the server. The client includes application specific code and a general purpose Sharemind HI client library, used to interface with the Sharemind HI server, encrypt data, and perform remote attestation. The server is split into trusted and untrusted components. The untrusted components contain functionality for coordinating work, network communication, file system interaction, and creating, running and delivering messages to the enclaves in the trusted part of the server. As the name implies, the untrusted components do not run inside SGX and therefore cannot be allowed to access any confidential information [28].

The trusted components are the solution-specific task enclaves and three management enclaves: the attestation enclave, the key enclave, and the core enclave. The management enclaves provide all functionality for secure communication and task execution. They are split into distinct enclaves in order to isolate the functionality into easily auditable pieces with the smallest possible attack surface. The attestation enclave is responsible for remote attestation and setting up secure communication channels between the client and the other enclaves. The only purpose of the key enclave is to store and manage access to

the keys required to use any confidential data. The core enclave handles coordinating data storage and retrieval, stores and manages the solution state, and creates the audit log. While the core enclave is in the trusted section of the server, it does not have access to any confidential data, except for the secure communication channel secrets. The task enclaves implement any and all solution-specific code that actually processes the confidential user data. They acquire input data from the core enclave and the respective keys from the key enclave [28].

2.6.3 Security

Sharemind HI relies and builds on the security guarantees of SGX, discussed in Section 2.5.4. While on a low-level the attack model of Sharemind HI is identical to that of Intel SGX, it introduces additional technical and organisational measures to reduce the risk of any high-level attacks caused by incorrect key management, flawed access controls, or mistakes in the programming or deployment of SGX-based applications. A thorough attack model, listing all considered threats and corresponding control measures, is provided in the Sharemind HI white paper [27]. The following gives a brief overview of the information from the white paper.

The security goal of Sharemind HI is that any confidential data uploaded to the Sharemind HI platform should only be accessed by those stakeholders and task enclaves that have the necessary permissions. Any other parties, like the server host or malware present in the system, should not be able to access the data in a unencrypted format. The security model of Sharemind HI focuses on attacks against confidentiality and integrity. Attacks against availability are considered to be the responsibility of stakeholders and out-of-scope for Sharemind HI. An illustration of the security model of Sharemind HI is given on Figure 7.

Input data is encrypted by the data owner at their premises using the Sharemind HI client. The encrypted data is sent to the Sharemind HI server, where it is stored in a topic, while the encryption keys are sent to the key enclave using secure authenticated channels. Similarly, output data is encrypted inside of a task enclave and stored on the server in a (possibly different) topic. When a stakeholder requests to download the output data, their authorization to access the data is confirmed, and the key enclave securely transfers the encryption keys to the authorized party.

Enforcers are required to verify that a task enclave is configured and deployed as agreed upon by the stakeholders. Each input provider and output consumer can select a subset (including the empty or full set) of enforcers they trust. Access control measures implemented in Sharemind HI ensure that the input providers can only upload data to and output consumers can download data from tasks which have been approved by their trusted enforcers. This trust chain ensures that any data uploaded to Sharemind HI is not made available to undesired task enclaves, as well as that the data is only received from approved task enclaves.

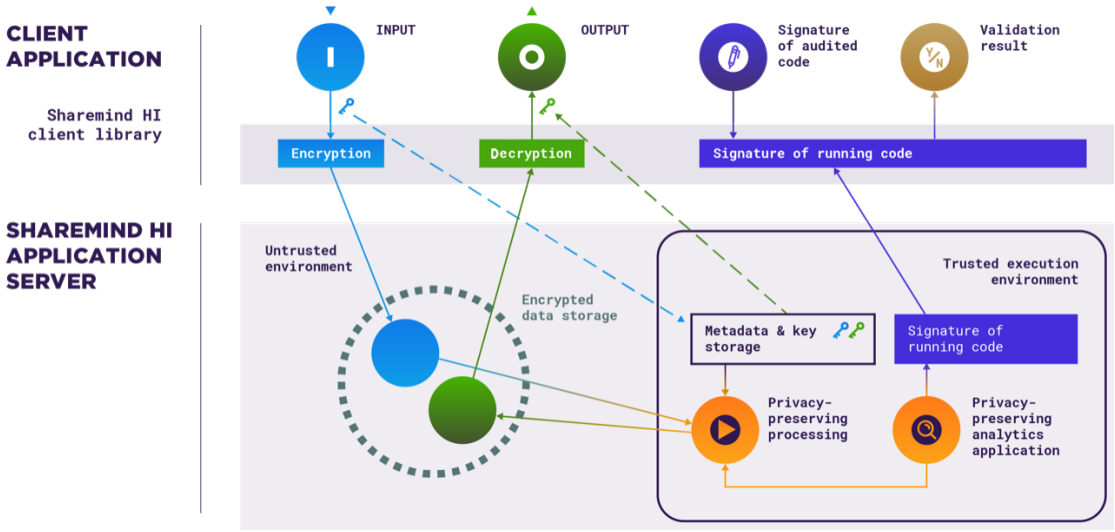

Figure 7. Security model of Sharemind HI [27].

Prior to deployment an auditor is required to validate the enclaves, ensuring they are secure and privacy-preserving, resulting in a cryptographic proof of the audited code. A client can, at any point after deployment, compare that proof against any of the deployed enclaves, ensuring the integrity of the server.

The coordinator has to generate a deployment specific asymmetric key pair for each Sharemind HI deployment. The public key certificate is signed by the Cybernetica Deployment Root CA for Sharemind HI, binding the coordinator to the key pair. Similarly, each client who needs to communicate with the Sharemind HI server, generates an asymmetric key pair, and their public key certificate is signed with the coordinator's private key. The coordinator's signed public key certificate is loaded into the server during deployment and used to authenticate clients in remote attestation. Analogously, the Cybernetica Deployment Root CA certificate is embedded into the server and verifies the validity of the coordinator's public key certificate. This ensures that only the parties explicitly added by the coordinator are allowed to access the deployment, and facilitates authenticating the stakeholders and enforcing access controls.

Sharemind HI also enforces all available measures provided by SGX to combat side-channel attacks. While SGX only provides the option to enable the mitigations with the cost of slower execution speeds, Sharemind HI takes a conservative approach and enforces using all available mitigatory measures.

3 Privacy-preserving data synthesis service

The DANCE project, which this thesis is based on, aims to develop a proof-of-concept (PoC) privacy-preserving data synthesis service, which enables data owners to outsource the synthesis process while being in compliance with data protection regulations. Ultimately, to fully comply with GDPR, the data controller has to decide, whether or not the organizational and technical measures offered by the service are sufficient for their specific use-case. However the service is designed with the goal to be suitable for most use-cases, by ensuring that the data is protected even from the service provider. It achieves this through the use of Sharemind HI.

At the highest level, the whole process of synthesizing data with the service can be separated into three steps. First the data owner encrypts their CSV file and uploads it to the Sharemind HI server. Secondly the task enclaves in the server decrypt the CSV file, use it to create and train a synthesizer model, and synthesize data based on the trained model. Lastly the data owner can download and decrypt the final synthesized data using the Sharemind HI client.

3.1 Use cases and motivation

As mentioned in Section 2.4, there are many use-cases for using synthetic data in general, like stress-testing of systems, data sharing, and training machine learning models. The main hurdle in generating synthetic data is the need to create a synthesizer model, which requires significant resources and knowledge of the subject matter. While multiple companies offering commercial solutions for data synthesis already exist, like Statice[5], Mostly.ai[6], Hazy[7], Replica Analytics[8], and Datagen[9], none of them offer data synthesis where the input data would be hidden from the service provider themselves. The DANCE project attempts to fill that void by creating a PoC privacy-preserving data synthesis service.

As an example use-case one can imagine a financial institution that wishes to create new machine learning models to combat fraud. If the institution does not have in-house capabilities to research and develop such models, they need to outsource the process. However since training, developing, and testing the machine learning models requires large amounts of relevant data, in this case the financial data collected by the institution, it would have to be shared to the outsourced research and development (R&D) company as well. As financial data is highly private and subject to the strongest clauses of data protection, banking secrecy, and confidentiality regulations, sharing it

[5] https://www.statice.ai/
[6] https://mostly.ai/
[7] https://hazy.com/
[8] https://www.replica-analytics.com/
[9] https://datagen.tech/

would require additional contracts between the parties, notifying data subjects of external processing of their private data, analysing compliance to the relevant regulations and implementing all appropriate technical and organizational measures required by data protection regulations. The technical and organizational measures required by privacy regulations can even become too difficult or expensive to be viable, making the sharing impossible. The issue would be solved by providing synthetic data to the R&D company, and only doing minimal fine-tuning of the models in-house. If the financial institution does not have the capability to synthesize data in house, which is likely, given they did not have the capability for building the original models, they would use a commercial service for it. Similarly, as original data is needed to generate the synthetic data, the same issues arise as with giving the data straight to the outsourced R&D company.

The PoC privacy-preserving data synthesis service developed during the thesis provides a solution to the issues outlined above. The service is designed with privacy in mind, providing strong technical and organizational measures to protect the data of the user. Once privacy regulators deem trusted execution environment (TEE) to be a mature enough technology, the service would enable minimizing the legal and operational overhead that comes with sharing personal data.

The financial institution can use the service to encrypt and upload their financial data to Sharemind HI, where the synthesized data is generated securely inside the protected SGX enclaves. The synthetic data can then be downloaded and decrypted, and shared to the R&D company creating the machine learning models.

The example use-case is also highly relevant as financial institutions have been looking into using synthetic data to drive innovation in banking as well as to include it in their artificial intelligence and machine learning pipelines [29]. Mostly.ai claims that machine learning models trained using synthetic data compare with models trained using real data with up to 99% accuracy, and can, in some cases, even provide better models[10]. The idea of using synthetic data to train better models is further supported by Andrew White[11], who also claims that in a few years time a majority of artificial intelligence systems will be trained using synthetic data.

In addition to hypothetical examples, Elering AS[12], the Estonian Rescue Board[13], and the Information Technology Centre for the Estonian Ministry of Finance[14] have all shown interest in the results of the project.

[10] https://mostly.ai/blog/15-synthetic-data-use-cases-in-banking/

[11] https://blogs.gartner.com/andrew_white/2021/07/24/by-2024-60-of-the-data-used-for-the-development-of-ai-and-analytics-projects-will-be-synthetically-generated/

[12] https://elering.ee/en

[13] https://www.rescue.ee/

[14] https://www.rmit.ee/

3.2 Design of the service

3.2.1 Stakeholders and roles

The service includes a number of stakeholders with different roles. As a direct result of using the Sharemind HI platform, the stakeholders can be described through the roles they have in the solution. Table 1 illustrates the roles assigned to each stakeholder. The end-users have all of the interactive roles: they have the input provider role as they are the data owners that have to encrypt and upload the original data; they have the runner role as they have to specify when to run the tasks and provide input parameters; they have the output consumer role to download and decrypt the final synthetic data. Cybernetica AS is in the role of the coordinator, as the developer of the service and the tasks. There also has to exist at least one auditor and one enforcer, who ensure that the service is safe to use for the end-users and that it complies with any and all specified security requirements. The physical server, where Sharemind HI is deployed, can either be hosted by Cybernetica AS, the external auditor, or some new stakeholder (e.g., a cloud service provider) with the appropriate hardware (a server equipped with an SGX-enabled processor). It is important to note that the server host should not have the runner role, as this would facilitate easier side-channel attacks. A host with the runner role can run a task and rollback the server state to before running the task, allowing them to easily generate many side-channel measurements.

Currently every end-user's access rights and roles have to be configured separately in the dataflow configuration file (DFC). This means that all end-users have to be configured in the DFC prior to the launch of the service, as the DFC has to be approved by the enforcers and additional parties cannot be added later without resetting the service. In order to facilitate adding many end-users in a scalable way, a new feature has to be introduced to Sharemind HI. It should be possible to configure a stakeholder to act as a certification authority (CA) that can issue certificates to end-users, who will then automatically have the same rights and roles as the issuing party (henceforth the rootCA stakeholder).

3.2.2 Components

In addition to the stakeholders, the service also involves multiple technical components inside the Sharemind HI server, namely data topics and task enclaves. There are three primary data topics involved in the service:

- an `InputData` topic, which stores the preprocessed contents of the CSV files uploaded by end-users,

[15] While the external auditors are important to ensure the security of the solution, for the PoC phase there are no external auditors and the enforcer and auditor roles will be assigned to the end-users and Cybernetica AS.

		Stakeholders		
		Cybernetica AS	**End-user**	**External auditor(s)**[15]
Roles	**Server host**	+		
	Coordinator	+		
	Enforcer			+
	Input provider		+	
	Output consumer		+	
	Runner		+	
	Developer	+		
	Auditor			+

Table 1. Assignment of roles to stakeholders.

- a `Model` topic, which stores the trained synthesizer models, and
- a `SynthesizedData` topic, which stores the synthesized data.

There are also two auxiliary data topics:

- a `Metadata` topic, which stores sensitive metadata about the data items in the primary topics, like filenames, and
- a `Preprocessing` topic, which stores parameters related to the preprocessing of user CSV files, like normalization ranges and encodings of any categorical fields in the original data.

The processing of data is split into two task enclaves:

- a `Model` task, which creates and trains a synthesizer model, and
- a `Synthesis` task, which generates synthetic data.

The service also includes a web based user interface (UI). The UI acts as a wrapper for the Sharemind HI client, displays all relevant metadata about the data items currently in the topics, and is tasked with preprocessing the CSV files before uploading. From this list of responsibilities the UI can be separated into three components:

- the Sharemind HI client,
- visualization, and
- a CSV processor, which parses the CSV files and preprocesses their contents.

```
Stakeholders:
    - Name: rootCA
      CertificateFile: path/to/rootCA.crt
    - Name: externalAuditor
      CertificateFile: path/to/externalAuditor.crt
Auditors:
    - externalAuditor
Enforcers:
    - externalAuditor
Tasks:
    - Name: model_task
      EnclaveFingerprint: "..."
      SignerFingerprint: "..."
      Runners:
        - rootCA
    - Name: synthesis_task
      EnclaveFingerprint: "..."
      SignerFingerprint: "..."
      Runners:
        - rootCA
Topics:
    - Name: InputData
      Producers:
        - rootCA
      Consumers:
        - model_task
    - Name: Model
      Producers:
        - model_task
      Consumers:
        - synthesis_task
    - Name: SynthesizedData
      Producers:
        - synthesis_task
      Consumers:
        - rootCA
    - Name: Preprocessing
      Producers:
        - rootCA
      Consumers:
        - rootCA
    - Name: Metadata
      Producers:
        - rootCA
      Consumers:
        - rootCA
```

Listing 1. DFC file of the service.

3.2.3 Access rights and the DFC

Prior to any data being uploaded or processed, all stakeholders, tasks, topics, and access rights have to be formalized in the DFC file, as outlined in Section 2.6.1. An example of a DFC file used in the service can be seen on Listing 1 and is illustrated by a corresponding dataflow configuration graph on Figure 8.

Note that while the end-users are not explicitly listed in the DFC file, the rootCA stakeholder is. As a result everyone who is issued a certificate by the rootCA, will have all the same access rights as the rootCA. The end-users and the rootCA are allowed to upload data into the `InputData` topic, download data from the `SynthesizedData` topic, and both upload and download data from the `Metadata` and `Preprocessing` topics. They are also allowed to run both the `Model` and `Synthesis` tasks. The `Model` task is only allowed to read data from the `InputData` topic and upload data to the `Model` topic. Similarly, the `Synthesis` task is only allowed to read data from the `Model` topic and upload data to the `SynthesizedData` topic.

The external auditor does not interact with the launched service, and as such, is not depicted on the graph in Figure 8. However, due to having the enforcer role, they are allowed to and responsible for approving the DFC before the service is launched. Additionally, due to having the auditor role, they are allowed to download and decrypt the audit logs at any point after the service has been launched.

3.2.4 Service lifecycle

The lifecycle of the service can be separated to pre- and post-launch phases. In the pre-launch phase the stakeholders generate asymmetric key pairs and corresponding certificates, the DFC is constructed and approved, and enclaves are initialized. In the post-launch phase, only the rootCA, end-users, and the server will be involved with the service. The whole process flow of the deployed service can be separated into 11 distinct steps as shown on Figure 8.

1. The rootCA stakeholder issues a certificate to an end-user, giving them access to the service.
2. The end-user uses the Sharemind HI client to encrypt and upload their CSV file to the `InputData` topic on the Sharemind HI server.
3. The end-user uses the Sharemind HI client to trigger the `Model` task to run, and to provide the necessary input arguments to the task (e.g., which CSV file to use, model specific parameters, metadata).
4. The `Model` task downloads the specified CSV file from the `InputData` topic.
5. The `Model` task decrypts the CSV file, and uses it to create and train the synthesizer model.

6. The `Model` task encrypts the trained model and uploads it to the `Model` topic.
7. The end-user uses the Sharemind HI client to trigger the `Synthesis` task to run, and to provide the necessary input arguments to the task (e.g., which model to use, how many rows of data to synthesize, metadata).
8. The `Synthesis` task downloads the specified model from the `Model` topic.
9. The `Synthesis` task decrypts the model, and uses it to generate synthesized data.
10. The `Synthesis` task encrypts the synthesized data and uploads it to the `SynthesizedData` topic.
11. The end-user uses the Sharemind HI client to download and decrypt a synthesized CSV file from the `SynthesizedData` topic on the Sharemind HI server.

More detailed Business Process Model and Notation (BPMN) diagrams of uploading a CSV file, running the `Model` task, running the `Synthesis` task, and downloading synthetic data, can be found in Appendix A on Figures 15, 16, 17, and 18, respectively.

3.2.5 User interface

For a more streamlined user experience, the service includes a web application that handles the preprocessing of CSV files, provides a user interface (UI) to the Sharemind HI client, and displays the data present in Sharemind HI servers to the user.

The CSV preprocessing consists of converting all categorical text values in the CSV file to numeric values using label-encoding, standardizing all numeric values column-wise, and serializing the resulting matrix. By default, standardization uses the minimum and maximum values of each column. However these ranges are also displayed to the user, who can modify the ranges to better suit their needs, and the user provided values are used for standardization instead. The user-provided ranges are also used during synthesis to ensure no values outside the specified range are generated. This filtering is needed to ensure that, for example, the synthesized data does not include negative values in the column representing a person's age. While some existing models, like the truncated mixture models, can handle such limitations natively, the Gaussian mixture modeling (GMM) included in the service does not. To avoid limiting the selection of models that could be added to future iterations of the service, the filtering was added instead of changing the model.

The preprocessing is done entirely inside the end-user's browser, due to security, performance, and usability considerations. From the security side, parsing text fields in a side-channel safe manner is difficult and requires more sophisticated methods instead of naïvely parsing the file row-by-row. For example, the length of each row in the CSV file can already leak a substantial amount of information, if timing attacks are used. From the

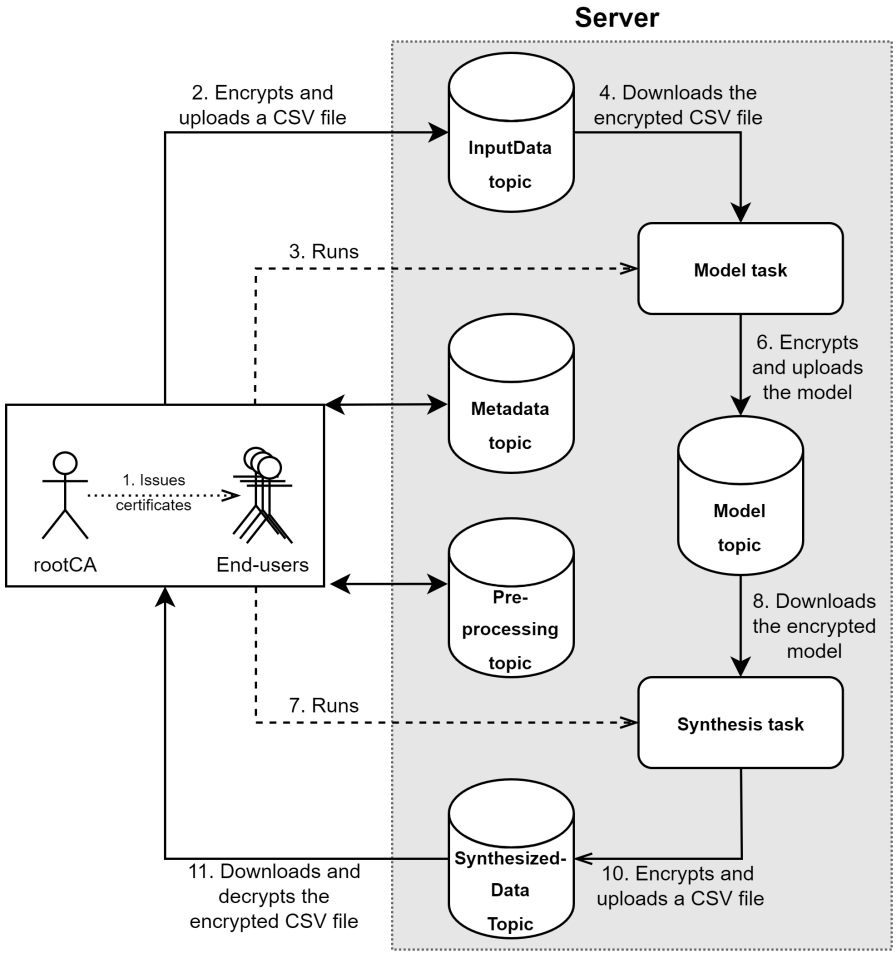

Figure 8. Dataflow configuration graph describing the access control and process flow of the service.

performance perspective, parsing large sections of data inside the enclave is significantly slower than doing it outside the enclave due to memory limitations. While the memory restrictions can be circumvented by using synthesizer models that allow using streams, or incrementally training the models chunk-by-chunk, doing so was considered out-of-scope for the PoC service. Lastly, preprocessing inside the task enclave would incur slight usability issues, as the Sharemind HI task enclaves are designed to only take inputs before starting the task. Hence, changing the column ranges based on user input would require terminating the currently running task and starting a new one with the new ranges.

The web application provides convenient buttons for uploading and downloading data, as well as for starting the model training and data synthesis tasks. The user does not need to know anything about the Sharemind HI client working in the background. The web application also automatically queries the Sharemind HI server to display any

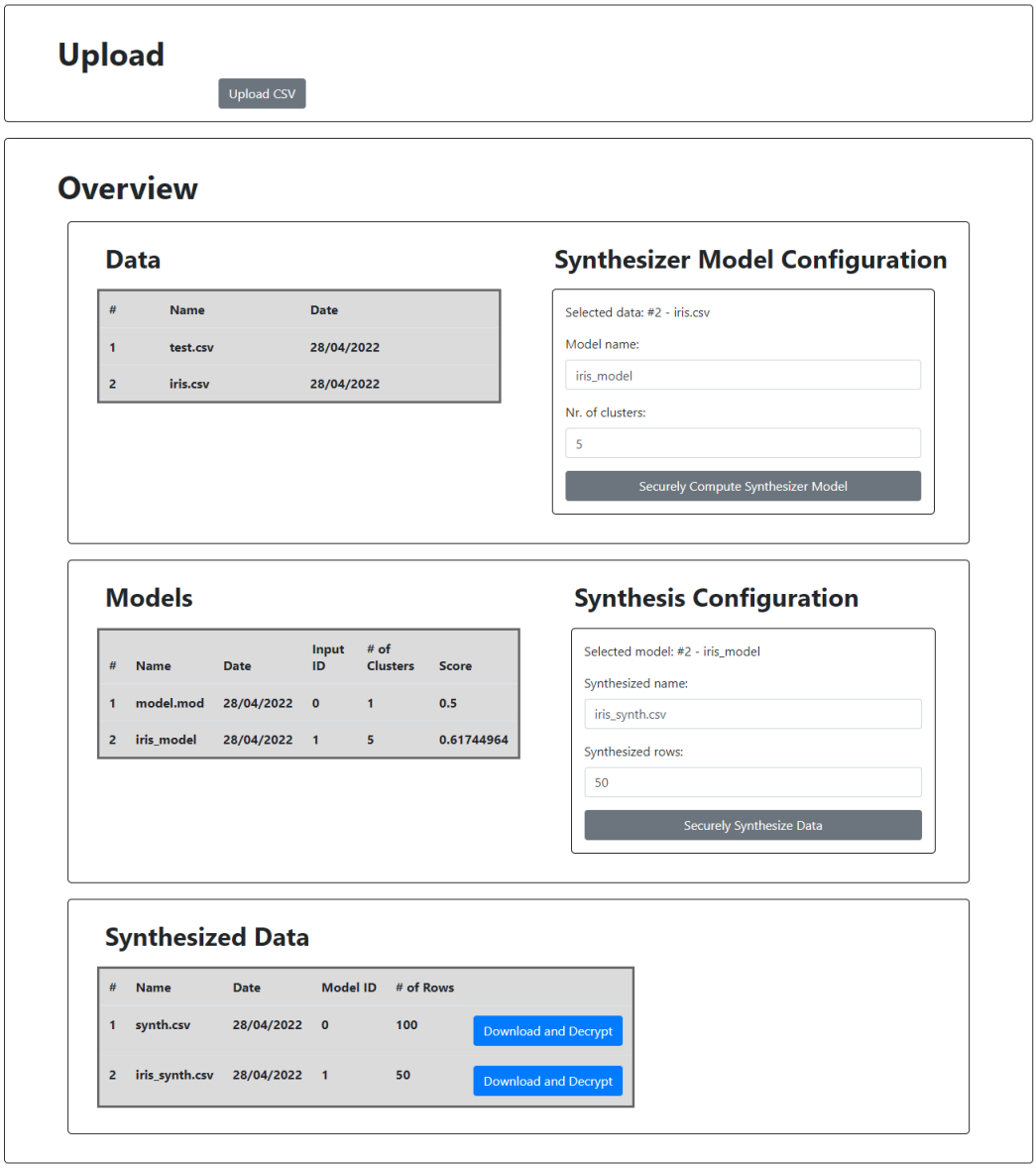

Figure 9. A screenshot of the user interface.

upload or created files, providing the user with an overview of available files. The user interface is shown on Figure 9.

The expected user flow through the service can be separated into four steps: uploading a CSV file, training a model, synthesizing data, and downloading the synthesized data.

To upload a CSV file, the user clicks the "Upload CSV" button and selects a file from their local file system. The web application parses the file and asks the user to confirm or change the inferred column ranges, after which the CSV file is preprocessed and uploaded to the `InputData` topic in Sharemind HI server. The file name and preprocessing info is uploaded to their corresponding topics, respectively the `Metadata` and `Preprocessing` topics. The table containing uploaded files is updated, confirming that the file was indeed uploaded to Sharemind HI and is available for further actions.

To train a synthesizer model based on the newly uploaded file, the user selects the file from the UI, enters a name for the model, specifies the number of clusters the GMM model should be trained with, and clicks the "Securely Compute Synthesizer Model" button. The web application consolidates the user inputs into metadata and task arguments, sends the task run request to the Sharemind HI server, and uploads the metadata to the `Metadata` topic. Once the model is successfully trained, the model along with its metadata appears in the table containing all models.

Similarly, in the data synthesis step, the user selects a model, enters a name for the synthesized data, specifies how many rows the output should have, and clicks the "Securely Synthesize Data" button. The web application acts the same way as it did in the model training step and the synthesized data file appears in the table containing synthesized data.

Finally the user can click the "Download and decrypt" button next to a synthesized data file, which prompts the web application to download the file from the `SynthesizedData` topic and the preprocessing information from the `Preprocessing` topic, convert the data back to the original data ranges, format the data as a CSV file, and initiate the download to the user's local file system.

While the project is intended to act as a commercial service involving many end-users, for the proof-of-concept (PoC) phase we only allow for a single user. Supporting multiple users requires implementing login and user management systems to the web application, which is not in the scope of the PoC privacy-preserving data synthesis service and is intended as future work.

4 Service Architecture

The proof-of-concept (PoC) privacy-preserving data synthesis service developed as a part of this thesis is separated into a client-side and a server-side architecture, which communicate with each other using strictly defined application programming interfaces (API). Both sides also contain generic libraries for performing specific tasks, with the goal of reusing them in future projects.

4.1 Server-side architecture

The server-side architecture wholly consists of the Sharemind HI server. The general architecture of the server is discussed in Section 2.6.2. As the management enclaves are independent of the solution, we will not discuss them further and consider them a black-box in terms of the architecture of the service. The task enclaves, however, are solution-specific and contain the most critical parts of the service.

As described in Section 3, there are two sequential tasks that need to use confidential information. The synthesizer model is created and trained in the `Model` task enclave and data is synthesized in the `Synthesis` task enclave. Having two smaller isolated tasks allows for easier auditing and reasoning about the code, which reduces the risk of task developers introducing vulnerabilities and enforcers not noticing said vulnerabilities. Additionally, separating the tasks isolates them and hence minimizes the attack surface and limits the damage of any potential breaches caused by unforeseen attacks. From the usability side having the tasks separate gives the end-users more options by allowing them to easily re-use a model to generate multiple synthetic datasets from a single model, as well as allowing to create multiple models with different parameters prior to synthesizing any data. Both task enclaves are written in the Rust programming language[16] and make use of a custom built data synthesis library, also written in Rust, which provides all necessary functionality for data synthesis.

The actual solution-specific code, which runs inside the task enclaves of the PoC service, was a contribution of the author. At the beginning of the project this involved creating and deploying empty task enclaves, written in C++, to facilitate the development of other parts of the service and ensure the future development of the enclaves can be done rapidly along with the other parts. At later stages of development, task enclaves were iteratively upgraded to include basic input and output for testing the functionality of the web application and the deployment of the service. The main implementation of the task enclaves was only started once the CSV processing library (Section 4.2.1), and the data synthesis library (Section 4.1.1), were completed. While the logic that has to run inside the task enclaves is not too complex, implementing it inside the task enclaves required utilizing and understanding all other parts of the service. Hence the

[16] `https://www.rust-lang.org/`

main difficulties in implementing the logic involved fully understanding the service architecture and the implementations of all the included parts.

4.1.1 Data synthesis library

The data synthesis library defines and implements an API that allows to easily create and train a synthesizer model, serialize and deserialize a trained model for storage or transfer, use the model to generate synthetic data, and compute various metrics from the model and synthetic data. The library has a modular design enabling easy inclusion of new synthesis algorithms. Currently only Gaussian mixture modeling (GMM) and generative adversarial network (GAN) methods are implemented, however only the GMM method is available for use through the API.

The implementation of GMM in the synthesis library requires the input data for training a model to be in the format of a two-dimensional matrix of 32-bit floating-point numbers. Synthesizing data using the model returns a matrix of the same format. Additionally, training the model requires the number of clusters to be specified by the user.

The synthesis library consolidates the functionality of multiple external libraries. However, since all functionality has to be run inside the enclaves, the code of the external libraries is modified to accommodate the restricted environment. This is done by removing their dependency of the Rust Standard Library[17] (libstd), or in other words making them *no-std* compatible. This is needed as code running inside the enclaves does not have access to the primitive operations, like interfacing with file systems and managing memory, which the Rust Standard Library relies on to implement its functionality and which are usually provided by the operating system [30].

4.1.2 Model task enclave

The `Model` task enclave takes as input the ID of a CSV file in the `InputData` topic and the number of clusters the GMM model should use. It then reads the specified data item from the `InputData` topic, and parses and formats it into a structure usable by the data synthesis library. The parsed data is fed to the data synthesis library to create and train a model based on the inputs. The trained model is serialized, encrypted and written to the `Model` topic.

As all data transport into and out of enclaves happens as bytes, the data is serialized as shown on Figure 10. This format allows for easy and fast serialization from a CSV encoding and deserialization to a matrix of floating-point numbers as required by the data synthesis library.

The first four bytes (each byte is denoted as u8) are the big-endian encoding (meaning the first byte holds the most-significant bits of the number and the last byte holds the

[17] `https://doc.rust-lang.org/std/`

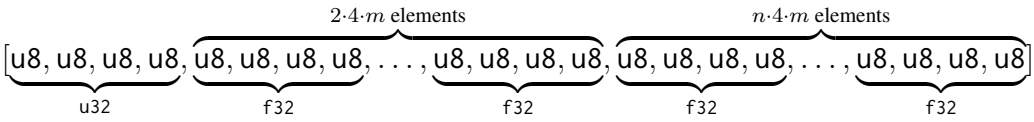

Figure 10. The memory layout of serialized input data.

least-significant bits of the number) of a 32-bit unsigned integer (denoted u32) and the rest of the bytes are big-endian encodings of 32-bit floating-point numbers (f32).

The unsigned integer represents the number of columns in the CSV file, denoted as m. The following floating-point values are separated into two sections. The first $2m$ values are the normalized ranges for each column, where the first half are the minimum and second half are the maximum values. All floating-point values after the ranges are the normalized values of the input file. Hence the byte array consists of $4 \cdot (n+2) \cdot m + 4$ bytes, where n is the number of rows and m is the number of columns in the CSV file.

The enclave reads the whole byte array at the specified ID in the `InputData` topic into the enclave memory, extracts the number of columns from the first four bytes, and parses the rest into corresponding vectors of minimum and maximum values, and a 2-dimensional matrix usable by the data synthesis library.

An argument could be made to further split the `Model` task enclave into smaller tasks, like a model initialization and model training tasks, isolating them and making the service more robust against potential attacks. However, this would introduce more complexity into the overall structure of the service, making it a trade-off between the cohesion of the process as a whole and isolation of specific tasks.

4.1.3 Synthesis task enclave

The `Synthesis` task enclave takes as input the ID of a model in the `Model` topic and the number of rows to be synthesized. The specified model is read from the `Model` topic into enclave memory, and used to synthesize new data row-by-row using the data synthesis library. The synthesized rows are filtered based on the user specified ranges, dropping any row that includes a value that is out of range. The synthesis process is repeated until the requested number of valid rows is reached, at which point the resulting matrix is serialized and written to the `SynthesizedData` topic.

4.1.4 Adding support for the Rust language

The choice to write parts of the service in Rust had multiple advantages over writing them in the C++ language, although only the latter is natively supported by Sharemind HI for writing task enclaves. First and foremost, the low-level nature of Rust combined

with its security oriented design promising to remove many common programming pitfalls that come with manual memory management and raw pointer handling, makes it an attractive choice for writing security critical embedded code [30]. Secondly, using the Rust language was intended as a secondary research objective by the project team, allowing them to gather experience with the language and to test whether writing task enclave code in Rust is feasible.

One of the goals of Sharemind HI is to minimize the risk of introducing errors and bugs when implementing business logic, that may influence the security of the solution. A common source of such errors come from the complexity and power of the C++ programming language [24]. The Rust programming language is designed to prevent many common pitfalls, while retaining the low-level capabilities of other common systems programming languages [30]. The alignment of these design goals leads to the natural conclusion to allow implementing the solution-specific logic used in Sharemind HI task enclaves in the Rust programming language, further reducing the risk of introducing security critical errors.

Task enclaves have access to a limited set of operations to interface with the rest of the Sharemind HI server. Specifically these operations are reading data (and public metadata) from the topics for which the task enclave has an output consumer role, writing data (and public metadata) to the topics for which the task enclave has an input provider role, reading the public data present in the dataflow configuration file (DFC), and receiving the task invocation-specific arguments.

In order to support developing task enclaves in Rust, a foreign function interface (FFI) was implemented. The FFI serves as a bridge between the existing limited operation set written in C++ and the task enclaves written in Rust. From the viewpoint of the task enclave developer, the only thing that has changed is the programming language. When a Rust task enclave calls a function included in the FFI, the Rust side of the FFI deconstructs all the arguments given to the function into raw pointers and passes the pointers over to C++ side of the FFI, which translates the pointers back into arguments usable by the original task enclave operations in C++. Sending information back to the Rust enclave happens analogously, where the raw pointers to the memory location of the information are passed over the FFI, and read back in the appropriate format usable by Rust.

While the basic interface for downloading data from data topics into the enclave was created by a senior developer in the Sharemind HI team, the author of this thesis was responsible for testing and fixing the created interface, adding other critical functionality needed in the DANCE project, like uploading data to topics and adding public metadata, and ensuring the code can be successfully deployed and run in Sharemind HI enclaves.

The implementation required the author to learn the Rust programming language, including advanced features like “unsafe Rust” and managing of raw pointers [30], and to fully understand how the task enclaves are created and deployed in Sharemind HI.

4.2 Client-side architecture

The client-side web application consists of a CSV processing library, the Sharemind HI client library, and the user interface (UI). The Sharemind HI client library was discussed in Section 2.6.2 and can be considered a simple API for interfacing with the Sharemind HI server. For the full version of the service, a user management system is also planned, however it is not implemented during the PoC phase.

The UI also makes heavy use of the public metadata feature, which was added to Sharemind HI by the author of this thesis, to display information to the user about the data items present in the server and to ensure the correct data items are uploaded and downloaded. For example the public metadata attached to the models includes the date when the model was trained, the number of clusters used in the GMM model, the ID of the input data in the `InputData` topic used for training, the ID of the private metadata of the model in the `Metadata` topic, and ID of the preprocessing information in the `Preprocessing` topic.

4.2.1 CSV processing library

The CSV processing library contains all of the functionality to parse, preprocess, and serialize a CSV file. Similarly to the data synthesis library discussed in Section 4.1.1, the CSV processing library is written in the Rust programming language, and does not depend on the Rust Standard Library.

While the CSV processing library is not used inside an enclave in the DANCE project, it was designed with the goal of being re-usable in future projects, which may require using it inside enclaves as well. As the CSV processing library is written to be re-usable, it also contains many functionalities that are not used in the DANCE project, like nullable fields and fields containing dates.

Furthermore, the CSV processing library is written to be compatible with WebAssembly[18]. Compiling the library with WebAssembly allows it to be run in most browsers, which can not normally run compiled Rust code.

4.2.2 User interface

The user interface has two intertwined tasks. First and foremost it provides the user with all the required information about the process through the use of buttons, forms, and tables. Secondly it joins together the user inputs, the CSV processing library, and the Sharemind HI client, and translates between them.

When a user first connects to the web application, their certificate and cryptographic keys are used to negotiate a temporary session key with the Sharemind HI server. The session key is used for all communication between the Sharemind HI client and server.

[18] `https://webassembly.org/`

For the PoC phase, however, the user's certificate and keys are hardcoded into the application. The functionality to generate, store, and import the certificates and keys is planned to be included in the full version of the service.

After the session has been established with the server, and after each user action involving the server, the web application queries the server for the metadata of all data items available to the user. The results are displayed to the user in the form of tables. Quering the metadata after each action also allows to display a fully consistent state to the user. In the PoC phase the user has access to the metadata of all data items, however once the functionality for multiple users is added, each user should only see the metadata of the data items, which they have uploaded or created.

While retrieving all of the metadata after each query incurs a large overhead in communication, ease of implementation was preferred in the PoC phase, as all processes can be optimized in future iterations of the service.

The UI is written in the TypeScript[19] programming language and uses the React[20] framework.

The author of this thesis was fully responsible for everything related to the user interface, from its design, to the implementation and deployment. This includes creating the graphical user interface, including and using all involved libraries, such as the CSV processing library and Sharemind HI client library, implementing the communication with the Sharemind HI server and managing the data items, handling the user credentials, and locally deploying and testing the whole service.

4.2.3 Public metadata

As Sharemind HI is intended as a general purpose platform, it needs to provide flexible functionality to account for as many use-cases as possible. In many situations an input provider may wish to include additional public information with the confidential data they upload. While some information is infeasible to hide completely, like the upload dates or file sizes, others may be public by design, like solution-specific variables or who the original data owner is. Prior to the DANCE project the Sharemind HI platform did not include functionality to handle such cases. The public information had to either be handled externally or serialized along with the confidential data, which complicated the design and analysis of task enclaves.

To simplify the development and analysis of the data synthesis service, the functionality to include public metadata was added. This public metadata is implemented as a list of key-value pairs for each data item, which can be defined arbitrarily by an input provider when uploading the data.

Sharemind HI already included some technical metadata in the DFC file about the data items present on the platform. For example, the DFC included the IDs of all data

[19] `https://www.typescriptlang.org/`
[20] `https://reactjs.org/`

items in all topics as well as some technical details, like the hash values of the data items. Furthermore, the DFC was visible to all stakeholders at all times, and updated during runtime when new data items were uploaded. Hence the new user defined metadata was added to the DFC, to explicitly display the metadata as public, and to implicitly group together all public information about the data items uploaded to Sharemind HI.

Implementing the public metadata feature required the author to acquire a deep level of understanding on how the DFC is constructed and updated in the Sharemind HI platform, and how the client-server communication works.

5 Security of the service

The PoC privacy-preserving data synthesis service developed as a part of this thesis mostly relies on the security properties of Intel SGX and Sharemind HI, discussed respectively in Sections 2.5.4 and 2.6.3. This section will focus on the threat model of the PoC privacy-preserving data synthesis service developed as a part of this thesis, and access rights of each party as defined by the dataflow configuration file (DFC).

The main goal of the service is to synthesize data based on confidential input data, without the host of the server ever seeing the original values. This primary goal can be formalized as a security requirement, where a malicious server host should not be able to learn anything about the uploaded CSV files of the users.

In addition to the server host and the end-user, the service also includes multiple other parties, like the auditors or other end-users in the final version of the service, each of whom has their own access rights. It is important to ensure that no information about a given end-user's confidential data leaks to any of the other parties. Hence we can generalize the security requirement to include any malicious party instead of just the server host.

In addition to the actual confidential contents of the CSV files, the service uses auxiliary metadata. While some metadata, like file sizes and upload dates, are considered to be public information (as the effort spent to protect these from trivial side-channel attacks would exceed their potential sensitivity), other pieces, like filenames, have to be kept secret.

The above considerations are summarized in Table 3 in Appendix A, which shows the full list of all data items involved in the client-server communication, along with who has access to which data items. Furthermore, the flow of the data items between parties is visible in the BPMN graphs on Figures 15, 16, 17, and 18. The table and BPMN graphs mirror the dataflow configuration file (DFC) contents shown on Listing 1, which is the basis for access control management in Sharemind HI. Sharemind HI enforces the access controls as listed in the DFC file, ensuring that each data item is only accessible as shown in Table 3.

A malicious end-user is not considered in the PoC phase, as they are the only party whose data we are protecting, and maliciousness would directly imply that all confidential data has leaked. The security guarantees of Intel SGX protect against a malicious server host through the use of enclaves. The security guarantees of Sharemind HI protect against malicious task enclave developers, as all task enclaves have to be first checked by auditors and approved by enforcers, and are later confirmed to run the approved code using the attestation process. Similarly, the code of the Sharemind HI client should be confirmed by the enforcers and can be checked by the auditors. However, the case of a malicious web application developer is not considered in the PoC phase. While the code of the web application (and the code of Sharemind HI client included within the web client) can be checked by the enforcers and auditors, there are no ways to confirm if the checked version

is the same as the one sent to the end-user. The issue is somewhat mitigated by the fact that the code runs fully in the end-user's browser and can be checked by the end-user each time they interact with the service. This, however, is infeasible for most end-users and, as such, does not provide a strong protection. Ensuring that the code sent out by the web server is the same that was received by the client is a critical known weakness of web applications in general [31]. As such this is considered to be an acceptable risk and the end-user needs to trust the developer of the web application. The cases of malicious enforcers and auditors can be mitigated by having multiple non-colluding stakeholders with these roles. Hence the security of the service requires trusting the developer of the web application and some subset of enforcers. In turn, the enforcers need to trust Intel and the attestation process.

The trust assumptions and potential attack vectors applicable for each component in the PoC privacy-preserving data synthesis service developed as a part of this thesis is given in Table 2. The table shows, that in order to use the service, an end-user needs to ultimately trust two separate parties: Intel, as the developer of SGX, and Cybernetica AS, as the developer of Sharemind HI and the web application. Additionally, trust in the enforcers and auditors reduces the relience on Cybernetica AS, as they can verify that the service fulfills the promised security requirements. The end-user is also expected to trust their browser, as the web-application has no protections against a corrupted browser. This limitation could be removed by, for example, providing a binary executable file, the integrity of which could be verified with a checksum provided through trusted channels. This approach, however, requires the end-user to download, verify, and run the executable, which is significantly less user-friendly than the current method, making it unsuitable to be the primary means of accessing service.

Component	Requires trust in	Potential attack vectors
Intel CPU	Intel	Physical access to the CPU.
SGX instruction set	Intel CPU	Attacks on trusted components. Side-channel leakage from the host machine. Exploiting flaws in the implementation of SGX. Breaking cryptographic primitives.
Intel's pre-made enclaves	SGX instruction set, Intel	Attacks on trusted components. Exploiting flaws in the implementation of the pre-made enclaves.
Local attestation	Intel's pre-made enclaves	Attacks on trusted components. Corrupting communication between enclaves.
Remote attestation	Intel Attestation Service, local attestation	Attacks on trusted components. Corrupting communication between the verifier and the enclave. Corrupting communication between the verifier and the IAS.
Sharemind HI server	Cybernetica AS, remote attestation, enforcers, auditors	Attacks on trusted components. Exploiting flaws in the implementation of the Sharemind HI server.
Sharemind HI client	Cybernetica AS, enforcers, auditors	Attacks on trusted components.
Task enclaves	Sharemind HI server, enforcers, auditors	Attacks on trusted components. Exploiting flaws in the implementation of the Sharemind HI client.
Web Application	Cybernetica AS, Sharemind HI client, browser	Attacks on trusted components. Exploiting flaws in the implementation of the application. Exploiting vulnerabilities in the dependencies of the application. Corrupting the user's browser or machine. Corrupting the communication between the user and the web server.
PoC service	Web application, task enclaves	Attacks on trusted components.

Table 2. List of trust assumptions and attack vectors by component.

6 Service performance and benchmarks

The PoC privacy-preserving data synthesis service developed as a part of this thesis was deployed on a server with the following technical specifications:

- CPU: Intel® Xeon® CPU E3-1225 v5 @ 3.30GHz
- RAM: 4x8GiB DDR4, ECC UDIMM, 2133 Mbps
- Storage: Samsung SSD 850 PRO 1TB

The server was running the Sharemind HI server and serving the web application. The service was accessed through a 100 MB/s wireless connection using the Chrome browser on a client machine, which had the following technical specifications:

- CPU: Intel® Core™ i5-7300U @ 2.60GHz.
- RAM: 2x8GiB DDR3, SODIMM, 2133 Mbps
- Storage: Toshiba SSD KXG5AZNV 256GB

Currently, the maximum size of CSV files that can be processed and uploaded is 64 KiB, due the limitations of the CSV parsing library. This limitation could be overcome with optimizations to the CSV parsing library, and as such will not be considered in further tests. For measuring the performance of other sections of the service, parsing the CSV files was circumvented by generating randomized raw data and uploading it directly to Sharemind HI.

Benchmarking various metrics for assessing the accuracy or goodness-of-fit of the trained model is considered out-of-scope for this thesis, as the synthesis method and its implementation are only used in the service and have not been developed by any member of the project team.

All results are reported as the average run-time over ten independent runs, where time is measured from two pairs of start and end points. The end-to-end times are measured on the client-side from the moment of starting a task in the user interface until receiving confirmation of task completion. The enclave times are measured from inside the task enclaves from the moment the task enclave code starts running until exiting the enclave. Input data consisted of random values between zero and one and was regenerated for every run.

The performance of the `Model` task enclave is described by Figures 11 and 12. Figure 11 describes the relation between the run-time of the `Model` task and the size of the input dataset, while the number of clusters used to train the models was fixed at three. Figure 12 describes the relation between the run-time of the `Model` task and the number of clusters used to train the synthesis model, while the size of the input data was fixed at 4 kB. The `Model` enclave was allocated 50 MiB of stack and 352 MiB of heap for tests.

The performance of the Synthesis task enclave is similarly described by Figures 13 and 14. Figure 13 describes the relation between the run-time of the Synthesis task and the size of the output dataset, while the number of clusters used to train the models was fixed at three and the size of the input dataset was fixed at 4 kB. Figure 14 describes the relation between the run-time of the Synthesis task and the number of clusters used to train the synthesis model, while the size of both the input and output data was fixed at 4 kB. The Synthesis enclave was allocated 256 Kib of stack and 16 Mib of heap for both tests.

Running the tests showed that the maximum size of the input datasets was directly dependant on the size of the stack allocated to the Model enclave during the deployment phase of the service. This limitation, however, could be reduced by adding synthesis methods that allow for streamable or batch-based training of models. The relatively small maximum size of output data was caused by the lack of optimization in the Synthesis enclave causing it to run out of memory when bigger sets were generated.

Comparing the end-to-end and enclave run times shows that the overhead of communication between the client and the server, and of creating and destroying the enclave takes constant time, regardless of the size of input data or the arguments supplied. It is also important to note that the amount of time needed to create an enclave is heavily dependant on the amount of stack and heap allocated to the enclave, which is fixed during the deployment phase of the service. The Sharemind HI team has plans to further improve the enclave creation process, further reducing the overhead.

It is clear that the number of clusters used to train the model increases the run-time of the Model task exponentially, however has minimal effect on the Synthesis task, the run-time of which increased linearly in relations to the number of clusters used. As expected, however, increasing the input size for the Model task and output size for the Synthesis task increases the workload of both tasks linearly.

The performance results brought attention to the need of further optimizing the service in terms of memory management. One of the main limitations of the service is the relatively small size of data that can be processed across all parts, starting from the 64 KiB limit of the CSV parsing library, to the 1 MB limit of output data in the Synthesis enclave. Removing these limitations and improving the general performance of the service is intended as future work. Significant optimizations could be made by adding synthesis methods that support streamable or batch-based training of the models or that can utilize multi-threaded computation environments. Such methods would be able to better utilize the low-memory environment of the enclaves, leading to a smaller memory footprint and reducing slowdowns caused by memory paging.

However, disregarding the current limitations and extrapolating from the performance results indicates that a synthesizer model for an input dataset of size 1 GB could be trained in less than two hours. Furthermore, there is hope that the planned optimizations would reduce the training time significantly, proving that synthesizing data in a privacy-

preserving manner by using trusted execution environments is not only possible, but also feasible in practice.

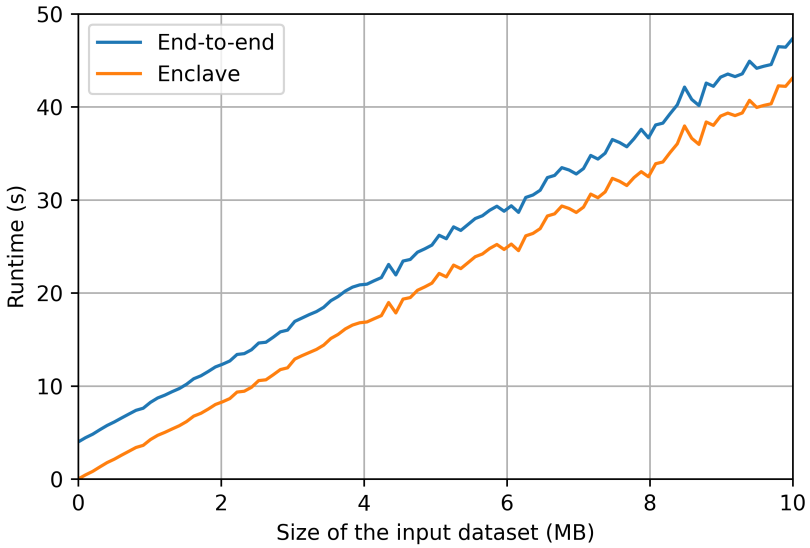

Figure 11. Run-time of the Model task relative to input size.

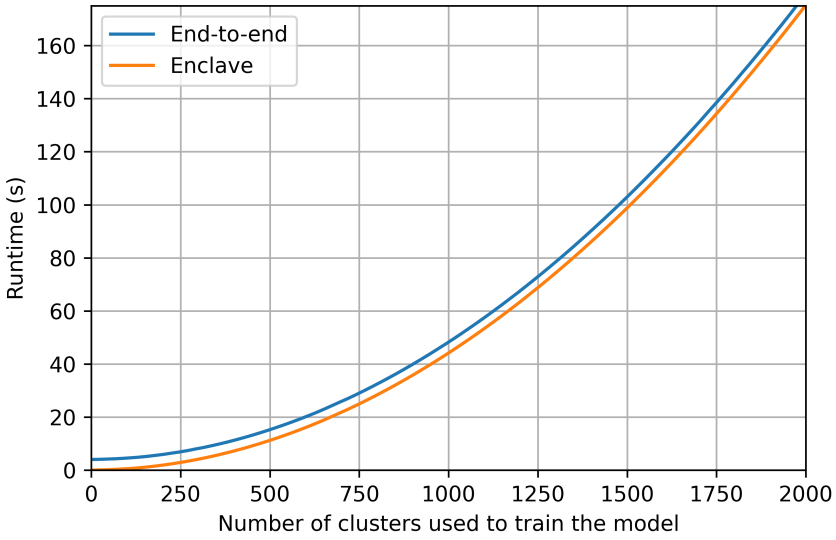

Figure 12. Run-time of the Model task relative to the number of clusters.

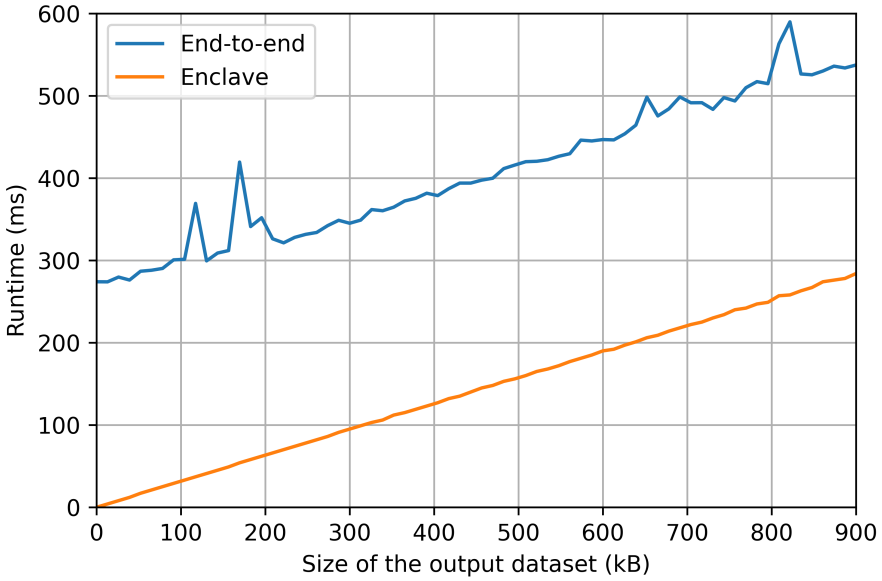

Figure 13. Run-time of the Synthesis task relative to output size.

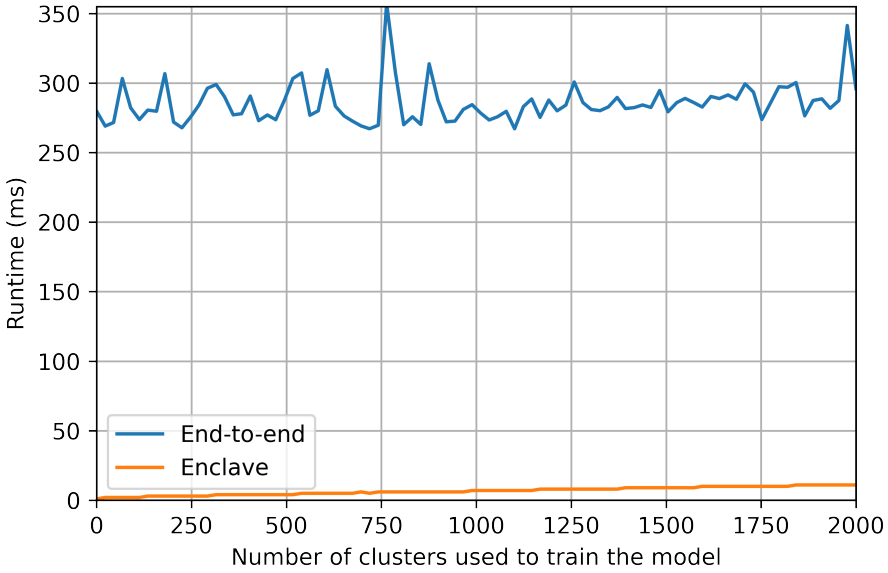

Figure 14. Run-time of the Synthesis task relative to the number of clusters.

7 Conclusion

The goal of this thesis is to develop a proof-of-concept service showing that it is possible to use trusted execution environments to perform data synthesis in a privacy-preserving manner. The thesis gives an overview of the design, architecture, and security properties of the prototype for a privacy-preserving data synthesis service.

The service enables outsourcing the data synthesis process to an untrusted remote server, by ensuring that both the original and synthesized data are fully hidden from the untrusted server host throughout the lifecycle of the service. To achieve the required security goals the service prototype uses trusted execution environment technology, specifically the Sharemind HI development platform, which is in turn based on the Intel Software Guard Extentions (SGX) platform.

The author of the thesis worked as a member of the team responsible for developing the service prototype. Over the course of development the author contributed in three general fields: developing missing features for the Sharemind HI platform, designing and implementing the user interface, and implementing the logic that joins all the various parts of the solution into a coherent working service.

The developed service prototype proves that it is indeed possible to synthesize data in a privacy-preserving manner by using trusted execution environments. Furthermore, benchmarks of the service show that doing so is practically feasible.

The development of the service prototype is still ongoing at the time of writing and is intended to be completed in August 2022. As such the thesis only covers the completed functionality of the service, which is still missing various key features, such as allowing multiple users, and has many limitations, such as only being able to process relatively small datasets due to a lack of optimizations. Cybernetica AS has plans to further develop the prototype and significant work is required before it could be offered as a commercial service. Adding support for multiple users and additional data synthesis methods, as well as optimizing the service, to allow for bigger input and output datasets and to enable faster training of synthesizer models, are planned as future work.

References

[1] *General Data Protection Regulation*
https://eur-lex.europa.eu/eli/reg/2016/679/oj
(Visited on 17.05.2022)

[2] I. Anati, S. Gueron, S. P. Johnson, V. R. Scarlata. Intel Corporation (2013).
Innovative Technology for CPU Based Attestation and Sealing,
https://www.intel.com/content/www/us/en/developer/articles/technical/innovative-technology-for-cpu-based-attestation-and-sealing.html
(Visited on 17.05.2022)

[3] J. Katz, Y. Lindell.
Introduction to Modern Cryptography.
CRC Press, 2020

[4] *California Consumer Privacy Act of 2018*
https://leginfo.legislature.ca.gov/faces/codes_displayText.xhtml?division=3.&part=4.&lawCode=CIV&title=1.81.5
(Visited on 17.05.2022)

[5] *Health Insurance Portability and Accountability Act of 1996*
https://aspe.hhs.gov/reports/health-insurance-portability-accountability-act-1996
(Visited on 17.05.2022)

[6] T. Siil.
Legal status of privacy technologies.
https://courses.cs.ut.ee/LTAT.04.007/2022_spring/uploads/Main/2022-02-16-PET-course-legal-Triin-Siil.pdf
(Visited on 17.05.2022)

[7] G. W. van Blarkom, J. J. Borking, J. G. E. Olk.
Handbook of Privacy and Privacy-Enhancing Technologies - The case of Intelligent Software Agents.
The Hague, 2003.

[8] Big Data UN Global Working Group.
UN Handbook on Privacy-Preserving Computation Techniques.
https://unstats.un.org/bigdata/task-teams/privacy/UN%20Handbook%20for%20Privacy-Preserving%20Techniques.pdf
(Visited on 17.05.2022)

[9] K. El Emam, L. Mosquera, R. Hoptroff.
Practical Synthetic Data Generation: Balancing Privacy and the Broad Availability of Data.
1st edition. O'Reilly Media, 2020.

[10] R. Lebaredian.
SYNTHETIC DATA / AI.
https://developer.download.nvidia.com/video/gputechconf/gtc/2019/presentation/s9943-synthetic-data-will-drive-next-wave-of-business-applications.pdf
(Visited on 17.05.2022)

[11] D. A. Reynolds.
A Gaussian Mixture Modeling Approach to Text-Independent Speaker Identification.
PhD thesis, Georgia Institute of Technology, 1992.

[12] M. Y. Liu, X. Huang, J. Yu, T. C. Wang, A. Mallya.
Generative Adversarial Networks for Image and Video Synthesis: Algorithms and Applications.
In Proceedings of the IEEE, vol. 109, no. 5, pp. 839-862, 2021.

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio.
Generative Adversarial Nets.
Advances in Neural Information Processing Systems, vol. 27, 2014.

[14] B. S. Everitt, D. J. Hand.
Finite Mixture Distributions.
Chapmann and Hall, 1981.

[15] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery
Numerical Recipes: The Art of Scientific Computing (3rd ed.).
New York: Cambridge University Press, 2007.

[16] Intel Corporation.
Intel® Software Guard Extensions Developer Guide
`https://download.01.org/intel-sgx/sgx-linux/2.16/docs/Intel_SGX_Developer_Guide.pdf`
(Visited on 17.05.2022)

[17] V. Costan and S. Devadas.
Intel SGX Explained,
IACR Cryptol. ePrint Arch., 2016, 86.

[18] S. Gueron.
A Memory Encryption Engine Suitable for General Purpose Processors.
Cryptology ePrint Archive, Report 2016/204.

[19] Intel Corporation.
Intel Xeon Scalable Platform Built for Most Sensitive Workloads
`https://www.intel.com/content/www/us/en/newsroom/news/xeon-scalable-platform-built-sensitive-workloads.html#gs.x4jyrv`
(Visited on 17.05.2022)

[20] Intel Corporation.
Intel® Total Memory Encryption White Paper
`https://www.intel.com/content/dam/www/central-libraries/us/en/documents/white-paper-intel-tme.pdf`
(Visited on 17.05.2022)

[21] Intel Corporation.
Intel® SGX Data Center Attestation Primitives
`https://download.01.org/intel-sgx/latest/dcap-latest/linux/docs/DCAP_ECDSA_Orientation.pdf`
(Visited on 17.05.2022)

[22] Systems Software & Security Lab, Georgia Institute of Technology.
SGX 101.
https://sgx101.gitbook.io/sgx101/sgx-bootstrap/overview
(Visited on 17.05.2022)

[23] S. Laur.
MTAT.07.003 Cryptology II, Sigma Protocols
https://courses.cs.ut.ee/MTAT.07.003/2021_fall/uploads/Main/lecture-ix.pdf
(Visited on 17.05.2022)

[24] J. Randmets.
An Overview of Vulnerabilities and Mitigations of Intel SGX Applications
Cybernetica research report D-2-116, 2021.

[25] J. V. Bulck, F. Piessens, R. Strackx.
Nemesis: Studying microarchitectural timing leaks in rudimentary CPU interrupt logic.
ACM Conference on Computer and Communications Security, pp. 178–195, 2018.

[26] Z. Chen, G. Vasilakis, K. Murdock, E. Dean, D. Oswald, F. D. Garcia.
VoltPillager: Hardware-based fault injection attacks against Intel SGX enclaves using the SVID voltage scaling interface.
30th USENIX Security Symposium (USENIX Security 21), Vancouver, B.C., 2021.

[27] Cybernetica AS (2022).
Sharemind HI White Paper.
https://cyber.ee/uploads/sharemind_hi_white_paper_ec24e8189a.pdf
(Visited on 17.05.2022)

[28] Cybernetica AS (2022).
Sharemind HI Overview.
Internal document.

[29] Financial Conduct Authority *Synthetic data to support financial services innovation.*
https://www.fca.org.uk/publication/call-for-input/synthetic-data-to-support-financial-services-innovation.pdf
(Visited on 17.05.2022)

[30] Embedded Rust documentation
https://docs.rust-embedded.org/
(Visited on 17.05.2022)

[31] M. Freudenthal, M. Oruaas.
Secure Programming Techniques - Software Integrity, Supply Chain security.
https://courses.cs.ut.ee/MTAT.07.015/2022_spring/uploads/Main/secureprogramming-04-software-integrity-supply-chain.pdf
(Visited on 17.05.2022)

A BPMN

This appendix provides a detailed overview, in the form of Business Process Model and Notation (BPMN) diagrams, of each step in the main flow of the PoC privacy-preserving data synthesis service developed as a part of this thesis. The main flow consists of four separate steps:

- an end-user uploading a CSV file to Sharemind HI (Figure 15),
- training a synthesizer model using the uploaded CSV file as training data (Figure 16),
- synthesizing data using the synthesizer model (Figure 17), and
- downloading the synthesized data to the end-user's local file system (Figure 18).

All of the distinct data items involved in the service are listed in Table 3. The table also shows which party involved in the service can access each of the data items. The ID and Name columns in the table match the names shown on the BPMN diagrams, allowing to easily cross-reference between the table and diagrams. For each data item, a "+" symbol in the corresponding row in the table, shows that a given party, as indicated by the column header, has access to and visibility of the data item.

For example, the first data item listed in the table, the "End-user's CSV file", can only be seen and accessed by the end-user and the web application. At no point in the service is the CSV file in its raw form accessed or seen by any of the other parties. However the serialized format of the CSV file, named "Serialized CSV file" in the table, is seen and accessed by the web application, the Sharemind HI client, and the `Model` encalve, but is never accessed by the end-user. While they both encode the same information, they are viewed as separate items and are accessed separately.

It is important to note, that the table does not include whether or not a given party can derive some data item from the information they can access. Clearly the end-user could serialize their CSV file to the same format as is used by the `Model` enclave and see the data item, but they do not access the data item directly using the service.

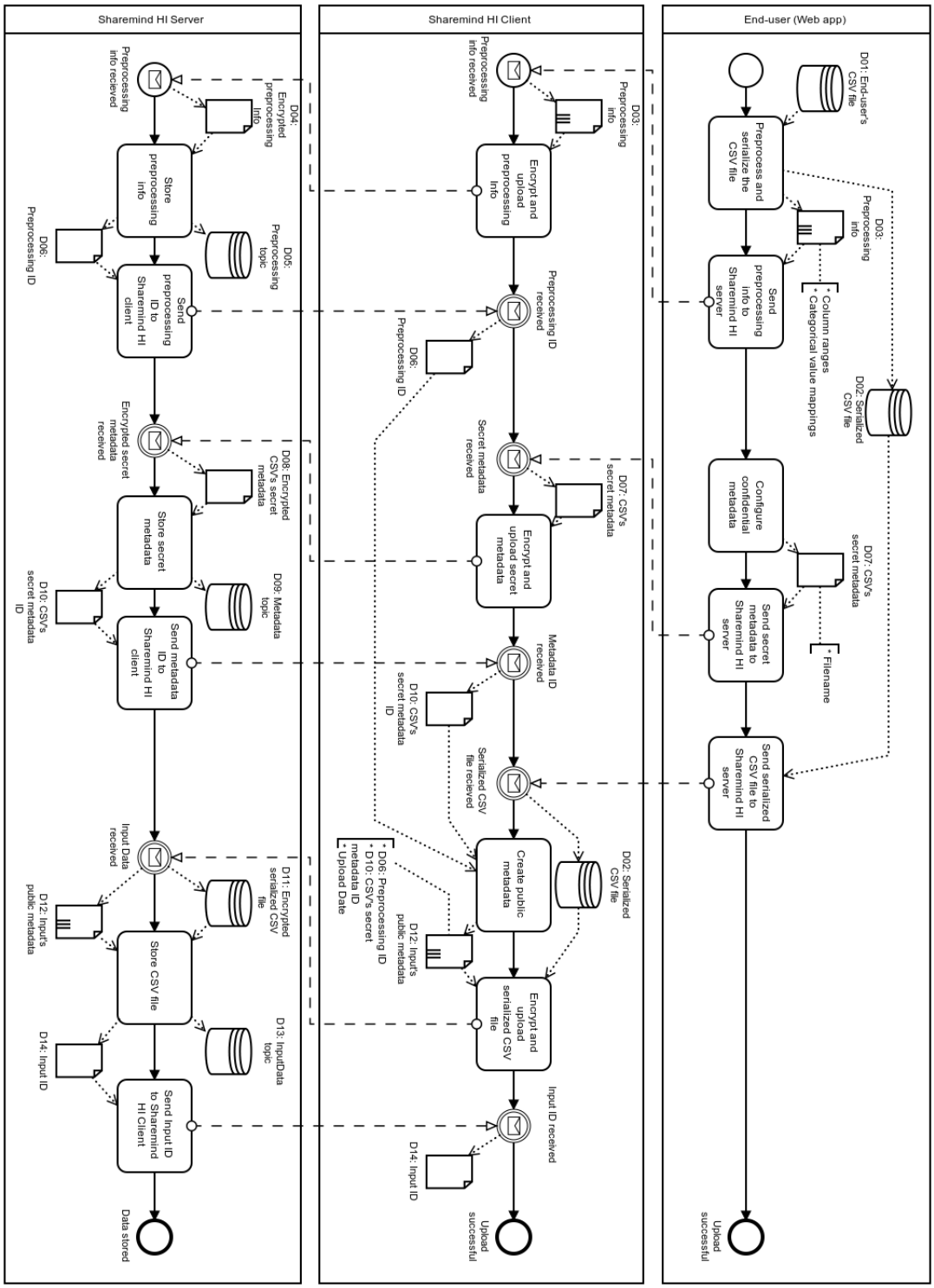

Figure 15. BPMN diagram of the upload process.

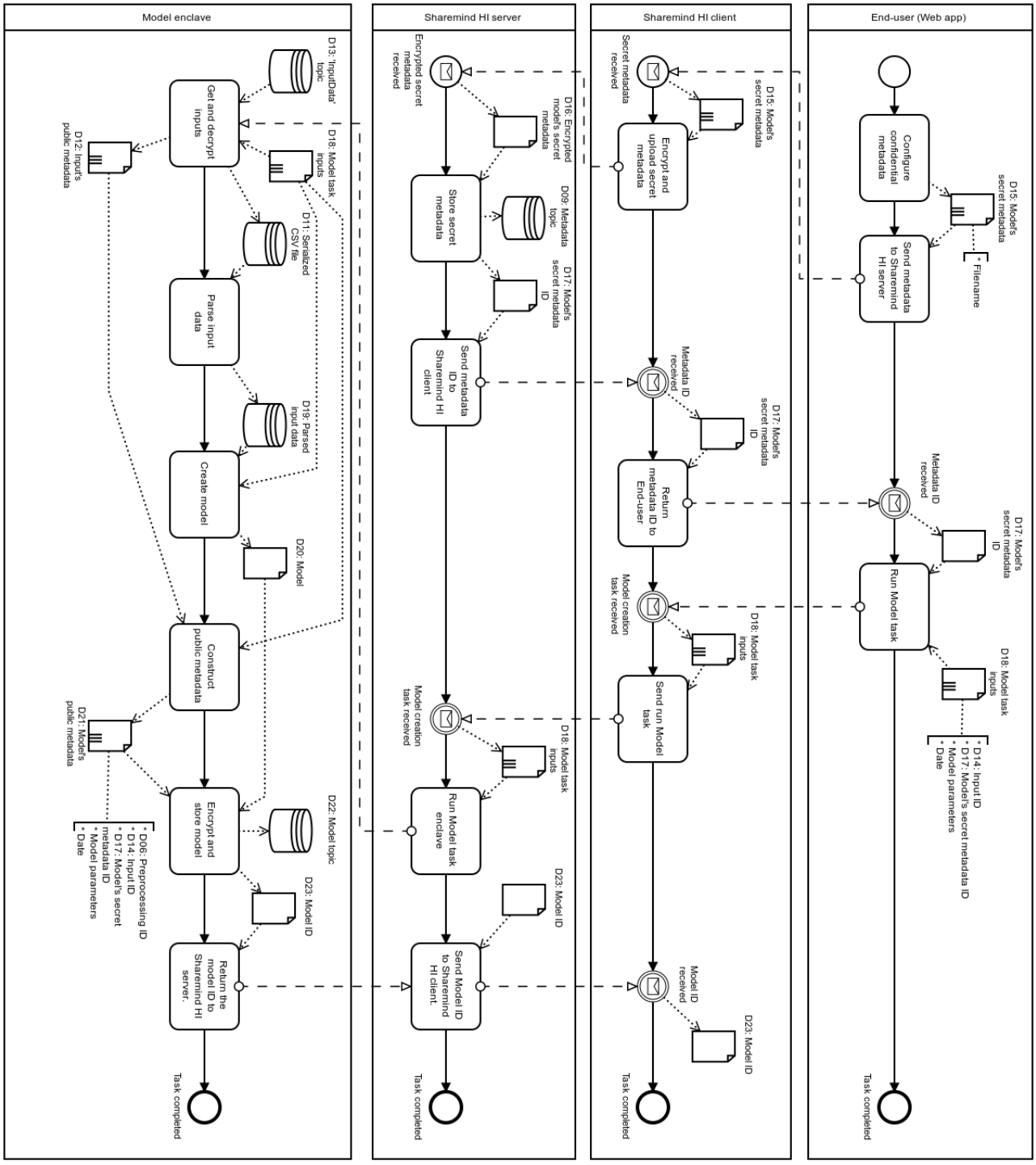

Figure 16. BPMN diagram of the `Model` task.

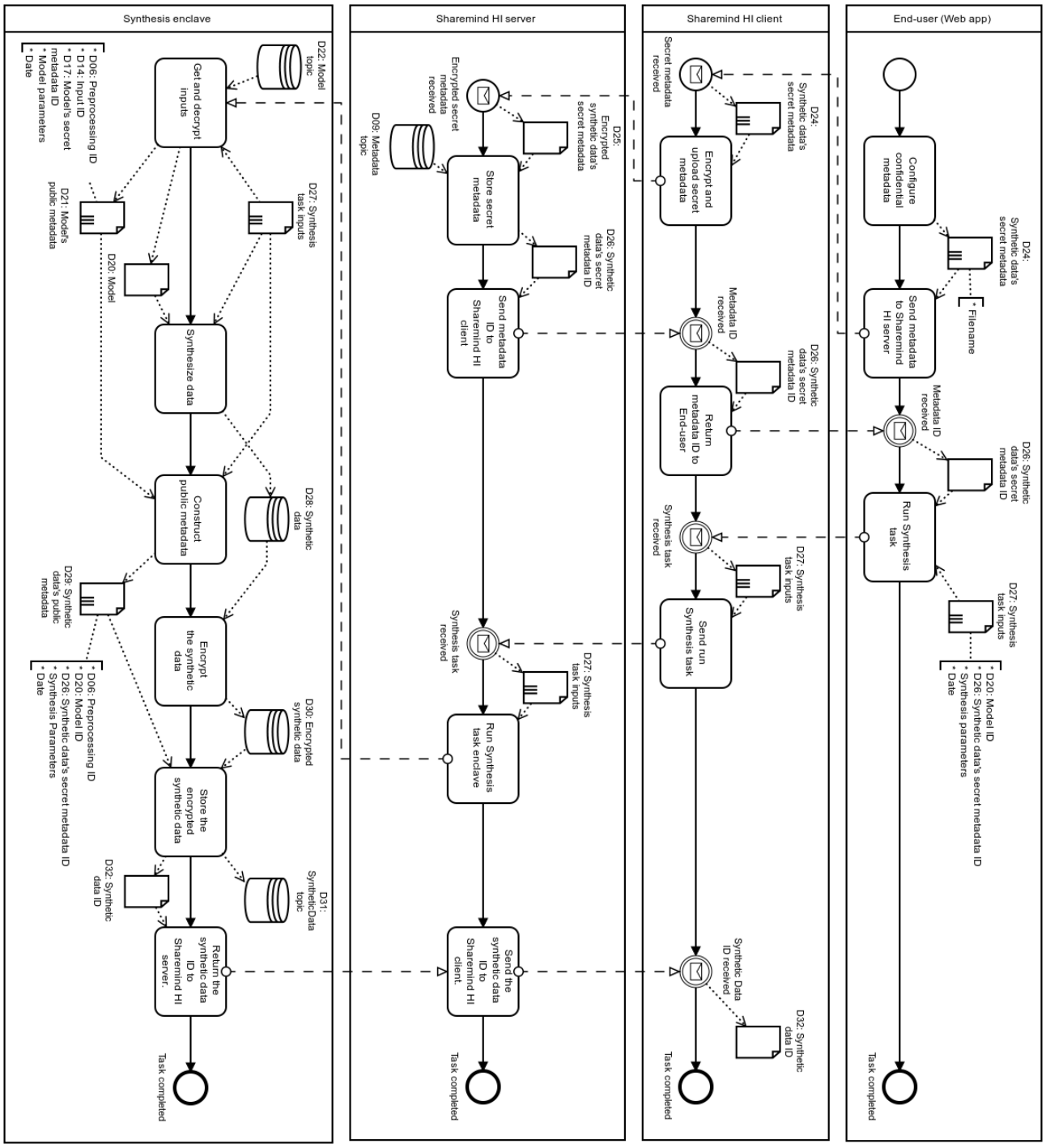

Figure 17. BPMN diagram of the `Synthesis` task.

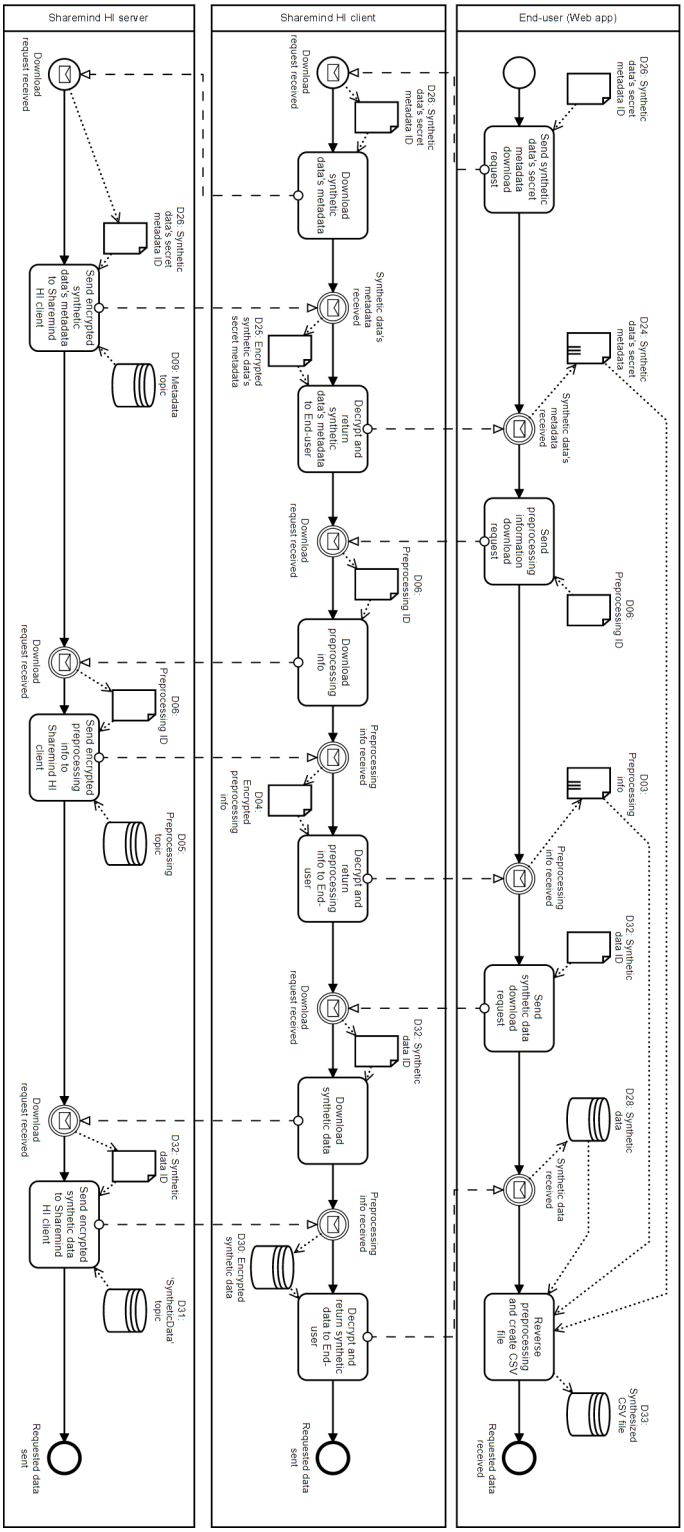

Figure 18. BPMN diagram of the download process.

ID	Name	End-user	Web application	Sharemind HI client	Sharemind HI server	`Model` enclave	`Synthesis` enclave
D01	End-user's CSV file	+	+				
D02	Serialized CSV file		+	+		+	
D03	Preprocessing info		+	+			
D04	Encrypted preprocessing info			+	+		
D05	`Preprocessing` topic				+		
D06	Preprocessing ID	+	+	+	+	+	+
D07	CSV's secret metadata	+	+	+			
D08	Encrypted CSV's secret metadata			+	+		
D09	`Metadata` topic				+		
D10	CSV's secret metadata ID	+	+	+	+		
D11	Encrypted serialized CSV file			+	+	+	
D12	Input's public metadata	+	+	+	+	+	+
D13	`InputData` topic				+		
D14	Input ID	+	+	+	+	+	+
D15	Model's secret metadata	+	+	+			
D16	Encrypted model's secret metadata			+	+		
D17	Model's secret metadata ID	+	+	+	+	+	+
D18	Model task inputs	+	+	+	+	+	
D19	Parsed input data					+	
D20	Model					+	+
D21	Model's public metadata	+	+	+	+	+	+
D22	`Model` topic				+		
D23	Model ID	+	+	+	+	+	+
D24	Synthetic data's secret metadata	+	+	+			
D25	Encrypted synthetic data's secret metadata			+	+		
D26	Synthetic data's secret metadata ID	+	+	+	+		+
D27	`Synthesis` task inputs	+	+	+	+		+
D28	Synthetic data	+	+	+			+
D29	Synthetic data's public metadata	+	+	+	+		+
D30	Encrypted synthetic data			+	+		+
D31	`SyntheticData` topic				+		
D32	Synthetic data ID	+	+	+	+		+

Table 3. List of all data objects involved in the process and who can access them.

Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, Karl Hannes Veskus,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Privacy-Preserving Data Synthesis using Trusted Execution Environments,

   supervised by Liina Kamm and Sven Laur.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Karl Hannes Veskus
17/05/2022