Collaborative Private Decision-Tree Evaluation using (Multi-Key) Fully Homomorphic Encryption

with applications to Risk-Adaptive Access Control

D. G. van der Ende

TU Delft

Collaborative Private Decision-Tree Evaluation using (Multi-Key) Fully Homomorphic Encryption

with applications to Risk-Adaptive Access Control

by

D. G. van der Ende

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on April 13th/14th, 2021 at 10:00 AM.

Student number:	4292014	
Project duration:	February 10th, 2020 – April 1st, 2021	
Thesis committee AM:	Prof. dr. D. C. Gijswijt,	TU Delft, supervisor,
	Dr. ir. M. van Gijzen,	TU Delft,
	Dr. G. Spini,	TNO.
Thesis committee CS:	Dr. Z. Erkin,	TU Delft, supervisor,
	Dr. M.M. de Weerdt,	TU Delft,
	Dr. ir. R.M. Seepers,	TNO.

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

Preface

If someone told me at the start of my studies that I would write my Master's thesis on homomorphic encryption, I would have never believed this, let alone understood what homomorphic encryption is. Only 1.5 years ago, I followed my first cryptography course in Melbourne, taught by an inspiring professor that showed me the magical properties of homomorphic encryption. This immediately sparked my interest. Cryptography is a research field on the intersection of mathematics and computer science and is therefore the perfect research area for my thesis; this thesis is the final requirement to obtain a degree in both Computer Science and Applied Mathematics. Next to this, cryptography has a clear and important application in real life. Therefore, I am grateful for the knowledge I have obtained during the past year. All in all, with handing in this thesis, my life as a student at Delft University of Technology comes to an end. But without my supervisors, friends and family around me, this thesis would not have been as it is now. Therefore, I would like to thank some people in particular.

First of all, I would like to thank my two TNO supervisors, Gabriele Spini and Robert Seepers. Reserving time every two weeks to discuss the progress of this thesis always resulted in interesting new insights or welcome non-thesis-related discussions. While Gabriele was always there to help me with technical, mathematical challenges, Robert made sure that I kept seeing the whole picture. Next, I would like to thank both my supervisors at the TU Delft: Dion Gijswijt from Applied Mathematics and Zekeriya Erkin from Computer Science. I want to thank you both for all the time you took to guide me in this research, the insightful discussions and the weekly check-ups of how I was doing, since working from home has not always been easy. Also, a big thanks to all members of the thesis committees for taking the time to evaluate my work. As I always will be, I am grateful to my family and boyfriend whose support knows no limits and to all my friends who made my student life as exciting as it was. Last, but not least, I would like to thank you for having finished the first page of my thesis. I hope you will enjoy reading the rest of it as much as I enjoyed conducting this research.

Dieuwke van der Ende
Delft, April 2021

Abstract

Decision-tree evaluation is a widely-used classification approach known for its simplicity and effectiveness. Decision-tree models are shown to be helpful in classifying instances of fraud, malware, or diseases and can be used to make dynamic, flexible access decisions within an access-control system. These applications often require sensitive data of more than one party, like financial or health-related records. It is important to keep this data private, especially when the decision-tree evaluation is done in a collaborative manner where more than one party provides sensitive input data.

Current privacy-preserving solutions only consider scenarios in which input data originates from a single source. However, collaboration for decision-tree evaluation tasks is needed more and more since these collaborations often bear fruit. Therefore, in this work we address Private Decision-Tree Evaluation in a collaborative setting. We assume one entity, called the server, that holds the decision tree and multiple users that provide private data on which the decision tree is evaluated. The focus of our research lies on solutions that make use of homomorphic encryption. We give ten different protocols that each take place in a different setting; either the holder of the decision tree receives the evaluation result or the users that provide the input. The protocols use Multi-Key Fully Homomorphic Encryption (FHE) or normal FHE with a Semi-Trusted Third Party (STTP). Additionally, we introduce a novel key-switching method within two of the STTP protocols such that the dependency on the STTP is greatly reduced. All protocols are proven to be secure in the semi-honest model and compared in terms of run-time complexity and communication costs.

Due to the high computational overhead for the Multi-Key FHE schemes, the protocols that make use of these schemes are not yet feasible. Therefore, the protocols that use an STTP are the most promising. All protocols take no more than 4 communication rounds. Assuming that the implementation can be parallelized and given an input bit length of 4, the decision-tree evaluation in our protocols takes in the worst case between 60 and 160 hours, executed on an Intel Core Processor at 2.4 GHz with 16 GB memory.

Contents

1 Introduction

For proper treatment, it is crucial that doctors have access to medical information about their patients. Moreover, in emergency situations access to the correct medical records should be given instantaneously in case these records are essential for proper treatment. In the Netherlands, information is shared between medical providers via the General Practitioner (GP) of the patient, on the condition that this patient gave permission sometime in the past. This information stored at the GP is often not up-to-date or thorough, since medical providers often store medical data only on their own systems and do not share it [89]. Therefore, to cope with the lack of information, medical files are sometimes transported together with the patient in the ambulance via a USB-stick or even sent via e-mail or fax, not secured or encrypted [103]. As a consequence, the privacy of the documents is not guaranteed and in emergency situations those additional documents can often not be shared on time. This can cause wrong or repeated treatment which in turn can lead to severe or fatal consequences [113]. On top of this, since the current platform is based on 'all-or-nothing' according to the permission of patients, healthcare providers can view *all* or *no* medical files once permission is given or not given. In April 2020, to tackle the delay in treatment caused by non-specified consents, the government of the Netherlands registered 8 million Dutch citizens as 'Covid-19-opt-in' [78]. This means that all people that did not yet specify their permission status, are registered as if they gave this permission sometime in the past. Technically, all the Emergency Health Care (EHC) providers in the Netherlands now have the possibility to see the medical information of all these people without the patient giving permission explicitly [110]. The aggregated security risk of this large amount of access points can become significant as these points can be used for phishing or shoulder surfing attacks [113].

To address these problems, doctors and consultants in the field advocate the deployment of Electronic Health Records (EHRs) at the GP to which all medical information of each healthcare provider is synchronised automatically [89]. This system needs to be coupled with an *access-control system* that determines to whom – and in which circumstances – access to certain medical files is granted. Such a system should be able to take into account the current context and change access decisions depending on the severity of the situation [30]. Next to this, the doctors should only gain access to the files that are in line with the treatment or symptoms or their current task; this is called the principle of *need-to-know* or the principle of *least privilege* [113].

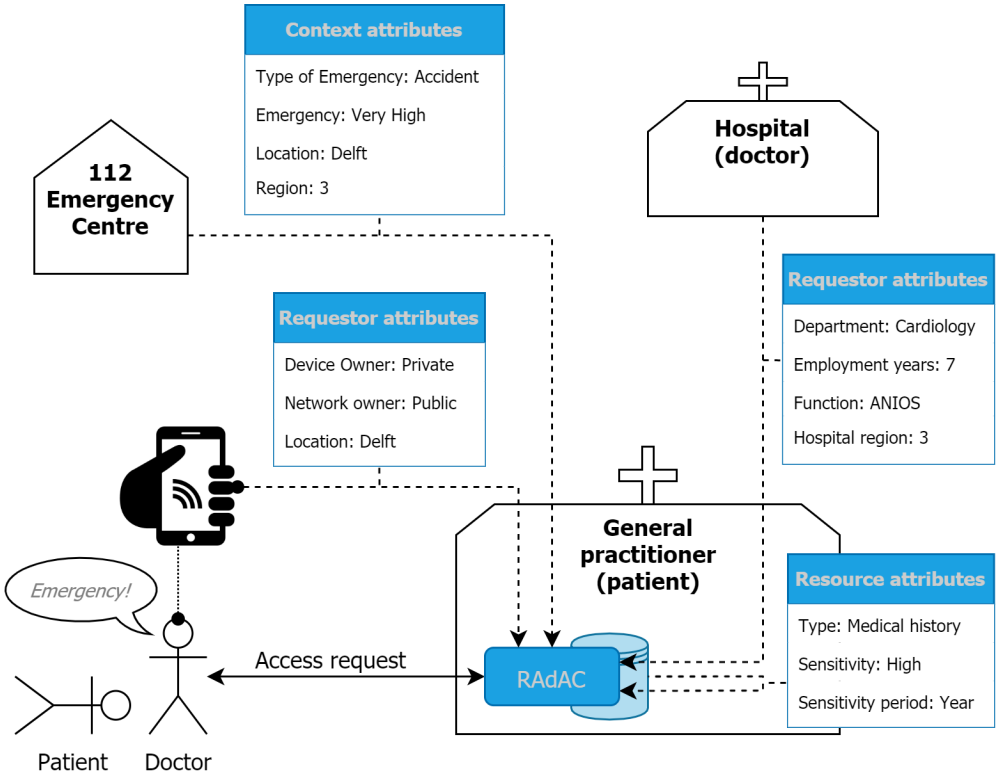


Figure 1.1: An example of a RAdAC use-case.

1.1. Risk-Adaptive Access Control

Healthcare is only one of the domains where such a specialised access-control system is required. Within the military, for example, there is also a need to exchange information to successfully complete missions. On the theatre of war, circumstances can change drastically and emergency conditions can occur. In these conditions, unplanned access decisions often need to be taken and the consequences of not sharing information can be more grave than those of sharing it [75]. Also, access to information may have to be granted in ways that were not previously foreseen. Other examples of environments where specialised access-control systems are needed are within air traffic control, multinational partnerships, or when multiple governmental institutions work together.

Several access-control systems are proposed in the literature [81, 99]. Many of them are not capable of handling complex, dynamic environments and their associated challenges since they employ rigid and static access control policies [43]. Fortunately, research has been done into new access-control models that tackle the above limitations. In 2009, McGraw [75] proposed a novel access-control model that dynamically provides access to information: Risk-Adaptive Access Control (RAdAC). McGraw sees this model as an emulation of real-world decision making where operational need, security risks and the influence of situational conditions on these factors are taken into account. RAdAC bases the access decision on the determined security risk instead of only a strict comparison of the input with the system's rules. It grants access in case the risk is lower than the acceptable risk associated with the operational benefit of the access request. In this way, the system can handle more complex and dynamic environments where risks can be accepted more often in case these are outweighed by the operational benefit of granting access.

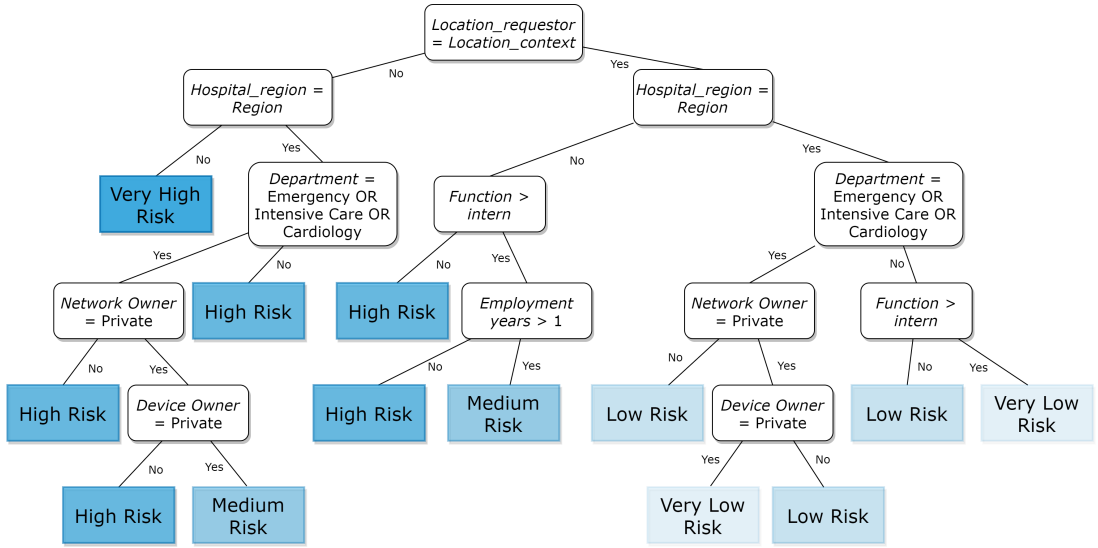


Figure 1.2: An example of a RAdAC decision tree that can be used within the use-case of Figure 1.1; the input is classified towards risk levels and in case the operational benefit outweighs this risk, access is granted.

As an example, in Figure 1.1 a use-case is given in which RAdAC can be applied. It demonstrates an emergency situation in which a doctor tries to access some medical files of the patient. The doctor works in a hospital close to the location of the accident, but this hospital is not the usual healthcare provider of the patient. RAdAC can then base the access decision on attributes originating from multiple organisations; the hospital of the doctor, the GP and the 112 emergency centre. RAdAC translates all attributes to a risk value related to the access request and checks how it compares to the operational benefit of granting the access request.

1.2. Decision-tree evaluation

This thesis is the first work that proposes to use a decision tree as the function to determine risk within RAdAC. In Figure 1.2, this is demonstrated for the use-case in Figure 1.1. The RAdAC system is given by a decision tree that classifies the input attributes towards different risk levels. Access is granted if this risk level is outweighed by the operational benefit regarding granting access to the file in the current context. By using a decision tree, the system can be designed by associating sets of attributes to a risk value. Other RAdAC systems need to assign a numeric risk score to every specific attribute value and require a function that combines these scores to determine the total risk. Translating all attribute values to risks and finding the correct aggregation function is known to be a hard task [19, 43].

Next to the application within RAdAC, decision-tree evaluation has shown to be very useful in many other impactful areas, due to its effectiveness and low training cost [73, 104]. It is a valuable tool that has a positive impact on our daily lives. Decision trees are used to provide users with proper recommendations on products, are deployed to detect malware, and are even able to predict bacterial infections [55, 92, 94]. Other applications include fraud detection, spam filtering, intrusion detection, and remote diagnosis [56, 91, 98, 105]. Decision-tree evaluation is a preferred method in many real-time applications due to the fact that it is easy to visualise and understand [12].

1.3. Privacy concerns

Our RAdAC system comprises a decision-tree model for risk assessment. The decision tree requires sensitive (external) data about the requestor, file, and context in order to do a proper risk assessment of the access request [75]. This input information comes from multiple entities that all send their data to one party that determines the access decision. The more detailed the input data, the better the system can do the risk assessment. Recently, the privacy issues related to access-control mechanisms have been acknowledged in the literature [10, 115, 116, 119]. One might understand that individuals or entities do not want to share all the input information due to privacy constraints. In collaborative environments where parties work together towards a common goal but share no mutual trust, this problem becomes even more significant. Solving this issue makes the application of RAdAC more attractive.

Also in the other application areas, access to sensitive input data, like financial or health records, is required in order for the decision-tree evaluation to be successful. In these fields, collaboration between parties for classification tasks is needed more and more since these collaborations often bear fruit [44]. As an example, collaborations between financial institutions can improve the detection of fraud [97]. This is only possible when the privacy of the data of all these institutions is guaranteed. Next to this, the decision-tree model that is used for the classification might be a valuable asset to the holder and should not be publicly known. Furthermore, it is shown that making the model public could violate the privacy of the training data due to model-inversion attacks [50].

Therefore, it can be concluded that decision-tree evaluation within RAdAC and other contexts raises many privacy concerns. Ideally, the evaluation should be done while keeping both the input data and the model private.

1.4. Potential solutions

Related to our problem setting, namely the privacy concerns regarding decision-tree evaluation, multiple methods have been proposed that preserve the confidentiality of the input data and the decision tree itself while evaluating a decision tree. These are called Private Decision-Tree Evaluation (PDTE) protocols. The existing PDTE protocols make use of one of these techniques: homomorphic encryption [13, 59, 104, 107, 114], garbled circuits [106] or secret sharing [38, 40].

All the current PDTE protocols are placed in the two-party setting where a server holds a decision-tree model and a user holds all inputs. These protocols try to reach the goal of outputting the evaluation result to the user, where both the user and the server do not gain any knowledge about the decision tree or input respectively. These protocols are therefore in a promising direction with regard to solving our privacy concerns.

1.5. Research questions

None of the current works focusing on PDTE propose privacy-preserving solutions when the input data that needs to be classified is coming from more than one authority or organisation. However, within RAdAC, the input that needs to be classified comes from different external sources, like a hospital, GP or the device of the requestor. As mentioned earlier, collaboration for decision-tree evaluation is needed more and more. Sometimes, even collaboration between mutually untrusted parties or parties that have conflicts of interests, such as in the context of military operations, is required. This can only happen if the privacy of each of those parties is guaranteed.

Clearly, there is a need for solutions that address both the necessity of collaboration in the field of decision-tree evaluation and the privacy issues that coexist with this collaboration. Our meaning of *collaboration* is that the input variables originate from more than one party. These privacy issues are addressed when both the input attributes and the decision-tree model stay private to the owners. Therefore, in this thesis we try to answer the following research question:

> How can we collaboratively evaluate a decision tree in a privacy-preserving way using homomorphic encryption?

Our work focuses on using homomorphic encryption as the main technique for our solutions. Homomorphic encryption is a type of encryption where computations can be performed on ciphertexts without accessing the plaintexts. As a result, the number of required communication rounds stays low and independent of the size of the tree. In addition, within homomorphic encryption many different research areas exist, which makes the number of potential research directions widespread.

We compare our protocols in terms of run-time complexity and communication costs. It is assumed that all protocols take place in the *semi-honest* model which means that all parties execute the protocols correctly. In order to check the feasibility of our protocols regarding the use-case, we discuss how different types of variables can be handled in our protocols. Therefore, we define the following sub-questions:

1. How do we prove the correctness and the security in the semi-honest model of these protocols?
2. How do these protocols compare in terms of run-time complexity and communication?
3. How can numerical and categorical variables be handled in these protocols?

1.6. Contributions

This thesis takes a step towards feasible solutions for collaboratively evaluating a decision tree in a privacy-preserving way. To our knowledge, this is the first work that addresses Private Decision Tree Evaluation in a collaborative setting.

First of all, we give ten different collaborative PDTE protocols that each take place in a different setting; the access-control use-case where the *server*, that holds the decision tree, receives the result or the use-case where the decision-tree model is used as an external resource and where all users receive the evaluation result. The protocols either use multi-key FHE or normal FHE. Multi-key FHE allows homomorphic calculations on ciphertexts that are not encrypted under the same key. Six of our protocols use an additional party that participates in the protocol but does not gain any knowledge about the tree or the input. In this thesis we call this additional party a Semi-Trusted Third Party (STTP), since the amount of trust that needs to be put into this party is very low. Additionally, we introduce a novel key-switching method within two of the STTP protocols such that the dependency on the STTP is greatly reduced.

Next to this, we show that all our protocols are secure in the semi-honest model. We give an elaborate analysis of the correctness of the FHE schemes and the noise propagation during all the operations in the protocols, which is essential for a correct implementation. Also, we introduce some adaptations and new operations within the FHE schemes that are required for the functionality of our protocols and to make our schemes more intuitive and efficient. We give a first implementation of the protocols in order to do an overall comparison of the protocols in terms of run-time complexity and communications costs and give suggestions on how to implement these protocols in the use-cases mentioned earlier.

Concisely, our contributions are the following:

- We propose the first collaborative PDTE protocols using (multi-key) fully homomorphic encryption, an STTP and/or a key-switching procedure.
- We provide extensive analyses of the used homomorphic encryption schemes, which are required for implementation of the protocols. This includes the introduction of new homomorphic operations, adaptations to bigger plaintext space, and detailed description of the parameter setting of the schemes and noise propagation per homomorphic operation. Additionally, we provide proofs of correctness and elaborate upon security proofs from the literature.
- We implement all protocols ourselves, including all used encryption schemes and building blocks, and compare them in terms of computational complexity and communications costs (number of rounds and communication sizes).
- We are the first work that proposes the use of decision trees as a way of determining the risk of an access request within RAdAC.

As a final remark, since our work is the first work to propose private decision-tree evaluation where the input originates from more than one user, we introduce a new line of research within private decision-tree evaluation.

1.7. Outline

The rest of this thesis is organised as follows. We present the related work in Chapter 2. Next, in Chapter 3, we cover the preliminaries necessary for understanding the subsequent chapters including some techniques and schemes that are used in the rest of this thesis.

Due to the fact that this is a thesis to obtain a degree in both Applied Mathematics and Computer Science, the next chapters have a clear distinction between the two different aspects of this research. Therefore, we introduce two parts in this thesis, where the focus of the first part lies within mathematics and the second part within computer science. Since the thesis is set up around one topic, which addresses both research fields, we advice the reader to read this thesis as a whole.

The first part consist of two chapters. In Chapter 4, we give detailed mathematical analyses of the used encryption schemes. This includes some adaptations required for the implementation, the introduction of additional operations, and correctness proofs. We give our collaborative PDTE protocols in Chapter 5, with several different approaches per scenario.

In the second part of this thesis, we analyse the protocols and implement them. In Chapter 6, we give complexity and security analyses of all protocols. We give details of the implementation and present the results regarding the run-times and communications costs of our protocols, in Chapter 7.

Finally, in Chapter 8 we discuss the results and give conclusions and recommendations for future work. For the interested reader, the Appendix finishes this thesis with an additional proof draft, the parameter setting per protocol, additional results and the evaluation of an example use-case tree.

2

Related work

This chapter describes the literature related to this thesis, including the research within RAdAC and existing privacy-preserving techniques within access control. Secure Multi-Party Computation (MPC) is a field within cryptography that has the goal to design methods in which multiple parties can compute an agreed function on their inputs in a secure way, where the correctness of the output is proved and the input of every party is kept private [35]. This means that PDTE protocols are secure MPC protocols. We build upon existing PDTE protocols, by extending the setting to multiple parties that provide the input. Therefore, our work falls in the category of MPC using homomorphic encryption, specifically where the evaluated function is a decision tree and the input comes from more than two parties. Therefore, this chapter introduces some general secure MPC techniques and MPC methods that use homomorphic encryption. Additionally, the existing PDTE protocols are described where there is only one user that provides the input.

2.1. Towards Risk-Adaptive Access Control

Access control of data is defined as a way to protect the data against unauthorised access by checking that a user presents proper certification before allowing them access [11]. The two commonly used mechanisms are Role Based Access Control (RBAC) and Attribute Based Access Control (ABAC). While RBAC makes use of predefined roles that carry a specific set of privileges associated with these roles, ABAC grants access to those users with a certain set of attributes specifying for example the network connection, role, and department [58].

As mentioned in the introduction, complex environments like healthcare or air traffic control demand a dynamic and flexible access-control system, that is able to adapt itself to the current context and can incorporate uncertainties in the access decision. An example is when an external doctor acquires access to sensitive medical files in order to save the life of a patient. The traditional access-control schemes can not offer all of this [48]. Gasparani [52] summarises the limitations of the current schemes in three points: (i) they can only handle a certain amount of complexity making them incapable of handling exceptional situations, (ii) they are risk-averse and therefore give little room to support the dynamic and situational conditions in the real world, and (iii) they are not flexible enough to handle changing behaviour of users or context.

In 2009, McGraw [75] proposed RAdAC as an emulation of real-world decision making where both operational need, security risks, and the influence of situational conditions on these factors are taken

into account. RAdAC incorporates a determination of security risk in the access decision instead of only using a hard comparison of attributes [48]. In this way, the system can handle more complex and dynamic environments where risks associated to granting access are accepted more often in case this risk is outweighed by the operational benefit of the access request.

Since McGraw proposed this new idea of doing access control, several models have been developed to make an access-control system more based on risks. Some of these models were already developed before McGraw officially defined RAdAC [27, 42]. Still, they try to incorporate risk in an access-control system and therefore we discuss them here as well. All models are characterised by the determination of the input factors and the function that assigns a risk value to each request [43, 48]. Kandala et al. [60] and dos Santos et al. [43] propose a framework in which an access request is defined as a set of attributes and values to which a risk quantification function is applied. In [19] possible attributes are discussed that can be taken into account within this framework. These are also used in the work of dos Santos et al. [43]. In [42] the final risk score is a weighted mean of the risk values in terms of availability, confidentiality, and integrity using the several input attributes. The above works use a framework that is the closest to our approach in which we evaluate a decision tree towards risks. Instead of assigning to each attribute a risk value on its own, we assign a risk level to a combination of attributes. This makes it easier to design the system. Namely, defining the amount of risk contributed by one attribute value and expressing this by a number is a very hard task [19].

Other approaches take only a few specific input factors on which the risk value is based. These can be scores related to either the sensitivity of the document or the level of need-to-know of the requestor. Wang et al. [112] propose a solution in this direction using a health information systems use-case. Cheng et al. [27] and Ni et al. [83] use this approach and apply fuzzy logic to these parameters to quantify risk values. A slightly different approach is proposed in [62, 100] where the focus lies on the impact that a granted access can have, which is related to document sensitivity. Molloy et al. [79], on the other hand, train classifiers on the history of made access decisions and use the confidence of the classifiers as an indication of the risk of the access request.

2.2. Privacy-preserving techniques within access control

Recently, the privacy issues related to access-control systems have been acknowledged in the literature. According to Zhang et al. [119], frequent disclosure of the attributes will inevitably lead to leaks of sensitive information. Xu et al. [115, 116] agree; they argue that the exposure of sensitive information to the access-control system has an impact on the users' privacy. Next to this, the access-control model is often not hidden. Having this model publicly available can lead to disclosure of private data [10]. These issues reduce the public trust in these access-control models and therefore affect the models' development [116]. Research has been done to tackle these privacy issues within access control. The works that try to tackle the privacy issue regarding the access-control model require the model to be invisible to the server that makes the access decision, for example when the access policy is a combination of the private policies of multiple parties. To answer our research question, we are looking for a solution that keeps the input and model private to the *owners*, where the access-control model is defined by the access-control system itself. We present all proposed solutions in this section, although some are not closely related to our research.

2.2.1. Anonymous credentials

Anonymous credentials can be seen as digitally-signed attributes that allow the owner to prove statements about attribute values without revealing additional information [6]. The major advantage of anonymous credentials is that it is not needed to disclose all attributes in the credential in order to

authenticate yourself; the system allows for selective disclosure. Anonymous credentials work by engaging the user and verifier in an interactive protocol during which the user proves that she owns a valid credential in which some attribute values satisfy certain statements. These statements can consist of inequalities (e.g. age > 18), ranges (e.g. 18 > age > 12) or set-membership (e.g. role $\in$ [approvedroles]) [93]. The basic principle was put forward by Chaum in 1985 [23]. Efficient schemes were proposed e.g. by Brands [17] and Camenish et al. [20].

Anonymous credentials have been applied in the context of access control, to preserve the privacy of input attributes [6, 7, 21, 63, 93]. All these methods use selective attribute disclosure. The focus lies on which attributes have to be revealed or what conditions have to hold over the attributes and, based on this, certain attributes are hidden. The downside of using anonymous credentials within our decision-tree model is that for every decision node the result of the comparison, set-membership or equality test is still revealed.

2.2.2. Encryption-based methods

In [102] a method is proposed that preserves the privacy of an access-control model, using homomorphic encryption such that the model can stay hidden. In this method, first all different private policies are homomorphically evaluated for a certain access request; then, the private decisions are combined according to some multi-party policy. The final decision is then translated to plaintext.

To our knowledge, Xu et al. [115] propose the only cryptographic method that preserves the privacy of the input attributes. By using homomorphic encryption, each rule of the access-control policy can be privately evaluated in two communication rounds. As for the anonymous credentials, the evaluator can see the outcome of every specific rule, so not all information leakage is counteracted by this method.

Next to preserving the privacy of the input attributes or the access-control model, encryption can also be used differently within access control. Encryption can serve as the access-control system itself [2, 68]. These methods work by encrypting the resources, and only users with a specific set of attributes have the secret key to decrypt these resources. These methods can be extended to also address the privacy concerns. In [47] a one-way anonymous key agreement protocol is added to this resource encryption scheme, to anonymize each attribute that is used for the key generation. Moreover, research has been done into ways of keeping the model hidden in these schemes [67, 85, 117, 120].

2.2.3. Other privacy-preserving techniques

Next to using anonymous credentials or encryption, alternative approaches focus on disclosing less attributes or using an additional party in the middle that is trusted more than the access-control party and can provide evidence of the attributes.

Zhang et al. [119] have created a method in which the sensitivity of the attributes is compared to the destination's trust level and based on this, attributes are disclosed or not. Park and Chung [86] use a similar approach: they define an optimal set of attributes that is essential to access a resource, based on the attribute sensitivity and trust level of the resource provider. In [66] multiple decision points are introduced within the access-control system. This means that users can choose which decision point to use and disclose their attributes that are required by this decision point. Another approach by Esmaeeli et al. [46] protects the user attributes by splitting the access-control system in a 'home' and 'destination' party using certificates. Only the first party sees the content of the attributes and makes the access decision. Based on this, the system provides the destination party with a certificate that denotes if access should be granted or not. Lin et al. [70] use attribute fuzzy grouping done by a trusted third party so that a resource provider does not gain any information.

All of these solutions result in the access-control system having access to less attributes, depending on the sensitivity of the attributes and on the trust level of the access-control provider. To be able to handle complex environments, the access-control system should have access to detailed information. Neglecting attributes can therefore decrease the usability of the whole system. Some of the above solutions introduce an additional party. In case the inputs come from more than one party, finding such a third party that is trusted by everyone only shifts the problem: namely, now the trusted party can see all the input attributes of all parties. Next to this, one of the solutions [46] shifts the access decision to the party that owns the attributes, such that this party knows the exact access policy rules.

2.3. Secure Multi-Party Computation

Secure MPC protocols enable multiple parties to compute an agreed-upon function on their inputs in a secure way, where the correctness of the output is proved and the input of every party is kept private [35]. An example is the well-known millionaire's problem of Yao, where two millionaires want to find out who is richer. The function they agree upon to privately evaluate is thus $x_1 <^? x_2$ with x_1, x_2 the amount of money the millionaires own. For constructing secure MPC protocols there are three main building blocks that can be used; garbled circuits, secret sharing and homomorphic encryption. The focus of this thesis lies on homomorphic encryption. Therefore, we first introduce garbled circuits and secret sharing and how these can be used for secure MPC. After this, we discuss secure MPC using homomorphic encryption more elaborately.

GCT
$\mathrm{Enc}_{k_a^0}(\mathrm{Enc}_{k_b^0}(k_c^0))$
$\mathrm{Enc}_{k_a^0}(\mathrm{Enc}_{k_b^1}(k_c^0))$
$\mathrm{Enc}_{k_a^1}(\mathrm{Enc}_{k_b^0}(k_c^0))$
$\mathrm{Enc}_{k_a^1}(\mathrm{Enc}_{k_b^1}(k_c^1))$

Figure 2.1: An example of a garbled table of an AND-gate [1].

Garbled circuits Yao [118] introduced *garbled circuits* in 1986. Garbled circuits can be applied in a two-party setting and the function to be evaluated should be described as a Boolean circuit. One of the parties garbles the Boolean circuit that represents the function, which is then evaluated by the other party [1]. During the circuit garbling, every wire in the circuit is associated with a random encryption key. The input keys are used to encrypt the value of the output key using a symmetric encryption scheme, which can be sent to the other party in the form of a randomised table. As an example, in Figure 2.1 such a table of an AND-gate is shown. The second party can then evaluate the circuit using the encrypted input keys of the other party and his own input keys. The output key is gained by finding out which value in the table can be decrypted. The two parties can then communicate with each other to find the result of the boolean circuit.

Secret sharing *Secret sharing* divides secrets in a set of shares that do not reveal any information on their own. One of the most commonly used secret-sharing schemes is the linear scheme of Shamir [101], which is based on polynomial interpolation. This scheme works by choosing a random polynomial f of degree at most t, such that $f(0) = s$ with s the secret, and distributing the share $f(j)$ to party j. The correctness is proved by the fact that any set of t or less shares does not give away any information about the secret s, but with at least $t+1$ shares the secret can be reconstructed [35]. Evaluating a non-linear function, such as a multiplication, using a linear secret-sharing scheme requires some

additional communication [1]. Therefore, the round complexity of a secret-sharing protocol is linear in the multiplicative depth of the protocol. Multiple secret-sharing protocols have been developed that take place in the preprocessing model where some pre-computation required for non-linear operations is done in an 'offline' phase. An example is the SPDZ protocol [37].

2.3.1. Secure Multi-Party Computation using homomorphic encryption

Homomorphic encryption can be used for secure MPC. In contrast to secret sharing, the communication complexity of MPC with homomorphic encryption is independent of the size of the circuit [1]. Homomorphic encryption allows computations on ciphertexts without accessing the plaintexts. The protocols that make use of homomorphic encryption either make use of FHE or additively-homomorphic encryption. While the last one only allows additive operations on the ciphertexts without accessing the plaintexts, the first allows both multiplications and additions. In contrast to protocols using FHE, the protocols that use additively-homomorphic encryption often require more communication.

When making use of FHE, it is straightforward to construct a secure two-party MPC protocol. Namely, one party can encrypt its input, send it to the other party that evaluates the function on the encryptions. FHE was first introduced in 2009 by Gentry [53]. It is suitable for contexts where one server does all the computations on encrypted input from a client [1]. When more parties are involved, creating a secure MPC protocol becomes more complicated. Ideally, all parties encrypt under their own key, but this makes doing homomorphic operations impossible.

One way to tackle this problem is to use threshold FHE, in which the parties have the same common public key and each has a share of the secret key. A ciphertext encrypted under the public key can only be decrypted when all parties work together. One main result in this area came from Asharov et al. [8] that use the FHE constructions from [14]. This protocol is proven to be secure in the semi-honest setting and works in 3 communication rounds in the Common Random String (CRS) model, that assumes that there is a common random string that is known by all parties.

Mukherjee and Wichs propose the first 2-round MPC protocol that is secure in the CRS model, based on Multi-Key FHE [80]. A *Multi-Key FHE* scheme allows homomorphic operations on ciphertexts encrypted under different keys. As the scheme of Asharov et al. [8], this protocol is proven to be secure in the semi-honest setting. The scheme can be used for on-the-fly MPC. This has the advantage that parties can go online at any moment, send their encrypted input and go offline again.

2.4. Multi-Key Fully Homomorphic Encryption

The first Multi-Key FHE scheme based on a commonly-used hard mathematical problem, namely Learning with Errors (LWE), is from Clear and McGoldrick [31] and is based on the GSW FHE scheme. This scheme was simplified by Mukherjee and Wichs who built the 2-round multi-party computation protocol around it, as mentioned in Section 2.3.1 [80]. These two schemes are static, so no additional keys can be added during the protocol. So, either the protocol has to restart again or some expensive bootstrapping procedure has to take place, that 'refreshes' the ciphertext, in case more parties are added.

In the same year, both Peikert et al. and Brakerski et al. solved this problem by creating schemes that are *multi-hop* for keys [15, 88]. The work of Brakerski et al. minimises the ciphertext size by using an expensive bootstrapping procedure which is necessary for every homomorphic operation. This work uses the extension algorithm from [80] to extend secret key encryptions to more keys, which can then be used for this bootstrapping. Peikert et al.'s [88] solution has bigger ciphertexts but the

homomorphic operations only require some matrix operations like in the GSW FHE scheme. Several other multi-hop multi-key schemes have been proposed as well, extending other FHE schemes from the literature [24–26].

2.5. Private Decision-Tree Evaluation protocols

The research within privacy-preserving methods for decision trees can be split in two parts: either it focuses on creating the decision-tree model while keeping the training data private [72] or it proposes a privacy-preserving evaluation method [13, 39, 40]. Since our work lies in the second category, we discuss the current studies in that field.

The existing PDTE protocols make use of one of the MPC techniques: garbled circuits, secret sharing or homomorphic encryption. Our works follows the works within homomorphic encryption. All existing protocols are placed in the two-party setting where a *server* holds a decision-tree model and a *user* holds all input, while we are looking for a solution in case the input originates from multiple parties. All the protocols described below try to reach the goal of outputting the classification to the user, where the user and the server do not gain any knowledge about the decision tree or the input, respectively. We now introduce all protocols briefly per used technique. All the protocols are secure in the semi-honest setting.

2.5.1. Protocols based on garbled circuits

The first PDTE protocols were introduced by Brickell et al. [18] and Barni et al. [9] in 2007 and 2011. These protocols are based on garbled circuits and were quickly outperformed by some of the schemes using homomorphic encryption in terms of complexity and communication. Tueno et al. [106] propose the first protocol that has sub-linear complexity of the size of the trees by representing the trees as arrays and implementing *oblivious array indexing*, a protocol that makes it possible to access an element of an array where the index stays private, together with garbled circuits. The tree is represented as an array with node elements that contain the threshold value, the indices of the child nodes and the index within the attribute vector. In this protocol the server does not know which node it is evaluating. In each iteration, first an oblivious array indexing protocol is executed in which the corresponding attribute value from the attribute vector is shared between the parties. After this, a garbled circuit evaluates the index of the next child node, which is secretly shared between the parties. The corresponding node is found by using oblivious array indexing (but then with the server holding the tree array). A garbled circuit can be evaluated to check if the new node is a leaf. This results in only D necessary comparisons, with D the depth of the tree. The number of communication rounds and communication sizes are also dependent on the depth of the tree, given by D^2+3D and $\mathcal{O}(D^4)$.

2.5.2. Protocols based on secret sharing

A protocol using secret sharing was constructed by de Cock et al. [40]. It makes use of three subprotocols: an oblivious input selection for each node, a secure distributed-comparison protocol and a secure multiplication-of-integers protocol. For each internal node, the corresponding attribute is compared to a threshold. By using the oblivious input selection, which is similar to oblivious array indexing, the user does not know which attribute is used at each node. The decision tree is then evaluated by expressing it as a polynomial and using secure multiplication and local addition of the secret shares. The final answer is shared between the two parties and the results can be reconstructed by combining the shares of all parties. The user only gets to know the depth of the tree. The secure multiplication protocol is placed in the commodity-based model where pre-distributed data is made

available to the players by a trusted initialiser. This can be replaced by running pre-computations during a setup phase. The complexity is linear in the number of nodes in the tree. The protocol makes use of a comparison protocol that requires $\lceil \log_2 \ell \rceil + 1$ communication rounds with ℓ the bit-size of the input that is compared. Each secure multiplication requires 1 round as well.

Damgard et al. [38] adapt the model of de Cock et al. [40] by using a new secret-sharing based comparison protocol that works over rings instead of field. This results in a small increase in communication costs but a bigger improvement in efficiency. They also show how to make their model actively secure.

2.5.3. Protocols based on homomorphic encryption

The homomorphic encryption protocols make use of FHE or additively-homomorphic encryption. The schemes using additively-homomorphic encryption [59, 104, 114] require more communication rounds than the schemes using FHE [13, 107], but are often computationally less expensive.

In [13] a privacy-preserving decision-tree protocol based on fully homomorphic encryption is proposed, where the decision tree is represented by a polynomial whose output is the result of the classification. Wu et al. [114] improved this protocol two years later by using different techniques such that the protocol only requires homomorphic additions and no homomorphic multiplications. In [104] again an improvement, mostly for large trees, is gained by exploiting the structure of the decision tree. In 2018, Joye et al. proposed another protocol based on the protocol of Wu et al. [59] by optimising the comparison protocol and reducing the number of comparisons, but making the communication and number of rounds dependent on the tree depth. They make use of additively-homomorphic encryption and oblivious transfer. Tueno et al. [107] give a client-server protocol that delegates the evaluation to the server while preserving privacy by evaluating the tree on ciphertexts encrypted under the client's public key. Below, we describe the ideas of all main protocols using homomorphic encryption. The two methods of Tai et al. [104] and Tueno et al. [107] require a low, constant number of rounds.

The protocol of Wu et al. [114] uses a comparison protocol (for the internal nodes) that requires an additive homomorphic encryption scheme. The answers of the comparisons are secretly shared between the two parties. The user sends all its shares in encrypted form to the server, which then sends back the comparison results, after permuting the tree. The user then decrypts and finds the index of the correct leaf node in the permuted tree. By engaging in an oblivious transfer protocol, the client gets to know the value of the leaf. The protocol takes 6 rounds; its complexity is exponential in the depth of the tree, and the number of comparisons is equal to the number of decision nodes.

Tai et al. also use additively-homomorphic encryption [104]. Their protocol takes 4 rounds and the computational complexity is linear in the number of decision nodes. At each decision node the user and the server interact in a comparison protocol [108], the results of which are secretly shared between the two parties. After this, the user sends these shares in encrypted form to the server who then homomorphically evaluates every path. This results in a cost value for each path, which is only equal to zero if it corresponds to the actual evaluation path. These encrypted path costs, together with the encrypted labels, are sent to the user in a random order. The user can then decrypt all costs and find the classification label.

Joye et al. use the first scheme of Wu et al. as a baseline but permute at every level of the tree [59]. The number of comparisons is reduced to one per level, but the number of rounds depend on the depth of the tree. It executes an oblivious-transfer protocol multiple times between the user and server. The workload is reduced for both the user and the server compared to the other schemes, but it is still linear in the number of decision nodes.

Tueno et al. [107] propose the first non-interactive PDTE protocol. It makes use of fully homomorphic encryption. The protocol is initialised by a one-time key generation, after which the user encrypts

his input bit-wise and sends it to the server. The server then computes the decision bit for each comparison node by using the comparison scheme in [28]. In this scheme, the comparison is carried out by homomorphic bit additions and multiplications. The result of each comparison is stored at both child nodes. For every path the comparison bits are aggregated and combined with the classification labels, which results in the encrypted classification label of the tree. The server then sends this to the client who is able to decrypt it. In this way, the protocol only has one round of communication.

3

Theoretical background

In this chapter the background knowledge is discussed that is required for understanding the subsequent chapters. First, the general preliminaries are stated, followed by an explanation of the Learning with Errors problem, a mathematical problem that defines the security of our encryption schemes. After this, a description of two (Multi-Key) Fully Homomorphic Encryption schemes are given that are implemented and used as encryption schemes for our protocols. Lastly, we formally define a decision tree and give a decision-tree evaluation algorithm. We give two methods from the literature for privately evaluating a decision tree. Our protocols build upon these methods.

3.1. Preliminaries

The set of integers modulo q, namely $\{0,1,\ldots,q-1\}$, is denoted by $\mathbb{Z}_q$. A column vector and a matrix are written in bold letters, for example $\mathbf{x} \in \mathbb{Z}^n$ and $\mathbf{M} \in \mathbb{Z}^{n\times m}$. A row vector of length n with elements $x_1, x_2, \ldots, x_n$ is written as $\begin{pmatrix} x_1 & x_2 & \ldots & x_n \end{pmatrix}$. The inner product of two vectors $\mathbf{x}, \mathbf{y}$ is denoted by $\mathbf{x} \cdot \mathbf{y}$. The absolute value of an element $x \in \mathbb{Z}_q$ is given by $|x| = \min\{x, q-x\}$. Let ϕ be a discrete probability distribution over a countable set Ω. Then $\omega \leftarrow \phi$ denotes that ω is sampled at random according to ϕ and $\omega \leftarrow \Omega$ means that it is sampled uniformly at random from the set Ω.

Definition 3.1 ($\mathcal{O}$-notation and $\tilde{\mathcal{O}}$-notation [32])**.** *For a given function $g : N \to \mathbb{R}_+$ where N is an unbounded subset of $\mathbb{R}_+$, the set of functions $\mathcal{O}(g(n))$ is given by $\{f : N \to \mathbb{R}_+ \mid \exists c > 0, n_0 \in \mathbb{R}$ such that $f(n) \leq cg(n) \forall n \geq n_0\}$. The set of functions $\tilde{\mathcal{O}}(g(n))$ is given by $\bigcup_k \mathcal{O}(g(n)\log^k g(n))$, meaning that the logarithmic factors are ignored in comparison to the set $\mathcal{O}(g(n))$.*

Definition 3.2 (Negligible function in n [95])**.** *The function $f(n)$ is a* negligible function in n *if for all $c > 0$ it holds that $\lim_{n\to\infty} n^c f(n) = 0$. Often, a negligible function in n is denoted by* $\mathrm{negl}(n)$.

The encryption schemes that are implemented in this thesis make use of the tensor product and require the definition of a gadget vector with a computable deterministic inverse function. The definitions of these can be found below.

Definition 3.3 (Tensor product [88])**.** *The* tensor product $\mathbf{A} \otimes \mathbf{B}$ *of a matrix $\mathbf{A} \in \mathbb{Z}_q^{n_1 \times m_1}$ with a matrix $\mathbf{B} \in \mathbb{Z}_q^{n_2 \times m_2}$ is the $n_1 n_2 \times m_1 m_2$ matrix that is defined as*

$$\mathbf{A} \otimes \mathbf{B} = \begin{pmatrix} a_{1,1}\mathbf{B} & \ldots & a_{1,m_1}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{n_1,1}\mathbf{B} & \ldots & a_{n_1,m_1}\mathbf{B} \end{pmatrix}.$$

The following property holds: $(\mathbf{A}\otimes\mathbf{B})\cdot(\mathbf{C}\otimes\mathbf{D}) = (\mathbf{AC})\otimes(\mathbf{BD})$ *for any matrices* $\mathbf{A},\mathbf{B},\mathbf{C}$ *and* $\mathbf{D}$ *with the appropriate dimensions.*

Definition 3.4 (Gadget vector and matrix [88])**.** *For any integers* $q \geq 2$*,* $n \geq 1$*, the* gadget (column) vector $\mathbf{g}$ *is defined as*

$$\mathbf{g} = \begin{pmatrix} 1 \\ 2 \\ 4 \\ \vdots \\ 2^{\ell-1} \end{pmatrix} \in \mathbb{Z}_q^{\ell}$$

with $\ell = \lceil \log_2 q \rceil$*. The* gadget matrix $\mathbf{I}_n \otimes \mathbf{g}^\intercal$ *is given by*

$$\mathbf{I}_n \otimes \mathbf{g}^\intercal = \begin{pmatrix} \mathbf{g}^\intercal & & & & \\ & \mathbf{g}^\intercal & & & \\ & & \ddots & & \\ & & & \mathbf{g}^\intercal & \\ & & & & \mathbf{g}^\intercal \end{pmatrix} \in \mathbb{Z}_q^{n\times n\ell}.$$

There exists an efficiently computable function denoted by $(\mathbf{I}_n \otimes \mathbf{g}^{-1})[\cdot]$*, that on input matrix* $\mathbf{M} \in \mathbb{Z}_q^{n\times m'}$ *for any* m'*, gives as output* $(\mathbf{I}_n \otimes \mathbf{g}^{-1})[\mathbf{M}] \in \{0,1\}^{n\ell\times m'}$ *such that* $(\mathbf{I}_n \otimes \mathbf{g}^\intercal)\cdot(\mathbf{I}_n \otimes \mathbf{g}^{-1})[\mathbf{M}] = \mathbf{M}$ *[77]. The function* $(\mathbf{I}_n \otimes \mathbf{g}^{-1})[\cdot]$ *can be seen as the* bit-decomposition function *that decomposes every element of the matrix in a length* ℓ *column* $\{0,1\}$*-vector.*

A common way to describe the security of an asymmetric, or public-key, encryption scheme is by proving that the scheme is Indistinguishable under Chosen Plaintext Attack (IND-CPA) secure. This definition of security is given by a game in which a challenger that holds the secret key engages with an adversary that has certain capabilities. The adversary then tries to break the scheme. The idea behind the game is that, even though the adversary can encrypt many plaintexts, he still does not gain any additional knowledge seeing all those encryptions. Please note here that the encryption is a randomised process, so that an encryption of one specific plaintext is different each time. The official definition of this IND-CPA security is given below.

Definition 3.5 (IND-CPA security [61])**.** *Consider the following security game:*

1. *Challenger: generates the public and private key with security parameter* κ *and sends the public key to the adversary.*
2. *Adversary: can generate any polynomial number of encryptions and can do additional operations in polynomial time.*
3. *Adversary: generates two distinct messages* m_0*,* m_1 *of the same length and sends them to the challenger.*
4. *Challenger: chooses a random bit* $b \in \{0,1\}$*, encrypts the message* m_b *and sends it to the adversary.*
5. *Adversary: is free to do additional encryptions or operations in polynomial time. He guesses a bit* b' *and the output of the experiment is* 1 *of* $b' = b$*, and* 0 *otherwise.*

The asymmetric encryption scheme is IND-CPA secure *if the outcome of the above experiment, for any probabilistic polynomial adversary, is* 1 *with probability at most* $1/2 + \mathrm{negl}(\kappa)$*, with* κ *the security parameter.*

The analyses and proofs regarding the encryption schemes in this thesis make use of definitions and theorems from probability theory. Therefore, we now state the definitions and theorems from this field that are required to understand the subsequent chapters. We assume the reader is familiar with some basic definitions from probability theory. For the definitions of continuous or discrete random variables and probability (density) functions, we refer the reader to [41].

Definition 3.6 (Normal (or Gaussian) probability distribution [69])**.** *The* normal probability distribution *over* $\mathbb{R}$ *is defined by the probability density function*

$$\rho(x) = \frac{e^{-(x-\mu)^2/(2\sigma^2)}}{\sqrt{2\pi\sigma^2}} \tag{3.1}$$

for $x \in \mathbb{R}$ *with* $\mu, \sigma > 0 \in \mathbb{R}$ *the mean and standard deviation respectively. A* normal variable X *is a random variable that is distributed according to the normal probability density function. This is denoted by* $X \sim \mathcal{N}(\mu, \sigma^2)$.

Definition 3.7 (Discrete normal (Gaussian) probability distribution [76])**.** *For* S *a countable set of elements in* $\mathbb{R}$ *and* $\mu, \sigma^2 \geq 0 \in \mathbb{R}$*, the* discrete normal probability function *with mean* μ *and variance* σ^2 *over* S *is defined by the probability distribution*

$$\mathcal{D}_{S,\mu,\sigma^2}(x) = \frac{\rho_{\mu,\sigma^2}(x)}{\sum_{y \in S} \rho_{\mu,\sigma^2}(y)}$$

where $\rho_{\mu,\sigma^2}(x) = \exp\left(-(x-\mu)^2/(2\sigma^2)\right)$ *for* $x \in S$*. When* $\mu = 0$*, the function is denoted by* $\mathcal{D}_{S,\sigma^2}$.

Theorem 3.8 (Normal linear transform theorem [69])**.** *Given a normal variable* $X \sim \mathcal{N}(\mu, \sigma^2)$*, a* linear function $\alpha + \beta X$ *of this normal variable is another normal variable with modified mean and variance distributed according to*

$$\alpha + \beta.\mathcal{N}(\mu, \sigma^2) = \mathcal{N}(\alpha + \beta\mu, \beta^2\sigma^2). \tag{3.2}$$

Definition 3.9 (Statistically independent [69])**.** *Two random variables* X, Y *are* statistically independent *if*

$$\forall x, y : P\left(X = x, Y = y\right) = P(X = x)P(Y = y). \tag{3.3}$$

This means that the realisation of X *does not affect the probability density distribution of* Y *and vice versa.*

In case two normal variables are statistically independent, it can be shown that the distribution of the sum of these two variables again follows a normal distribution, as given by the normal sum theorem below.

Theorem 3.10 (Normal sum theorem [69])**.** *The* sum of two statistically independent normal variables $X \sim \mathcal{N}_1(\mu_1, \sigma_1^2)$ *and* $Y \sim \mathcal{N}_2(\mu_2, \sigma_2^2)$ *is again a normal variable distributed with mean* $\mu_1 + \mu_2$ *and variance* $\sigma_1^2 + \sigma_2^2$*, so*

$$X + Y \sim \mathcal{N}_1(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2).$$

For the difference *between* X *and* Y *it holds that*

$$X - Y \sim \mathcal{N}_1(\mu_1 - \mu_2, \sigma_1^2 + \sigma_2^2).$$

Definition 3.11 (Statistical distance [76])**.** *Let* ϕ_1 *and* ϕ_2 *be two probability functions over the same countable set* S*. The* statistical distance *or the* total variation distance *between* ϕ_1 *and* ϕ_2 *is given by* $\frac{1}{2}\sum_{s \in S}\left|\phi_1(s) - \phi_2(s)\right|$*. We denote this distance by* $d(\phi_1, \phi_2)$.

Clearly, the statistical distance ranges in $[0, 1]$, since $\sum_{s \in S} \phi_1(s) = 1$ and $\sum_{s \in S} \phi_2(s) = 1$. It can also be noted that this statistical distance is a metric since for all ϕ_1, ϕ_2 and ϕ_3 probability functions on a

countable set S, the following conditions are satisfied: (i) $d(\phi_1,\phi_2) \geq 0$, (ii) $d(\phi_1,\phi_2) = 0 \Leftrightarrow \phi_1 = \phi_2$, (iii) $d(\phi_1,\phi_2) = d(\phi_2,\phi_1)$ and (iv) $d(\phi_1,\phi_3) \leq d(\phi_1,\phi_2) + d(\phi_2,\phi_3)$. The last condition holds since

$$d(\phi_1,\phi_3) = \frac{1}{2}\sum_{s\in S}\left|\phi_1(s) - \phi_3(s)\right| = \frac{1}{2}\sum_{s\in S}\left|\phi_1(s) - \phi_2(s) + \phi_2(s) - \phi_3(s)\right|$$

$$\leq \frac{1}{2}\sum_{s\in S}\left|\phi_1(s) - \phi_2(s)\right| + \frac{1}{2}\sum_{s\in S}\left|\phi_2(s) - \phi_3(s)\right| = d(\phi_1,\phi_2) + d(\phi_2,\phi_3).$$

3.2. Learning with Errors

The LWE problem is a mathematical problem underlying the security of many cryptographic systems. The encryption schemes we implemented during this research are also based on the LWE problem. Therefore, this problem plays an important role in the security proofs of this thesis. The LWE problem is closely related to lattice problems. Below, the definitions of lattices and two lattice problems are touched upon, after which the LWE problem is defined together with a hardness theorem. The lattice-related definitions and problems can be found in [32, 77, 90].

Definition 3.12 (Lattice)**.** *A subset Λ of $\mathbb{R}^m$ is a* lattice *if there exists a set of n linearly independent vectors $\{\mathbf{b}_1,\ldots,\mathbf{b}_n\}$ in $\mathbb{R}^m$ such that $\Lambda = \{\sum_{i=1}^n x_i\mathbf{b_i} \mid x_i \in \mathbb{Z}\}$. The set of vectors $\{\mathbf{b}_1,\ldots,\mathbf{b}_n\}$ is called a* basis *of Λ. The n, m are the* rank *and* dimension *of the lattice, respectively. The lattice is* full-rank *if and only if $m = n$.*

For ease of notation, sometimes a basis is denoted by a matrix $\mathbf{B} = \begin{pmatrix}\mathbf{b}_1 & \mathbf{b}_2 & \ldots & \mathbf{b}_n\end{pmatrix}$. The lattice generated by $\mathbf{B}$, denoted by $\Lambda(\mathbf{B})$, refers to the lattice with the columns of $\mathbf{B}$ as a basis.

Definition 3.13 (Shortest non-zero vector of a lattice)**.** *Let $\Lambda \subseteq \mathbb{R}^m$ be a lattice. The* shortest non-zero vector *is the vector of $\Lambda \setminus \{\mathbf{0}\}$ with the smallest Euclidean norm. We denote the length of this vector by $\lambda_1(\Lambda)$.*

Such a shortest non-zero vector does exist since a lattice is discrete. We now define two lattice problems that have hardness results that are related to the hardness of LWE. The first problem is a *promise* problem which means that the outcome sets do not exhaust all possible outcomes and when the input does not belong to one of the output sets, there are no requirements on the output of the algorithm.

Problem 3.14 (The γ-Gap Shortest Vector Problem (GapSVP$_\gamma$))**.** *Given $\mathbf{B}$ a basis generating a lattice of dimension m and $\gamma \geq 1$ a function of the dimension, the GapSVP$_\gamma$ promise problem is defined as follows. For $d > 0$ a real number, decide between $\lambda_1(\Lambda(\mathbf{B})) \leq d$ and $\lambda_1(\Lambda(\mathbf{B})) > \gamma d$.*

Problem 3.15 (The Bounded Distance Decoding (BDD) problem)**.** *Given $\mathbf{B}$ a basis generating a lattice of dimension m, the BDD problem is defined as follows. Let $\mathbf{x} \in \mathbb{R}^m$ be a vector whose Euclidean distance to the lattice $\Lambda(\mathbf{B})$ is at most d for $d > 0$ a real number, find the vector in $\Lambda(\mathbf{B})$ closest to $\mathbf{x}$.*

Below, we state the LWE problem and give an important result which shows that solving LWE is at least as hard as solving a worst-case lattice problem.

Problem 3.16 (Decisional and Search LWE [87])**.** *Let n and $q \geq 2$ be positive integers and χ a probability distribution over $\mathbb{Z}_q$. The* Decisional LWE$_{n,m,q,\chi}$ problem *is defined as follows. Given m independent samples drawn from one of the two distributions below, decide from which one:*

1. $(\mathbf{a}, b)$ *uniformly from* $\mathbb{Z}_q^{n+1}$,
2. $(\mathbf{a}, b) \in \mathbb{Z}_q^{n+1}$ *with* $\mathbf{a} \leftarrow \mathbb{Z}_q^n$ *uniformly and* $b = \boldsymbol{a}\cdot\boldsymbol{s} + e \mod q$ *with* $e \leftarrow \chi$ *and* $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ *uniformly.*

Here, $\mathbf{s} \in \mathbb{Z}_q^n$ *is the same for every sample from the second distribution. The* Search $\text{LWE}_{n,m,q,\chi}$ problem *is to recover the secret* $\mathbf{s}$ *given* m *independent samples only from the second distribution. The problems are solved if the correct answer is given with probability exponentially (in* n*) close to* 1.

The Search LWE problem is closely related to the BDD problem [96]. Let $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ and denote the m LWE samples by $(\mathbf{a}_i, b_i)$ with $\mathbf{a}_i \leftarrow \mathbb{Z}_q^n$ uniformly and $b_i = \mathbf{a}_i \cdot \mathbf{s} + e_i \mod q$ with $e_i \leftarrow \chi$. For $\mathbf{A} = \begin{pmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_m \end{pmatrix}$ and $\mathbf{b}$, $\mathbf{e}$ the column vectors containing all b_i and e_i respectively, it holds that $\mathbf{b} = \mathbf{A}^\intercal \mathbf{s} + \mathbf{e} \mod q$. So, $\mathbf{b}$ can be seen as an element of the lattice $\{\mathbf{z} \in \mathbb{Z}^m | \mathbf{z} = \mathbf{A}^\intercal \mathbf{s} \mod q\}$ subject to an error. Finding $\mathbf{s}$ is then related to solving a BDD on this lattice with $\mathbf{b}$ as input vector.

It is shown that the LWE problem is as hard as a worst-case lattice problem, using a quantum reduction. Specifically, solving a *random* LWE instance implies an efficient quantum algorithm for *all* instances of this lattice problem. This quantum reduction was given by Regev in 2009, in the following main theorem:

Theorem 3.17 (Theorem 1.1 from [95])**.** *Let* n, q *be integers and* $\alpha \in (0,1)$ *a real number be such that* $\alpha q > 2\sqrt{n}$*. Take* χ *as the distribution* $\bar{\Psi}_q^\alpha$ *(see definition below). If there exists an algorithm that solves the Search* $\text{LWE}_{n,m,q,\chi}$ *problem in polynomial (in* n*) time then there exists a polynomial time (in* n*) quantum algorithm that approximates any* GapSVP_γ *problem on arbitrary* n*-dimensional lattices to within* $\gamma = \tilde{\mathcal{O}}(n/\alpha)$.

Here, $\bar{\Psi}_q^\alpha$ is defined as the distribution on $\mathbb{Z}_q$ gained by sampling from a Gaussian distribution with mean 0 and standard deviation $\alpha/\sqrt{2\pi}$ and rounding to the closest integer modulo q. A brief description of the proof of Theorem 3.17 is given in the appendix in Section A.1.

According to theorem 3.17, solving a random $\text{LWE}_{n,m,q,\chi}$ instance in polynomial time is at least as hard as solving any n-dimensional GapSVP_γ instance within $\gamma = \tilde{\mathcal{O}}(n/\alpha)$ by a polynomial-time quantum algorithm. Combining this with the fact that the best known classical algorithms for GapSVP_γ with polynomial approximation factors run in exponential time and that there are no known quantum algorithms that outperform the classical ones, the average-case hardness of the Search $\text{LWE}_{n,m,q,\chi}$ problem is established [95, 96].

Applebaum et al. [5] showed that this hardness still holds in case the elements of secret $\mathbf{s}$ are drawn from the same distribution as the error distribution χ. Furthermore, Regev [95] proved that Decisional and Search LWE are equivalent, given a *prime* q of size polynomially bounded in n. Thereafter, this was shown for essentially all values for q [87]. Accordingly, the specification 'Decisional' or 'Search' is often neglected in the context of the LWE problem.

Important to note here is that the above reduction is not tight; the polynomial time of the quantum algorithm is much higher than the polynomial time of solving the corresponding LWE problem [22]. This is called a tightness-gap. Therefore, for the same polynomial time, the parameters of the LWE problem are higher than the parameters of the lattice problem. The exact tightness-gap of the reduction is not known. Therefore, the exact parameter setting is often chosen such that all possible attacks take at least 2^λ bit operations, with λ a security parameter. Albrecht et al. [3] created a model that can estimate the running time of possible algorithms that can break a specific LWE instance. The security parameter λ is normally chosen to be equal to 110 to guarantee security [24].

3.3. (Multi-Key) Fully Homomorphic Encryption schemes

Homomorphic encryption schemes allow mathematical operations on ciphertexts without first decrypting them. The result is an encryption of the outcome that the mathematical operations would give when applied to plaintexts. This property makes these schemes useful for outsourcing computations to a cloud when it is important to preserve the privacy of the data. Mathematically, a homomorphic encryption scheme is defined by two groups $(C,\cdot),(P,*)$, a key generation scheme that generates a key pair (pk, sk), an encryption procedure Enc and a decryption procedure Dec. Here, P and C are the plaintext and ciphertext space respectively. To be (partially) homomorphic it should hold that for any $m_1, m_2 \in P$ and $c_1, c_2 \in C$ with $m_1 = Dec_{\text{sk}}(c_1)$ and $m_2 = Dec_{\text{sk}}(c_2)$, there exists an efficient operation f on C such that $Dec_{\text{sk}}(f(c_1,c_2)) = m_1 * m_2$ [61]. Typically, the operator $*$ is either addition or multiplication. A *Fully* Homomorphic Encryption scheme is homomorphic regarding both addition and multiplication.

The first FHE scheme was given by Gentry in 2009 [53]. FHE schemes are able to evaluate arbitrary circuits of unbounded depth and can therefore be used for running *any* program by an untrusted party. Below we state the formal definition of a FHE scheme to demonstrate its functionality. A detailed description of a scheme is given later, together with its homomorphic operations, correctness and security proofs. In the definition below, it is assumed that the plaintext space is $\{0,1\}$, i.e. only bits can be encrypted and decrypted.

Definition 3.18 (Fully Homomorphic Encryption scheme [8])**.** *A public-key* Fully Homomorphic Encryption *(FHE) scheme is a set of 4 algorithms* {*FHE.Keygen, FHE.Enc, FHE.Dec, FHE.Eval*}*, defined as follows:*

- *FHE.Keygen*$(1^\kappa) \to$ (pk, sk)*: Outputs a public encryption key* pk *and a secret decryption key* sk *on input the security parameter* κ*.*
- *FHE.Enc*$_{\text{pk}}(\mu) \to c$*: Encrypts a bit* $\mu \in \{0,1\}$ *under public key* pk*. It outputs a ciphertext* c*.*
- *FHE.Dec*$_{\text{sk}}(c) \to \mu^*$*: Decrypts the ciphertext* c *using* sk*. It outputs plaintext bit* $\mu^* \in \{0,1\}$*.*
- *FHE.Eval*$(f, c_1, \ldots, c_l) \mapsto c_f$*: The* homomorphic evaluation algorithm *takes as input a boolean circuit* $f : \{0,1\}^l \to \{0,1\}$ *and a set of* l *ciphertexts* $c_1, \ldots c_l$*. It outputs the result ciphertext* c_f *which is the encryption of* f (*FHE.Dec*$_{\text{sk}}(c_1), \ldots,$ *FHE.Dec*$_{\text{sk}}(c_l)$)*.*

The FHE scheme is shown to be correct if $\text{FHE.Dec}_{\text{sk}}\left(\text{FHE.Enc}_{\text{pk}}(\mu)\right) = \mu$ for any message $\mu \in \{0,1\}$ and sk, pk a generated key-pair. Also, it should hold that

$$\text{FHE.Dec}_{\text{sk}}\left(\text{FHE.Eval}(f, c_1, \ldots, c_l)\right) = \mu_f = f\left(\text{FHE.Dec}_{\text{sk}}(c_1), \ldots, \text{FHE.Dec}_{\text{sk}}(c_l)\right)$$

with f a boolean circuit with $f(\mu_1, \ldots \mu_l) = \mu_f$ and $c_1, \ldots, c_l$ the encryptions of $\mu_1, \ldots \mu_l \in \{0,1\}$ plaintexts. We require the FHE scheme to be IND-CPA secure as defined in Definition 3.5.

While a normal FHE scheme allows homomorphic operations on its ciphertexts when encrypted under the same key, a *Multi-Key* FHE scheme allows this even when the ciphertexts are encrypted under *different* keys. This is done by 'expanding' the ciphertext to multiple keys. The formal definition of a Multi-Key FHE scheme and its functionality is given by:

Definition 3.19 (Multi-Key Fully Homomorphic Encryption scheme [80])**.** *A public-key* Multi-Key Fully Homomorphic Encryption *scheme is a set of 5 algorithms* {*MKFHE.Keygen,MKFHE.Enc, MKFHE.Expand, MKFHE.Eval, MKFHE.Dec*}*, defined as follows:*

- *MKFHE.Keygen*$(1^\kappa) \to$ (pk, sk)*: Outputs a public encryption and extension keys denoted by* pk *and a secret decryption key* sk *using the input security parameter* κ*.*

- *MKFHE.Enc*$_{\text{pk}}(\mu) \rightarrow c$*: Encrypts a bit* $\mu \in \{0,1\}$ *under public key* pk. *Outputs ciphertext* c.
- *MKFHE.Expand*$_{\text{pk}_1,\ldots,\text{pk}_k}(i,c) \rightarrow \hat{c}$*: Given a sequence of* k *public keys of different parties and a fresh ciphertext* c *encrypted under the* i*-th public key* pk_i*, it outputs an expanded ciphertext* $\hat{c}$ *encrypted under all* k *public keys.*
- *MKFHE.PartDec*$_{\text{sk}_i}(\hat{c}) \rightarrow p_i$*: Partially decrypts the extended ciphertext* $\hat{c}$ *that is an encryption under* $\text{pk}_1,\ldots,\text{pk}_k$ *using a secret key* sk_i. *The output is a partial decryption* p_i.
- *MKFHE.FinDec*$(p_1,\ldots,p_k) \rightarrow \mu^*$ *Combines partial decryptions* $p_1,\ldots,p_k$ *to find an output plaintext* $\mu^* \in \{0,1\}$.
- *MKFHE.Eval*$(f,\hat{c}_1,\ldots,\hat{c}_l) \rightarrow \hat{c}_f$*: The* homomorphic evaluation algorithm *takes as input a boolean circuit* $f : \{0.1\}^l \mapsto \{0,1\}$ *and a set of* l *expanded ciphertexts* $\hat{c}_1,\ldots\hat{c}_l$*. It outputs the result ciphertext* $\hat{c}_f$ *which is the encryption of* $f(\mu_1,\ldots\mu_l)$ *where* $\mu_1,\ldots\mu_l$ *are the plaintext values of the extended ciphertexts* $\hat{c}_1,\ldots\hat{c}_l$.

As for the normal FHE scheme, a Multi-Key FHE scheme is shown to be correct if

$$\text{MKFHE.FinDec}\left(\text{MKFHE.PartDec}_{\text{sk}_1}(\hat{c}),\ldots,\text{MKFHE.PartDec}_{\text{sk}_k}(\hat{c})\right) = \mu$$

for any message $\mu \in \{0,1\}$ encrypted by any party i and extended to $\text{pk}_1,\ldots,\text{pk}_k$ with as result

$$\hat{c} = \text{MKFHE.Expand}_{\text{pk}_1,\ldots,\text{pk}_k}\left((i,\text{MKFHE.Enc}_{\text{pk}_i}(\mu)\right).$$

We also require the Multi-Key FHE scheme to be IND-CPA secure. In all FHE schemes, homomorphic operations on ciphertexts result in an increase in ciphertext noise. Therefore, the parameters of the scheme should be set high enough such that the final ciphertext after the complete evaluation can still be decrypted.

In this thesis, the GSW FHE scheme and its multi-key extension are used and implemented. The GSW scheme is conceptually simpler than other schemes and has a natural homomorphic addition and multiplication procedure [54]. This scheme was transformed to a single-hop multi-key scheme in [80]. The meaning of single-hop is that the extension to multiple keys can only be done once for a set of public keys known beforehand. Therefore, multi-hop solutions were proposed, in which ciphertexts can again be extended to additional keys after homomorphic computations have been done already. We use the scheme of Peikert et al. as this is the most feasible multi-hop scheme that extends the GSW scheme [88]. Below, we describe these schemes in detail. We use the simplified notation of Alperin-Sheriff et al. [4]. The security proofs and analysis of the correctness and the homomorphic operations are given in subsequent chapters as this is not done elaborately in the literature, but is necessary for implementation. Namely, the amount of noise that is propagated during our protocols requires the parameters of our scheme to be high enough such that decryption is still possible.

3.3.1. GSW Fully Homomorphic Encryption scheme

Let n be the dimension, $\alpha \in (0,1)$ a real number, q the modulus and $\chi = \bar{\Psi}_q^\alpha$ over $\mathbb{Z}_q$ the probability distribution of the *noise* gained by sampling from a Gaussian distribution with mean 0 and standard deviation $\alpha/\sqrt{2\pi}$ and rounding to the closest integer modulo q. Also choose another dimension parameter value m which is specified later on and let $\ell = \lceil \log_2 q \rceil$. It is assumed that a random common public matrix is defined, namely $\mathbf{B} \leftarrow \mathbb{Z}_q^{(n-1)\times m}$ uniformly. The gadget vector $\mathbf{g}$ is given in Definition 3.4. The GSW scheme consists then of the following set of algorithms [4, 54]:

- *(Key generation)* The secret key $\mathbf{t}$ is chosen as $\mathbf{t} = \begin{pmatrix} -\mathbf{s}^\intercal & 1 \end{pmatrix}^\intercal \in \mathbb{Z}_q^n$ where $\mathbf{s} \leftarrow \chi^{n-1}$ uniformly. The public key is defined as the matrix

$$\mathbf{A} = \begin{pmatrix} \mathbf{B} \\ \mathbf{b} \end{pmatrix} \in \mathbb{Z}_q^{n\times m}$$

where $\mathbf{b} = \mathbf{s}^\intercal \mathbf{B} + \mathbf{e}^\intercal$ with $\mathbf{e} \leftarrow \chi^m$. It holds that $\mathbf{t}^\intercal \mathbf{A} = -\mathbf{s}^\intercal \mathbf{B} + \mathbf{b} = -\mathbf{s}^\intercal \mathbf{B} + \mathbf{s}^\intercal \mathbf{B} + \mathbf{e}^\intercal = \mathbf{e}^\intercal$.

- *(Encryption)* First, a random binary matrix is chosen $\mathbf{R} \leftarrow \{0,1\}^{m \times n\ell}$ uniformly. The encryption of a message $\mu \in \mathbb{Z}_q$ is then given by the ciphertext matrix $\mathbf{C} = \mathbf{AR} + \mu(\mathbf{I}_n \otimes \mathbf{g}^\intercal) \in \mathbb{Z}_q^{n \times n\ell}$.
- *(Decryption)* We give here the decryption procedure for small plaintext space $\{0,1\}$, since we build upon this procedure in the rest of our thesis. The procedure for plaintext space $\mathbb{Z}_q$ can be found in [54]. Let $\mathbf{c}$ be the penultimate column of $\mathbf{C}$ and calculate the inner product $\mu' = \mathbf{c} \cdot \mathbf{t}$. If $\mu' \mod q$ is closer to 0 in $\mathbb{Z}_q$ than to $2^{\ell-2}$ return 0, else return 1. Please note here that $q-1$ is also very close to 0 since $q \mod q \equiv 0$.
- *(Homomorphic Addition)* Homomorphic addition of two ciphertexts $\mathbf{C_1}, \mathbf{C_2} \in \mathbb{Z}_q^{n \times n\ell}$ is done by calculating $\mathbf{C_1} + \mathbf{C_2} \in \mathbb{Z}_q^{n \times n\ell}$. We denote this addition by $\mathbf{C_1} \boxplus \mathbf{C_2}$. The result is an encryption of $\mu_1 + \mu_2$ where μ_i is the plaintext of ciphertext $\mathbf{C}_i$ for $i = 1,2$.
- *(Homomorphic Multiplication)* Homomorphic multiplication of two ciphertexts $\mathbf{C_1}, \mathbf{C_2} \in \mathbb{Z}_q^{n \times n\ell}$ is done by calculating $\mathbf{C_1}(\mathbf{I}_n \otimes \mathbf{g}^{-1})(\mathbf{C_2}) \in \mathbb{Z}_q^{n \times n\ell}$. We denote this multiplication by $\mathbf{C_1} \boxdot \mathbf{C_2}$. The result is an encryption of $\mu_1 \cdot \mu_2$ where μ_i is the plaintext of ciphertext $\mathbf{C}_i$ for $i = 1,2$.

Intuitively, when $\mathbf{t}^\intercal \mathbf{C} = \mathbf{t}^\intercal \mathbf{AR} + \mu \mathbf{t}^\intercal (\mathbf{I}_n \otimes \mathbf{g}^\intercal) = \mathbf{e}^\intercal \mathbf{R} + \mu(\mathbf{t}^\intercal \otimes \mathbf{g}^\intercal)$ for some $\mathbf{e}$ with small enough values (we make this amount explicit in the next chapters), it holds that, for $\mathbf{c}$ the penultimate column of $\mathbf{C}$, $\mu' = \mathbf{c} \cdot \mathbf{t} = \mu 2^{\ell-2} + e'$ for some e' again small. So indeed, the decryption procedure outputs the right value for a small enough e'.

Regarding the homomorphic operations, take $\mathbf{t}^\intercal \mathbf{C}_1 = \mathbf{e}_1^\intercal \mathbf{R}_1 + \mu_1(\mathbf{t}^\intercal \otimes \mathbf{g}^\intercal)$ and $\mathbf{t}^\intercal \mathbf{C}_2 = \mathbf{e}_2^\intercal \mathbf{R}_2 + \mu_2(\mathbf{t}^\intercal \otimes \mathbf{g}^\intercal)$ for some $\mathbf{e}_1$ and $\mathbf{e}_2$ with small enough values. After a homomorphic addition we find

$$\mathbf{t}^\intercal (\mathbf{C}_1 + \mathbf{C}_2) = \mathbf{e}_1^\intercal \mathbf{R}_1 + \mathbf{e}_2^\intercal \mathbf{R}_2 + \left(\mu_1 + \mu_2\right)(\mathbf{t}^\intercal \otimes \mathbf{g}^\intercal).$$

Applying a homomorphic multiplication gives

$$\begin{aligned}\mathbf{t}^\intercal \left(\mathbf{C_1}(\mathbf{I}_n \otimes \mathbf{g}^{-1})(\mathbf{C_2})\right) &= \mathbf{e}_1^\intercal \mathbf{R}_1 (\mathbf{I}_n \otimes \mathbf{g}^{-1})(\mathbf{C_2}) + \mu_1 (\mathbf{t}^\intercal \otimes \mathbf{g}^\intercal)(\mathbf{I}_n \otimes \mathbf{g}^{-1})(\mathbf{C_2}) \\ &= \mathbf{e}_1^\intercal \mathbf{R}_1 (\mathbf{I}_n \otimes \mathbf{g}^{-1})(\mathbf{C_2}) + \mu_1 \mathbf{e}_2^\intercal \mathbf{R}_2 + \mu_1 \mu_2 (\mathbf{t}^\intercal \otimes \mathbf{g}^\intercal).\end{aligned}$$

The ciphertexts can therefore be decrypted towards $\mu_1 + \mu_2$ or $\mu_1 \mu_2$ for small enough noise values since $\mathbf{R}_1, \mathbf{R}_2$ and $(\mathbf{I}_n \otimes \mathbf{g}^{-1})(\mathbf{C_2})$ are binary matrices.

3.3.2. Multi-Key Fully Homomorphic Encryption scheme

In this section, the multi-hop Multi-Key FHE scheme of Peikert and Shiehian [88] is described. Now the key generation is more elaborate, since every public key contains an encryption of its secret key and an encryption of the randomness used for these encryptions. The parameter setup, decryption and homomorphic operations are the same as in the previous section. However, a different random common public matrix is now defined: $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$ uniformly. The extension procedure translates a ciphertext under one specific key to a ciphertext of the same plaintext, that can only be decrypted by the concatenation of the old secret key and the new secret key. In order to extend the ciphertext for k keys, this procedure should be applied multiple times; once per new key. The homomorphic operations can be done in the similar way as for the GSW FHE scheme. The Multi-Key FHE scheme of Peikert et al. includes the following additional procedures, next to the ones introduced in the previous section [88]:

- *(Key generation)* The secret key $\mathbf{t}$ is chosen as $\mathbf{t} = \begin{pmatrix} -\mathbf{s} & 1 \end{pmatrix}^\intercal \in \mathbb{Z}_q^n$ where $\mathbf{s} \leftarrow \chi^{n-1}$ uniformly. The public *extension* key is set to $(\mathbf{b}, \mathbf{P}, \mathbf{D})$, that is defined as follows:
  - $\mathbf{b}^\intercal = \mathbf{t}^\intercal \mathbf{A} + \mathbf{e}_\mathbf{b}^\intercal$ with $\mathbf{e_b} \leftarrow \chi^m$

- $\mathbf{P} = \mathbf{AR} + (\mathbf{I}_n \otimes \mathbf{t}^\intercal \otimes \mathbf{g}^\intercal) \in \mathbb{Z}_q^{n \times n^2 \ell}$ with $\mathbf{R} \leftarrow \{0,1\}^{m \times n^2 \ell}$ uniformly random. This is the commitment to the secret key $\mathbf{t}$.
- $\mathbf{D} = \bar{\mathbf{D}} + (\mathbf{R} \otimes \mathbf{g} \otimes \mathbf{e}_n)$ with $\mathbf{e}_n \in \{0,1\}^n$ the n-th standard basis vector, $\mathbf{R}$ the random matrix used for generating $\mathbf{P}$ and $\bar{\mathbf{D}}$ defined as

$$\bar{\mathbf{D}} = \begin{pmatrix} \mathbf{d}_1 & \dots & \mathbf{d}_{(n^2\ell-1)ml+1} \\ \vdots & \ddots & \vdots \\ \mathbf{d}_{ml} & \dots & \mathbf{d}_{(n^2\ell)ml} \end{pmatrix} \in \mathbb{Z}_q^{nml \times n^2\ell}$$

where

$$\mathbf{d}_j = \begin{pmatrix} \mathbf{a}_j \\ \mathbf{s}^\intercal \cdot \mathbf{a}_j + e_j \end{pmatrix}$$

with $e_j \leftarrow \chi$ and $\mathbf{a}_j \leftarrow \mathbb{Z}_q^{n-1}$ uniformly. Clearly, the d_j are $(n^2\ell)ml$ independent LWE samples. It can be seen that

$$(\mathbf{I}_{ml} \otimes \mathbf{t}^\intercal) \cdot \bar{\mathbf{D}} = \begin{pmatrix} e_1 & \dots & e_{(n^2\ell-1)ml+1} \\ \vdots & \ddots & \vdots \\ e_{ml} & \dots & e_{(n^2\ell)ml} \end{pmatrix} := \mathbf{E_D} \in \mathbb{Z}_q^{ml \times n^2\ell}$$

such that $(\mathbf{I}_{ml} \otimes \mathbf{t}^\intercal) \cdot \mathbf{D} = \mathbf{E_D} + (\mathbf{I}_{ml} \otimes \mathbf{t}^\intercal) \cdot (\mathbf{R} \otimes \mathbf{g} \otimes \mathbf{e}_n) = \mathbf{E_D} + (\mathbf{I}_{ml}(\mathbf{R} \otimes \mathbf{g})) \otimes (\mathbf{t}^\intercal \mathbf{e}_n) = \mathbf{E_D} + (\mathbf{R} \otimes \mathbf{g})$. $\mathbf{D}$ can be seen as the encryption of the random matrix $\mathbf{R}$ under the secret key.

– *(Encryption)* We now describe the encryption procedure under secret key $\mathbf{t} = \begin{pmatrix} -\mathbf{s} & 1 \end{pmatrix}^\intercal$. Sample nl independent LWE samples

$$\mathbf{c}_j = \begin{pmatrix} \mathbf{a}_j \\ \mathbf{s}^\intercal \cdot \mathbf{a}_j + e_j \end{pmatrix}$$

with $e_j \leftarrow \chi$ and $\mathbf{a}_j \leftarrow \mathbb{Z}_q^{n-1}$ uniformly. Concatenate these samples as follows: $\bar{\mathbf{C}} = \begin{pmatrix} \mathbf{c}_1 & \dots & \mathbf{c}_{n\ell} \end{pmatrix} \in \mathbb{Z}_q^{n \times n\ell}$. The encryption of $\mu \in \mathbb{Z}_q$ is then given by: $\mathbf{C} = \bar{\mathbf{C}} + \mu(\mathbf{I}_n \otimes \mathbf{g}^\intercal) \in \mathbb{Z}_q^{n \times n\ell}$. Notice that $\mathbf{t}^\intercal \mathbf{C} = \begin{pmatrix} e_1 & \dots & e_{n\ell} \end{pmatrix} + \mu(\mathbf{t}_i^\intercal \otimes \mathbf{g}^\intercal)$.

– *(Extension to new keys)* Here we demonstrate how we extend a (multi-key) ciphertext to a new key, so we assume we have a ciphertext that is an encryption under the keys of k parties already. We denote the secret key and public extension key of party i by $\mathbf{t}_i$ and $(\mathbf{b}_i, \mathbf{P}_i, \mathbf{D}_i)$. Let $\mathbf{C} \in \mathbb{Z}_q^{nk \times nk\ell}$ be a ciphertext that encrypts μ under a key

$$\mathbf{t} = \begin{pmatrix} \mathbf{t}_1 \\ \vdots \\ \mathbf{t}_k \end{pmatrix} \in \mathbb{Z}_q^{nk}$$

for $k \geq 1$ the number of parties. We want to extend this ciphertext to an additional key, namely $\mathbf{t}_{\text{new}}$, of party 'new'. Define

$$\mathbf{Y}_1 := \mathbf{I}_k \otimes \mathbf{P}_{\text{new}} = (\mathbf{I}_k \otimes \mathbf{AR}_{\text{new}}) + \left(\mathbf{I}_{nk} \otimes \mathbf{t}_{\text{new}}^\intercal \otimes \mathbf{g}^\intercal\right) \in \mathbb{Z}_q^{nk \times n^2 k\ell}$$

and

$$\mathbf{Y}_2 := \left(\left(\left(\mathbf{I}_{km} \otimes \mathbf{g}^{-1}\right)[-\mathbf{b}]\right)^\intercal \otimes \mathbf{I}_n\right) \cdot (\mathbf{I}_k \otimes \mathbf{D}_{\text{new}}) \in \mathbb{Z}_q^{n \times n^2 k\ell}$$

where $\mathbf{b}$ is the concatenation of all public keys $\mathbf{b}_i$ for $i \in \{1, \dots, k\}$ all parties connected to ciphertext $\mathbf{C}$. So for the creation of $\mathbf{Y}_1$ and $\mathbf{Y}_2$, all public extension keys of the new party are used, namely $(\mathbf{b}_{\text{new}}, \mathbf{P}_{\text{new}}, \mathbf{D}_{\text{new}})$. Set

$$\mathbf{Y} = \begin{pmatrix} \mathbf{Y}_1 \\ \mathbf{Y}_2 \end{pmatrix} \in \mathbb{Z}_q^{n(k+1) \times n^2 k\ell}$$

and calculate the matrix

$$\mathbf{X} = \mathbf{Y}\left((\mathbf{I}_{n^2k} \otimes \mathbf{g}^{-1})[\hat{\mathbf{C}} \otimes \mathbf{I}_n]\right)\Pi \in \mathbb{Z}_q^{n(k+1)\times n\ell}$$

where $\Pi \in \{0,1\}^{n\ell \times n\ell}$ the permutation matrix such that $(\mathbf{g}^\top \otimes \mathbf{t}_{\text{new}}^\top)\Pi = (\mathbf{t}_{\text{new}}^\top \otimes \mathbf{g}^\top)$ and $\hat{\mathbf{C}} = \mathbf{C}\cdot(\mathbf{e}_{nk} \otimes \mathbf{I}_l) \in \mathbb{Z}_q^{nk\times l}$ as the last ℓ columns of the ciphertext $\mathbf{C}$. Now output

$$\mathbf{C}' = \begin{pmatrix} \mathbf{C} & \mathbf{X} \\ \mathbf{0} & \end{pmatrix} \in \mathbb{Z}_q^{n(k+1)\times n(k+1)\ell}.$$

Let $\mathbf{t}$ be the concatenation of the k secret keys of the parties associated to the input ciphertext $\mathbf{C}$ that encrypts μ under $\mathbf{t}$. Then $\mathbf{C}'$ is the ciphertext that extends $\mathbf{C}$ with one extra key $\mathbf{t}_{\text{new}}$ and is therefore an encryption of μ under

$$\mathbf{t}' = \begin{pmatrix} \mathbf{t} \\ \mathbf{t}_{\text{new}} \end{pmatrix}.$$

This is due to the fact that the matrices and public keys are constructed in such a way that $\mathbf{t}'^\top\mathbf{X} \approx \mu(\mathbf{t}_{\text{new}}^\top \otimes \mathbf{g}^\top) + \mathbf{e}_X^\top$. So, $\mathbf{t}'^\top\mathbf{C}' = \left(\mathbf{t}^\top\mathbf{C} \quad \mathbf{t}'^\top\mathbf{X}\right) \approx \mu\left((\mathbf{t}^\top \otimes \mathbf{g}^\top) \quad (\mathbf{t}_{\text{new}}^\top \otimes \mathbf{g}^\top)\right) = \mu(\mathbf{t}'^\top \otimes \mathbf{g}^\top)$ which corresponds to a correct extension of the ciphertext with one additional key.

3.4. Decision trees

In this section we give the definition of a decision tree and define how it can be evaluated given an input. First, we formally define a graph and a tree.

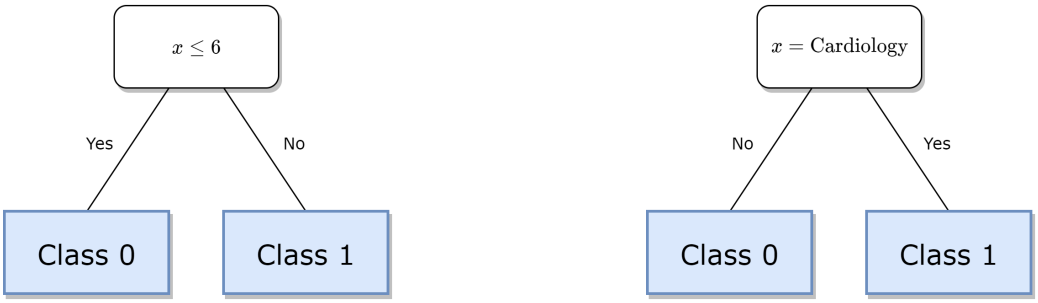
Definition 3.20 (Graph [33])**.** *A* graph *G consists of disjoint finite sets V of vertices and E of edges where every edge is an unordered pair of nodes (informally, the edge connects these two nodes). The graph is denoted by $G = (V, E)$.*

We write an edge e as $e = (\text{start}_e, \text{end}_e)$ where start_e and end_e are the begin and end node that are connected by e. A *path* in a graph $G = (V, E)$ is a sequence of edges $e_1, e_2, \ldots, e_n$ such that for all i it holds that $\text{end}_{e_i} = \text{start}_{e_{i+1}}$ and all vertices are distinct. A *connected graph* is a graph where for every pair of vertices u and v there is a path from u to v.

Definition 3.21 (Tree [33])**.** *A* tree *is a graph in which any two vertices are connected by exactly one path.*

Equivalently, a tree is a connected graph without any cycles, where a cycle is, loosely speaking, a path where the first and last vertex coincide. A proof for this can be found in [33]. A *rooted tree* is a tree where one of the vertices has been labeled as the *root* of the tree. The *parent* of a vertex v is then the first vertex w on the path from v to the root. We call v the *child* of vertex w. A *leaf* is a vertex without children. The *depth* of a vertex is the number of edges on the path from the root to this vertex.

Definition 3.22 (Decision tree [107])**.** *A* decision tree *$\mathcal{T}$ is a rooted tree $\mathcal{T} = (\mathcal{D} \cup \mathcal{L}, E)$ where the set of vertices is split in the set of* decision nodes *$\mathcal{D}$ (which includes the root) and the set of* leaf nodes *$\mathcal{L}$. Additionally, the decision tree includes the following functions: (i)* $\mathrm{t} : \mathcal{D} \to \mathbb{Z}$ *a function that assigns to each decision node a value from $\mathbb{Z}$, (ii)* $\mathrm{m} : \mathcal{D} \to \{1, \ldots, A\}$ *a function that assigns to each decision node an index value from $\{1, \ldots, A\}$, (iii)* $\mathrm{c} : \mathcal{L} \to \{c_1, \ldots, c_v\}$ *a function that assigns to each leaf node a label from the set of leaf labels $\{c_1, \ldots, c_v\}$.*



(a) A threshold decision node.

(b) A categorical decision node.

Figure 3.1: Two types of decision node in a decision tree.

Let $\mathscr{T} = (\mathscr{D} \cup \mathscr{L}, E)$ be a decision tree with functions t, m and c which needs to be evaluated on input $\mathbf{x} \in \mathbb{Z}^A$. This evaluation is denoted by the function $\mathscr{T} : \mathbb{Z}^A \mapsto \{c_1, \ldots, c_v\}$ with $\{c_1, \ldots, c_v\}$ the finite set of v classification labels. In Algorithm 1, the tree evaluation algorithm is given. Given v a node, v.left returns the left child node and v.right the right child node.

The algorithm traverses the tree by evaluating the decision nodes that he comes across. For every decision node, the functions m and t are called. The first function returns the index of the input value that is being evaluated at the decision node within the input vector $\mathbf{x}$. Every decision node is either a threshold decision node or a categorical decision node. Examples of these type of nodes can be seen in Figure 3.1. We assume that both threshold decision nodes and categorical decision nodes have only two children. Please note here that every categorical node with more than two child nodes can be translated to a tree with categorical decision nodes with only two children.

A threshold decision node compares an input value to a certain threshold, while a categorical decision node tests if the input value is equal to a certain category. The function t gives this threshold or category of a decision node. At each decision node the corresponding input variable is compared to the decision node's threshold or categorical value. Based on this, either the right or left child node is picked as the next node. The traversal ends at a certain leaf node, which tells the classification label of the input according to the decision tree, which is given by calling the function c.

Algorithm 1

```
1:  function 𝒯(x)
2:      v ← root
3:      while v not leaf node do
4:          if v threshold then
5:              if x_m(v) ≤ t(v) then
6:                  v ← v.left
7:              else v ← v.right
8:          if v categorical then
9:              if x_m(v) = t(v) then
10:                 v ← v.right
11:             else v ← v.left
12:     return c(v)
```

3.5. Private path-evaluation techniques

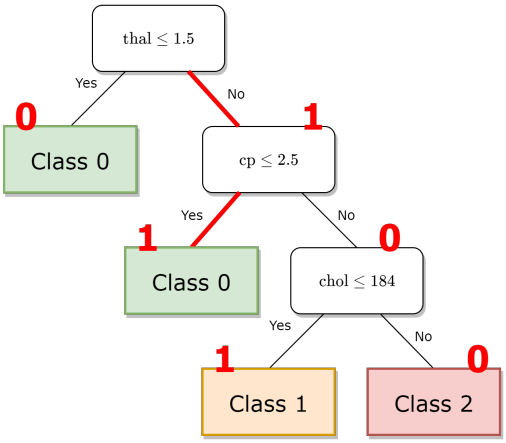
In the literature, several techniques are proposed to evaluate a decision tree while keeping the input and decision tree private. In this thesis, the focus lies on solutions that use homomorphic encryption. The methods based on homomorphic encryption use a protocol to evaluate every decision node homomorphically. This results in an encrypted decision bit for each decision node. Either an encrypted value of 1 or an encrypted value of 0 is stored at the child nodes to denote the appropriate decision. The fact that these outputs are encrypted shows the necessity of evaluating *all* decision nodes. After this, each path needs to be evaluated. Below, two important approaches for this homomorphic path evaluation are given. They either use homomorphic multiplications as their basis or homomorphic additions. We call those two methods the *multiplicative approach* and the *additive approach* respectively.

3.5.1. Multiplicative approach

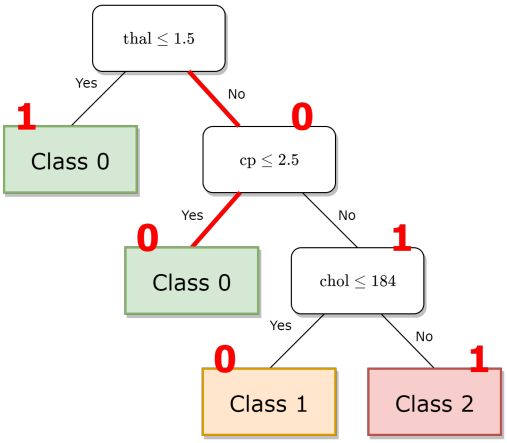
Tueno et al. [107] use the idea that by storing an encryption of 1 at the correct child node (and 0 at the wrong child node) and homomorphically *multiplying* all these bits per path, only the correct path has a multiplication value equal to 1. This method can be seen in Algorithm 2. multiplications of decision bits are done per level of the tree starting at the root. In this way, it is avoided that for each new path the knowledge of previous multiplications within the path gets lost. This is implemented by a queue with the functions `enqueue` to add nodes to the back of the queue and the function `dequeue` to remove nodes from the front of the queue [107]. In the algorithm, $[\![\ldots]\!]$ denotes an encryption of the value within the brackets according to a fully homomorphic encryption scheme. For $d \in \mathscr{D}$ a decision node, d.cmp denotes the decision bit that is stored at this node which is during the path evaluation changed to the multiplication of the decision bit itself with all decision bits on the path towards the root. The decision bit of the root node is set to 1 before running this protocol. Multiplying the final decision bit of the leaf nodes (which is only 1 for the correct leaf node) with the corresponding label and adding all these results, gives the output label. In Figure 3.2a an example of such a path evaluation is given.

Algorithm 2

1: **function** EVALPATHS($\mathscr{D}, \mathscr{L}$)
2: Let Q be an empty queue
3: Q.`enqueue`(root)
4: **while** Q.`empty`() = false **do**
5: $v \leftarrow Q$.`dequeue`()
6: $[\![v.\text{left.cmp}]\!] \leftarrow [\![v.\text{left.cmp}]\!] \boxdot [\![v.\text{cmp}]\!]$
7: $[\![v.\text{right.cmp}]\!] \leftarrow [\![v.\text{right.cmp}]\!] \boxdot [\![v.\text{cmp}]\!]$
8: **if** v.left $\in \mathscr{D}$ **then**
9: Q.`enqueue`(v.left)
10: **if** v.right $\in \mathscr{D}$ **then**
11: Q.`enqueue`(v.right)
12: [[result]] $\leftarrow$ [[0]]
13: **for all** $l \in \mathscr{L}$ **do**
14: $[\![\text{result}]\!] \leftarrow [\![\text{result}]\!] \boxplus \left([\![l.\text{cmp}]\!] \boxdot [\![\text{c}(l)]\!]\right)$
15: **return** [[result]]



(a) Path evaluation method of Tueno et al. [107]. Multiplying all decision bits on the correct path returns the value 1 while for the other paths this returns 0.



(b) Path evaluation method of Tai et al. [104]. Adding all decision bits on the correct path returns the value 0 while for the other paths this returns a value ≥ 1.

Figure 3.2: An example of a decision tree trained on a heart-disease dataset [45] that is evaluated on input tal = 2, cp = 3 and chol = 120 using different path evaluation methods. In these figures, the stored decision bits are denoted in red and the correct path according to the input is given by red edges. These numbers could also be encrypted and homomorphically evaluated.

3.5.2. Additive approach

Tai et al. 's [104] approach uses only homomorphic additions. The algorithm can be seen in Algorithm 3. Their method makes sure that a value of 0 is stored at the correct next child node (instead of 1 as in the previous method). The paths are evaluated by homomorphically *adding* the encrypted comparison bits of all nodes on a path. This summation is called the path cost. Now, only if the path cost evaluates to 0, this path corresponds to the correct leaf node and classification label. In contrast to Algorithm 2, in order to find the classification label, all path costs are decrypted so that the zero-cost path can be found. The leaf node labels can be sent by adding the label to the path cost homomorphically. Tai et al. [104] make sure that no more information is given away by multiplying the path costs with a random value. In Figure 3.2b the method of Tai et al. is demonstrated.

Algorithm 3

function EVALPATHS+($\mathscr{D}, \mathscr{L}$)
  Let Q be an empty queue
  Q.enqueue(root)
  while Q.empty() = false **do**
    $v \leftarrow Q$.dequeue()
    $[\![v.\text{left.cmp}]\!] \leftarrow [\![v.\text{left.cmp}]\!] \boxplus [\![v.\text{cmp}]\!]$
    $[\![v.\text{right.cmp}]\!] \leftarrow [\![v.\text{right.cmp}]\!] \boxplus [\![v.\text{cmp}]\!]$
    if v.left $\in \mathscr{D}$ **then**
      Q.enqueue(v.left)
    if v.right $\in \mathscr{D}$ **then**
      Q.enqueue(v.right)
  for all $l \in \mathscr{L}$ **do**
    $r, r' \leftarrow$ uniformly random
    $[\![\text{cost}_l]\!] \leftarrow r \boxdot [\![l.\text{cmp}]\!]$
    $[\![\text{label}_l]\!] \leftarrow \left(r' \boxdot [\![l.\text{cmp}]\!]\right) \boxplus [\![\text{c}(l)]\!]$
  return $([\![\text{cost}_l]\!], [\![\text{label}_l]\!])$ for all $l \in \mathscr{L}$

I

Mathematical analyses of the encryption schemes and formulation of the protocols

4

Detailed analyses and adaptations of encryption schemes

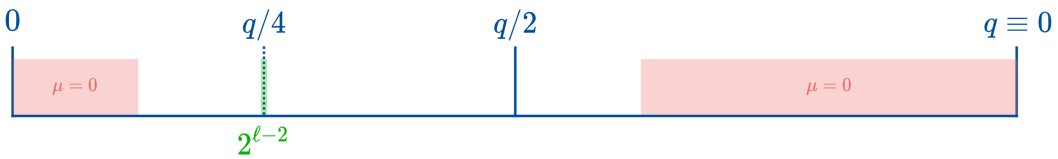
The aim of this chapter is to introduce some adaptations and new procedures essential for the functionality of our protocols and to analyse the correctness and security of the encryption schemes used in this thesis. Our protocols require some additional homomorphic operations, an encryption method using a public key, a collaborative-decryption procedure and the possibility of a bigger plaintext space. In this chapter these contributions are described. The correctness analysis needs to be done elaborately, since for implementation the exact noise propagation in our protocols has to be known for setting the parameters of the schemes. We are not aware of any literature that has done this, since all previous works focus on the theory behind the encryption schemes and do not implement it. Firstly, we describe the adaptations we propose in order for our protocols to work. Next, we give a detailed description of the correctness of the schemes. We end this chapter by proving the security of the encryption schemes, based on proof drafts from the literature.

4.1. Adaptations of Fully Homomorphic Encryption schemes

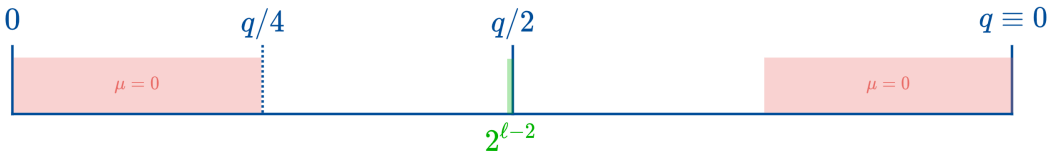
For usage within our protocols, the (Multi-Key) FHE schemes described in Sections 3.3.1 and 3.3.2 are not sufficient. Namely, in our protocols we need some additional homomorphic operations and we require the decryption to be done in a collaborative way. Some of our protocols use the path evaluation technique of Section 3.5.2, where the path cost is the sum of the comparison bits on the path. This can reach values higher than 1. Therefore, the schemes should be extended such that an encryption of an element that is not a bit can still be decrypted. First of all, we describe our method of setting the modulus q in the encryption schemes, which has an effect on the decryption procedure and its effectiveness. Also, this is essential for the parts of our protocols that take place in $\mathbb{Z}_2$ or $\{0,1\}$.

4.1.1. Modulus setting

During our research, we concluded that in the literature no requirements are given regarding the value of q, only to set it high enough such that the ciphertext noise does not obstruct a correct decryption. But, it is important in case the scheme needs to operate in the $\mathbb{Z}_2$ plaintext space. This holds for both the GSW and the Multi-Key FHE scheme. In order to understand why the modulus setting is important, we first describe the original decryption procedure.



(a) As an example, say $q = 2^\ell$ such that $2^{\ell-2} = q/4$, then the red area denotes the possible values of μ' that lead to a decryption to 0.



(b) As an example, say $q = 2^{\ell-1} + 1$ such that $2^{\ell-2} = \frac{q}{2} - \frac{1}{2}$, then the red area denotes the possible values of μ' that lead to a decryption to 0.

Figure 4.1: The $\mathbb{Z}_q$ space demonstrating the original $\{0,1\}$ decryption procedure. Since $\ell = \lceil \log_2 q \rceil$, we know that $2^{\ell-2} \in [q/4, q/2)$ (denoted in green).

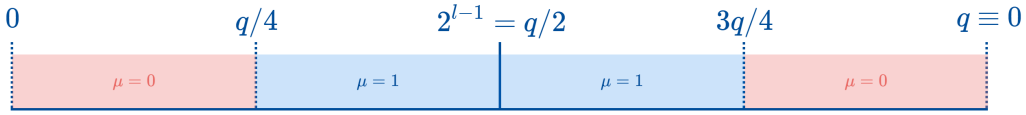


Figure 4.2: The $\mathbb{Z}_q$ space demonstrating the new $\{0,1\}$ decryption procedure. Since $q = 2^\ell$, we know that $2^{\ell-1} = q/2$. The red and blues area denote the possible values of μ' that lead to a decryption to 0 or 1 respectively.

As mentioned in Section 3.3.1, the output of the first step of the decryption is the value μ'; the inner product between the key and the *penultimate* column of the ciphertext. After choosing the value for q, we can calculate $\ell = \lceil \log_2 q \rceil$ that determines the size of the gadget vector used in the encryption schemes. It holds that $\mu' \approx \mu 2^{\ell-2}$ with μ the encrypted message of the ciphertext from the plaintext space $\{0,1\}$. In case μ' is closer to 0 than to $2^{\ell-2}$, we decrypt to 0, else to 1. In case we add two ciphertexts that are both an encryption of 1, the result should be a ciphertext encrypting 0, since $1+1 \bmod 2 \equiv 0$ in the plaintext space $\{0,1\}$. After such a homomorphic addition $\mu' \approx 2^{\ell-2} + 2^{\ell-2} = 2 \cdot 2^{\ell-2}$. Here, we know that $q \leq 2^\ell$ since $\ell = \lceil \log_2 q \rceil$ and therefore that $q/4 \leq 2^{\ell-2} < q/2$. In case $2^{\ell-2}$ is between $q/4$ and $q/3$ this gives the wrong result. For $2^{\ell-2} = q/4$ we find $\mu' = 2 \cdot 2^{\ell-2} = q/2$ which is closer to $2^{\ell-2}$ than it is to 0 and is therefore decrypted towards 1, as shown in Figure 4.1a. For $2^{\ell-2} = q/3$ we find $\mu' = 2 \cdot 2^{\ell-2} = 2q/3$ which is as close to $2^{\ell-2}$ as it is to 0. Correctly decrypting when two values of 1 are added in $\mathbb{Z}_2$, is impossible for these values of q. In case $2^{\ell-2}$ is higher than but close to $q/3$, a first addition can still be decrypted, but multiple additions after each other can result in the same problem. In Figure 4.1a, it can be seen that optimally, $2^{\ell-2}$ is close to $q/2$.

The GSW FHE scheme of Gentry et al. [54] is designed for the plaintext space $\mathbb{Z}_q$. They propose to evaluate a boolean circuit in $\mathbb{Z}_2$ by NAND-gates out of which any boolean circuit can be created. This means that the sum of two messages $\mu_1 + \mu_2$ in $\{0,1\}$ is evaluated by multiple NAND-gates that each contain a homomorphic multiplication. Namely, a homomorphic multiplication does not have the problem as above. For a more efficient calculation of a bit addition in $\mathbb{Z}_2$ we take q a power of 2 such that $q = 2^\ell$ with $\ell = \log_2 q$ such that the ciphertext addition does not give any problems. This is demonstrated in Figure 4.2. Gentry et al. [54] propose to take $\ell = \log_2 q + 1$ in case q is a power of 2, such that the inner product of the penultimate column with the secret key will be close to $q/2$. Instead of taking the penultimate column (and define ℓ differently), we take the *last* column of the ciphertext for decryption, since then $\mu' \approx \mu 2^{\ell-1} = q/2$ and we can keep our definition of ℓ. In the rest of this thesis, it is assumed q is chosen as a power of 2. The new decryption procedure is given below.

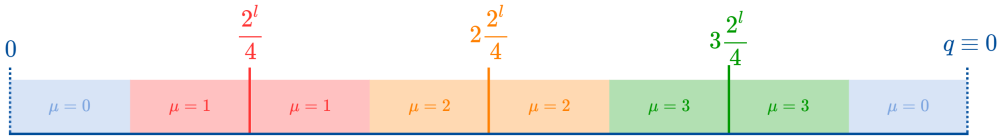


Figure 4.3: The $\mathbb{Z}_q$ space demonstrating the $\{0,1,\ldots,3\}$ decryption procedure for $D = 2^2 - 1$. Here, $q = 2^\ell$. The areas denote the possible values of μ' that lead to a decryption of the denoted values for μ.

– *(New decryption for bit plaintext space)* Let $q = 2^\ell$ for some integer ℓ and let $\mathbf{c}$ be the *last* column of the ciphertext $\mathbf{C}$ and calculate the inner product $\mu' = \mathbf{c} \cdot \mathbf{t}$. If $\mu' \mod q$ is closer to 0 than to $2^{\ell-1} = q/2$ return 0, else return 1.

4.1.2. Extended plaintext space

For some of our protocols non-bit values need to be encrypted, thus for both the GSW FHE scheme and the Multi-Key FHE a decryption procedure for non-bit values is required as well. Gentry et al. [54] propose, instead of taking one column for decryption, to multiply the secret key with the last ℓ columns of the ciphertext and 'decode' this to the appropriate output value μ such that the found row vector is the closest to $\mu \cdot \mathbf{g}^\intercal$.

Since we know the maximum value of our plaintext space we can approach this problem a bit differently such that only one multiplication with a column is required. Say we have a plaintext space equal to $\{0,1,\ldots,D\} \subseteq \mathbb{Z}_q$ with D a fixed value. We divide the space $\mathbb{Z}_q$ into $D+1$ intervals. Again we take q as a power of 2, namely $q = 2^\ell$. We assume that $D < q$ is given by $2^\kappa - 1$ for some positive integer κ (else take the next such value). The decryption for this plaintext space is given below. By taking the κ-th column from the right, we have $\mu' \approx \mu 2^{\ell-\kappa} = \mu \frac{2^\ell}{D+1}$. Then the output is 0 if μ' is closest to 0 and D if μ' is the closest to $D\frac{2^\ell}{D+1}$. In Figure 4.3 this is demonstrated for $D = 3$.

– *(Decryption for extended plaintext space)* Let $q = 2^\ell$ and $\{0,1,\ldots,D\} \subseteq \mathbb{Z}_q$ the plaintext space with $D = 2^\kappa - 1$ for some positive integer κ. Take $\mathbf{c}$ as the κ-th column from the right of $\mathbf{C}$ and calculate the inner product $\mu' = \mathbf{c} \cdot \mathbf{t}$. Return the value $i \in \{0,1,\ldots,D\}$ for which $i\frac{2^\ell}{D+1}$ is the closest to the value of μ'.

4.1.3. Encryption and decryption

In Table 4.1 a summary of the constructions of the Multi-Key FHE scheme of Peikert et al. [87] is shown, together with corresponding noises per equation. This scheme uses new LWE samples for every encryption (which is denoted by the matrix $\bar{\mathbf{C}}$). In order to make it possible to encrypt plaintexts under the secret key of another party by using their public key, we introduce a new public-key encryption procedure for the multi-key scheme. Additionally, this makes it possible to do a *re-randomisation* of a ciphertext, which will be explained in more detail in the next section. This new encryption procedure is given below. It is based on the encryption procedure of the GSW FHE scheme, but using different notation.

– *(Encryption by party i)* Let $\mathbf{A}'$ be the matrix given by the first $n-1$ rows of the common public matrix $\mathbf{A}$ and $\mathbf{a} \in \mathbb{Z}_q^m$ the vector containing the elements of the last row of $\mathbf{A}$, i.e.

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}' \\ \mathbf{a}^\intercal \end{pmatrix}.$$

Also, a random binary matrix is chosen $\mathbf{R} \leftarrow \{0,1\}^{m \times n\ell}$ uniformly. The encryption of a message $\mu \in \{0,1,\ldots,D\} \subseteq \mathbb{Z}_q$ by party i with secret key $\mathbf{t}_i = \begin{pmatrix} -\mathbf{s}_i & 1 \end{pmatrix}^\top \in \mathbb{Z}_q^n$ is then given by the ciphertext matrix

$$\mathbf{C} = \begin{pmatrix} \mathbf{A}' \\ -\mathbf{b}_i^\top + \mathbf{a}^\top \end{pmatrix} \mathbf{R} + \mu(\mathbf{I}_n \otimes \mathbf{g}^\top) \in \mathbb{Z}_q^{n \times n\ell}.$$

Notice that $-\mathbf{b}_i^\top + \mathbf{a}^\top = -\mathbf{t}_i^\top \mathbf{A} - \mathbf{e}_{\mathbf{b}_i}^\top + \mathbf{a}^\top = \mathbf{s}_i^\top \mathbf{A}' - \mathbf{a}^\top - \mathbf{e}_{\mathbf{b}_i}^\top + \mathbf{a}^\top = \mathbf{s}_i^\top \mathbf{A}' - \mathbf{e}_{\mathbf{b}_i}^\top$. Therefore, it holds that $\mathbf{t}_i^\top \mathbf{C} = \mu(\mathbf{t}_i^\top \otimes \mathbf{g}^\top) + (-\mathbf{s}_i^\top \mathbf{A}' - \mathbf{b}_i^\top + \mathbf{a}^\top)\mathbf{R} = \mu(\mathbf{t}_i^\top \otimes \mathbf{g}^\top) - \mathbf{e}_{\mathbf{b}_i}^\top \mathbf{R}$ and therefore encrypts μ correctly. Next to this, all elements in the matrix

$$\begin{pmatrix} \mathbf{A}' \\ -\mathbf{b}_i^\top + \mathbf{a}^\top \end{pmatrix}$$

are publicly known and can therefore be seen as the public key.

In the protocols using Multi-Key FHE, the decryption should be done partially since all parties posses their own key. Peikert et al. [87] do not introduce a way in which the decryption can be done in a collaborative manner where all parties do not give away any information about their secret key. Mukherjee et al. [80] do give such a procedure by introducing a 'smudging noise' that hides the partial decryption which is the result of multiplying one of the secret keys with the corresponding ciphertext rows. This decryption procedure is introduced below. We slightly adjusted it to incorporate our way of doing decryption, as defined above. Note that we can do a collaborative decryption for an extended plaintext space in a similar way.

– *(Collaborative decryption)* Let $q = 2^\ell$ and $\mathbf{C} \in \mathbb{Z}_q^{nk \times nk\ell}$ be a ciphertext associated to k parties, an encryption under the key $\mathbf{t}$, which is the concatenation of the k secret keys of all parties. Now, take $\mathbf{c}$ as the last column of $\mathbf{C}$ and for each party i, let $\mathbf{c}_i$ be the concatenation of the n elements with the same indices in $\mathbf{c}$ as the elements of $\mathbf{t}_i$ have in $\mathbf{t}$. Every party i then calculates the partial decryption $p_i = \mathbf{t}_i \cdot \mathbf{c}_i + e_i^{\text{smudge}} \in \mathbb{Z}_q$ with $e_i^{\text{smudge}} \leftarrow [0, B_{\text{smudge}}] \subseteq \mathbb{Z}_q$ uniformly. The purpose of the smudging noise e_i^{smudge} is to hide the value of the inner product, such that no information about the party's secret keys is shared. The exact value of B_{smudge} is given in the security analysis. The partial decryptions are then combined by calculating $\mu' = \sum_i p_i$. Output 0 if μ' is closer to 0 and 1 if μ' is closer to $2^{\ell-1}$.

As mentioned before, the encryption procedure of the Multi-Key FHE scheme given in Table 4.1 uses new LWE samples for every fresh encryption. This has the advantage that these fresh ciphertexts have a lower noise than fresh ciphertext encrypted using the public key. Namely, the noise vector is not multiplied with a random matrix. Therefore, it can be beneficial to also use this encryption procedure within GSW FHE when possible.

Table 4.1: A summary of the constructions of the Multi-Key FHE scheme from Peikert et al. [87] (the used notation can be found in Section 3.3.2). Here, n and m are the dimensions and $\chi = \bar{\Psi}_q^\alpha$ over $\mathbb{Z}_q$ with $\alpha \in (0,1)$ a real number and q the modulus. Also, $\ell = \lceil \log_2 q \rceil$ and $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$ uniformly the random common public matrix.

Name	Construction	Equation	Noise
Key generation	$\mathbf{b}^\intercal = \mathbf{t}^\intercal \mathbf{A} + \mathbf{e}_\mathbf{b}^\intercal$ with $\mathbf{e}_\mathbf{b} \leftarrow \chi^m$	$\mathbf{b}^\intercal \approx \mathbf{t}^\intercal \mathbf{A}$	$\mathbf{e}_\mathbf{b}^\intercal$
	$\mathbf{P} = \mathbf{A}\mathbf{R} + (\mathbf{I}_n \otimes \mathbf{t}^\intercal \otimes \mathbf{g}^\intercal)$	-	-
	$\mathbf{D} = \bar{\mathbf{D}} + (\mathbf{R} \otimes \mathbf{g} \otimes \mathbf{e}_n)$	-	-
Encryption	$\mathbf{C} = \bar{\mathbf{C}} + \mu(\mathbf{I}_n \otimes \mathbf{g}^\intercal)$	$\mathbf{t}^\intercal \mathbf{C} \approx \mu(\mathbf{t}^\intercal \otimes \mathbf{g}^\intercal)$	$\mathbf{e}_\mathbf{C}^\intercal$
Extension	$\mathbf{b}^\intercal = \begin{pmatrix} \mathbf{b}_1^\intercal & \dots & \mathbf{b}_k^\intercal \end{pmatrix}$	$\mathbf{b}^\intercal \approx \mathbf{t}^\intercal \cdot (\mathbf{I}_k \otimes \mathbf{A})$	$\mathbf{e}_\mathbf{b}^\intercal = \begin{pmatrix} \mathbf{e}_{\mathbf{b}_1}^\intercal & \dots & \mathbf{e}_{\mathbf{b}_k}^\intercal \end{pmatrix}$
	$\mathbf{Y}_1 = \mathbf{I}_k \otimes \mathbf{P}_{\text{new}}$	-	-
	$\mathbf{Y}_2 = \left(\left(\left(\mathbf{I}_{km} \otimes \mathbf{g}^{-1}\right)[-\mathbf{b}]\right)^\intercal \otimes \mathbf{I}_n\right) \cdot (\mathbf{I}_k \otimes \mathbf{D}_{\text{new}})$	-	-
	$\mathbf{Y} = \begin{pmatrix} \mathbf{Y}_1 \\ \mathbf{Y}_2 \end{pmatrix}$	$\mathbf{t}'^\intercal \mathbf{Y} \approx (\mathbf{t}^\intercal \otimes \mathbf{t}_{\text{new}}^\intercal \otimes \mathbf{g}^\intercal)$	$\mathbf{e}_Y^\intercal = \left(\left(\mathbf{I}_{km} \otimes \mathbf{g}^{-1}\right)[-\mathbf{b}]\right)^\intercal \cdot (\mathbf{I}_k \otimes \mathbf{E}_{\mathbf{D}_{\text{new}}}) - \mathbf{e}_\mathbf{b}^\intercal \cdot (\mathbf{I}_k \otimes \mathbf{R}_{\text{new}})$
	$\mathbf{X} = \begin{pmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \end{pmatrix} = \mathbf{Y}\left(\left(\mathbf{I}_{n^2 k} \otimes \mathbf{g}^{-1}\right)[\hat{\mathbf{C}} \otimes \mathbf{I}_n]\right)\Pi$	$\mathbf{t}'^\intercal \mathbf{X} \approx \mu(\mathbf{t}_{\text{new}}^\intercal \otimes \mathbf{g}^\intercal)$	$\mathbf{e}_\mathbf{X}^\intercal = \mathbf{e}_{\mathbf{X},1}^\intercal + \mathbf{e}_{\mathbf{X},2}^\intercal$ with $\mathbf{e}_{\mathbf{X},1}^\intercal = \mathbf{e}_\mathbf{Y}^\intercal\left((\mathbf{I}_{n^2 k} \otimes \mathbf{g}^{-1})[\hat{\mathbf{C}} \otimes \mathbf{I}_n]\right)\Pi$ and $\mathbf{e}_{\mathbf{X},2}^\intercal = (\mathbf{e}_\mathbf{C}^\intercal(\mathbf{e}_{nk} \otimes \mathbf{I}_\ell) \otimes \mathbf{t}_{\text{new}}^\intercal)\Pi$
	$\mathbf{C}' = \begin{pmatrix} \mathbf{C} & \mathbf{X}_1 \\ & \mathbf{X}_2 \end{pmatrix}$	$\mathbf{t}'^\intercal \mathbf{C}' \approx \mu(\mathbf{t}'^\intercal \otimes \mathbf{g}^\intercal)$	$\mathbf{e}_{\mathbf{C}'}^\intercal = \begin{pmatrix} \mathbf{e}_\mathbf{C}^\intercal & \mathbf{e}_\mathbf{X}^\intercal \end{pmatrix}$

4.1.4. Additional homomorphic operations

In this thesis' protocols, ciphertexts also need to be added to and multiplied with plaintexts. Therefore, we add the following homomorphic operations to the standard description of both FHE scheme. The homomorphic multiplication with a plaintext method is shortly described by Gentry et al. [54]. We add the re-randomisation step which is important when the ciphertext is directly shared with other parties after the homomorphic operation. Other parties can then possibly derive the value of the plaintext. This is especially the case when the matrix is multiplied by a value 0, which results in a ciphertext with only 0-values.

- *(Homomorphic addition with a plaintext)* Homomorphic addition of a ciphertext $\mathbf{C} \in \mathbb{Z}_q^{nk\times nk\ell}$ with a plaintext message $\zeta \in \{0,1,\ldots,D\} \subseteq \mathbb{Z}_q$ is done by calculating $\mathbf{C}+\zeta(\mathbf{I}_{nk}\otimes\mathbf{g}^\intercal) \in \mathbb{Z}_q^{nk\times nk\ell}$. We denote this addition by $\mathbf{C}\boxplus\zeta$ or by $\zeta\boxplus\mathbf{C}$.
- *(Homomorphic multiplication with a plaintext)* Homomorphic multiplication of a ciphertext $\mathbf{C} \in \mathbb{Z}_q^{nk\times nk\ell}$ encrypted under the public key $\mathbf{A} \in \mathbb{Z}_q^{nk\times m}$ with a plaintext message $\zeta \in \{0,1,\ldots,D\} \subseteq \mathbb{Z}_q$ is done by calculating $\zeta\mathbf{C} \in \mathbb{Z}_q^{nk\times nk\ell}$. We denote this multiplication by $\zeta\boxdot\mathbf{C}$.
- *(Re-randomisation)* The ciphertext $\mathbf{C} \in \mathbb{Z}_q^{nk\times nk\ell}$ is re-randomised by adding a matrix $\mathbf{AR}'$ as follows: $\zeta\mathbf{C}+\mathbf{AR}' \in \mathbb{Z}_q^{nk\times nk\ell}$ with $\mathbf{R}' \leftarrow \{0,1\}^{m\times nk\ell}$ a random binary matrix.

Also, since the encryptions and extensions are not always done in the same order during the execution of the protocols, a re-ordering of the ciphertext rows is done such that all ciphertexts have the same common extended key which is necessary for doing homomorphic operations.

- *(Re-ordering)* Let $\mathbf{C} \in \mathbb{Z}_q^{nk\times nk\ell}$ be a ciphertext associated to k parties, encrypted under the key $\mathbf{t}$, which is the concatenation of the k secret keys of all parties. The order of the keys within $\mathbf{t}$ is in the order of the encrypted party followed by the extensions. Re-ordering the rows results in a ciphertext $\mathbf{C}' \in \mathbb{Z}_q^{nk\times nk\ell}$ that encrypts under a key $\mathbf{t}' \in \mathbb{Z}_q^{nk}$ given by $\mathbf{t}' = \begin{pmatrix}\mathbf{t}_0 & \mathbf{t}_1 & \ldots & \mathbf{t}_{k-1}\end{pmatrix}^\intercal$ for parties $i \in \{0,\ldots,k-1\}$ with corresponding keys $\mathbf{t}_i$.

4.1.5. Key-switching

We introduce a procedure called *key-switching* in the FHE schemes. Key-switching transforms a ciphertext to a different ciphertext that encrypts the same message, but is now decryptable under a different secret key. This makes it possible to translate multiple ciphertexts, encrypted under different keys, to the same key such that they can be homomorphically combined. We make use of this procedure in our last two protocols.

Key-switching was introduced by Brakerski et al. [16]. One of the inputs of the switching procedure is a switching key, which is the encryption of 0 under the new key to which some information about the old key is added. Therefore, the party holding the old key, the 'old' party, and the party holding the new key, the 'new' party, have to communicate with each other in order to create this switching key. The method in [16] assumes the ciphertext to be a vector instead of a matrix. The correctness and security is proved in Lemma 3 in [16]. We describe the procedure below. Let $\mathbf{c}_{\text{old}} \in \mathbb{Z}_q^n$ be the decryption column of a ciphertext under the secret key $\mathbf{t}_{\text{old}} \in \mathbb{Z}_q^n$, that needs to be translated to an encryption $\mathbf{c}_{\text{new}}$ under the secret key $\mathbf{t}_{\text{new}} \in \mathbb{Z}_q^n$. Here $\mathbf{t}_{\text{old}}$ is the secret key of the old party and $\mathbf{t}_{\text{new}}$ the secret key of the new party.

- *(Switching key generation)* First, the new party with key $\mathbf{t}_{\text{new}}$, creates an encryption $\bar{\mathbf{S}} \in \mathbb{Z}_q^{n\times nl}$ of 0 using this secret key. It now holds that $\mathbf{t}_{\text{new}}^\intercal\bar{\mathbf{S}} = \begin{pmatrix}e_1 & \ldots & e_{nl}\end{pmatrix}$ with $e_i \leftarrow \bar{\Psi}_q^\alpha$ from the noise

distribution. This new party then send this encryption to the old party. The old party then calculates the switching key

$$\mathbf{S}_{\mathbf{t}_{\text{old}}\to\mathbf{t}_{\text{new}}} = \bar{\mathbf{S}} + \begin{pmatrix} \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{t}_{\text{old}}^{\mathsf{T}} \otimes \mathbf{g}^{\mathsf{T}} \end{pmatrix} \mathbb{Z}_q^{n\times nl}.$$

– *(Switch the key)* The party that does the key-switching, should not be the same party as the new party, since then this new party can find the secret key of the old party upon receiving the switching key. The party that does the key switching calculates the new ciphertext column by

$$\mathbf{c}_{\text{new}} = \mathbf{S}_{\mathbf{t}_{\text{old}}\to\mathbf{t}_{\text{new}}} \cdot (\mathbf{I}_n \otimes \mathbf{g}^{-1})[\mathbf{c}_{\text{old}}].$$

In our protocols, after key-switching, still some homomorphic operations on the full ciphertexts are performed. Therefore, we need the whole ciphertext, and not just the decryption column, during the rest of the protocol. Luckily, the procedure above works in exactly the same way when the ciphertexts are matrices $\mathbf{C}_{\text{old}}, \mathbf{C}_{\text{new}} \in \mathbb{Z}_q^{n\times nl}$. In this case, the procedure of switching consists of one homomorphic multiplication between the switching key and the ciphertext given by $\mathbf{S}_{\mathbf{t}_{\text{old}}\to\mathbf{t}_{\text{new}}} \cdot (\mathbf{I}_n \otimes \mathbf{g}^{-1})[\mathbf{C}_{\text{old}}]$.

4.1.6. New gadget vector

In our protocols that need an extended plaintext space $\{0,1,\ldots,D\}$ for D some integer, we need to randomise some ciphertexts by multiplying them with a random value. This can be done by homomorphically multiplying the ciphertexts by a plaintext. The functionality of our protocols requires the result to be non-zero values. In case $D+1$ is not a prime, we see that the multiplication with a random value can result in a zero value which is not desired. As an example, take $D+1=4$, then when a result of 2 is multiplied by a random value of 2, we find $2\cdot 2 = 4 \equiv 0 \mod D+1$. Therefore, we need $D+1$ to be prime. We take the plaintext space $\{0,1,\ldots,D'\}$ with $D'+1$ the first prime number that is bigger than D. Dividing the $\mathbb{Z}_q$ space in $D'+1$ parts can be done by choosing $q=(D'+1)^\ell$ and using a different gadget vector, namely

$$\mathbf{g} = \begin{pmatrix} 1 \\ D'+1 \\ (D'+1)^2 \\ \vdots \\ (D'+1)^{l-1} \end{pmatrix} \in \mathbb{Z}_q^\ell \text{ with } l = \log_{D'+1} q. \tag{4.1}$$

Micciancio and Peikert [77] show that this gadget vector can securely be used within FHE and show that it has computable 'inverse' function as in Definition 3.4. This inverse function translates elements from $\mathbb{Z}_q$ to a vector in $\{0,1,\ldots,D'\}^\ell$ which is a representation of the elements by expressing them into the terms of the new gadget vector. Formally, the new inverse function is now from a matrix $\mathbf{M} \in \mathbb{Z}_q^{n\times m'}$ for any n, m' to a matrix in $\{0,1,\ldots,D'\}^{nl\times m'}$ with $\ell = \lceil \log_{D'+1} q \rceil$ such that $(\mathbf{I}_n \otimes \mathbf{g}^{\mathsf{T}}) \cdot (\mathbf{I}_n \otimes \mathbf{g}^{-1})[\mathbf{M}] = \mathbf{M}$. The decryption is now again defined by the inner product between the secret key and the *last* column of the ciphertext, as given below.

– *(Decryption for new gadget vector)* Let $q=(D'+1)^\ell$ for some integer ℓ and let $\mathbf{c}$ be the *last* column of the ciphertext $\mathbf{C}$ and calculate the inner product $\mu' = \mathbf{c}\cdot\mathbf{t}$. Return the value $i \in \{0,1,\ldots,D'\}$ for which $i\frac{q}{D'+1}$ is the closest to the value of μ'.

4.2. Correctness

To be able to prove correctness of the used encryption schemes, we need to analyse the encryption and decryption procedures together with the noise of each ciphertext and its bound needed for decryption. This is also crucial in the implementation phase, where the exact noise propagation should be defined. In order to do so, we define a noisy ciphertext in the same way as Mukherjee et al. [80] in combination with the definition of a bounded distribution from [88]. This definition of a bounded distribution is given in Definition 4.1.

Definition 4.1 (N_τ-bounded distribution [88])**.** *A distribution χ over $\mathbb{Z}_q$ or $\mathbb{R}$ is called N_τ-*bounded *if for $x \leftarrow \chi$ the following holds: $P(|x| > N) \leq 2^{-\tau}$ with τ, N positive integers and $|x| = \min\{x, q-x\}$ if $x \in \mathbb{Z}_q$.*

Mukherjee et al. [80] define a noisy ciphertext by a strict bound on the noise where the probability of exceeding this bound should be equal to 0. To make the noise analysis easier and more flexible, we slightly adjust this definition of a noisy ciphertext to the more flexible bound on the noise distribution given by Peikert et al. [88] in the definition above; the probability of exceeding the bound is now $2^{-\tau}$ for some positive integer τ. The resulting definition can be seen in Definition 4.2.

Definition 4.2 (N_τ-noisy ciphertext [80])**.** *A N_τ-*noisy ciphertext *of a message μ under secret key $\mathbf{t}$, which is a concatenation of the secret keys of $k \geq 1$ parties, is a matrix $\mathbf{C} \in \mathbb{Z}_q^{nk \times nk\ell}$ such that $\mathbf{t}^\intercal \mathbf{C} = \mu(\mathbf{t}^\intercal \otimes \mathbf{g}^\intercal) + \mathbf{e}^\intercal$ with the elements of $\mathbf{e}$ distributed according to a N_τ-bounded zero-mean normal distribution.*

Let q be an integer, $\alpha \in (0,1)$ a real number. We denote the normal distribution on $\mathbb{R}$ with mean 0 and standard deviation $\frac{\alpha q}{\sqrt{2\pi}}$ by Ψ_q^α. The noise distribution $\bar{\Psi}_q^\alpha$ of the encryption schemes is then given by rounding the samples from the distribution Ψ_q^α to the closest integer modulo q. We now show that if the distribution Ψ_q^α is bounded by some value, this also holds for the distribution after rounding to the closest integer modulo q. The corresponding proposition and proof is given below.

Proposition 4.1. *Let q be an integer, $\alpha \in (0,1)$ a real number and Ψ_q^α the normal distribution with mean 0 and standard deviation $\frac{\alpha q}{\sqrt{2\pi}}$. If Ψ_q^α is N_τ-bounded for τ, N positive integers, then also $\bar{\Psi}_q^\alpha$ is N_τ-bounded.*

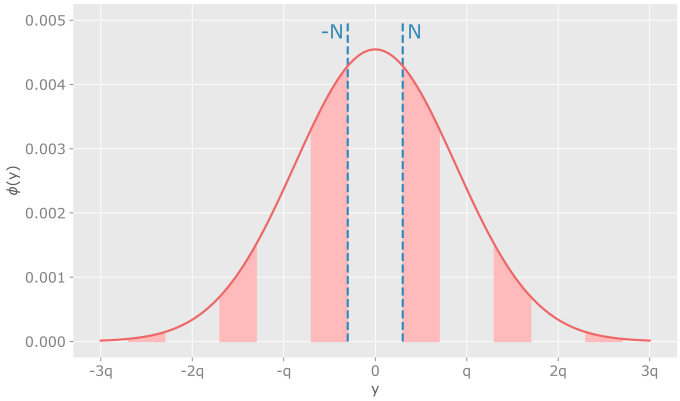


Figure 4.4: The probability density function ϕ of the continuous Gaussian distribution on $\mathbb{R}$ with mean 0 and standard deviation $\alpha q/\sqrt{2\pi}$ with example values $q = 100, \alpha = 2.2$. In blue the example boundary value of $N = 30 < q/2$ is shown. The light red area underneath the curve denotes the chance of values x for which $|x| > N$, which is clearly lower then $P(|y| > N)$ which is the total area underneath the curve on the outside of the blue lines.

Proof. Take and $y \leftarrow \Psi_q^\alpha \sim \mathcal{N}\left(0, \frac{(\alpha q)^2}{2\pi}\right)$ and let x take the value of y rounded to the closest integer modulo q. If the distribution Ψ_q^α is N_τ-bounded it holds that $P(|y| > N) \leq 2^{-\tau}$ with τ, N positive integers. We have two possibilities:

- $N \geq q/2$, which means that $P(|x| > N) = 0 \leq 2^{-\tau}$ or
- $0 < N < q/2$, which results in $P(|x| > N) < P(|y| > N) \leq 2^{-\tau}$.

In Figure 4.4, the reasoning of the second possibility is given in the description. So we can conclude that $\bar{\Psi}_q^\alpha$ is indeed N_τ-bounded. □

To be able to formalise the noise of ciphertexts, we first prove that the distribution Ψ_q^α is B_τ-bounded for certain positive integers B, τ. This immediately gives that $\bar{\Psi}_q^\alpha$ is also B_τ-bounded given Proposition 4.1. The proof mainly depends on Lemma 4 in [3] which is a standard fact about a Gaussian distribution. The proposition uses the Lambert W function which is a multi-valued function that describes the inverse of the relation $f(w) = w\exp(w)$ where $w \in \mathbb{R}$.

Proposition 4.2. *The distribution Ψ_q^α on $\mathbb{R}$ is a N_τ-bounded distribution for $N = \alpha q\sqrt{\frac{W(2^{2\tau+1}/\pi)}{2\pi}}$ with $W(\cdot)$ the Lambert W function and τ a positive integer.*

Proof. Let $y \leftarrow \Psi_q^\alpha$. We have to prove that $P(|y| > N) \leq 2^{-\tau}$ with $N = \alpha q\sqrt{\frac{W(2^{2\tau+1}/\pi)}{2\pi}}$. According to Lemma 4 in [3], for Ψ_q^α a normal distribution with mean 0 and standard deviation $\frac{\alpha q}{\sqrt{2\pi}}$, it holds for all $C > 0$ that

$$P\left(|y| > C\frac{\alpha q}{\sqrt{2\pi}}\right) \leq \frac{2}{C\sqrt{2\pi}}\exp\left(-C^2/2\right). \tag{4.2}$$

Take $N = C\frac{\alpha q}{\sqrt{2\pi}}$ such that $C = \frac{N\sqrt{2\pi}}{\alpha q}$. We can make the inequality $P(|y| > N) \leq 2^{-\tau}$ hold by isolating N as follows:

$$\frac{\alpha q}{N\pi}\exp\left(-\frac{N^2\pi}{\alpha^2 q^2}\right) = 2^{-\tau} \iff \left(\frac{N\pi}{\alpha q}\right)^2\exp\left(\frac{2N^2\pi}{\alpha^2 q^2}\right) = 2^{2\tau} \iff \left(\frac{2N^2\pi}{\alpha^2 q^2}\right)\exp\left(\frac{2N^2\pi}{\alpha^2 q^2}\right) = 2^{2\tau+1}/\pi. \tag{4.3}$$

This gives the required result with $W(\cdot)$ the Lambert W function. □

In order to analyse the propagation of noise during homomorphic operations, we need to prove some statements about adding or transforming several normally-distributed variables, since the initial noise is distributed according to a normal distribution. These are given in Propositions 4.3 and 4.4. These proofs are based on some probability theory theorems introduced in Chapter 3.

Proposition 4.3. *Let q be an integer, $\alpha \in (0,1)$ a real number and Ψ_q^α the normal distribution on $\mathbb{R}$ with mean 0 and standard deviation $\frac{\alpha q}{\sqrt{2\pi}}$ that is U_τ-bounded for τ, U positive integers. Take $e \leftarrow \Psi_q^\alpha$. For n a positive integer, $n \cdot e$ is distributed according to the $(nU)_\tau$-bounded zero-mean normal distribution given by $\Psi_q^{n\alpha}$.*

Proof. According to the linear transform theorem in Theorem 3.8, $n \cdot e$ is again normally distributed with zero mean and standard deviation equal to $n\sigma_\chi = \frac{n\alpha q}{\sqrt{2\pi}}$. So we can conclude that $n \cdot e$ is indeed distributed according to a zero-mean normal distribution, namely $\Psi_q^{n\alpha}$. From Proposition 4.2 we can deduce that $\Psi_q^{n\alpha}$ is W_τ-bounded with $W = n\alpha q\sqrt{\frac{W(2^{2\tau+1}/\pi)}{2\pi}} \leq nU$ since $U \geq \alpha q\sqrt{\frac{W(2^{2\tau+1}/\pi)}{2\pi}}$. So $\Psi_q^{n\alpha}$ is indeed $(nU)_\tau$-bounded. This can also be seen by $P(|n\cdot e| > nU) = P(n\cdot|e| > nU) = P(|e| > U) \leq 2^{-\tau}$. □

Proposition 4.4. *Let q be an integer, $\alpha_1, \alpha_2 \in (0,1)$ real numbers and $\Psi_q^{\alpha_1}, \Psi_q^{\alpha_2}$ normal distributions on $\mathbb{R}$ with mean 0 and standard deviations $\frac{\alpha_1 q}{\sqrt{2\pi}}, \frac{\alpha_2 q}{\sqrt{2\pi}}$ that are U_τ-bounded and V_τ-bounded respectively for τ, U and V positive integers. Take $e_1 \leftarrow \Psi_q^{\alpha_1}$ and $e_2 \leftarrow \Psi_q^{\alpha_2}$ independently. Then $e_1 + e_2$ and $e_1 - e_2$*

are distributed according to a zero-mean normal distribution that is $(U+V)_\tau$*-bounded given by* $\Psi_q^{\alpha'}$ *with* $\alpha' = \sqrt{\alpha_1^2+\alpha_2^2}$*. In case* $\alpha_1 = \alpha_2$*, the distributions of* e_1+e_2 *and* e_1-e_2 *are both* $(\sqrt{2}U)_\tau$*- and* $(\sqrt{2}V)_\tau$*-bounded.*

Proof. According to the normal sum theorem in Theorem 3.10, e_1+e_2 and e_1-e_2 are normal variables with mean 0 and variance $\frac{\alpha_1^2q^2}{2\pi}+\frac{\alpha_2^2q^2}{2\pi} = \frac{(\alpha_1^2+\alpha_2^2)q^2}{2\pi}$. Thus, they are distributed according to $\Psi_q^{\alpha'}$ with $\alpha' = \sqrt{\alpha_1^2+\alpha_2^2}$. Proposition 4.2 shows that $\Psi_q^{\alpha'}$ is W_τ-bounded for

$$W = \alpha' q\sqrt{\frac{W(2^{2\tau+1}/\pi)}{2\pi}} = \sqrt{\alpha_1^2+\alpha_2^2}\,q\sqrt{\frac{W(2^{2\tau+1}/\pi)}{2\pi}} \leq (\alpha_1+\alpha_2)q\sqrt{\frac{W(2^{2\tau+1}/\pi)}{2\pi}} \leq U+V$$

since $U \geq \alpha_1 q\sqrt{\frac{W(2^{2\tau+1}/\pi)}{2\pi}}$ and $V \geq \alpha_2 q\sqrt{\frac{W(2^{2\tau+1}/\pi)}{2\pi}}$. Since $P(|e_1+e_2| > U+V) \leq P(|e_1+e_2| > W) \leq 2^{-\tau}$, we can conclude that $\Psi_q^{\alpha'}$ is indeed $(U+V)_\tau$-bounded. In case $\alpha_1 = \alpha_2$, we find $\alpha' = \sqrt{2}\alpha_1 = \sqrt{2}\alpha_2$. Then, $\alpha' q\sqrt{\frac{W(2^{2\tau+1}/\pi)}{2\pi}} = \sqrt{2}\alpha_1 q\sqrt{\frac{W(2^{2\tau+1}/\pi)}{2\pi}} \leq \sqrt{2}U$. The same holds for α_2 and V. Therefore, $\Psi_q^{\alpha'}$ is both $(\sqrt{2}U)_\tau$- and $(\sqrt{2}V)_\tau$-bounded. □

Below, we describe the noise after encryption, homomorphic operations, extensions and decryptions and prove that the schemes return the correct decryption. Drafts of the proofs in the following subsections can be found in [4, 54, 80, 88]. To the best of our knowledge, we are the first work that gives a detailed description of the noise during these procedures. The newly introduced procedures and homomorphic operations can be found in the sections above. For the other homomorphic operations, Sections 3.3.1 and 3.3.2 can be consulted.

It is important to note here that in order to make the derivations more straightforward, from now on the intermediate rounding in the noise analysis is neglected. By doing that, the combined noises can be analysed by the combination of normal distributions (not rounded). This induces only a small discrepancy in the final noise distribution, but still gives a proper image of the noise distribution and its bound.

4.2.1. Encryption

A *fresh* ciphertext is a ciphertext which is not yet extended and on which no homomorphic operations are applied yet. We can formalise the noise of a fresh ciphertext for both the encryption procedure from the Multi-Key FHE scheme and the encryption procedure using a public key $\mathbf{A}$. This formalisation is given in Propositions 4.5 and 4.6.

Proposition 4.5. *Let* n, m *be the dimensions and* q *the modulus of the cryptosystem and* $\bar{\Psi}_q^\alpha$ *over* $\mathbb{Z}_q$ *the noise distribution that is* B_τ*-bounded for* τ *a positive integer and* $\alpha \in (0,1)$ *a real number. A fresh single-key ciphertext* $\mathbf{C} = \mathbf{AR} + \mu(\mathbf{I}_n \otimes \mathbf{g}^\intercal) \in \mathbb{Z}_q^{n\times n\ell}$ *of message* μ *with public key* $\mathbf{A}$*, random matrix* $\mathbf{R} \in \{0,1\}^{m\times n\ell}$ *for secret key* $\mathbf{t}$ *is a* $(mB)_\tau$*-noisy ciphertext.*

Proof. We have $\mathbf{t}^\intercal\mathbf{C} = \mu(\mathbf{t}^\intercal\otimes\mathbf{g}^\intercal) + \mathbf{t}^\intercal\mathbf{AR} = \mu(\mathbf{t}^\intercal\otimes\mathbf{g}^\intercal) + \mathbf{e}^\intercal\mathbf{R}$ where $\mathbf{e}^\intercal = \begin{pmatrix} e_1 & \dots & e_m \end{pmatrix}$ with $e_i \leftarrow \bar{\Psi}_q^\alpha$. Assuming $e_i \leftarrow \Psi_q^\alpha$ (so not rounded), Proposition 4.4 proves that the elements in $\mathbf{e}^\intercal\mathbf{R}$ are distributed according to a $(mB)_\tau$-bounded zero-mean normal distribution. So neglecting the intermediate rounding and given Proposition 4.1, we can conclude that this fresh single-key ciphertext is indeed $(mB)_\tau$-noisy. □

Proposition 4.6. *Let* n *be the dimensions and* q *the modulus of the cryptosystem and* $\bar{\Psi}_q^\alpha$ *over* $\mathbb{Z}_q$ *the noise distribution that is* B_τ*-bounded for* τ *a positive integer and* $\alpha \in (0,1)$ *a real number. A fresh single-key ciphertext* $\mathbf{C} = \bar{\mathbf{C}} + \mu(\mathbf{I}_n \otimes \mathbf{g}^\intercal) \in \mathbb{Z}_q^{n\times n\ell}$ *of message* μ *with* $\bar{\mathbf{C}}$ *a concatenation of* $n\ell$ *independent LWE samples for secret key* $\mathbf{t}$ *is a* B_τ*-noisy ciphertext.*

Proof. We have $\mathbf{t}^\intercal \mathbf{C} = \mu(\mathbf{t}^\intercal \otimes \mathbf{g}^\intercal) + \mathbf{t}^\intercal \bar{\mathbf{C}} = \mu(\mathbf{t}^\intercal \otimes \mathbf{g}^\intercal) + \mathbf{e}^\intercal$ where $\mathbf{e}^\intercal = \begin{pmatrix} e_1 & \dots & e_{n\ell} \end{pmatrix}$ with $e_i \leftarrow \bar{\Psi}_q^\alpha$. Assuming $e_i \leftarrow \Psi_q^\alpha$ (so not rounded), Proposition 4.4 proves that the elements in $\mathbf{e}^\intercal$ are distributed according to a B_τ-bounded zero-mean normal distribution. So neglecting the intermediate rounding and given Proposition 4.1, we can conclude that such a fresh single-key ciphertext is indeed B_τ-noisy. □

4.2.2. Homomorphic operations

In this subsection we analyse the noise propagation after a homomorphic addition, multiplication, addition or multiplication with a plaintext, and re-randomisation of a ciphertext.

Proposition 4.7. *Let n be the dimension and q the modulus of the cryptosystem. Also, let $\mathbf{C_1}, \mathbf{C_2} \in \mathbb{Z}_q^{nk \times nk\ell}$ be two ciphertexts of messages μ_1 and μ_2 from the plaintext space $\{0, 1, \dots, D\} \subseteq \mathbb{Z}_q$ under the secret key $\mathbf{t}$, which is a concatenation of the secret keys of $k \geq 1$ parties, that are U_τ- and V_τ-noisy respectively. When doing homomorphic addition $(\mathbf{C_1} \boxplus \mathbf{C_2})$, the resulting ciphertext is a $(U+V)_\tau$-noisy ciphertext of $\mu_1 + \mu_2$.*

Proof. We have $\mathbf{C}_1, \mathbf{C}_2 \in \mathbb{Z}_q^{nk \times nk\ell}$ such that $\mathbf{t}^\intercal \mathbf{C}_i = \mu_i(\mathbf{t}^\intercal \otimes \mathbf{g}^\intercal) + \mathbf{e}_i^\intercal$ for $i = 1, 2$ with the elements of $\mathbf{e}_i$ distributed according to a (assuming continuous) zero-mean normal distribution that is either U_τ- or V_τ-bounded. Therefore, $\mathbf{t}^\intercal(\mathbf{C_1} \boxplus \mathbf{C_2}) = \mu_1 \left(\mathbf{t}^\intercal \otimes \mathbf{g}^\intercal\right) + \mathbf{e}_1^\intercal + \mu_2 \left(\mathbf{t}^\intercal \otimes \mathbf{g}^\intercal\right) + \mathbf{e}_2^\intercal = (\mu_1 + \mu_2)\left(\mathbf{t}^\intercal \otimes \mathbf{g}^\intercal\right) + (\mathbf{e}_1^\intercal + \mathbf{e}_2^\intercal)$. Therefore, the resulting ciphertext is indeed $(U+V)_\tau$-noisy since the distribution of $(\mathbf{e}_1^\intercal + \mathbf{e}_2^\intercal)$ is $(U+V)_\tau$-bounded according to Propositions 4.1 and 4.4. □

Proposition 4.8. *Let n be the dimension and q the modulus of the cryptosystem. Also, let $\mathbf{C_1}, \mathbf{C_2} \in \mathbb{Z}_q^{nk \times nk\ell}$ be two ciphertexts of messages μ_1 and μ_2 from the plaintext space $\{0, 1, \dots, D\} \subseteq \mathbb{Z}_q$ under the secret key $\mathbf{t}$, which is a concatenation of the secret keys of $k \geq 1$ parties, that are U_τ- and V_τ-noisy respectively. When doing homomorphic multiplication $(\mathbf{C_1} \boxdot \mathbf{C_2})$, the resulting ciphertext is a $(nk\ell U + DV)_\tau$-noisy ciphertext of $\mu_1\mu_2$.*

Proof. We have $\mathbf{C}_1, \mathbf{C}_2 \in \mathbb{Z}_q^{nk \times nk\ell}$ such that $\mathbf{t}^\intercal \mathbf{C}_i = \mu_i(\mathbf{t}^\intercal \otimes \mathbf{g}^\intercal) + \mathbf{e}_i^\intercal$ for $i = 1, 2$ with the elements of $\mathbf{e}_i$ distributed according to a zero-mean normal distribution that is either U_τ- or V_τ-bounded. Thus, $\mathbf{t}^\intercal(\mathbf{C_1} \boxdot \mathbf{C_2}) = \mathbf{t}^\intercal \mathbf{C_1}(\mathbf{I}_{nk} \otimes \mathbf{g}^{-1})[\mathbf{C_2}] = (\mu_1 \left(\mathbf{t}^\intercal \otimes \mathbf{g}^\intercal\right) + \mathbf{e}_1^\intercal)(\mathbf{I}_{nk} \otimes \mathbf{g}^{-1})[\mathbf{C_2}] = \mu_1 \mathbf{t}^\intercal \mathbf{C_2} + \mathbf{e}_1^\intercal(\mathbf{I}_{nk} \otimes \mathbf{g}^{-1})[\mathbf{C_2}] = \mu_1 \left(\mu_2 \left(\mathbf{t}^\intercal \otimes \mathbf{g}^\intercal\right) + \mathbf{e}_2^\intercal\right) + \mathbf{e}_1^\intercal(\mathbf{I}_{nk} \otimes \mathbf{g}^{-1})[\mathbf{C_2}] = \mu_1\mu_2 \left(\mathbf{t}^\intercal \otimes \mathbf{g}^\intercal\right) + \mu_1 \mathbf{e}_2^\intercal + \mathbf{e}_1^\intercal(\mathbf{I}_{nk} \otimes \mathbf{g}^{-1})[\mathbf{C_2}]$. Also, $\mu_1 \in \{0, 1, \dots, D\}$ and $(\mathbf{I}_{nk} \otimes \mathbf{g}^{-1})[\mathbf{C_2}]$ a binary $nk\ell \times nk\ell$ matrix. Neglecting the rounding and using Propositions 4.1, 4.3 and 4.4, $\mu_1 \mathbf{e}_2^\intercal + \mathbf{e}_1^\intercal(\mathbf{I}_{nk} \otimes \mathbf{g}^{-1})[\mathbf{C_2}]$ is $(nk\ell U + DV)_\tau$-bounded distributed. □

Proposition 4.9. *Let n be the dimension and q the modulus of the cryptosystem. Also, let $\mathbf{C} \in \mathbb{Z}_q^{nk \times nk\ell}$ be a U_τ-noisy ciphertext of message μ from the plaintext space $\{0, 1, \dots, D\} \subseteq \mathbb{Z}_q$ under the secret key $\mathbf{t}$, which is a concatenation of the secret keys of $k \geq 1$ parties. When doing homomorphic addition of the ciphertext with a plaintext $\zeta \in \{0, 1, \dots, D\}$ $(\mathbf{C} \boxplus \zeta)$, the resulting ciphertext is a U_τ-noisy ciphertext of $\zeta + \mu$.*

Proof. We have $\mathbf{C} \in \mathbb{Z}_q^{nk \times nk\ell}$ such that $\mathbf{t}^\intercal \mathbf{C} = \mu(\mathbf{t}^\intercal \otimes \mathbf{g}^\intercal) + \mathbf{e}^\intercal$ with the elements of $\mathbf{e}$ distributed according to a zero-mean normal distribution that is U_τ-bounded. Then, $\mathbf{t}^\intercal(\mathbf{C} \boxplus \zeta) = \mathbf{t}^\intercal \left(\mathbf{C} + \zeta(\mathbf{I}_{nk} \otimes \mathbf{g}^\intercal)\right) = \mu(\mathbf{t}^\intercal \otimes \mathbf{g}^\intercal) + \mathbf{e}^\intercal + \mathbf{t}^\intercal \zeta(\mathbf{I}_{nk} \otimes \mathbf{g}^\intercal) = \mu(\mathbf{t}^\intercal \otimes \mathbf{g}^\intercal) + \mathbf{e}^\intercal + \zeta(\mathbf{t}^\intercal \otimes \mathbf{g}^\intercal) = (\mu + \zeta)(\mathbf{t}^\intercal \otimes \mathbf{g}^\intercal) + \mathbf{e}^\intercal$ which shows that the resulting ciphertext is indeed a U_τ-noisy ciphertext of $\zeta + \mu$. □

Proposition 4.10. *Let n be the dimension and q the modulus of the cryptosystem. Also, let $\mathbf{C} \in \mathbb{Z}_q^{nk \times nk\ell}$ be a U_τ-noisy ciphertext of message μ from the plaintext space $\{0, 1, \dots, D\} \subseteq \mathbb{Z}_q$ under the secret key $\mathbf{t}$, which is a concatenation of the secret keys of $k \geq 1$ parties. When doing homomorphic multiplication of the ciphertext with a plaintext $\zeta \in \{0, 1, \dots, D\}$ $(\zeta \boxdot \mathbf{C})$, the resulting ciphertext is a $(\zeta U)_\tau$-noisy ciphertext of $\zeta \cdot \mu$ for $\zeta > 0$.*

Proof. We have $\mathbf{C} \in \mathbb{Z}_q^{nk \times nk\ell}$ such that $\mathbf{t}^\intercal \mathbf{C} = \mu(\mathbf{t}^\intercal \otimes \mathbf{g}^\intercal) + \mathbf{e}^\intercal$ with the elements of $\mathbf{e}$ distributed according to a zero-mean normal distribution that is U_τ-bounded. Then for $\zeta > 0$, $\mathbf{t}^\intercal(\zeta \boxdot \mathbf{C}) = \mathbf{t}^\intercal(\zeta \mathbf{C}) = \mu\zeta(\mathbf{t}^\intercal \otimes \mathbf{g}^\intercal) + \zeta\mathbf{e}^\intercal$. Neglecting the rounding and using Propositions 4.1 and 4.3, we can conclude that $\zeta\mathbf{e}^\intercal$ is $(\zeta U)_\tau$-bounded distributed. □

Proposition 4.11. *Let n, m be the dimensions and q the modulus of the cryptosystem and $\bar{\Psi}_q^\alpha$ over $\mathbb{Z}_q$ the noise distribution that is B_τ-bounded for τ a positive integer and $\alpha \in (0,1)$ a real number. Also, let $\mathbf{C} \in \mathbb{Z}_q^{nk \times nk\ell}$ be a U_τ-noisy ciphertext of message μ from the plaintext space $\{0,1,\ldots,D\} \subseteq \mathbb{Z}_q$ under the public key $\mathbf{A} \in \mathbb{Z}_q^{nk \times m}$ and the secret key $\mathbf{t}$, which is a concatenation of the secret keys of $k \geq 1$ parties. When doing re-randomisation of the ciphertext, the resulting ciphertext is a $(U + mB)_\tau$-noisy ciphertext of μ.*

Proof. We have $\mathbf{C} \in \mathbb{Z}_q^{nk \times nk\ell}$ such that $\mathbf{t}^\intercal \mathbf{C} = \mu(\mathbf{t}^\intercal \otimes \mathbf{g}^\intercal) + \mathbf{e}^\intercal$ with the elements of $\mathbf{e}$ distributed according to a zero-mean normal distribution that is U_τ-bounded. After re-randomisation, the result is $\mathbf{C} + \mathbf{AR} \in \mathbb{Z}_q^{nk \times nk\ell}$ with $\mathbf{R} \leftarrow \{0,1\}^{m \times nk\ell}$ a random binary matrix. So we find $\mathbf{t}^\intercal(\mathbf{C} + \mathbf{AR}) = \mu(\mathbf{t}^\intercal \otimes \mathbf{g}^\intercal) + \mathbf{e}^\intercal + \mathbf{t}^\intercal\mathbf{AR}$. So, an additional noise term $\mathbf{e}'^\intercal\mathbf{R}$ is added where $\mathbf{e}'^\intercal = \begin{pmatrix} e_1 & \ldots & e_m \end{pmatrix}$ with $e_i \leftarrow \bar{\Psi}_q^\alpha$. According to Propositions 4.1 and 4.4, this results in a $(U + mB)_\tau$-noisy ciphertext. □

4.2.3. Decryption

We now prove that the decryption procedure returns the correct decryption of a U_τ-noisy ciphertext with probability at least $1 - 2^{-\tau}$ for a certain bound on U. We do this for both the extended and $\{0,1\}$ plaintext space. The definition of B_{smudge} can be found in Section 4.1.3 where we explained the collaborative decryption procedure. In case GSW FHE is used in the protocols, it can be assumed that this smudging noise is equal to 0 since these protocols do not require collaborative decryption.

Proposition 4.12. *Let n be the dimension and $q = 2^\ell$ the modulus of the cryptosystem. Also, let $\mathbf{C} \in \mathbb{Z}_q^{nk \times nk\ell}$ be a U_τ-noisy ciphertext of a bit message μ under the secret key $\mathbf{t}$, which is a concatenation of the secret keys of $k \geq 1$ parties. The ciphertext $\mathbf{C}$ can be decrypted with probability at least $1 - 2^{-\tau}$ when $U < \frac{q}{4} - kB_{\text{smudge}}$.*

Proof. Let $\mathbf{c}$ be the last column of $\mathbf{C}$ and for each party i, let $\mathbf{c}_i$ be the n rows of $\mathbf{c}$ that have the corresponding positions as $\mathbf{t}_i$ has in $\mathbf{t}$. We have

$$\mu' = \sum_i p_i = \sum_i \left(\mathbf{t}_i \cdot \mathbf{c}_i + e_i^{\text{smudge}}\right) = \mathbf{t}^\intercal \cdot \mathbf{c} + \sum_i e_i^{\text{smudge}} = \mu 2^{\ell-1} + e + \sum_i e_i^{\text{smudge}}$$

with $e \leftarrow \chi$ for a distribution χ that is U_τ-bounded. We know that $\sum_i e_i^{\text{smudge}} \leq kB_{\text{smudge}}$. Looking at Figure 4.2, the decryption procedure works if $e + \sum_i e_i^{\text{smudge}} < q/4$. This happens with probability at least $1 - 2^{-\tau}$ if $U + kB_{\text{smudge}} < q/4$ or $U < \frac{q}{4} - kB_{\text{smudge}}$. This concludes the proof. □

Proposition 4.13. *Let n be the dimension and $q = 2^\ell$ the modulus of the cryptosystem. Also, let $\mathbf{C} \in \mathbb{Z}_q^{nk \times nk\ell}$ be a U_τ-noisy ciphertext of a message $\mu \in \{0,1,\ldots,D\} \subseteq \mathbb{Z}_q$ under the secret key $\mathbf{t}$, which is a concatenation of the secret keys of $k \geq 1$ parties. The ciphertext $\mathbf{C}$ can be decrypted with probability at least $1 - 2^{-\tau}$ when $U < \frac{q}{2(D+1)} - kB_{\text{smudge}}$.*

Proof. Let $\mathbf{c}$ be the $\left(\log_2(D+1)\right)$-th column of the right of $\mathbf{C}$ and for each party i, let $\mathbf{c}_i$ be the n rows of $\mathbf{c}$ that have the corresponding positions as $\mathbf{t}_i$ has in $\mathbf{t}$. We have

$$\mu' = \sum_i p_i = \mu \frac{2^\ell}{D+1} + e + \sum_i e_i^{\text{smudge}}$$

with $e \leftarrow \chi$ for a distribution χ that is U_τ-bounded. Again, $\sum_i e_i^{\text{smudge}} \leq kB_{\text{smudge}}$. Looking at Figure 4.3, the decryption procedure works if $e + \sum_i e_i^{\text{smudge}} < \frac{q}{2(D+1)}$. This happens with probability at least $1 - 2^{-\tau}$ if $U + kB_{\text{smudge}} < \frac{q}{2(D+1)}$ or $U < \frac{q}{2(D+1)} - kB_{\text{smudge}}$. This concludes the proof. □

4.2.4. Extension

We now show how the error propagates during one ciphertext extension in the Multi-Key FHE scheme. In order to do this, we first need to show the correctness of the extension procedure within the multi-key scheme, which is also given in [88].

Proposition 4.14. *Let n, m be the dimensions and q the modulus of the cryptosystem and $\bar{\Psi}_q^\alpha$ over $\mathbb{Z}_q$ the noise distribution that is B_τ-bounded for τ a positive integer and $\alpha \in (0,1)$ a real number. Let $\mathbf{C} \in \mathbb{Z}_q^{nk \times nk\ell}$ be a U_τ-noisy ciphertext of message μ under secret key $\mathbf{t}$, which is a concatenation of the secret keys of $k \geq 1$ parties. After extension to one extra key $\mathbf{t}_{\text{new}}$ of party* new, *the resulting ciphertext $\mathbf{C}' \in \mathbb{Z}_q^{n(k+1) \times n(k+1)\ell}$ is approximately a $\left((\sqrt{n} + nk)\ell\sqrt{km}B + UB\right)_\tau$-noisy multi-key ciphertext of message μ under secret key $\mathbf{t}' = \begin{pmatrix}\mathbf{t}^\intercal & \mathbf{t}_{\text{new}}^\intercal\end{pmatrix}^\intercal$.*

Proof. In Table 4.1, a summary of the extension procedure within the Multi-Key FHE scheme is shown. We first show that this procedure indeed results in a ciphertext $\mathbf{C}' \in \mathbb{Z}_q^{n(k+1) \times n(k+1)\ell}$ that correctly encrypts μ under the secret key $\mathbf{t}'$, or $\mathbf{t}'^\intercal \mathbf{C}' = \mu(\mathbf{t}'^\intercal \otimes \mathbf{g}^\intercal) + \mathbf{e}_{\mathbf{C}'}$ with the noise values normally distributed with a zero mean. This derivation can also be found in [88]. We have $\mathbf{C} \in \mathbb{Z}_q^{nk \times nk\ell}$ such that $\mathbf{t}^\intercal \mathbf{C} = \mu(\mathbf{t}^\intercal \otimes \mathbf{g}^\intercal) + \mathbf{e}_{\mathbf{C}}^\intercal$ with the elements of $\mathbf{e}$ distributed according to a zero-mean normal distribution that is U_τ-bounded. The method includes the creation of a matrix $\mathbf{X} = \begin{pmatrix}\mathbf{X}_1^\intercal & \mathbf{X}_2^\intercal\end{pmatrix}^\intercal$ with $\mathbf{X}_1 \in \mathbb{Z}_q^{nk \times n\ell}$ and $\mathbf{X}_2 \in \mathbb{Z}_q^{n \times n\ell}$ such that

$$\mathbf{t}'^\intercal \mathbf{X} = \mathbf{t}^\intercal \mathbf{X}_1 + \mathbf{t}_{\text{new}}^\intercal \mathbf{X}_2 = \mu(\mathbf{t}_{\text{new}}^\intercal \otimes \mathbf{g}^\intercal) + \mathbf{e}_{\mathbf{X}}^\intercal \tag{4.4}$$

for some $\mathbf{e}_{\mathbf{X}} \in \mathbb{Z}_q^{n\ell}$ of which the elements are normally distributed with zero mean. Choosing

$$\mathbf{C}' = \begin{pmatrix}\mathbf{C} & \mathbf{X}_1 \\ & \mathbf{X}_2\end{pmatrix}$$

then gives the required extended matrix, namely $\mathbf{t}'^\intercal \mathbf{C}' = \mu\left((\mathbf{t}^\intercal \otimes \mathbf{g}^\intercal) + \mathbf{e}_{\mathbf{C}}^\intercal \quad (\mathbf{t}_{\text{new}}^\intercal \otimes \mathbf{g}^\intercal) + \mathbf{e}_{\mathbf{X}}^\intercal\right) = \mu(\mathbf{t}'^\intercal \otimes \mathbf{g}^\intercal) + \left(\mathbf{e}_{\mathbf{C}}^\intercal \quad \mathbf{e}_{\mathbf{X}}^\intercal\right)$. The construction of this $\mathbf{X}$ consists of the following two steps.

– First, a matrix $\mathbf{Y} = \begin{pmatrix}\mathbf{Y}_1 \\ \mathbf{Y}_2\end{pmatrix} \in \mathbb{Z}_q^{n(k+1) \times n^2 k\ell}$ is created that satisfies

$$\mathbf{t}'^\intercal \mathbf{Y} = \mathbf{t}^\intercal \mathbf{Y_1} + \mathbf{t}_{\text{new}}^\intercal \mathbf{Y_2} = (\mathbf{t}^\intercal \otimes \mathbf{t}_{\text{new}}^\intercal \otimes \mathbf{g}^\intercal) + \mathbf{e}_{\mathbf{Y}}^\intercal \tag{4.5}$$

with $\mathbf{e}_{\mathbf{Y}} \in \mathbb{Z}_q^{n^2 k\ell}$.

For constructing the matrix $\mathbf{Y}$, we need the public extension keys of the parties. Say $\mathbf{b}_i$ for $i \in \{1, \ldots, k\}$ are the first extension keys of the k parties and $\mathbf{b}$ the concatenation of all these $\mathbf{b}_i$. Then, $\mathbf{b}^\intercal = \mathbf{t}^\intercal \cdot (\mathbf{I}_k \otimes \mathbf{A}) + \mathbf{e}_{\mathbf{b}}^\intercal$ with $\mathbf{A}$ the common public matrix and the elements in $\mathbf{e}_{\mathbf{b}} \in \mathbb{Z}_q^{mk}$ distributed according to $\bar{\Psi}_q^\alpha$. Let $\mathbf{P}_{\text{new}}$ and $\mathbf{D}_{\text{new}}$ be the extension keys of the new party, associated with the private key $\mathbf{t}_{\text{new}}$ and the random binary matrix $\mathbf{R}_{\text{new}}$. By construction, $\mathbf{P}_{\text{new}} = \mathbf{A}\mathbf{R}_{\text{new}} + (\mathbf{I}_n \otimes \mathbf{t}_{\text{new}}^\intercal \otimes \mathbf{g}^\intercal)$. Now define $\mathbf{Y}_1 = \mathbf{I}_k \otimes \mathbf{P}_{\text{new}} = (\mathbf{I}_k \otimes \mathbf{A}\mathbf{R}_{\text{new}}) + \left(\mathbf{I}_{nk} \otimes \mathbf{t}_{\text{new}}^\intercal \otimes \mathbf{g}^\intercal\right) \in \mathbb{Z}_q^{nk \times n^2 k\ell}$ and $\mathbf{Y}_2 := \left(\left(\left(\mathbf{I}_{km} \otimes \mathbf{g}^{-1}\right)[-\mathbf{b}]\right)^\intercal \otimes \mathbf{I}_n\right) \cdot (\mathbf{I}_k \otimes \mathbf{D}_{\text{new}}) \in \mathbb{Z}_q^{n \times n^2 k\ell}$. So,

$$\begin{aligned}
\mathbf{t}^\intercal \mathbf{Y}_1 &= \mathbf{t}^\intercal \cdot (\mathbf{I}_k \otimes \mathbf{A}\mathbf{R}_{\text{new}}) + (\mathbf{t}^\intercal \otimes 1 \otimes 1) \cdot \left(\mathbf{I}_{nk} \otimes \mathbf{t}_{\text{new}}^\intercal \otimes \mathbf{g}^\intercal\right) \\
&= \mathbf{t}^\intercal \cdot (\mathbf{I}_k \otimes \mathbf{A}) \cdot (\mathbf{I}_k \otimes \mathbf{R}_{\text{new}}) + \left(\mathbf{t}^\intercal \otimes \mathbf{t}_{\text{new}}^\intercal \otimes \mathbf{g}^\intercal\right) \\
&= (\mathbf{b}^\intercal - \mathbf{e}_{\mathbf{b}}^\intercal) \cdot (\mathbf{I}_k \otimes \mathbf{R}_{\text{new}}) + \left(\mathbf{t}^\intercal \otimes \mathbf{t}_{\text{new}}^\intercal \otimes \mathbf{g}^\intercal\right)
\end{aligned}$$

and

$$\begin{aligned}\mathbf{t}_{\text{new}}^{\intercal}\mathbf{Y}_2 &= \mathbf{t}_{\text{new}}^{\intercal}\cdot\left(\left(\left(\mathbf{I}_{km}\otimes\mathbf{g}^{-1}\right)[-\mathbf{b}]\right)^{\intercal}\otimes\mathbf{I}_n\right)\cdot(\mathbf{I}_k\otimes\mathbf{D}_{\text{new}})\\ &= \left(\left(\mathbf{I}_{km}\otimes\mathbf{g}^{-1}\right)[-\mathbf{b}]\right)^{\intercal}\cdot(\mathbf{I}_{km\ell}\otimes\mathbf{t}_{\text{new}}^{\intercal})\cdot(\mathbf{I}_k\otimes\mathbf{D}_{\text{new}})\\ &= \left(\left(\mathbf{I}_{km}\otimes\mathbf{g}^{-1}\right)[-\mathbf{b}]\right)^{\intercal}\cdot\left((\mathbf{I}_k\otimes\mathbf{R}_{\text{new}}\otimes\mathbf{g})+(\mathbf{I}_k\otimes\mathbf{E}_{\mathbf{D}_{\text{new}}})\right)\\ &= -\mathbf{b}^{\intercal}(\mathbf{I}_k\otimes\mathbf{R}_{\text{new}})+\left(\left(\mathbf{I}_{km}\otimes\mathbf{g}^{-1}\right)[-\mathbf{b}]\right)^{\intercal}\cdot(\mathbf{I}_k\otimes\mathbf{E}_{\mathbf{D}_{\text{new}}})\end{aligned}$$

since by construction $(\mathbf{I}_{ml}\otimes\mathbf{t}_{\text{new}}^{\intercal})\cdot\mathbf{D}_{\text{new}} = (\mathbf{R}_{\text{new}}\otimes\mathbf{g})+\mathbf{E}_{\mathbf{D}_{\text{new}}}$ where the elements in the noise matrix are sampled from $\bar{\Psi}_q^{\alpha}$. So indeed, Equation 4.5 holds for $\mathbf{e}_{\mathbf{Y}}^{\intercal} = \left(\left(\mathbf{I}_{km}\otimes\mathbf{g}^{-1}\right)[-\mathbf{b}]\right)^{\intercal}\cdot(\mathbf{I}_k\otimes\mathbf{E}_{\mathbf{D}_{\text{new}}})-\mathbf{e}_{\mathbf{b}}^{\intercal}\cdot(\mathbf{I}_k\otimes\mathbf{R}_{\text{new}})$.

– In the second step the matrix $\mathbf{X}$ is derived from the matrix $\mathbf{Y}$. First, take $\hat{\mathbf{C}} = \mathbf{C}\cdot(\mathbf{e}_{nk}\otimes\mathbf{I}_\ell)\in\mathbb{Z}_q^{nk\times\ell}$ as the last ℓ columns of the ciphertext $\mathbf{C}$ and $\Pi\in\{0,1\}^{n\ell\times n\ell}$ the permutation matrix such that $(\mathbf{g}^{\intercal}\otimes\mathbf{t}_{\text{new}}^{\intercal})\Pi = (\mathbf{t}_{\text{new}}^{\intercal}\otimes\mathbf{g}^{\intercal})$. Take

$$\mathbf{X} = \mathbf{Y}\left((\mathbf{I}_{n^2k}\otimes\mathbf{g}^{-1})[\hat{\mathbf{C}}\otimes\mathbf{I}_n]\right)\Pi\in\mathbb{Z}_q^{n(k+1)\times n\ell}.$$

Then,

$$\begin{aligned}\mathbf{t}'^{\intercal}\mathbf{X} &= \mathbf{t}'^{\intercal}\mathbf{Y}\left((\mathbf{I}_{n^2k}\otimes\mathbf{g}^{-1})[\hat{\mathbf{C}}\otimes\mathbf{I}_n]\right)\Pi\\ &= (\mathbf{t}^{\intercal}\otimes\mathbf{t}_{\text{new}}^{\intercal}\otimes\mathbf{g}^{\intercal})\left((\mathbf{I}_{n^2k}\otimes\mathbf{g}^{-1})[\hat{\mathbf{C}}\otimes\mathbf{I}_n]\right)\Pi+\mathbf{e}_{\mathbf{X},1}^{\intercal}\\ &= (\mathbf{t}^{\intercal}\otimes\mathbf{t}_{\text{new}}^{\intercal})(\hat{\mathbf{C}}\otimes\mathbf{I}_n)\Pi+\mathbf{e}_{\mathbf{X},1}^{\intercal}\\ &= ((\mathbf{t}^{\intercal}\hat{\mathbf{C}})\otimes\mathbf{t}_{\text{new}}^{\intercal})\Pi+\mathbf{e}_{\mathbf{X},1}^{\intercal}\\ &= (((\mu(\mathbf{t}^{\intercal}\otimes\mathbf{g}^{\intercal})+\mathbf{e}_{\mathbf{C}}^{\intercal})(\mathbf{e}_{nk}\otimes\mathbf{I}_\ell))\otimes\mathbf{t}_{\text{new}}^{\intercal})\Pi+\mathbf{e}_{\mathbf{X},1}^{\intercal}\\ &= \mu(\mathbf{g}^{\intercal}\otimes\mathbf{t}_{\text{new}}^{\intercal})\Pi+\mathbf{e}_{\mathbf{X},2}^{\intercal}+\mathbf{e}_{\mathbf{X},1}^{\intercal}\\ &= \mu(\mathbf{t}_{\text{new}}^{\intercal}\otimes\mathbf{g}^{\intercal})+\mathbf{e}_{\mathbf{X},2}^{\intercal}+\mathbf{e}_{\mathbf{X},1}^{\intercal}\\ &= \mu(\mathbf{t}_{\text{new}}^{\intercal}\otimes\mathbf{g}^{\intercal})+\mathbf{e}_{\mathbf{X}}^{\intercal}\end{aligned}$$

with $\mathbf{e}_{\mathbf{X}}^{\intercal} = \mathbf{e}_{\mathbf{X},1}^{\intercal}+\mathbf{e}_{\mathbf{X},2}^{\intercal}$ with $\mathbf{e}_{\mathbf{X},1}^{\intercal} = \mathbf{e}_{\mathbf{Y}}^{\intercal}\left((\mathbf{I}_{n^2k}\otimes\mathbf{g}^{-1})[\hat{\mathbf{C}}\otimes\mathbf{I}_n]\right)\Pi$ and $\mathbf{e}_{\mathbf{X},2}^{\intercal} = (\mathbf{e}_{\mathbf{C}}^{\intercal}(\mathbf{e}_{nk}\otimes\mathbf{I}_\ell)\otimes\mathbf{t}_{\text{new}}^{\intercal})\Pi$. This is exactly given in Equation 4.4.

So the only thing left to prove is that $\mathbf{e}_{\mathbf{X}}$ is distributed according to a zero-mean normal distribution with the given bound. We have

$$\begin{aligned}\mathbf{e}_{\mathbf{X}}^{\intercal} &= \left(\left(\left(\mathbf{I}_{km}\otimes\mathbf{g}^{-1}\right)[-\mathbf{b}]\right)^{\intercal}\cdot(\mathbf{I}_k\otimes\mathbf{E}_{\mathbf{D}_{\text{new}}})-\mathbf{e}_{\mathbf{b}}^{\intercal}\cdot(\mathbf{I}_k\otimes\mathbf{R}_{\text{new}})\right)\left((\mathbf{I}_{n^2k}\otimes\mathbf{g}^{-1})[\hat{\mathbf{C}}\otimes\mathbf{I}_n]\right)\Pi\\ &\quad+\left(\mathbf{e}_{\mathbf{C}}^{\intercal}(\mathbf{e}_{nk}\otimes\mathbf{I}_\ell)\otimes\mathbf{t}_{\text{new}}^{\intercal}\right)\Pi\\ &= \left(\left(\mathbf{I}_{km}\otimes\mathbf{g}^{-1}\right)[-\mathbf{b}]\right)^{\intercal}\cdot(\mathbf{I}_k\otimes\mathbf{E}_{\mathbf{D}_{\text{new}}})\cdot\left((\mathbf{I}_{n^2k}\otimes\mathbf{g}^{-1})[\hat{\mathbf{C}}\otimes\mathbf{I}_n]\right)\Pi-\mathbf{e}_{\mathbf{b}}^{\intercal}\cdot(\mathbf{I}_k\otimes\mathbf{R}_{\text{new}})\left((\mathbf{I}_{n^2k}\otimes\mathbf{g}^{-1})[\hat{\mathbf{C}}\otimes\mathbf{I}_n]\right)\Pi\\ &\quad+\left(\mathbf{e}_{\mathbf{C}}^{\intercal}(\mathbf{e}_{nk}\otimes\mathbf{I}_\ell)\otimes\mathbf{t}_{\text{new}}^{\intercal}\right)\Pi.\end{aligned}$$

We know that $\left(\mathbf{I}_{km}\otimes\mathbf{g}^{-1}\right)[-\mathbf{b}]\in\{0,1\}^{km\ell}$, so $\left(\left(\mathbf{I}_{km}\otimes\mathbf{g}^{-1}\right)[-\mathbf{b}]\right)^{\intercal}\cdot(\mathbf{I}_k\otimes\mathbf{E}_{\mathbf{D}_{\text{new}}})\in\mathbb{Z}_q^{n^2k\ell}$ where the elements are distributed according to $\Psi_q^{\alpha'}$ with in the worst case $\alpha' = \sqrt{ml}\alpha$ that is $(\sqrt{ml}B)_\tau$ bounded according to Proposition 4.4 and ignoring the intermediate rounding. Also, $\left((\mathbf{I}_{n^2k}\otimes\mathbf{g}^{-1})[\hat{\mathbf{C}}\otimes\mathbf{I}_n]\right)\Pi\in\{0,1\}^{n^2k\ell\times n\ell}$ with maximum of $nk\ell$ non-zeroes per column. With the same reasoning above this results in the elements of $\left(\left(\mathbf{I}_{km}\otimes\mathbf{g}^{-1}\right)[-\mathbf{b}]\right)^{\intercal}\cdot(\mathbf{I}_k\otimes\mathbf{E}_{\mathbf{D}_{\text{new}}})\cdot\left((\mathbf{I}_{n^2k}\otimes\mathbf{g}^{-1})[\hat{\mathbf{C}}\otimes\mathbf{I}_n]\right)\Pi$ distributed in the worst case according to $\Psi_q^{\alpha'}$ with $\alpha' = \sqrt{nkm\ell^2}\alpha = \ell\sqrt{nkm}\alpha$ that is $(\ell\sqrt{nkm}B)_\tau$ bounded.

For the second term, we have $(\mathbf{I}_k\otimes\mathbf{R}_{\text{new}})\left((\mathbf{I}_{n^2k}\otimes\mathbf{g}^{-1})[\hat{\mathbf{C}}\otimes\mathbf{I}_n]\right)\Pi$ a $mk\times n\ell$ matrix where the elements have a maximum value of $nk\ell$. Neglecting the rounding and using Propositions 4.3 and 4.4, the elements of $\mathbf{e}_{\mathbf{b}}^{\intercal}\cdot(\mathbf{I}_k\otimes\mathbf{R}_{\text{new}})\left((\mathbf{I}_{n^2k}\otimes\mathbf{g}^{-1})[\hat{\mathbf{C}}\otimes\mathbf{I}_n]\right)\Pi$ are distributed according to $\Psi_q^{\alpha'}$ with in the worst case $\alpha' = nk\ell\sqrt{mk}\alpha$ that is $(nk\ell\sqrt{kmB})_\tau$ bounded.

The row vector $\left(\mathbf{e}_{\mathbf{C}}^{\top}(\mathbf{e}_{nk}\otimes\mathbf{I}_{\ell})\otimes\mathbf{t}_{\text{new}}^{\top}\right)\Pi$ of size $n\ell$ consists of elements that are the multiplication of a secret key element from B_τ-bounded $\bar{\Psi}_q^\alpha$ and an element from $\mathbf{e}_{\mathbf{C}}$ that is distributed according to a zero-mean normal distribution that is U_τ-bounded. We wish to express the noise by a normal distribution and therefore we approximate the bound on the elements by assuming that all the elements in $\mathbf{e}_{\mathbf{C}}$ are equal to U. By doing that, we can say that this final vector's elements are distributed according to $\bar{\Psi}_q^{\alpha'}$ with $\alpha' = U\alpha$ and is therefore $(UB)_\tau$-bounded according to Proposition 4.3. This affects the final bound only slightly, since for an element in $\mathbf{e}_{\mathbf{C}}$, say e, it holds that $P(e > U) \leq 2^{-\tau}$ and therefore $P(|et| > U|t|) = P(|e||t| > U|t|) \leq 2^{-\tau}$ for a given $t \leftarrow \bar{\Psi}_q^\alpha$.

To conclude, from neglecting the intermediate rounding and proposition 4.4 and 4.1 the final noise elements can be approximated in the worst case by a zero-mean normal distribution that is Q_τ-bounded with $Q = \sqrt{n}\ell\sqrt{km}B + nk\ell\sqrt{km}B + UB = (\sqrt{n}+nk)\ell\sqrt{km}B + UB$. So we can conclude that the ciphertext $\mathbf{C}'$ is a Q_τ-noisy ciphertext of message μ under secret key $\mathbf{t}'$. □

4.2.5. Key-switching

In this section we prove the correctness of the key-switching procedure, which we slightly adapted such that it can also be applied on ciphertext matrices. Also, we show how the error propagates during this procedure. The results are given by the proposition and proof below.

Proposition 4.15. *Let n be the dimensions and q the modulus of the cryptosystem and $\bar{\Psi}_q^\alpha$ over $\mathbb{Z}_q$ the noise distribution that is B_τ-bounded for τ a positive integer and $\alpha \in (0,1)$ a real number. Also, let $\mathbf{C} \in \mathbb{Z}_q^{n\times nl}$ be a ciphertext of message μ from the plaintext space $\{0,1,\ldots,D\} \subseteq \mathbb{Z}_q$ under the secret key $\mathbf{t}$ that is U_τ-noisy. After key-switching to $\mathbf{t}_{\text{new}}$, the resulting ciphertext is a $(U+\sqrt{nl}B)_\tau$-noisy ciphertext of μ under $\mathbf{t}_{\text{new}}$.*

Proof. We have $\mathbf{t}^{\top}\mathbf{C} = \mu(\mathbf{t}^{\top}\otimes\mathbf{g}^{\top}) + \mathbf{e}^{\top}$ with the elements of $\mathbf{e}$ distributed according to a (assuming continuous) zero-mean normal distribution that is U_τ-bounded. Let

$$\mathbf{S}_{\mathbf{t}\to\mathbf{t}_{\text{new}}} = \bar{\mathbf{S}} + \begin{pmatrix} \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{t}^{\top}\otimes\mathbf{g}^{\top} \end{pmatrix} \mathbb{Z}_q^{n\times nl}$$

be the switching-key. The switched ciphertext is then given by $\mathbf{S}_{\mathbf{t}\to\mathbf{t}_{\text{new}}}\cdot(\mathbf{I}_n\otimes\mathbf{g}^{-1})[\mathbf{C}]$. Multiplying the new key with the switched ciphertext gives

$$\mathbf{t}_{\text{new}}\left(\mathbf{S}_{\mathbf{t}\to\mathbf{t}_{\text{new}}}\cdot(\mathbf{I}_n\otimes\mathbf{g}^{-1})[\mathbf{C}]\right) = \left(\mathbf{t}_{\text{new}}^{\top}\bar{\mathbf{S}} + \mathbf{t}_{\text{new}}^{\top}\begin{pmatrix} \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{t}^{\top}\otimes\mathbf{g}^{\top} \end{pmatrix}\right)\cdot(\mathbf{I}_n\otimes\mathbf{g}^{-1})[\mathbf{C}] = \begin{pmatrix} e_1 & \ldots & e_{nl} \end{pmatrix}\cdot(\mathbf{I}_n\otimes\mathbf{g}^{-1})[\mathbf{C}] + \mathbf{t}^{\top}\mathbf{C}$$

with $e_i \leftarrow \bar{\Psi}_q^\alpha$ and $\bar{\mathbf{S}} \in \mathbb{Z}_q^{n\times nl}$ an encryption of 0 under the new key $\mathbf{t}_{\text{new}}$. Since $(\mathbf{I}_n\otimes\mathbf{g}^{-1})[\mathbf{C}]$ is a matrix in $\{0,1\}^{nl\times nl}$, we find that the above is equal to $\mu(\mathbf{t}^{\top}\otimes\mathbf{g}^{\top}) + \mathbf{e}^{\top} + \mathbf{e}'^{\top}$ where the elements of $\mathbf{e}'$ are distributed according to a (assuming continuous) zero-mean normal distribution that is $(\sqrt{nl}B)_\tau$-bounded, given Proposition 4.4. Therefore, the resulting ciphertext is indeed $(U+\sqrt{nl}B)_\tau$-noisy since the distribution of $(\mathbf{e}^{\top}+\mathbf{e}'^{\top})$ is $(U+\sqrt{nl}B)_\tau$-bounded according to Propositions 4.1 and 4.4. □

4.2.6. New gadget vector

We now analyse the decryption procedure that makes use of the new gadget vector introduced in Section 4.1.6. The result is given in Proposition 4.16. The definition of B_{smudge} can be found in Section 4.1.3 where we explained the collaborative decryption procedure. In case GSW FHE is used in the protocols, it can be assumed that this smudging noise is equal to 0 since these protocols do not require collaborative decryption.

Proposition 4.16. *Let n, m be the dimensions and $q = (D'+1)^\ell$ the modulus of the cryptosystem with $D'+1$ a prime number. Also, let $\mathbf{C} \in \mathbb{Z}_q^{nk \times nkl}$ be a U_τ-noisy ciphertext of a message $\mu \in \{0,1,\ldots,D'\} \subseteq \mathbb{Z}_q$ under the secret key $\mathbf{t}$, which is a concatenation of the secret keys of $k \geq 1$ parties. The ciphertext $\mathbf{C}$ can be decrypted with probability at least $1-2^{-\tau}$ when $U < \frac{q}{2(D'+1)} - kB_{\text{smudge}}$.*

Proof. Let $\mathbf{c}$ be the last column of $\mathbf{C}$ and for each party i, let $\mathbf{c}_i$ be the n rows of $\mathbf{c}$ that have the corresponding positions as $\mathbf{t}_i$ has in $\mathbf{t}$. We have

$$\mu' = \sum_i p_i = \mu \frac{q}{D'+1} + e + \sum_i e_i^{\text{smudge}}$$

with $e \leftarrow \chi$ for a distribution χ that is U_τ-bounded. We know that $\sum_i e_i^{\text{smudge}} \leq kB_{\text{smudge}}$. Concluding, the decryption procedure works if $e + \sum_i e_i^{\text{smudge}} < q/\left(2(D'+1)\right)$. This happens with probability at least $1-2^{-\tau}$ if $U + kB_{\text{smudge}} < q/\left(2(D'+1)\right)$ or $U < \frac{q}{2(D'+1)} - kB_{\text{smudge}}$. □

Using the new gadget vector, the result matrix after the inverse gadget operation is not a binary matrix anymore, which impacts the noise propagation of a homomorphic multiplication, the extension and key-switching procedure, as given in the propositions below. The proofs of these propositions are straightforward given the proofs of Propositions 4.8, 4.14 and 4.15 by changing the definition of the function $(\mathbf{I}_{nk} \otimes \mathbf{g}^{-1})[\cdot]$. We therefore do not give these proofs.

Proposition 4.17. *Let n be the dimension and q the modulus of the cryptosystem, using the new gadget vector. Also, let $\mathbf{C_1}, \mathbf{C_2} \in \mathbb{Z}_q^{nk \times nk\ell}$ be two ciphertexts of messages μ_1 and μ_2 from the plaintext space $\{0,1,\ldots,D'\} \subseteq \mathbb{Z}_q$ under the secret key $\mathbf{t}$, which is a concatenation of the secret keys of $k \geq 1$ parties, that are U_τ- and V_τ-noisy respectively. When doing homomorphic multiplication $(\mathbf{C_1} \boxdot \mathbf{C_2})$, the resulting ciphertext is a $(nk\ell D'U + D'V)_\tau$-noisy ciphertext of $\mu_1\mu_2$.*

Proposition 4.18. *Let n, m be the dimensions and q the modulus of the cryptosystem, using the new gadget vector. Let $\bar{\Psi}_q^\alpha$ over $\mathbb{Z}_q$ be the noise distribution that is B_τ-bounded for τ a positive integer and $\alpha \in (0,1)$ a real number. Let $\mathbf{C} \in \mathbb{Z}_q^{nk \times nk\ell}$ be a U_τ-noisy ciphertext of message $\mu \in \{0,1,\ldots,D'\} \subseteq \mathbb{Z}_q$ under secret key $\mathbf{t}$, which is a concatenation of the secret keys of $k \geq 1$ parties. After extension to one extra key $\mathbf{t}_{\text{new}}$ of party* new, *the resulting ciphertext $\mathbf{C}' \in \mathbb{Z}_q^{n(k+1) \times n(k+1)\ell}$ is approximately a $\left((\sqrt{n}D' + nk)\ell\sqrt{km}D'B + UB\right)_\tau$-noisy multi-key ciphertext of message μ under secret key $\mathbf{t}' = \begin{pmatrix}\mathbf{t}^\intercal & \mathbf{t}_{\text{new}}^\intercal\end{pmatrix}^\intercal$.*

Proposition 4.19. *Let n be the dimension and q the modulus of the cryptosystem, using the new gadget vector. Let $\bar{\Psi}_q^\alpha$ over $\mathbb{Z}_q$ be the noise distribution that is B_τ-bounded for τ a positive integer and $\alpha \in (0,1)$ a real number. Also, let $\mathbf{C} \in \mathbb{Z}_q^{n \times nl}$ be a ciphertext of message μ from the plaintext space $\{0,1,\ldots,D'\} \subseteq \mathbb{Z}_q$ under the secret key $\mathbf{t}$ that is U_τ-noisy. After key-switching to $\mathbf{t}_{\text{new}}$, the resulting ciphertext is a $(U + \sqrt{nl}D'B)_\tau$-noisy ciphertext of μ under $\mathbf{t}_{\text{new}}$.*

4.3. Security

In this section we prove the IND-CPA security of the two FHE schemes, according to Definition 3.5. Recall that an encryption scheme is IND-CPA secure, if the outcome of the game in Definition 3.5, for any probabilistic, polynomial time-bounded adversary, is 1 with a probability at most $1/2 + \text{negl}(\kappa)$ with κ the security parameter. Drafts of the proofs can be found in [54, 80, 88]. Our proofs are slightly more detailed. First, the Leftover Hash Lemma is introduced. The proof of the Leftover Hash Lemma can be found in [95].

Lemma 4.3 (Leftover Hash Lemma [95])**.** *Let G be some finite Abelian group and let j be some integer. For any j elements $g_1, \ldots, g_j \in G$ consider the distribution given by the sum of the elements in a random subset of $\{g_1, \ldots, g_j\}$. Then, the expectation of the statistical distance between this distribution and the uniform distribution on G, for a uniform choice of $g_1, \ldots, g_j \in G$, is at most $\sqrt{|G|/2^j}$.*

In an ideal game the public keys and ciphertext are picked uniformly at random and independent of the message. In order to prove the IND-CPA security, it is sufficient to show that an adversary in the ideal game can learn as much as an adversary in the real game and vice versa [88]. The security proof of the GSW encryption scheme is given below.

Theorem 4.4 (IND-CPA security of the GSW scheme)**.** *Let n, m be the dimensions and q the modulus of the cryptosystem and $\bar{\Psi}_q^\alpha$ over $\mathbb{Z}_q$ the noise distribution with $\alpha \in (0,1)$ a real number such that $\alpha q \geq 2\sqrt{n-1}$ that define the GSW cryptosystem. If $m \geq n \log_2 q + 2\kappa$ and assuming the hardness of the $LWE_{n-1,q,\bar{\Psi}_\alpha}$ problem, the GSW scheme from Section 3.3.1 is IND-CPA secure with κ the security parameter.*

Proof. We prove that the view of the attacker in the real game is not different than his view when every ciphertext and all public keys are uniformly random, which is sufficient for proving the IND-CPA security. We define three games for which we prove that the view of the adversary is equivalent:

- *Game 0:* The real IND-CPA game,
- *Game 1:* The IND-CPA game where public keys $\mathbf{b}, \mathbf{A}$ and the common matrix $\mathbf{B}$ are chosen uniformly at random and sent to the adversary,
- *Game 2:* The same as the previous game, but now also the ciphertexts are chosen uniformly at random; the ideal game.

To prove that the adversary does not gain extra information in game 0 compared to game 1, we use the hardness of the LWE problem. Namely, according to Theorem 3.17 with $\alpha q \geq 2\sqrt{n-1}$, we can replace the columns of $\mathbf{A}$ with m uniformly random samples from $\mathbb{Z}_q^n$ since the adversary is polynomial time-bounded. It is clear that also $\mathbf{B}$ and $\mathbf{b}$ are uniformly random in that case. Therefore, the adversary has no advantage in game 0 compared to game 1.

In game 1, the encryption of any plaintext μ is given by $\mathbf{C} = \mathbf{AR} + \mu(\mathbf{I}_n \otimes \mathbf{g}^\intercal) \in \mathbb{Z}_q^{n \times n\ell}$ with $\mathbf{A}$ uniformly random. We can now apply the Leftover Hash Lemma. Take $G = \mathbb{Z}_q^n$ which is a finite Abelian group and $j = m$. The columns of $\mathbf{A}$ are j elements of G. When multiplying $\mathbf{A}$ with $\mathbf{R}$, the result is a matrix with column vectors that are the sum of random subsets of the j elements of G. According to Lemma 4.3, these are distributed with an expected statistical distance of at most $\sqrt{q^n/2^m}$ to a uniform distribution over $\mathbb{Z}_q^n$. We wish to make the statistical distance negligible in the security parameter κ. The dimension m has a certain lower bound in order to achieve this, namely

$$\sqrt{\frac{q^n}{2^m}} \leq 2^{-\kappa} \iff \frac{q^n}{2^m} \leq 2^{-2\kappa} \iff q^n \leq 2^{m-2\kappa} \iff n \log q \leq m - 2\kappa \iff m \geq n \log q + 2\kappa.$$

It can be concluded that for $m \geq n \log q + 2\kappa$ the expected statistical distance between the distribution of $\mathbf{AR}$ and the uniform distribution is negligible in κ. Therefore, the adversary has only a negligible (in κ) advantage in game 2, the ideal game, in comparison to game 1.

Combining the above conclusions, the adversary has the same view and can therefore learn the same in all three games up to a negligible amount in κ. This proves the IND-CPA security in the security parameter κ. □

Theorem 4.5 (IND-CPA security of the Multi-Key FHE scheme)**.** *Let n, m be the dimensions and q the modulus of the cryptosystem and $\bar{\Psi}_q^\alpha$ over $\mathbb{Z}_q$ the noise distribution with $\alpha \in (0,1)$ a real number such that $\alpha q \geq 2\sqrt{n-1}$ that define the encryption scheme. If $m \geq n \log_2 q + 2\kappa$ and assuming the hardness of the $LWE_{n-1,q,\bar{\Psi}_\alpha}$ problem, the Multi-Key FHE scheme from Section 3.3.2 with the encryption procedure from Section 4.1.3 is IND-CPA secure with κ the security parameter.*

Proof. We again define several games for which we prove that the view of the adversary is equivalent:

- *Game 0:* The real IND-CPA game,
- *Game 1:* The IND-CPA game where public keys $\mathbf{b}, \mathbf{D}$ and the common matrix $\mathbf{A}$ are chosen uniformly at random and send to the adversary,
- *Game 2:* The same as the previous game, but now also the ciphertexts and the public key $\mathbf{P}$ are chosen uniformly at random; the ideal game.

It can be seen that in the real game, $\mathbf{A}$ is chosen to be uniformly random and therefore also $\mathbf{b}$ is uniformly random due to the way it is constructed. According to Theorem 3.17 with $\alpha q \geq 2\sqrt{n-1}$, we can replace the samples out of which $\bar{\mathbf{D}}$ is constructed by uniformly random samples in $\mathbb{Z}_q^n$ since the adversary is polynomial time-bounded. Concluding, the adversary has no advantage in game 0 compared to game 1.

The encryption of any ciphertext μ is given by $\mathbf{C} = \begin{pmatrix} \mathbf{A}'^\intercal & -\mathbf{b}+\mathbf{a} \end{pmatrix}^\intercal \mathbf{R} + \mu(\mathbf{I}_n \otimes \mathbf{g}^\intercal) \in \mathbb{Z}_q^{n \times n\ell}$ with $\mathbf{R} \leftarrow \{0,1\}^{m \times n\ell}$ uniformly. Next to this, $\mathbf{P} = \mathbf{AR}' + (\mathbf{I}_n \otimes \mathbf{t}^\intercal \otimes \mathbf{g}^\intercal) \in \mathbb{Z}_q^{n \times n^2 l}$ with $\mathbf{R}' \leftarrow \{0,1\}^{m \times n^2 l}$ uniformly. Both the matrix $\begin{pmatrix} \mathbf{A}'^\intercal & -\mathbf{b}+\mathbf{a} \end{pmatrix}^\intercal$ and $\mathbf{A}$ are uniformly random since $\mathbf{b}$ is uniformly random and $\mathbf{A}'$ and $\mathbf{a}$ are parts of $\mathbf{A}$. Using the same reasoning as in the proof of Theorem 4.4, the Leftover Hash Lemma in Lemma 4.3 gives that the distributions of $\mathbf{C}$ and $\mathbf{P}$ are close to uniform. Namely, for $m \geq n \log q + 2\kappa$ the expected statistical distance between the distribution and the uniform distribution is negligible in κ. Therefore, the adversary has only a negligible (in κ) advantage in game 2, the ideal game, in comparison to game 1. So, the adversary has the same view and can therefore learn the same in all three games up to a negligible amount in κ. This proves the theorem. □

Additionally, it has to be proved that the partial decryptions in the collaborative decryption procedure do not give away additional information. Namely, in some of the protocols these partial decryptions are send to other parties such that they can be used to find the full decryption. This can be done by showing that the partial decryption of a party i can be simulated given the plaintext and the secret keys of all other parties. In other words, the partial decryption of party i can be simulated without knowing its secret key. This property tells us that the partial decryption does not reveal additional information about the secret key of party i [80]. We call this *simulation security*. Simulations are often used for proving security [71]. Below we define the simulation security of our collaborative decryption procedure. The proof can be found in [80].

Theorem 4.6 (Security of collaborative decryption procedure [80])**.** *Let $\mathbf{C} \in \mathbb{Z}_q^{nk \times nk\ell}$ be a U_τ-noisy ciphertext of message μ under the secret key $\mathbf{t}$, which is a concatenation of the secret keys of $k \geq 1$ parties. The collaborative decryption procedure within the Multi-Key FHE scheme, explained in Section 4.1.3, satisfies* simulation security *in the security parameter κ when $\frac{U}{B_{\text{smudge}}} = \text{negl}(\kappa)$.*

5

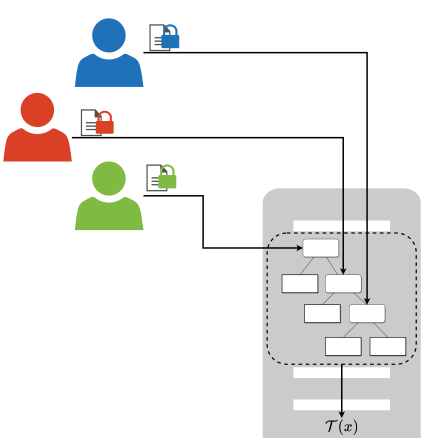
Collaborative Private Decision-Tree Evaluation protocols

In the previous chapter we gave detailed analyses of the correctness and security of the encryption schemes that are used in our protocols. In addition, we made some adaptations essential for the functionality of our protocols. In this chapter we describe our private decision-tree evaluation protocols. We consider a collaborative setting, where the input variables originate from more than one user. These users communicate with a server that holds the decision tree. The goal of our protocols is to perform the collaborative evaluation in a privacy-preserving way: the server should not get to know anything about the input attributes of all users (except for the output of the decision tree evaluation for some protocols), and the users should not get to know anything about the decision tree.

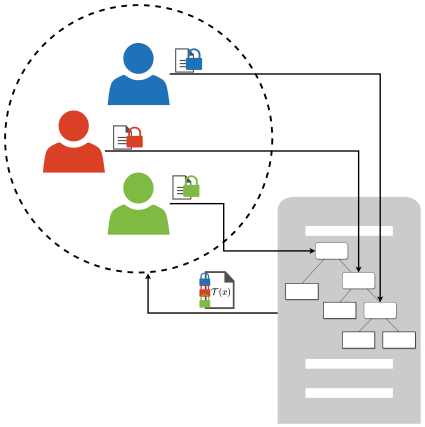
In the context of access control, the server that holds the decision tree does the evaluation and has to receive the evaluation result in order to make the access decision. In other contexts, where the decision tree is used as an external resource, the parties need to receive the evaluation result themselves. An example of such a scenario is when multiple banks work together to detect fraud. Therefore, two scenarios are considered that are defined in Definitions 5.1 and 5.2. Either the server gets the final classification result, or the users get the result. In the scenario where the server learns the classification label the server can deduce the possible tree paths taken by the users' input. This is inherent to the problem statement.

Definition 5.1 (Multi-Party Server Private Decision-Tree Evaluation)**.** *Let k users have private inputs $\mathbf{x}^{(i)}$ for $i \in \{0,1,\cdots,k-1\}$ and a server hold a decision tree $\mathcal{T} = (\mathcal{D} \cup \mathcal{L}, E)$. A* Multi-Party Server Private Decision-Tree Evaluation *of $\mathcal{T}$ is an evaluation of $\mathcal{T}$ on $\mathbf{x} = \left(\mathbf{x}^{(0)},\ldots,\mathbf{x}^{(k-1)}\right)$, where the users do not learn anything and the server only learns $\mathcal{T}(\mathbf{x})$.*

Definition 5.2 (Multi-Party User Private Decision-Tree Evaluation)**.** *Let k users have private inputs $\mathbf{x}^{(i)}$ for $i \in \{0,1,\cdots,k-1\}$ and a server hold a decision tree $\mathcal{T} = (\mathcal{D} \cup \mathcal{L}, E)$. A* Multi-Party User Private Decision-Tree Evaluation *of $\mathcal{T}$ is an evaluation of $\mathcal{T}$ on $\mathbf{x} = \left(\mathbf{x}^{(0)},\ldots,\mathbf{x}^{(k-1)}\right)$, where the server does not learn anything and the users only learn $\mathcal{T}(\mathbf{x})$.*



(a) The scenario of Definition 5.1 where the server receives the final evaluation result, as for the access-control use-case.



(b) The scenario of Definition 5.2 where the users receive the final evaluation result, for example when multiple banks work together for fraud detection.

Figure 5.1: Demonstration of the two different scenarios considered in this thesis, defined in Definitions 5.1 and 5.2.

We propose 6 different types of of protocols of which the first four are placed in both settings above. The fifth and sixth protocol can only be applied in the second setting. Our protocols either use the multiplicative [107] path evaluation or the additive path evaluation [104] described in Section 3.5. Either Multi-Key FHE is used or a Semi-Trusted Third Party (STTP), with our without key-switching. Each protocol has their own properties and trade-offs. While the advantage of the Multi-Key FHE protocols is that the users can encrypt under their *own* key and do not need to put their trust anywhere else, it requires a server with high computational power due to the higher complexity of the encryption scheme and the fact that it has to extend all incoming ciphertexts to multiple keys. The protocols using an STTP require such an additional party to be found that generates a key pair that is used by the users for encryption. Additionally, this party does the decryption. Finding such a party should not be a hard task since this party does not gain any knowledge about the users' input and server's tree due to randomisation. Lastly, the protocol that in addition uses a procedure called key-switching, makes the trust that needs to be put in the STTP even lower, since now the decryption can be done by the users themselves. It is only required that they generate some keys in communication with the server, which costs an additional key-exchange communication round. The protocols that use an STTP have the overall benefit that the GSW FHE encryption scheme can be used, which yields a lower complexity than the multi-key encryption scheme.

In this thesis it is assumed that the corrupted parties are passive (a.k.a. semi-honest or honest-but-curious), which means that these parties execute the protocol correctly. In our protocols, when a user colludes with the server, both do not gain any additional knowledge. This also holds when the STTP colludes with one of the users. Only in case the server colludes with the STTP, the server can gain knowledge about the input of the users.

We now describe all protocols in detail. First, we introduce the notation used in the protocols. In Sections 5.2 and 5.3, our protocols for both the multiplicative as the additive approach are given that use Multi-Key FHE. In Section 5.4 and 5.5 we give four protocols that an STTP with or without key-switching, both using the GSW FHE encryption scheme. Then, in Section 5.6, we compare several comparison protocols to perform the decision-node evaluations and give a private equality test. These are all building blocks of the first part of our protocols, which we treat as a black box in the first sections.

5.1. Notation

The input variables originate from $k \geq 2$ users that communicate with a server that holds the decision tree. The users' inputs are given as a vector $\mathbf{x}^{(i)} \in \mathbb{Z}_p^{a_i}$ for $i \in \{0,1,\cdots,k-1\}$ with a_i the number of input variables of user i and $p \geq$ some integer. In total there are $A = \sum_{i=0}^{k-1} a_i$ input variables. These are encrypted bit-wise by the k users. An encryption of an element $\mathbf{y} \in \mathbb{Z}_p$ with plaintext space $\mathbb{Z}_p$ for some integer $p \geq 2$ is denoted by $[[\mathbf{y}]]_p$. Homomorphic multiplication and addition are denoted by the symbols $\boxdot$ and $\boxplus$ respectively. Let $\mathbf{x} \in \mathbb{Z}_p^A$ be the concatenation of all $\mathbf{x}^{(i)}$, so $\mathbf{x} = \left(\mathbf{x}^{(0)},\ldots,\mathbf{x}^{(k-1)}\right)$. As described in Section 3.4, the evaluation of decision tree $\mathcal{T} = (\mathscr{D} \cup \mathscr{L}, E)$ on input $\mathbf{x} = \left(\mathbf{x}^{(0)},\ldots,\mathbf{x}^{(k-1)}\right)$ is given by $\mathcal{T}(\mathbf{x})$ with $\mathcal{T} : \mathbb{Z}_p^A \mapsto \{c_1,\ldots,c_v\}$ the decision-tree evaluation function with class labels $\{c_1,\ldots,c_v\}$. The bit-wise representation of a final evaluation label is denoted by $\mathcal{T}(\mathbf{x})_0\mathcal{T}(\mathbf{x})_1\ldots\mathcal{T}(\mathbf{x})_\rho$ with ρ the bit-length of the class label. The evaluation makes use of the functions t, m and c. We define $\sigma := |\mathscr{D}|$ and $\gamma := |\mathscr{L}|$. Also, for a decision node $d \in \mathscr{D}$, d.i denotes its i-th child node and d.children the set of child nodes of d. Additionally, d.cmp denotes the bit result of the decision-node evaluation by the black box comparison or equality protocol. The collaborative decryption of a multi-key ciphertext is split in two parts, explained in Section 4.1.3. In the protocols these parts are denoted by two algorithms, `PartialDecrypt` and `CombinePartials`. The first algorithm is the calculation of the partial decryption with the secret key of the specific party. The second algorithm sums these partial decryptions to find the full decryption.

5.2. Multiplicative protocols using Multi-Key Fully Homomorphic Encryption

Multi-Key FHE is an approach to build secure MPC protocols based on homomorphic encryption, and requires only two communication rounds. Therefore, we propose two collaborative PDTE protocols (for both scenarios one) using Multi-Key FHE. The advantage is that all users can encrypt their input under their *own* key and send it to the server together with their public extension key. They do not have to interact any more, except when the complete tree is evaluated and they have to partially decrypt the result. The server is able to combine the inputs of all users by extending all ciphertexts to all the keys of the users. The server does not have to generate keys himself. Since the multi-key encryption scheme requires a commonly known matrix, the protocols take place in the CRS model in which it is assumed that all parties have access to a common random string.

The protocols in this section make use of the multiplicative path evaluation [107] method, presented in Section 3.5. The complete tree evaluation algorithm `Evaluate` is given in Algorithm 4. As a reminder, only 1-values are stored on the correct evaluation path in the tree and 0's elsewhere. The final label is found by multiplying the path evaluation result with each leaf node label. Note here that we do this bit-wise to stay in the $\mathbb{Z}_2$ plaintext space; we first translate every class label to its bit representation, after which every bit is multiplied by the path evaluation result. By adding these values, only the correct path's label bits are counted and returned. The correctness of this method follows directly from Lemma 1 in [107]. We improve upon [107] by introducing the plaintext multiplication such that the server does not need to encrypt the leaf labels. By using plaintext multiplications instead of ciphertext multiplication, the noise increases less.

In Protocol 1, our Multi-Party User Private Decision-Tree Evaluation protocol based on Multi-Key FHE is given. Upon receiving all encrypted input variables, the server evaluates the tree. He sends the encrypted result to all users. Each user then partially decrypts the results and communicates the partial decryptions with all other users. Every user can combine these to get the evaluation result in plaintext.

Clearly, the encryptions of the users do not reveal anything to the server since these are encrypted by the IND-CPA secure Multi-Key FHE scheme. According to Theorem 4.6, the partial decryptions do not reveal any information about the secret key used for the partial decryption. Therefore, the users do not gain any information other than the final evaluation result.

The Multi-Party Server Private Decision-Tree Evaluation protocol is shown in Protocol 2. It works in the same way as the previous protocol. The difference is that now the users send their partial decryption to the server, such that the server can obtain the evaluation result in plaintext. The server does learn the possible paths taken in the decision tree, which is inherent to the problem statement. For additional security, the server adds a random value R to the encrypted result such that when the users share their partial encryption with each other, they still can not find the result.

Algorithm 4

function EVALUATE($\mathscr{D}, \mathscr{L}, [[\mathbf{x}]]_2$)
  for all $d \in \mathscr{D}$ **do**
    if d a threshold node **then**
      $[[d.\text{right.cmp}]]_2 \leftarrow [[\mathbf{x}_{\text{m}(d)} > \text{t}(d)]]_2$
      $[[d.\text{left.cmp}]]_2 \leftarrow [[\mathbf{x}_{\text{m}(d)} > \text{t}(d)]]_2 \boxplus 1$
    if d a categorical node **then**
      $[[d.\text{right.cmp}]]_2 \leftarrow [[\mathbf{x}_{\text{m}(d)} = \text{t}(d)]]_2$
      $[[d.\text{left.cmp}]]_2 \leftarrow [[\mathbf{x}_{\text{m}(d)} = \text{t}(d)]]_2 \boxplus 1$
  for all $d \in \mathscr{D}$ **do**
    If Multi-Key FHE scheme is used: extend $[[d.\text{cmp}]]_2$ to all users' keys
    If key-switching is used: switch $[[d.\text{cmp}]]_2$ to an encryption under the public key of the STTP
  Let Q be an empty queue
  for all $v \in$ root.children **do**
    Q.`enqueue`(v)
  while Q.`empty`() = false **do**
    $v \leftarrow Q$.`dequeue`()
    for all $w \in v$.children **do**
      $[[w.\text{cmp}]]_2 \leftarrow [[w.\text{cmp}]]_2 \boxdot [[v.\text{cmp}]]_2$
      Q.`enqueue`(w)
  $\rho \leftarrow$ highest bit-length of class labels
  for all $i \in \{0, 1, \ldots, \rho\}$ **do**
    $[[\text{result}_i]]_2 \leftarrow [[0]]_2$
  for all $l \in \mathscr{L}$ **do**
    $c_0 c_1 \ldots c_\rho \leftarrow$ bit representation of $\text{c}(l)$
    for all $i \in \{0, 1, \ldots, \rho\}$ **do**
      $[[\text{result}_i]]_2 \leftarrow [[\text{result}_i]]_2 \boxplus \left([[l.\text{cmp}]]_2 \boxdot c_i\right)$
  return $[[\text{result}_0]]_2, \ldots, [[\text{result}_\rho]]_2$

Protocol 1: Multi-Party User Private Decision-Tree Evaluation Protocol based on Multi-Key FHE.

User i		**Server**
Input: $\mathbf{x}^{(i)}, \mathrm{pk}_i, \mathrm{sk}_i$ **Output:** $\mathscr{T}(\mathbf{x})$		**Input:** $\mathscr{D}, \mathscr{L}$ **Output:** –
	$\xrightarrow{\mathrm{pk}_i, [[\mathbf{x}^{(i)}]]_2}$	
		$[[\mathbf{x}]]_2 \leftarrow \left([[\mathbf{x}^{(0)}]]_2, [[\mathbf{x}^{(1)}]]_2, \ldots, [[\mathbf{x}^{(k-1)}]]_2\right)$ $[[\mathscr{T}(\mathbf{x})_0]]_2, \ldots, [[\mathscr{T}(\mathbf{x})_\rho]]_2 \leftarrow \texttt{Evaluate}(\mathscr{D}, \mathscr{L}, [[\mathbf{x}]]_2)$
	$\xleftarrow{[[\mathscr{T}(\mathbf{x})_0]]_2, \ldots, [[\mathscr{T}(\mathbf{x})_\rho]]_2}$	
for $j = 0, \ldots, \rho$ **do:** $p_j^{(i)} \leftarrow \texttt{PartialDecrypt}([[\mathscr{T}(\mathbf{x})_j]]_2)$ Send $p_0^{(i)}, \ldots, p_\rho^{(i)}$ to all other users (not server) **for** $j = 0, \ldots, \rho$ **do:** $\mathscr{T}(\mathbf{x})_j \leftarrow \texttt{CombinePartials}(p_j^{(0)}, \ldots, p_j^{(k-1)})$		

Protocol 2: Multi-Party Server Private Decision-Tree Evaluation Protocol based on Multi-Key FHE.

User i		**Server**
Input: $\mathbf{x}^{(i)}, \mathrm{pk}_i, \mathrm{sk}_i$ **Output:** –		**Input:** $\mathscr{D}, \mathscr{L}$ **Output:** $\mathscr{T}(\mathbf{x})$
	$\xrightarrow{\mathrm{pk}_i, [[\mathbf{x}^{(i)}]]_2}$	
		$[[\mathbf{x}]]_2 \leftarrow \left([[\mathbf{x}^{(0)}]]_2, [[\mathbf{x}^{(1)}]]_2, \ldots, [[\mathbf{x}^{(k-1)}]]_2\right)$ $[[\mathscr{T}(\mathbf{x})_0]]_2, \ldots, [[\mathscr{T}(\mathbf{x})_\rho]]_2 \leftarrow \texttt{Evaluate}(\mathscr{D}, \mathscr{L}, [[\mathbf{x}]]_2)$ **for** $j = 0, \ldots, \rho$ **do:** $R_j \leftarrow$ uniformly random bit $[[\mathscr{T}'(\mathbf{x})_j]]_2 \leftarrow [[\mathscr{T}(\mathbf{x})_j]]_2 \boxplus R_j$
	$\xleftarrow{[[\mathscr{T}'(\mathbf{x})_0]]_2, \ldots, [[\mathscr{T}'(\mathbf{x})_\rho]]_2}$	
for $j = 0, \ldots, \rho$ **do:** $p_j^{(i)} \leftarrow \texttt{PartialDecrypt}([[\mathscr{T}'(\mathbf{x})]]_2)$		
	$\xrightarrow{p_0^{(i)}, \ldots, p_\rho^{(i)}}$	
		for $j = 0, \ldots, \rho$ **do:** $\mathscr{T}(\mathbf{x})_j \leftarrow \texttt{CombinePartials}(p_j^{(0)}, \ldots, p_j^{(k-1)}) - R_j$

5.3. Additive protocols using Multi-Key Fully Homomorphic Encryption

From Section 4.2.2 it can be concluded that the ciphertext noise is lower after a homomorphic addition than after a homomorphic multiplication. The lower the total noise propagation, the smaller we can set the parameters of the encryption scheme, which is beneficial for the efficiency of the protocol. We therefore propose two protocols using Multi-Key FHE, where the additive path evaluation approach from Section 3.5 is used. The main difference with the previous two protocols is that now the tree evaluation algorithm of Algorithm 5 is used, which uses additions instead of multiplications for the path evaluation.

As a reminder, this approach works by calculating for every path a cost; the addition of all decision bits on the path. Now, each path cost can have a value up to the depth of the tree. Therefore, the ciphertexts have the plaintext space $\{0,1,\ldots,D\}$ with D the depth of the tree $\mathcal{T}$. In case $D+1$ is not a prime, we see that the required multiplication with a random value can result in a zero value which is not desired. As an example, take $D+1=4$, then when $l.\text{cmp}=2$ and $r=2$ the result is $2\cdot 2=4\equiv 0 \mod D+1$. Therefore, we need $D+1$ to be prime. We extend the plaintext space to $\{0,1,\ldots,D'\}$ with $D'+1$ the first prime number that is bigger than D. Dividing the $\mathbb{Z}_q$ space in $D'+1$ parts can be done by choosing $q=(D'+1)^l$ and using a different gadget vector as introduced in Section 4.1.6.

After the path evaluation, only the correct path has zero-cost. Now for every leaf node the encrypted path cost and corresponding encrypted leaf label is communicated back to the users. The decryption works in the same way as for the previous two protocols. Randomising every path cost by a random value makes sure that all non-zero path costs do not give away any additional information [104]. Since we introduced the plaintext addition, we do not need to encrypt the label in contrast to the work of Tai et al. [104]. For the leaf node $l\in\mathscr{L}$ with path cost equal to 0, we have $[[l.\text{cmp}]]_{D'+1}=[[0]]_{D'+1}$ and therefore $[[\text{cost}_l]]_{D'+1}=[[0]]_{D'+1}$. Thus, once the final user decrypts the label of the zero-cost path this results in the result $\text{c}(l)$. This immediately shows the correctness of the protocol. The labels are again communicated bit-wisely, since the possible number of leaf nodes can be higher than $D'+1$, which is the plaintext space size.

Since these protocols take place in the plaintext space $\{0,1,\ldots,D'\}$ the decision bits have to be stored at the child nodes in a different way than for the protocols that use the multiplicative path evaluation. Namely, when the result of a comparison or equality protocol is equal to 1, we would like to store 0 at the right child node. But this can not be realised by adding a 1 since now $1+1\equiv 2 \mod D'+1$. We solve this by calculating $1\boxplus\big(D'\boxdot[[\mathbf{x}_{\text{m}(d)}>\text{t}(d)]]_{D'+1}\big)$. When $[[\mathbf{x}_{\text{m}(d)}>\text{t}(d)]]_{D'+1}=[[1]]_{D'+1}$ the result is $1\boxplus(D'\boxdot[[1]]_{D'+1})=[[D'+1]]_{D'+1}\equiv[[0]]_{D'+1}$. When $[[\mathbf{x}_{\text{m}(d)}>\text{t}(d)]]_{D'+1}=[[0]]_{D'+1}$ the result is $[[1]]_{D'+1}$. Therefore, the functionality is as desired. We can do the same for the categorical nodes.

Our Multi-Party User Private Decision-Tree Evaluation protocol according to the additive approach can be seen in Protocol 3. This protocol requires more communication, since for every leaf node both the encrypted path cost and label are send from the server to the users. Namely, in order to find the classification label of the zero-cost path, all path costs need to be decrypted. This does give away the number of leaf nodes in the tree. Randomly permuting the order of the communicated ciphertext makes sure that the location if the zero-cost path within the tree stays hidden for the users.

Our protocol in the second scenario, where the server learns the evaluation label instead of the users, can be seen in Protocol 4. Since the users do not have to receive the result, it is sufficient that only one of the users finds the path with a path cost of 0 using the partial decryption of all other users. Again, we introduce the random variable R which is only known by the server, to hide the exact value of the evaluation label. Once the final user decrypts the label of the zero-cost path, he finds the value $\text{c}(l)+R$ which is send to the server. Clearly, the server can then find $\mathcal{T}(\mathbf{x})$ by subtracting R.

Algorithm 5

function EVALUATE+($\mathscr{D}, \mathscr{L}, [\![\mathbf{x}]\!]_{D'+1}$)
  for all $d \in \mathscr{D}$ **do**
    if d a threshold node **then**
      $[\![d.\text{right.cmp}]\!]_{D'+1} \leftarrow 1 \boxplus \left(D' \boxdot [\![\mathbf{x}_{\text{m}(d)} > \text{t}(d)]\!]_{D'+1}\right)$
      $[\![d.\text{left.cmp}]\!]_{D'+1} \leftarrow [\![\mathbf{x}_{\text{m}(d)} > \text{t}(d)]\!]_{D'+1}$
    if d a categorical node **then**
      $[\![d.\text{right.cmp}]\!]_{D'+1} \leftarrow 1 \boxplus \left(D' \boxdot [\![\mathbf{x}_{\text{m}(d)} = \text{t}(d)]\!]_{D'+1}\right)$
      $[\![d.\text{left.cmp}]\!]_{D'+1} \leftarrow [\![\mathbf{x}_{\text{m}(d)} = \text{t}(d)]\!]_{D'+1}$
  for all $d \in \mathscr{D}$ **do**
    If Multi-Key FHE scheme is used: extend $[\![d.\text{cmp}]\!]_{D'+1}$ to all users' keys
    If key-switching is used: switch $[\![d.\text{cmp}]\!]_2$ to an encryption under the public key of the STTP
  Let Q be an empty queue
  for all $v \in$ root.children **do**
    Q.`enqueue`(v)
  while Q.`empty`() = false **do**
    $v \leftarrow Q$.`dequeue`()
    for all $w \in v$.children **do**
      $[\![w.\text{cmp}]\!]_{D'+1} \leftarrow [\![w.\text{cmp}]\!]_{D'+1} \boxplus [\![v.\text{cmp}]\!]_{D'+1}$
      Q.`enqueue`(w)
  $\rho \leftarrow$ highest bit-length of class labels
  for all $l \in \mathscr{L}$ **do**
    $c_0 c_1 \dots c_\rho \leftarrow$ bit representation of c(l)
    $r \leftarrow$ uniformly random from $\{1, 2, \dots, D'\}$
    $[\![\text{cost}_l]\!]_{D'+1} \leftarrow r \boxdot [\![l.\text{cmp}]\!]_{D'+1}$
    for all $i \in \{0, 1, \dots, \rho\}$ **do**
      $r' \leftarrow$ uniformly random from $\{1, 2, \dots, D'\}$
      $[\![\text{label}_{l,i}]\!]_{D'+1} \leftarrow \left(r' \boxdot [\![l.\text{cmp}]\!]_{D'+1}\right) \boxplus c_i$
  return $([\![\text{cost}_l]\!]_{D'+1}, [\![\text{label}_{l,0}]\!]_{D'+1}, \dots, [\![\text{label}_{l,\rho}]\!]_{D'+1})$ for all $l \in \mathscr{L}$

Protocol 3: Additive Multi-Party User Private Decision-Tree Evaluation Protocol based on Multi-Key FHE.

User i		**Server**
Input: $\mathbf{x}^{(i)}, \mathrm{pk}_i, \mathrm{sk}_i$ **Output:** $\mathscr{T}(\mathbf{x})$		**Input:** $\mathscr{D}, \mathscr{L}$ **Output:** –
	$\xrightarrow{\mathrm{pk}_i, [[\mathbf{x}^{(i)}]]_{D'+1}}$	
		$[[\mathbf{x}]]_{D'+1} \leftarrow \left([[\mathbf{x}^{(0)}]]_{D'+1}, [[\mathbf{x}^{(1)}]]_{D'+1}, \ldots, [[\mathbf{x}^{(k-1)}]]_{D'+1}\right)$ $\left([[\mathrm{cost}_l]]_{D'+1}, [[\mathrm{label}_{l,0}]]_{D'+1}, \ldots, [[\mathrm{label}_{l,\rho}]]_{D'+1}\right)_l \leftarrow \texttt{Evaluate+}(\mathscr{D}, \mathscr{L}, [[\mathbf{x}]]_{D'+1})$ $P \leftarrow$ Random permutation $\Phi := \left([[\mathrm{cost}_{P(l)}]]_{D'+1}, [[\mathrm{label}_{P(l),0}]]_{D'+1}, \ldots, [[\mathrm{label}_{P(l),\rho}]]_{D'+1}\right)_l$
	$\xleftarrow{\Phi}$	
for $j = 1, \ldots, \gamma$ **do:** $c_j^{(i)} \leftarrow \texttt{PartialDecrypt}([[\mathrm{cost}_j]]_{D'+1})$ **for** $x = 0, \ldots, \rho$ **do:** $r_{j,x}^{(i)} \leftarrow \texttt{PartialDecrypt}([[\mathrm{label}_{j,x}]]_{D'+1})$ Send $(c_j^{(i)}, r_{j,0}^{(i)}, \ldots, r_{j,\rho}^{(i)})_j$ to all other users (not server) Find j with $\texttt{CombinePartials}(c_j^{(0)}, \ldots, c_j^{(k-1)}) = 0$ **for** $x = 0, \ldots, \rho$ **do:** $\mathscr{T}(\mathbf{x})_x \leftarrow \texttt{CombinePartials}(r_{j,x}^{(0)}, \ldots, r_{j,x}^{(k-1)})$		

Protocol 4: Additive Multi-Party Server Private Decision-Tree Evaluation Protocol based on Multi-Key FHE.

User i		**Server**
Input: $\mathbf{x}^{(i)}, \mathrm{pk}_i, \mathrm{sk}_i$ **Output:** –		**Input:** $\mathscr{D}, \mathscr{L}$ **Output:** $\mathscr{T}(\mathbf{x})$
	$\xrightarrow{\mathrm{pk}_i, [[\mathbf{x}^{(i)}]]_{D'+1}}$	
		$[[\mathbf{x}]]_{D'+1} \leftarrow \left([[\mathbf{x}^{(0)}]]_{D'+1}, [[\mathbf{x}^{(1)}]]_{D'+1}, \ldots, [[\mathbf{x}^{(k-1)}]]_{D'+1}\right)$ $\left(\left[[\mathrm{cost}_l]\right]_{D'+1}, [[\mathrm{label}_{l,0}]]_{D'+1}, \ldots, [[\mathrm{label}_{l,\rho}]]_{D'+1}\right)_l \leftarrow \texttt{Evaluate+}(\mathscr{D}, \mathscr{L}, [[\mathbf{x}]]_{D'+1})$ $P \leftarrow$ Random permutation $R_0, \ldots, R_\rho \leftarrow$ uniformly random bit $\Phi := \left([[\mathrm{cost}_{P(l)}]]_{D'+1}, [[\mathrm{label}_{P(l),0}]]_{D'+1} \boxplus R_0, \ldots, [[\mathrm{label}_{P(l),\rho}]]_{D'+1} \boxplus R_\rho\right)_l$
	$\xleftarrow{\Phi}$	
for $j = 1, \ldots, \gamma$ **do:** $c_j^{(i)} \leftarrow \texttt{PartialDecrypt}([[\mathrm{cost}_j]]_{D'+1})$ Send $c_1^{(i)}, \ldots, c_\gamma^{(i)}$ to all other users (not server) Find j with $\texttt{CombinePartials}(c_j^{(0)}, \ldots, c_j^{(k-1)}) = 0$ **for** $x = 0, \ldots, \rho$ **do:** $r_x^{(i)} \leftarrow \texttt{PartialDecrypt}([[\mathrm{label}_x]]_{D'+1})$		
	$\xrightarrow{r_0^{(i)}, \ldots, r_\rho^{(i)}}$	
		for $j = 0, \ldots, \rho$ **do:** $\mathscr{T}(\mathbf{x})_j \leftarrow \texttt{CombinePartials}(r_j^{(0)}, \ldots, r_j^{(k-1)}) - R_j$

5.4. Semi-Trusted Third Party protocols using GSW Fully Homomorphic Encryption

The next four protocols we propose use an additional party. Only this party is required to do the key generation of the encryption scheme. All users encrypt under the public key pk of this party. This means that instead of a Multi-Key FHE scheme, we can use the GSW FHE scheme. Namely, the server receives encrypted input from the users that are all encrypted under the same key. The server sends the evaluation result to this additional party that does the decryption. Since the server randomised the result by adding a value that is only known by him and/or the users, this additional party does not gain any knowledge. We therefore call this additional party an STTP; this party is only involved in the protocol by performing the key generation and decryption properly. Even when the STTP colludes with one of the users, this user can not see the evaluation result since this is randomised by the server.

The result is a more straightforward and efficient protocol. Finding such an STTP should not be a hard task since the STTP does not get any insight into the input of the other parties. The protocols that make use of STTP in combination with the multiplicative approach in Algorithm 4, can be seen in Protocols 5 and 6. The protocols that make use of the additive approach of Algorithm 5, can be seen in 7 and 8. The setup and key generation does add an extra round compared to the previous protocols, since the public key pk of the STTP has to be shared with all users.

Protocol 5: Multi-Party User Private Decision-Tree Evaluation Protocol using GSW FHE and an STTP.

STTP		**User** i		**Server**
Input: pk, sk **Output:** –		**Input:** $\mathbf{x}^{(i)}$ **Output:** $\mathcal{T}(\mathbf{x})$		**Input:** $\mathcal{D}, \mathcal{L}$ **Output:** –
	$\xrightarrow{\text{pk}}$			
		Encrypt $\mathbf{x}^{(i)}$ under received pk		
			$\xrightarrow{[[\mathbf{x}^{(i)}]]_2}$	
				$[[\mathbf{x}]]_2 \leftarrow \left([[\mathbf{x}^{(0)}]]_2, [[\mathbf{x}^{(1)}]]_2, \ldots, [[\mathbf{x}^{(k-1)}]]_2\right)$ $[[\mathcal{T}(\mathbf{x})_0]]_2, \ldots, [[\mathcal{T}(\mathbf{x})_\rho]]_2 \leftarrow \texttt{Evaluate}(\mathcal{D}, \mathcal{L}, [[\mathbf{x}]]_2)$ **for** $j = 0, \ldots, \rho$ **do:** $R_j \leftarrow$ uniformly random bit $[[\mathcal{T}'(\mathbf{x})_j]]_2 \leftarrow [[\mathcal{T}(\mathbf{x})_j]]_2 \boxplus R_j$
			$\xleftarrow{R_0, \ldots, R_\rho}$	
	$\xleftarrow{[[\mathcal{T}'(\mathbf{x})_0]]_2, \ldots, [[\mathcal{T}'(\mathbf{x})_\rho]]_2}$			
for $j = 0, \ldots, \rho$ **do:** $\mathcal{T}'(\mathbf{x})_j \leftarrow \texttt{Decrypt}([[\mathcal{T}'(\mathbf{x})_j]]_2)$				
	$\xrightarrow{\mathcal{T}'(\mathbf{x})_0, \ldots, \mathcal{T}'(\mathbf{x})_\rho}$			
		for $j = 0, \ldots, \rho$ **do:** $\mathcal{T}(\mathbf{x})_j \leftarrow \mathcal{T}'(\mathbf{x})_j - R_j$		

Protocol 6: Multi-Party Server Private Decision-Tree Evaluation Protocol using GSW FHE and an STTP.

User i	**Server**	**STTP**
Input: $\mathbf{x}^{(i)}$ **Output:** –	**Input:** $\mathscr{D}, \mathscr{L}$ **Output:** $\mathscr{T}(\mathbf{x})$	**Input:** pk, sk **Output:** –

$\xleftarrow{\text{pk}}$

Encrypt $\mathbf{x}^{(i)}$ under received pk

$\xrightarrow{[[\mathbf{x}^{(i)}]]_2}$

$[[\mathbf{x}]]_2 \leftarrow \left([[\mathbf{x}^{(0)}]]_2, [[\mathbf{x}^{(1)}]]_2, \ldots, [[\mathbf{x}^{(k-1)}]]_2\right)$

$[[\mathscr{T}(\mathbf{x})_0]]_2, \ldots, [[\mathscr{T}(\mathbf{x})_\rho]]_2 \leftarrow \texttt{Evaluate}(\mathscr{D}, \mathscr{L}, [[\mathbf{x}]]_2)$

for $j = 0, \ldots, \rho$ **do:**

$R_j \leftarrow$ uniformly random bit

$[[\mathscr{T}'(\mathbf{x})_j]]_2 \leftarrow [[\mathscr{T}(\mathbf{x})_j]]_2 \boxplus R_j$

$\xrightarrow{[[\mathscr{T}'(\mathbf{x})_0]]_2, \ldots, [[\mathscr{T}'(\mathbf{x})_\rho]]_2}$

for $j = 0, \ldots, \rho$ **do:**

$\mathscr{T}'(\mathbf{x})_j \leftarrow \texttt{Decrypt}([[\mathscr{T}'(\mathbf{x})_j]]_2)$

$\xleftarrow{\mathscr{T}'(\mathbf{x})_0, \ldots, \mathscr{T}'(\mathbf{x})_\rho}$

for $j = 0, \ldots, \rho$ **do:**

$\mathscr{T}(\mathbf{x})_j \leftarrow \mathscr{T}'(\mathbf{x})_j - R_j$

Protocol 7: Additive Multi-Party User Private Decision-Tree Evaluation Protocol using GSW FHE and an STTP.

STTP		**User** i		**Server**
Input: pk, sk **Output:** –		**Input:** $\mathbf{x}^{(i)}$ **Output:** $\mathcal{T}(\mathbf{x})$		**Input:** $\mathscr{D}, \mathscr{L}$ **Output:** –
	$\xrightarrow{\text{pk}}$			
		Encrypt $\mathbf{x}^{(i)}$ under received pk		
			$\xrightarrow{[\![\mathbf{x}^{(i)}]\!]_{D'+1}}$	
				$[\![\mathbf{x}]\!]_{D'+1} \leftarrow \left([\![\mathbf{x}^{(0)}]\!]_{D'+1}, [\![\mathbf{x}^{(1)}]\!]_{D'+1}, \ldots, [\![\mathbf{x}^{(k-1)}]\!]_{D'+1}\right)$ $\left([\![\text{cost}_l]\!]_{D'+1}, [\![\text{label}_{l,0}]\!]_{D'+1}, \ldots, [\![\text{label}_{l,\rho}]\!]_{D'+1}\right)_l \leftarrow \texttt{Evaluate+}(\mathscr{D}, \mathscr{L}, [\![\mathbf{x}]\!]_{D'+1})$ $P \leftarrow$ Random permutation $R_0, \ldots, R_\rho \leftarrow$ uniformly random bit $\Phi := \left([\![\text{cost}_{P(l)}]\!]_{D'+1}, [\![\text{label}_{P(l),0}]\!]_{D'+1} \boxplus R_0, \ldots, [\![\text{label}_{P(l),\rho}]\!]_{D'+1} \boxplus R_\rho\right)_l$
			$\xleftarrow{R_0,\ldots,R_\rho}$	
	$\xleftarrow{\Phi}$			
Find j with $\texttt{Decrypt}([\![\text{cost}_j]\!]_{D'+1}) = 0$ **for** $x = 0, \ldots, \rho$ **do:** $\mathcal{T}'(\mathbf{x})_x \leftarrow \texttt{Decrypt}([\![\text{label}_{j,x}]\!]_{D'+1})$				
	$\xrightarrow{\mathcal{T}'(\mathbf{x})_0,\ldots,\mathcal{T}'(\mathbf{x})_\rho}$			
		for $j = 0, \ldots, \rho$ **do:** $\mathcal{T}(\mathbf{x})_j \leftarrow \mathcal{T}'(\mathbf{x})_j - R_j$		

Protocol 8: Additive Multi-Party Server Private Decision-Tree Evaluation Protocol using GSW FHE and an STTP.

User i		**Server**		**STTP**
Input: $\mathbf{x}^{(i)}$ **Output:** –		**Input:** $\mathscr{D}, \mathscr{L}$ **Output:** $\mathscr{T}(\mathbf{x})$		**Input:** pk, sk **Output:** –
		$\xleftarrow{\text{pk}}$		
Encrypt $\mathbf{x}^{(i)}$ under received pk				
	$\xrightarrow{[\![\mathbf{x}^{(i)}]\!]_{D'+1}}$			
		$[\![\mathbf{x}]\!]_{D'+1} \leftarrow \left([\![\mathbf{x}^{(0)}]\!]_{D'+1}, [\![\mathbf{x}^{(1)}]\!]_{D'+1}, \ldots, [\![\mathbf{x}^{(k-1)}]\!]_{D'+1}\right)$ $\left([\![\text{cost}_l]\!]_{D'+1}, [\![\text{label}_{l,0}]\!]_{D'+1}, \ldots, [\![\text{label}_{l,\rho}]\!]_{D'+1}\right)_l \leftarrow \texttt{Evaluate+}(\mathscr{D}, \mathscr{L}, [\![\mathbf{x}]\!]_{D'+1})$ $P \leftarrow$ Random permutation $R_0, \ldots, R_\rho \leftarrow$ uniformly random bit $\Phi := \left([\![\text{cost}_{P(l)}]\!]_{D'+1}, [\![\text{label}_{P(l),0}]\!]_{D'+1} \boxplus R_0, \ldots, [\![\text{label}_{P(l),\rho}]\!]_{D'+1} \boxplus R_\rho\right)_l$		
			$\xrightarrow{\Phi}$	
				Find j with $\texttt{Decrypt}([\![\text{cost}_j]\!]_{D'+1}) = 0$ **for** $x = 0, \ldots, \rho$ **do:** $\quad \mathscr{T}'(\mathbf{x})_x \leftarrow \texttt{Decrypt}([\![\text{label}_{j,x}]\!]_{D'+1})$
			$\xleftarrow{\mathscr{T}'(\mathbf{x})_0, \ldots, \mathscr{T}'(\mathbf{x})_\rho}$	
		for $j = 0, \ldots, \rho$ **do:** $\quad \mathscr{T}(\mathbf{x})_j \leftarrow \mathscr{T}'(\mathbf{x})_j - R_j$		

5.5. Semi-Trusted Third Party protocols using key-switching and GSW Fully Homomorphic Encryption

We propose two other protocols where the dependency on the STTP is even lower. In these protocols, no ciphertexts are communicated with the STTP. The keys of the STTP are only used by the server in order to translate the input ciphertexts of all parties to the same key. This is necessary since during the tree evaluation the ciphertexts can only be combined if they are encrypted under the same key. The STTP only needs to join the key generation in which he generates his public key and *switching keys* that can be used by the server. These switching keys can switch encryptions by user i to an encryption under the public key of the STTP. The advantage is that the trust that needs to be put into this additional party is even lower since it does not execute any part of the protocol. At the end of the protocols, the result ciphertexts are switched back to the users' keys. The key-switching procedure is described in Section 4.1.5.

Our protocols that make use of an STTP in combination with key-switching, can be seen in Protocols 9 and 10. In the protocols where the server receives the evaluation result, using key-switching is not possible. Namely, from the switching key, the secret key of the party can be determined and therefore all intermediate ciphertexts can be read by the server. Therefore, the STTP still has to be the decrypting party which is also the case in the other STTP protocols. The STTP approach has its advantage only in case the users are receiving the evaluation results; no communication of ciphertexts with the STTP is necessary. This makes trusting the STTP much easier. Protocol 9 makes use of the multiplicative path evaluation and Protocol 10 of the additive path evaluation.

The STTP only engages in the setup procedure to generate a public key and secret key pair $(\mathrm{pk}, \mathrm{sk})$ and k switching keys $\mathbf{S}_{\mathrm{sk}\to\mathrm{sk}_i}$; one together with each user. We denote an encryption of μ with plaintext space $\mathbb{Z}_2$ encrypted under secret key sk by $[[\mu]]_2^{\mathrm{sk}}$. Applying the key-switching procedure to change the public key of an encryption from sk to sk′ is denoted by $[[\mu]]_2^{\mathrm{sk}} \xrightarrow{\text{switch}} [[\mu]]_2^{\mathrm{sk}'}$. Please not that the public keys pk in the protocols are now an encryption of 0 under the corresponding secret key. First, the STTP and the users communicate with each other to share these public keys. Then, this public key can be used by the user i to create the switching key $\mathbf{S}_{\mathrm{sk}_i\to\mathrm{sk}}$. The STTP creates $\mathbf{S}_{\mathrm{sk}\to\mathrm{sk}_i}$ for all i and sends these to the server. This does require an additional round, in which the users communicate with the STTP to share their public keys. Note here that the evaluation algorithms now include the switching procedure to the STTP's secret key after every decision node is evaluated.

Protocol 9: Multi-Party User Private Decision-Tree Evaluation Protocol using GSW FHE, an STTP and key-switching.

STTP	**User** i	**Server**
Input: pk, sk	**Input:** $\mathbf{x}^{(i)}, \mathrm{pk}_i, \mathrm{sk}_i$	**Input:** $\mathscr{D}, \mathscr{L}$
Output: –	**Output:** $\mathscr{T}(\mathbf{x})$	**Output:** –

$\mathrm{pk}, \mathrm{pk}_i$ ⟷

$\mathbf{S}_{\mathrm{sk}\to\mathrm{sk}_1}, \ldots, \mathbf{S}_{\mathrm{sk}\to\mathrm{sk}_{k-1}}$ →

$\mathbf{S}_{\mathrm{sk}_i\to\mathrm{sk}}, [[\mathbf{x}^{(i)}]]_2^{\mathrm{sk}_i}$ →

$[[\mathbf{x}]]_2 \leftarrow \left([[\mathbf{x}^{(0)}]]_2^{\mathrm{sk}_0}, [[\mathbf{x}^{(1)}]]_2^{\mathrm{sk}_1}, \ldots, [[\mathbf{x}^{(k-1)}]]_2^{\mathrm{sk}_{k-1}}\right)$

$[[\mathscr{T}(\mathbf{x})_0]]_2^{\mathrm{sk}}, \ldots, [[\mathscr{T}(\mathbf{x})_\rho]]_2^{\mathrm{sk}} \leftarrow \mathtt{Evaluate}(\mathscr{D}, \mathscr{L}, [[\mathbf{x}]]_2)$

for $j = 0, \ldots, \rho$ **do:**

  for $i = 0, \ldots, k-1$ **do:**

    $[[\mathscr{T}(\mathbf{x})_j]]_2^{\mathrm{sk}} \xrightarrow{\text{switch}} [[\mathscr{T}(\mathbf{x})_j]]_2^{\mathrm{sk}_i}$

← $[[\mathscr{T}(\mathbf{x})_0]]_2^{\mathrm{sk}_i}, \ldots, [[\mathscr{T}(\mathbf{x})_\rho]]_2^{\mathrm{sk}_i}$

for $j = 0, \ldots, \rho$ **do:**

  $\mathscr{T}(\mathbf{x})_j \leftarrow \mathtt{Decrypt}([[\mathscr{T}(\mathbf{x})_j]]_2^{\mathrm{sk}_i})$

Protocol 10: Additive Multi-Party User Private Decision-Tree Evaluation Protocol using GSW FHE, an STTP and key-switching.

STTP	**User** i	**Server**
Input: pk, sk	**Input:** $\mathbf{x}^{(i)}, \mathrm{pk}_i, \mathrm{sk}_i$	**Input:** $\mathscr{D}, \mathscr{L}$
Output: $-$	**Output:** $\mathscr{T}(\mathbf{x})$	**Output:** $-$

$\mathrm{pk}, \mathrm{pk}_i$ ⟷

$\mathbf{S}_{\mathrm{sk}\to\mathrm{sk}_1}, \ldots, \mathbf{S}_{\mathrm{sk}\to\mathrm{sk}_{k-1}}$ →

$\mathbf{S}_{\mathrm{sk}_i\to\mathrm{sk}}, [[\mathbf{x}^{(i)}]]^{\mathrm{sk}_i}_{D'+1}$ →

$[[\mathbf{x}]]_{D'+1} \leftarrow \left([[\mathbf{x}^{(0)}]]^{\mathrm{sk}_0}_{D'+1}, [[\mathbf{x}^{(1)}]]^{\mathrm{sk}_1}_{D'+1}, \ldots, [[\mathbf{x}^{(k-1)}]]^{\mathrm{sk}_{k-1}}_{D'+1}\right)$

$\left([[\mathrm{cost}_l]]^{\mathrm{sk}}_{D'+1}, [[\mathrm{label}_{l,0}]]^{\mathrm{sk}}_{D'+1}, \ldots, [[\mathrm{label}_{l,\rho}]]^{\mathrm{sk}}_{D'+1}\right)_l \leftarrow \texttt{Evaluate+}(\mathscr{D}, \mathscr{L}, [[\mathbf{x}]]_{D'+1})$

for $i = 0, \ldots, k-1$ **do:**

$\left([[\mathrm{cost}_l]]^{\mathrm{sk}}_{D'+1}\right)_l \xrightarrow{\text{switch}} \left([[\mathrm{cost}_l]]^{\mathrm{sk}_i}_{D'+1}\right)_l$

for $j = 0, \ldots, \rho$ **do:**

$([[\mathrm{label}_{l,j}]]^{\mathrm{sk}}_{D'+1})_l \xrightarrow{\text{switch}} ([[\mathrm{label}_{l,j}]]^{\mathrm{sk}_i}_{D'+1})_l$

$P \leftarrow$ Random permutation

$\Phi_i := \left([[\mathrm{cost}_l]]^{\mathrm{sk}_i}_{D'+1}, [[\mathrm{label}_{l,0}]]^{\mathrm{sk}_i}_{D'+1}, \ldots, [[\mathrm{label}_{l,\rho}]]^{\mathrm{sk}_i}_{D'+1}\right)_l$

Φ_i ←

Find j with $\texttt{Decrypt}([[\mathrm{cost}_j]]^{\mathrm{sk}_i}_{D'+1}) = 0$

for $j = 0, \ldots, \rho$ **do:**

$\mathscr{T}(\mathbf{x})_j \leftarrow \texttt{Decrypt}([[\mathrm{label}_j]]^{\mathrm{sk}_i}_{D'+1})$

5.6. Decision-node evaluation

Every decision node needs to be evaluated, which amounts to either a comparison with a threshold value or an equality test with a categorical value. The comparison protocol should output 1 if the input exceeds the threshold and the equality protocol should output 1 if the input is the same as the node's category. We propose protocols that can be evaluated locally on the encrypted inputs. First we outline the comparison protocol, after which we give details of the equality protocol.

5.6.1. Comparison protocol

There are many comparison protocols that require multiple rounds and communication between the users. In [34], it is shown that there are two protocols that are the most promising in terms of communication and computational complexity. These protocols are given in [51, 65]. Both these protocols can be adapted such that they can be applied in our setting and do not require communication; they propose Boolean circuits that we are able to evaluate homomorphically with our (Multi-Key) FHE scheme. Tueno et al. [107] adopt another comparison protocol given by Cheon et al. [28]. The idea of all these protocols is to translate the comparison problem to some operations on the bits of the two input values. The work of Fischlin is one of the first that approaches the problem in this manner [49]. He notices that $x = x_n \dots x_0$ is greater than $y = y_n \dots y_0$, if there is a location i such that $x_i = 1$ and $y_i = 0$ and for all more significant bits $j = \{i+1, \dots, n\}$ it holds that $x_j = y_j$.

Below, we describe these three methods and evaluate which one is the best for our purpose. First, the functionalities are demonstrated and compared for plaintext space $\mathbb{Z}_2$. After that we show that the most suitable protocol can also be extended to a bigger plaintext space. It is assumed that we have two δ-bit integers $x, y \in \mathbb{Z}$ with corresponding bit representations $x_{\delta-1} \cdots x_0$ and $y_{\delta-1} \cdots y_0$ with $\delta \geq 1$. The x value is the user's input value that is bit-wise encrypted and y the non-encrypted threshold value known by the server. The result of the protocols should be the comparison bit $[\![x > y]\!]_2$ that encrypts 1 if $x > y$ and 0 otherwise. This output can then be used by the server for further computations and the server does not gain any knowledge about x or b (since both are encrypted).

First, we describe the comparison protocol of Cheon et al. [28]. They assume that both integers are bit-wise encrypted and propose a Boolean circuit. In our setting, the threshold values are not encrypted so that some homomorphic multiplications or additions between two ciphertexts can be replaced by homomorphic multiplications or additions with plaintexts. The circuit is given by

$$[\![x > y]\!]_2 = (1 \oplus y_{\delta-1}) \cdot [\![x_{\delta-1}]\!]_2 + \sum_{i=0}^{\delta-2} (1 \oplus y_i) \cdot [\![x_i]\!]_2 \boxdot [\![d_{i+1}]\!]_2 \boxdot [\![d_{i+2}]\!]_2 \boxdot \cdots \boxdot [\![d_{\delta-1}]\!]_2 \tag{5.1}$$

with $[\![d_j]\!]_2 = \left((1 \oplus y_j) \boxplus [\![x_j]\!]_2\right)$. It can be seen that this circuit includes in the worst case (when all $y_i = 0$) $\sum_{i=0}^{\delta-2}(\delta - 1 - i) = (\delta-1) + (\delta-2) + \cdots + 2 + 1 = \frac{1}{2}\delta(\delta-1)$ homomorphic multiplications and $\delta - 1$ homomorphic additions. The proof of correctness can be found in [28]. The intuition behind this scheme is the same as that of Fischlin [49]. It tries to find a location i where $x_i > y_i$ (so $x_i = 1$ and $y_i = 0$) and where for all $j > i$ it holds that $x_j = y_j$.

The next comparison protocol is from Garay et al. [51], where multiplications are done using so-called conditional gates (using some communication) based on a threshold additive homomorphic encryption scheme. Since we use fully homomorphic encryption, we can do multiplication over the ciphertexts and therefore no communication is necessary. The result is again a Boolean circuit that can be evaluated homomorphically. They propose to split the bit strings in equally long parts which are compared recursively. The Boolean circuit consists of two auxiliary functions; $t_{i,j}$ that denotes the value of $x_{i+j-1} \cdots x_i > y_{i+j-1} \cdots y_i$ and $z_{i,j}$ that is equal to 1 if the two substrings $x_{i+j-1} \cdots x_i$ and

$y_{i+j-1} \cdots y_i$ are equal, else 0. These functions are given by

$$[[t_{i,j}]]_2 = \begin{cases} [[x_i]]_2 \boxplus (y_i \cdot [[x_i]]_2), & j = 1; \\ [[t_{i+l,j-l}]]_2 \boxplus ([[z_{i+l,j-l}]]_2 \boxdot [[t_{i,l}]]_2), & j > 1. \end{cases} \tag{5.2}$$

and

$$[[z_{i,j}]]_2 = \begin{cases} (1 \oplus y_i) \boxplus [[x_i]]_2, & j = 1; \\ [[z_{i+l,j-l}]]_2 \boxdot [[z_{i,l}]]_2, & j > 1. \end{cases} \tag{5.3}$$

Here, $l = \lceil j/2 \rceil$. The encrypted comparison bit $[[x > y]]_2$ that encrypts 1 if $x > y$ and 0 else, is then given by $[[x > y]]_2 = [[t_{0,\delta}]]_2$. The calculation of a $z_{i,j}$ requires $j-1$ homomorphic multiplications and j plaintext additions. Each $t_{i,j}$ is calculated by again $j-1$ homomorphic multiplications, $2j-1$ homomorphic additions and j plaintext multiplications (not taking into account the operations required for the $z_{i+l,j-l}$ value). Thus, for $t_{0,\delta}$ a maximum of $2\delta-2$ homomorphic multiplications and $2\delta-1$ homomorphic additions have to be calculated. So for $\delta > 3$ less multiplications are needed compared to the previous protocol, but the noise propagation is much bigger since in the calculation of a $t_{i,j}$, two ciphertexts are multiplied that are both a result of another multiplication. In this way, the multiplicative depth is higher than for the previous method.

Lasty, we demonstrate the protocol of Koleskinov et al. [65]. Their solution is based on garbled circuits where the evaluation function is a Boolean comparison circuit. In our protocols we can again use this Boolean circuit and evaluate it homomorphically such that no communication has to take place. The Boolean comparison circuit sequentially compares each bit by computing

$$[[c_{i+1}]]_2 = \big(([[c_i]]_2 \boxplus y_i) \boxdot ([[c_i]]_2 \boxplus [[x_i]]_2)\big) \boxplus [[x_i]]_2 \tag{5.4}$$

where $c_0 = 0$. The result is then given by $[[x > y]]_2 = [[c_\delta]]_2$. Clearly, this method requires $\delta - 1$ homomorphic multiplications and $2(\delta - 1)$ homomorphic additions. Again, the multiplications take place between ciphertext that are again the result of multiplications (namely the c_i), which gives a high noise propagation.

To give an example of the noise propagation in the above three protocols, we take $[[x_i]]_2 \in \mathbb{Z}_q^{n \times nl}$ all U_τ-noisy ciphertexts with plaintext space $\mathbb{Z}_2$ and deduce the noise propagation for all three protocols, in case $\delta = 4$.

For the first protocol in Equation 5.1, we have

$$[[x > y]]_2 = (1 \oplus y_3) \cdot [[x_3]]_2 + \sum_{i=0}^{2} (1 \oplus y_i) \cdot [[x_i]]_2 \boxdot [[d_{i+1}]]_2 \boxdot \cdots \boxdot [[d_3]]_2.$$

We know that all $[[d_j]]_2$ terms are U_τ-noisy according to Proposition 4.9. The term behind the sum contains in the worst case (when $y_i = 0$ for all i) $\delta - 1 - i = 3 - i$ homomorphic multiplications of U_τ-noisy ciphertexts, resulting in $((3-i)nlU + U)_\tau$-noisy ciphertexts given Propositions 4.8 and 4.10. These are summed for $i = 0, 1, 2$ and, in the worst case that $y_3 = 0$, added to $[[x_3]]_2$, giving a $(6nlU + 4U)_\tau$-noisy ciphertext according to Propositions 4.7 and 4.9. So, 6 homomorphic multiplications and 3 homomorphic additions are performed.

According to Equations 5.2 and 5.3 we have for the second protocol that

$$[[x > y]]_2 = [[t_{0,4}]]_2 = [[t_{2,2}]]_2 \boxplus \big([[z_{2,2}]]_2 \boxdot [[t_{0,2}]]_2\big)$$

where

$$[[t_{2,2}]]_2 = [[t_{3,1}]]_2 \boxplus ([[z_{3,1}]]_2 \boxdot [[t_{2,1}]]_2),$$

$$[[z_{2,2}]]_2 = [[z_{3,1}]]_2 \boxdot [[z_{2,1}]]_2,$$

$$[\![t_{0,2}]\!]_2 = [\![t_{1,1}]\!]_2 \boxplus ([\![z_{1,1}]\!]_2 \boxdot [\![t_{0,1}]\!]_2).$$

All $[\![z_{i,1}]\!]_2$ are U_τ-noisy and all $[\![t_{i,1}]\!]_2$ are $(2U)_\tau$-noisy according to Propositions 4.7 and 4.9. Therefore, $[\![t_{2,2}]\!]_2$ is $(nlU+4U)_\tau$-noisy, $[\![z_{2,2}]\!]_2$ is $(nlU+U)_\tau$-noisy and $[\![t_{0,2}]\!]_2$ is $(nlU+4U)_\tau$-noisy. From this we can conclude that $[\![t_{0,4}]\!]_2$ and therefore $[\![x > y]\!]_2$ is $\left((nl)^2U+3nlU+8U\right)_\tau$-noisy. Here, we used Propositions 4.7 and 4.8. In total 4 homomorphic multiplications are done together with 7 homomorphic additions.

For the third protocol, according to Equation 5.4 the following equations hold:

$$[\![b]\!] = [\![c_4]\!]_2 = \left(([\![c_3]\!]_2 \boxplus y_3) \boxdot ([\![c_3]\!]_2 \boxplus [\![x_3]\!]_2)\right) \boxplus [\![x_3]\!]_2,$$

$$[\![c_3]\!]_2 = \left(([\![c_2]\!]_2 \boxplus y_2) \boxdot ([\![c_2]\!]_2 \boxplus [\![x_2]\!]_2)\right) \boxplus [\![x_2]\!]_2,$$

$$[\![c_2]\!]_2 = \left(([\![c_1]\!]_2 \boxplus y_1) \boxdot ([\![c_1]\!]_2 \boxplus [\![x_1]\!]_2)\right) \boxplus [\![x_1]\!]_2,$$

$$[\![c_1]\!]_2 = \left(y_0 \boxdot [\![x_0]\!]_2\right) \boxplus [\![x_0]\!]_2.$$

Clearly, 3 homomorphic multiplications take place and 6 homomorphic additions. Propositions 4.7 and 4.8 give that $[\![x > y]\!]_2 = [\![c_4]\!]_2$ is $\left(2(nl)^3U+8(nl)^2U+12nlU+8U\right)_\tau$-noisy. Every time δ increases by one, the maximum factor to which the nl terms are raised, increases by one as well.

It is clear that homomorphic multiplications of ciphertexts that are the result of a multiplication already, give a high increase in the noise. The higher the noise propagation, the higher the modulus q of the of the cryptographic scheme needs to be. The higher these parameters, the bigger the ciphertext matrices and the elements in the ciphertext. This results in a decrease of the efficiency of the operations on the ciphertexts. Since the comparisons are done before the rest of the protocol is executed, the aim is to keep this noise as low as possible. Therefore, in our protocols, we make use of the comparison protocol of Cheon et al. [28] in Equation 5.1.

Our additive protocols take place in a bigger plaintext space given by $\mathbb{Z}_{D'}$ with $D'+1$ the first prime higher than the depth of the tree D. The protocol of Cheon et al. [28] only works in the plaintext space $\{0,1\}$. Namely, if $x_j = 1$ and $y_j = 0$, the value $d_j = 2$ in $\mathbb{Z}_{D'}$, while this needs to be equal to 0. To solve this, we change the comparison circuit such that it always outputs the correct bit value. Our adapted Boolean circuit is given by

$$\begin{aligned}[\![x > y]\!]_{D'+1} = &\left((1 \oplus y_{\delta-1}) \mod 2\right) \cdot [\![x_{\delta-1}]\!]_{D'+1} + \\ &\sum_{i=0}^{\delta-2} \left((1 \oplus y_i) \mod 2\right) \cdot [\![x_i]\!]_{D'+1} \boxdot [\![d_{i+1}]\!]_{D'+1} \boxdot [\![d_{i+2}]\!]_{D'+1} \boxdot \cdots \boxdot [\![d_{\delta-1}]\!]_{D'+1}\end{aligned} \tag{5.5}$$

with

$$[\![d_j]\!]_{D'+1} = \begin{cases} [\![x_i]\!]_{D'+1}, & y_j = 1; \\ 1 \boxplus \left(D' \boxdot [\![x_i]\!]_{D'+1}\right), & y_j = 0. \end{cases} \tag{5.6}$$

It can be seen that this circuit has the desired functionality. Namely, in Table 5.1, it can be seen that $[\![d_j]\!]_{D'+1}$ in $\mathbb{Z}_{D'}$ gives exactly the same results as the original formula $[\![d_j]\!]_2 = \left((1 \oplus y_j) \boxplus [\![x_j]\!]_2\right)$ in $\mathbb{Z}_2$. So the output of the circuit is exactly the same, since the multiplications in $\mathbb{Z}_{D'}$ for 0,1 values give the same result as these multiplications in $\mathbb{Z}_2$.

According to the additive tree evaluation in Algorithm 5, if $[\![x > y]\!]_{D'+1} = [\![1]\!]_{D'+1}$, which is the case when $x > y$, the value $1 \boxplus \left(D' \boxdot [\![1]\!]_{D'+1}\right) = [\![0]\!]_{D'+1}$ is stored at the right child node and $[\![1]\!]_{D'+1}$ at the left child nodes, as desired. If $[\![x > y]\!]_{D'+1} = [\![0]\!]_{D'+1}$, the ciphertext $1 \boxplus \left(D' \boxdot [\![0]\!]_{D'+1}\right) = [\![1]\!]_{D'+1}$ is stored at the right child node and $[\![0]\!]_{D'+1}$ at the left child node. So the correct path has indeed a path cost equal to 0.

Table 5.1: Truth table for the homomorphic calculation of d_j in plaintext space $\mathbb{Z}_2$ and $\mathbb{Z}_{D'}$.

x_j	y_j	$[[d_j]]_{D'+1}$	$[[d_j]]_2$
0	0	$[[1]]_{D'+1}$	$[[1]]_2$
0	1	$[[0]]_{D'+1}$	$[[0]]_2$
1	0	$[[0]]_{D'+1}$	$[[0]]_2$
1	1	$[[1]]_{D'+1}$	$[[1]]_2$

5.6.2. Equality protocol

Again we have two δ-bit integers $x, y \in \mathbb{Z}$ with corresponding bit representations of $x_{\delta-1}\cdots x_0$ and $y_{\delta-1}\cdots y_0$ with $\delta \geq 1$. The x value is the user's input value that is bit-wise encrypted and y the non-encrypted threshold value known by the server. The result of the protocols should be the bit $[[x = y]]_2$ that encrypts 1 if $x = y$ and 0 else. Since $x = y$ if and only if $x_j = y_j$ for all j, in the plaintext space $\mathbb{Z}_2$ it is clear to see that

$$[[x = y]]_2 = \prod_{j=0}^{\delta-1} \left((1 \oplus y_j) \boxplus [[x_j]]_2\right).$$

This circuit consists in the worst case (when all $y_j = 0$) of $\delta - 1$ homomorphic multiplications and δ plaintext additions.

For bigger plaintext space, looking at Table 5.1 we find $[[x_j = y_j]]_{D'+1} = [[d_j]]_{D'+1}$ with $[[d_j]]_{D'+1}$ given by Equation 5.6. Therefore it holds that

$$[[x = y]]_{D'+1} = \prod_{j=0}^{\delta-1} \left([[d_j]]_{D'+1}\right).$$

This circuit contains again $\delta - 1$ homomorphic multiplications and in the worst case (when all $y_j = 0$) δ plaintext additions and plaintext multiplications.

II

Complexity and security analyses of the protocols, implementation, and results

6

Analyses of protocols

In the previous chapters we gave the mathematical formulations of our protocols and the underlying encryption schemes. In this chapter, we analyse the protocols in terms of run-time complexity and security. This is given in Section 6.1. Next to this, in Section 6.2, we describe the noise propagation during each part of the protocols, which is essential for setting the appropriate parameters of the encryption schemes.

6.1. Complexity and security analysis

In this section we discuss the complexity and security of all our protocols. As a reminder, in Table 6.1, the parameters of our protocols and encryption schemes are given. To get a clear overall picture, in Table 6.2, all our protocols are summarised. The additional rounds, denoted in between the brackets, are the rounds required for communication of the public and/or switching keys. These rounds are only required once, even when multiple trees are evaluated on different inputs.

Table 6.1: The parameters of the protocols and the encryption schemes.

σ	The number of decision nodes of the tree
γ	The number of leaf nodes of the tree
D	The depth of the tree
$D'+1$	The closest prime bigger than D
δ	The bit length of the input variables
a_i	The number of input variables of user i
A	The total number of input variables $\sum_i a_i$
k	The number of users
q	The modulus of the encryption scheme
ℓ	$\lceil \log_2 q \rceil$ for the multiplicative protocols and $\lceil \log_{D'+1} q \rceil$ for the additive protocols that together with n defines the ciphertext size
n	The secret-key size and the dimension of the underlying LWE problem that together with ℓ defines the ciphertext size
λ, τ	Security parameters
m	Size of the public key given by $m = n\lceil \log_2 q \rceil + 2\tau$

We can compare our work with other works within private decision-tree evaluation that also make use of homomorphic encryption, namely the works of Tueno et al. [107] and Tai et al. [104]. The complexity in terms of the homomorphic operations for the server, the communication sizes and number

of communication rounds of their protocols is given in Table 6.3. Please note here that these protocols only have one user that provides the input.

Table 6.2: Protocols summary.

	Page number	FHE scheme	Path evaluation	Result disclosed to	STTP	Rounds
Protocol 1	53	Multi-Key	Multiplicative	Users	No	3
Protocol 2	54	Multi-Key	Multiplicative	Server	No	3
Protocol 3	57	Multi-Key	Additive	Users	No	3
Protocol 4	58	Multi-Key	Additive	Server	No	4
Protocol 5	60	GSW	Multiplicative	Users	Yes	3 (+1)
Protocol 6	61	GSW	Multiplicative	Server	Yes	3 (+1)
Protocol 7	62	GSW	Additive	Users	Yes	3 (+1)
Protocol 8	63	GSW	Additive	Server	Yes	3 (+1)
Protocol 9	65	GSW	Multiplicative	Users	Yes	2 (+2)
Protocol 10	66	GSW	Additive	Users	Yes	2 (+2)

Table 6.3: Number of homomorphic operations of server and communication (expressed in the number of ciphertexts) of the protocols in [104] and [107] with the operations for the decision-node evaluations neglected (AHE := additively-homomorphic encryption, $U-S$:= user to server communication, $S-U$:= server to user communication).

	Complexity server			Communication		Rounds	Encryption scheme
	$\mathbf{C}_1 \boxplus \mathbf{C}_2$	$\mathbf{C}_1 \boxdot \mathbf{C}_2$	$\zeta \boxdot \mathbf{C}$	$U \to S$	$S \to U$		
Tai et al. [104]	$\sigma + 2\gamma - 3$	-	2γ	$\mathcal{O}(A\delta + \sigma)$	$\mathcal{O}(A\sigma + 2\gamma)$	4	AHE
Tueno et al. [107]	$\gamma\lceil\log_2\gamma\rceil$	$\sigma + \gamma - 3 + \gamma\lceil\log_2\gamma\rceil$	-	$\mathcal{O}(A\delta)$	$\mathcal{O}(\log_2\gamma)$	2	FHE

6.1.1. Complexity analysis

In Table 6.2, it can be seen that all our protocols require a constant, low number of rounds. All protocols differ in the number of homomorphic operations and amount of communication. The communication sizes, including the key-exchange, and number of homomorphic operations done by the server per protocol are given in Tables 6.4 and 6.6. The operations required for the evaluation of each decision node are neglected in these tables, because this can be chosen to be the same for every protocol. The complexity of the comparison and equality protocol in both the bit plaintext space and the extended plaintext space are given in Table 6.5.

Table 6.4: The communication (expressed in the number of ciphertexts) required for the protocols ($U-S$:= user to server communication, $S-U$:= server to user communication, $U-U$:= user to user communication, $S-T$:= server to STTP communication, $T-S$:= STTP to server communication).

	Communication					
	$U \to S$	$S \to U$	$U \to U$	$S \to T$	$T \to S$	$T \to U$
Protocol 1	$\mathcal{O}(a_i\delta + nml)$	$\mathcal{O}(k^2\log_2\gamma)$	$\mathcal{O}(1)^*$	-	-	-
Protocol 2	$\mathcal{O}(a_i\delta + nml)$	$\mathcal{O}(k^2\log_2\gamma)$	-	-	-	-
Protocol 3	$\mathcal{O}(a_i\delta + nml)$	$\mathcal{O}(k^2\gamma\log_2\gamma)$	$\mathcal{O}(1)^*$	-	-	-
Protocol 4	$\mathcal{O}(a_i\delta + nml)$	$\mathcal{O}(k^2\gamma\log_2\gamma)$	$\mathcal{O}(1)^*$	-	-	-
Protocol 5	$\mathcal{O}(a_i\delta)$	$\mathcal{O}(1)^*$	-	$\mathcal{O}(\log_2\gamma)$	-	$\mathcal{O}(1)^*$
Protocol 6	$\mathcal{O}(a_i\delta)$	-	-	$\mathcal{O}(\log_2\gamma)$	$\mathcal{O}(1)^*$	$\mathcal{O}(1)^*$
Protocol 7	$\mathcal{O}(a_i\delta)$	$\mathcal{O}(1)^*$	-	$\mathcal{O}(\gamma\log_2\gamma)$	-	$\mathcal{O}(1)^*$
Protocol 8	$\mathcal{O}(a_i\delta)$	-	-	$\mathcal{O}(\gamma\log_2\gamma)$	$\mathcal{O}(1)^*$	$\mathcal{O}(1)^*$
Protocol 9	$\mathcal{O}(a_i\delta + 1)$	$\mathcal{O}(\log_2\gamma)$	-	-	$\mathcal{O}(k)$	$\mathcal{O}(1)^*$
Protocol 10	$\mathcal{O}(a_i\delta + 1)$	$\mathcal{O}(\gamma\log_2\gamma)$	-	-	$\mathcal{O}(k)$	$\mathcal{O}(1)^*$

* This means that there is communication of an element from $\mathbb{Z}_q$, random numbers, the randomised class label bits or a public key of the GSW scheme.

In Table 6.4, it can be seen that the amount of communication from the users to the server depends on the bit-length δ and the number of input variables a_i of the users since the users always encrypt each bit of each input variable separately. This is also the case for the two protocols from the literature.

The main difference between the protocols in terms of communication is that the protocols that use the additive-path evaluation from Algorithm 5, communicate more ciphertexts, since for these protocols for every leaf node ciphertexts are communicated. The bit-length of the classification labels is $\mathcal{O}(\log_2 \gamma)$. For the multiplicative approach, for each label bit an encryption is sent. For the additive approach, $\mathcal{O}(\log_2 \gamma)$ encryptions (cost and label) per leaf node have to be sent, resulting in $\mathcal{O}(\gamma \log_2 \gamma)$ ciphertexts. This can also be seen in the in server-to-user communication in the protocols of [104] and [106], that use the additive and multiplicative approach, respectively. The first requires $\mathcal{O}(\gamma)$ communication, while the second requires only $\mathcal{O}(\log_2 \gamma)$ communication. The fact that the first is not dependent on $\log_2 \gamma$ is because they use an additively-homomorphic encryption scheme that has plaintext space equal to $\mathbb{Z}_q$, such that they can encrypt a label in one ciphertext.

In Protocols 9 and 10, the communication from the STTP to the server is higher, since this communication consists of k switching keys. Also, the communication from each user to the server is a bit higher due to the switching key that is sent. In the first four protocols, the ciphertexts are extended such that the ciphertext size is k^2 times bigger that the initial ciphertext; $nk \times nkl$ instead of $n \times nl$. This is denoted by the additional k^2 term. Also, a big public extension key is communicated between the user and server, with a size equal to $\mathcal{O}(nml)$ ciphertexts.

Table 6.5: Number of homomorphic operations required for the comparison and equality protocols used for the evaluation of one decision node. Here, $x \in \mathbb{Z}$ is the user's input variable and $y \in \mathbb{Z}$ the threshold or categorical variable that have both a bit length of δ.

	$\mathbf{C}_1 \boxplus \mathbf{C}_2$	$\mathbf{C}_1 \boxdot \mathbf{C}_2$	$\zeta \boxdot \mathbf{C}$	$\zeta \boxplus \mathbf{C}$
$[\![x > y]\!]_2$	$\leq \delta - 1$	$\leq \frac{1}{2}\delta(\delta-1)$	$\leq \delta$	$\leq \delta - 1$
$[\![x > y]\!]_{D'+1}$	$\leq \delta - 1$	$\leq \frac{1}{2}\delta(\delta-1)$	$\leq 2\delta - 1$	$\leq \delta - 1$
$[\![x = y]\!]_2$	–	$\leq \delta - 1$	–	$\leq \delta$
$[\![x = y]\!]_{D'+1}$	–	$\leq \delta - 1$	$\leq \delta$	$\leq \delta$

In Table 6.5, it can be seen that the comparison protocol requires more homomorphic plaintext multiplications if the plaintext space is bigger. This is also the case for the equality protocol. This is caused by a different calculation of d_j; when $y_j = 0$ it requires an additional homomorphic plaintext multiplication. On average, the equality protocol is less expensive than the comparison protocol since it requires less homomorphic multiplications. In Algorithms 4 and 5, the comparison or equality protocol is executed once for each decision node. So in case all decision nodes are threshold decision nodes, a maximum of $\frac{\sigma}{2}\delta(\delta-1)$ homomorphic multiplications and $\sigma(\delta-1)$ homomorphic additions are required for this. Our protocols use the same comparison protocol as in the protocol of Tueno et al. . The comparison procedure used by Tai et al. [104] requires two communication rounds, where the communication sizes are dependent on the number of decision nodes σ and the number of input variables A. The evaluation of one decision node is done by a maximum of δ^2 homomorphic additions.

The number of homomorphic operations required for the rest of the evaluation depends on the number of leaf nodes γ and decision nodes σ. This can be seen in Table 6.6. The `Evaluate` algorithm requires $\sigma + \gamma - 3$ multiplications for the path evaluation. Namely, for each of the total number of nodes $\sigma + \gamma$ a multiplication takes place with the parent node, except for the root node and its two children. The final evaluation of the leaf nodes takes (worst-case) $\gamma\lceil\log_2 \gamma\rceil$ homomorphic additions in the `Evaluate` algorithm, since each label is stored in a maximum of $\lceil\log_2 \gamma\rceil$ ciphertexts. To conclude, the server's number of homomorphic operations in Protocols 1, 2, 5 and 6 required for the path and leaf nodes evaluation are similar to the number of homomorphic operations in the proto-

Table 6.6: Number of homomorphic operations of server in the protocols with the decision-node evaluations neglected (γ:= the number of leaf nodes, σ:= the number of decision nodes, k := the number of users).

	$\mathbf{C}_1 \boxplus \mathbf{C}_2$	$\mathbf{C}_1 \boxdot \mathbf{C}_2$	$\zeta \boxdot \mathbf{C}$	$\zeta \boxplus \mathbf{C}$
Protocol 1	$\gamma\lceil\log_2\gamma\rceil$	$\sigma+\gamma-3$	$\lceil\log_2\gamma\rceil$	-
Protocol 2	$\gamma\lceil\log_2\gamma\rceil$	$\sigma+\gamma-3$	$\lceil\log_2\gamma\rceil$	-
Protocol 3	$\sigma+\gamma-3$	-	$\gamma(1+\lceil\log_2\gamma\rceil)$	$\gamma\lceil\log_2\gamma\rceil$
Protocol 4	$\sigma+\gamma-3$	-	$\gamma(1+\lceil\log_2\gamma\rceil)$	$\gamma\lceil\log_2\gamma\rceil$
Protocol 5	$\gamma\lceil\log_2\gamma\rceil$	$\sigma+\gamma-3$	$\lceil\log_2\gamma\rceil$	-
Protocol 6	$\gamma\lceil\log_2\gamma\rceil$	$\sigma+\gamma-3$	$\lceil\log_2\gamma\rceil$	-
Protocol 7	$\sigma+\gamma-3$	-	$\gamma(1+\lceil\log_2\gamma\rceil)$	$\gamma\lceil\log_2\gamma\rceil$
Protocol 8	$\sigma+\gamma-3$	-	$\gamma(1+\lceil\log_2\gamma\rceil)$	$\gamma\lceil\log_2\gamma\rceil$
Protocol 9†	$\gamma\lceil\log_2\gamma\rceil$	$2\sigma+\gamma-3+k\lceil\log_2\gamma\rceil$	$\lceil\log_2\gamma\rceil$	-
Protocol 10†	$\sigma+\gamma-3$	$\sigma+k\gamma(1+\lceil\log_2\gamma\rceil)$	$\gamma(1+\lceil\log_2\gamma\rceil)$	$\gamma\lceil\log_2\gamma\rceil$

† The key-switching procedure is seen as one homomorphic multiplication.

col of Tueno et al. [107]. Our approach does not require the label to be encrypted, so the evaluation of the leaf nodes is slightly more efficient; we do the multiplications with a plaintext label instead of homomorphically multiplying two ciphertexts and only when the leaf-label bit is non-zero a homomorphic addition takes places. Clearly, Protocol 9 requires more homomorphic operations due to the key-switching procedure. Namely, the key-switching procedure has to take place $\sigma + k\lceil\log_2\gamma\rceil$ times. The first term comes from the number of input ciphertexts and the second term from the translation of the $\lceil\log_2\gamma\rceil$ output ciphertexts for each party.

For Protocols 3, 4, 7 and 8, $\sigma+\gamma-3$ homomorphic additions are needed for the path evaluation, which is comparable with the path evaluation in [104]. The leaf node evaluation only requires some plaintext additions and multiplications. The difference with [104] is that they can encrypt the label in one ciphertext, since their encryption scheme has plaintext space $\mathbb{Z}_q$. We introduce plaintext addition, such that less homomorphic additions are required. Again, Protocol 10 requires more homomorphic operations due to the key-switching procedure that has to take place $\sigma + k\gamma(1+\lceil\log_2\gamma\rceil)$ times.

Additionally, the computational complexity of user i is defined by the cost of key generation, the encryption of $a_i\delta$ input variables and/or some decryptions. The STTP only participates in the key generation and decrypts the result ciphertexts for the protocols that do not use key-switching. The exact complexity of these partial decryptions depend on the underlying encryption scheme. The advantage of using an STTP is that the users only have to send a message once, both when they are decrypting and when the server is decrypting. Also, the complexity of all homomorphic operations using GSW FHE is much lower and no expensive extension method has to be used. When making use of key-switching, the involvement of the STTP is decreased, but the downside is that additional homomorphic operations are required.

We can therefore conclude that the number of homomorphic operations in our multiplicative protocols is similar to that of the protocol of Tueno et al. [107]. Also, our additive protocols require a similar number of homomorphic operations as for the protocol of Tai et al. [104]. However, since we make use of a comparison and equality protocol that do not require any communication rounds, the number of homomorphic operations required for the decision-node evaluations is higher. Next to this, in [104] an additively-homomorphic encryption scheme is used instead of a FHE, which is more efficient. Our first four protocols depend on Multi-Key FHE. Therefore, the computational complexity of the server in the first four protocols is, in addition to the homomorphic operations denoted in Table 6.6, also dependent on this ciphertext extension procedure.

6.1.2. Security analysis

The security of our protocols follows from the IND-CPA security of the encryption schemes and the fact that the partial decryptions do not give away any information about the secret key of the decrypting party. This is proved in Theorems 4.4, 4.5 and 4.6. Below, we shortly describe the security per protocol.

In Protocols 2 and 4, the protocols where the server receives the evaluation result, the users cannot learn anything from the server or other users, since only some ciphertexts or partial decryptions are sent to the users. In Protocol 2 only the bit-size of the classification label is revealed, since the output is encrypted bit-wisely. For Protocol 4 specifically, the ciphertexts per leaf node are randomly permuted such that the users do not know the exact location of the leaf nodes within the tree. For all leaf nodes which have non-zero path costs, a random value is added to the classification labels, such that it appears to be random to the user. For the leaf node with a path cost of 0, the classification can still not be found, since the server again randomised this. The only leakage from the server towards the users is the number of leaf nodes based on the number of ciphertexts returned. Since the users only send IND-CPA secure ciphertexts of their inputs, we can also conclude that the users privacy is guaranteed. Even when one of the users colludes with the server or when all users collude, no additional information is gained about the input of the other users or the evaluation result.

For Protocols 1 and 3, the same conclusions can be drawn. In these protocols the users receive the final evaluation result. Again, the users only see ciphertexts from the server and only learn the bit-size of the classification label. It is possible that the server returns a wrong classification result to the users by using a wrong tree. This can not be prevented since the decision tree is evaluated by the server that also holds the tree. In Protocol 3, the leaf nodes are randomly permuted and the path costs are randomised, so no other information than the number of leaf nodes, is revealed to the users. Again, when one of the users colludes with the server, no additional information is gained about the input of the other users.

For Protocols 5 to 8, the STTP does not learn the evaluation label because of the random value that is added. It is clear that the users' privacy is also kept, since only some ciphertexts are send to the server. When the STTP colludes with one of the users in Protocols 6 and 8 where the server receives the result, they gain no advantage due to randomisation. Possibly, in Protocols 5 and 7 the server can collude with the STTP to find the evaluation result or to decrypt the input of the users.

Protocols 9 and 10 partly counteract this by eliminating the STTP from communication during the protocol, such that the evaluation result is never communicated with the STTP. However, when colluding with the STTP, the public keys used for generating the switching keys can be shared with the server. The server is then able to calculate the users' secret keys from the switching keys that are sent to him. This makes the server able to decrypt the input of the users. Therefore, it should be ensured that the server is not colluding with the STTP.

6.2. Noise propagation and parameter setting

In this section we describe the noise propagation during each part of the protocols. This defines the parameter setting of the FHE schemes such that they can cope with this propagated noise. Namely, it should be possible to decrypt this final ciphertext with very high probability. To make this explicit, the probability of a proper decryption should be $1-\text{negl}(\tau)$ with τ a correctness parameter. Therefore, for each building block of the protocols we analyse the noise. It is clear that the noise is not increasing during (partial) decryption or during communication. Therefore, this analysis only focuses on the propagation during the server's tree evaluation, using one of the methods in Algorithms 4 or 5. Next to this, the noise propagation during key-switching is analysed as well. The noise is expressed in the number of decision nodes σ, the number of leaf nodes γ, the bit-size δ of the input and threshold variables, the depth D of the tree, the smallest prime $D'+1$ bigger than D, and the number k of users. Next, we assume all nodes to be worst-case threshold nodes, since the noise propagation is smaller for the equality protocol (due to less homomorphic operations), such that the parameter setting for worst-case threshold nodes is also valid for categorical nodes. We define $\bar{\Psi}_q^\alpha$ over $\mathbb{Z}_q$ to be the noise distribution of the underlying encryption scheme that is B_τ-bounded for

$$B = \alpha q \sqrt{\frac{W(2^{2\tau+1}/\pi)}{2\pi}}$$

with $W(\cdot)$ the Lambert W function, $\alpha \in (0,1)$ a real number and τ the security parameter, given by Propositions 4.1 and 4.2. For the multiplicative protocols we take $q = 2^\ell$ and for the additive protocols $q = (D'+1)^\ell$ for some positive integer ℓ (which can be different for the two approaches). A summary of the below results is given in Table 6.7.

We wrote a program that can determine the parameters q, α, n, m such that $\alpha q \geq 2\sqrt{n-1}$ and $m \geq n \log_2 q + 2\tau$ and the requirements on the modulus q are met and the security of the whole system is guaranteed according to Theorems 4.4 and 4.5[1]. Additionally, for the protocols using the Multi-Key FHE scheme, the security of the collaborative decryption is guaranteed by taking the appropriate value of B_{smudge} given by Theorem 4.6. Also, the parameters are checked by the model of [3], such that all possible attacks take at least 2^λ bit operations, for λ a security parameter. For determining these parameters, we took $\lambda, \tau = 110$ which is a common reference in the domain [24]. The parameters can be found in Appendix A.2. In the next sections, we describe the noise propagation of our protocols, on which we based the requirements on q given in Table 6.7.

6.2.1. Multiplicative protocols

We now describe the noise propagation of the multiplicative decision-tree evaluation method in Algorithm 4. Protocols 1, 2, 5, 6 and 9 make use of this evaluation method. We detail what happens with the noise during the decision node, path and leaf node evaluation. For Protocols 1 and 2, after the decision-node evaluations, all encryptions are extended to the keys of all users. This means that the ciphertexts are extended $k-1$ times. Below we also define the noise propagation for these ciphertext extensions. Initially, it is assumed that the initial users' input ciphertexts are all in $\mathbb{Z}_q^{n \times nl}$ and B_τ-noisy. We find:

- (*Threshold decision-node evaluation*) According to Equation 5.1, in the worst case $\delta - 1$ ciphertexts with the same noise values are summed. These ciphertexts are the result of $\delta - 1$ multiplications between ciphertexts. Then, these are again added to a B_τ-noisy ciphertext. Propositions 4.8 and 4.7 give that the resulting comparison bit encryption is a $\left((\delta-1)^2 nlB + \delta B\right)_\tau$-noisy-noisy.

[1] This code can be found at `https://github.com/Dieuwkevdende/Thesis-LWEparameters`.

– (*For the Multi-Key FHE protocols: ciphertext extension*) We have $Q_0 = (\delta-1)^2 nlB + \delta B$. Given Proposition 4.14, after one extension the ciphertext is $\left((\sqrt{n}+n)l\sqrt{m}B + Q_0 B\right)_\tau$-noisy. After the second extension the ciphertext is $\left((\sqrt{n}+2n)l\sqrt{2m}B + (\sqrt{n}+n)l\sqrt{m}B^2 + Q_0 B^2\right)_\tau$-noisy. So after $k-1$ extensions they are

$$\left((\sqrt{n}+(k-1)n)l\sqrt{(k-1)m}B + \cdots + (\sqrt{n}+2n)l\sqrt{2m}B^{k-2} + (\sqrt{n}+n)l\sqrt{m}B^{k-1} + Q_0 B^{k-1}\right)_\tau\text{-noisy.}$$

Since $\sum_{i=1}^{k-1} B^i = \frac{B^k-B}{B-1}$, we can bound this by $Q_1 := (\sqrt{n}+(k-1)n)l\sqrt{(k-1)m}\frac{B^k-B}{B-1} + Q_0 B^{k-1}$. So with this definition of Q_1, after extension the ciphertexts are $(Q_1)_\tau$-noisy.

– (*Path evaluation*) The path evaluation consists of at most $D-1$ multiplications. Proposition 4.8 proves that the result after the path evaluation is a $((D-1)nlQ_1 + Q_1)_\tau$-noisy ciphertext with Q_1 the bound on the noise of the ciphertext of previous step.

– (*Leaf node evaluation*) The leaf node evaluation consists of $\leq 2^D$ homomorphic additions of ciphertexts with independent noises, distributed according to a $((D-1)nlQ_1 + Q_1)_\tau$-bounded distribution. So, the final ciphertexts are $(\sqrt{2^D}Q_2)_\tau$-noisy ciphertext according to Proposition 4.7 with $Q_2 := (D-1)nlQ_1 + Q_1$. The plaintext multiplications and (for some protocols) the addition with a random plaintext do not increase the noise bound of the ciphertexts. Re-randomisation is only required when for a certain bit location, all the leaf labels have a zero bit on this location.

According to Proposition 4.12, the collaborative decryption procedure results in the correct decryption with probability higher than $1-2^{-\tau}$ when $\sqrt{2^D}Q_2 < \frac{q}{4} - kB_{\text{smudge}}$ with τ the security parameter. In Protocols 5, 6 and 9 the ciphertexts are single-key and therefore do not have to be decrypted collaboratively. Therefore, the decryption is correct with overwhelming probability if $\sqrt{2^D}Q_2 < \frac{q}{4}$.

6.2.2. Additive protocols

Protocols 3, 4, 7, 8 and 10 make use of the additive path evaluation approach given by Algorithm 5. We again assume that the initial users' input ciphertexts are all in $\mathbb{Z}_q^{n\times nl}$ and B_τ-noisy. Please note here that the noise propagation for homomorphic multiplication and extension is different than for the additive protocols, due to the new gadget vector. This is given by Propositions 4.17 and 4.18. The noise propagates then as follows:

– (*Threshold decision-node evaluation*) The difference with the previous approach is that now all encryptions of d_j are $(D'B)_\tau$-noisy when $y_j = 0$ by Proposition 4.10. Since we do know that all d_j are either equal to 0 or to 1, Proposition 4.8 shows that after the worst-case $\delta-1$ multiplications the ciphertexts are $\left(nlB + (\delta-2)nlD'^2B + D'^2B\right)_\tau$-noisy. After the additions, the result bit encryption of one threshold decision-node evaluation is then

$$\left((\delta D' - 2D' + 1)(\delta-1)nlD'B + (\delta D' - D' + 1)D'B\right)_\tau\text{-noisy}$$

by Proposition 4.7. Additionally, in Algorithm 5, the negation of this result bit requires another homomorphic plaintext multiplication resulting in a worst-case comparison bit encryption that is $(Q_0)_\tau$ noisy with $Q_0 : (\delta D' - 2D' + 1)(\delta-1)nlD'^2B + (\delta D' - D' + 1)D'^2B$.

– (*For the Multi-Key FHE protocols:: ciphertext extension*) According to Proposition 4.14, the noise propagation during the extension method is not dependent on the plaintext space. So we can again conclude that after extension the ciphertexts are $(Q_1)_\tau$-noisy with $Q_1 := (\sqrt{n}D' + (k-1)n)lD'\sqrt{(k-1)m}\frac{B^k-B}{B-1} + Q_0 B^{k-1}$.

– (*Path evaluation*) The additive path evaluation consists of maximal $D-1$ additions of the ciphertexts resulting from different decision-node evaluations with a noise that is distributed similarly but independent. Proposition 4.7 proves that the result after the path evaluation is a $\left(\sqrt{D-1}Q_1\right)_\tau$-noisy ciphertext.

– (*Leaf node evaluation*) During the leaf node evaluation one additional homomorphic plaintext addition and multiplication with plaintext value D' in the worst-case takes place. This results in a $\left(D'\sqrt{D-1}Q_1\right)_\tau$-noisy ciphertext, according to Propositions 4.9 and 4.10.

According to Proposition 4.16, the collaborative decryption procedure results in the correct decryption with probability higher than $1-2^{-\tau}$ when $D'\sqrt{D-1}Q_1 < \frac{q}{2(D'+1)} - kB_{\text{smudge}}$ for τ the security parameter.

6.2.3. Key-switching

In Protocols 9 and 10 the encryptions after the decision-node evaluations are switched to the key of the STTP. Next to this, after the whole evaluation the ciphertexts are once more switched back to the users' keys. In this section we analyse the noise during these tasks. We assume that all initial users' input ciphertexts and the zero encryptions as public keys are B_τ-noisy (fresh) ciphertexts in $\mathbb{Z}_q^{n\times nl}$.

For the multiplicative approach in Protocol 9, before the first key-switching procedure toward the STTP's public key, the input ciphertext is $\left((\delta-1)^2 nlB+\delta B\right)_\tau$-noisy. Let $Q_1 := (\delta-1)^2 nlB+\delta B+\sqrt{nl}B$. According to Proposition 4.15, the ciphertexts after key-switching are $(Q_1)_\tau$-noisy. Then after the path and leaf node evaluations, the ciphertexts are $(Q_2)_\tau$-noisy with $Q_2 = \sqrt{2^D}(D-1)nlQ_1 + \sqrt{2^D}Q_1$ as shown in Section 6.2.1. The last key switching gives final ciphertexts that are $\left(Q_2+\sqrt{nl}B\right)_\tau$-noisy.

The same derivation can be done for the additive approach in Protocol 10. We now use Proposition 4.15, since this protocol uses the new gadget vector. After the first key-switching the ciphertexts are

$$\left((\delta D'-2D'+1)(\delta-1)nlD'^2B+(\delta D'-D'+1)D'^2B+\sqrt{nl}D'B\right)_\tau\text{-noisy.}$$

Let $Q_1 := (\delta D'-2D'+1)(\delta-1)nlD'^2B+(\delta D'-D'+1)D'^2B+\sqrt{nl}D'B$. Then after the path and leaf node evaluations, the ciphertexts are $\left(D'\sqrt{D-1}Q_1\right)_\tau$-noisy as shown in Section 6.2.2. The last key switching gives final ciphertexts that are

$$\left(D'\sqrt{D-1}Q_1+\sqrt{nl}D'B\right)_\tau\text{-noisy.}$$

Table 6.7: Summary of the requirements on the modulus q of the FHE scheme of the several protocols. Here, δ is the bit-size of the users' input, D the depth of the tree, $D'+1$ the first prime number higher than D, $\ell = \lceil \log_2 q \rceil$ or $\ell = \lceil \log_{D'+1} q \rceil$ for the multiplicative or additive approach respectively, k the number of users and $B = \alpha q \sqrt{\frac{W(2^{2\tau+1}/\pi)}{2\pi}}$ for $\alpha \in (0,1)$.

	Bound on modulus
Protocols 1 and 2	$q > 4\sqrt{2^D}\left((D-1)nlQ+Q\right)+4kB_{\text{smudge}}$ with $Q := (\sqrt{n}+(k-1)n)l\sqrt{(k-1)m}\frac{B^k-B}{B-1}+\left((\delta-1)^2nlB+\delta B\right)B^{k-1}$
Protocols 3 and 4	$q > 2(D'+1)D'\sqrt{D-1}\left((\sqrt{n}D'+(k-1)n)lD'\sqrt{(k-1)m}\frac{B^k-B}{B-1}+QB^{k-1}\right)+2(D'+1)kB_{\text{smudge}}$ with $Q := (\delta D'-2D'+1)(\delta-1)nlD'^2B+(\delta D'-D'+1)D'^2B$
Protocols 5 and 6	$q > 4\sqrt{2^D}\left((D-1)nl+1\right)\left((\delta-1)^2nlmB+\delta mB\right)$
Protocols 7 and 8	$q > 2(D'+1)D'\sqrt{D-1}\left((\delta D'-2D'+1)(\delta-1)nlD'^2mB+(\delta D'-D'+1)D'^2mB\right)$
Protocol 9	$q > 4\sqrt{2^D}\left((D-1)nl+1\right)\left((\delta-1)^2nlB+\delta B+\sqrt{nl}B\right)+4\sqrt{nl}B$
Protocol 10	$q > 2(D'+1)D'\sqrt{D-1}\left((\delta D'-2D'+1)(\delta-1)nlD'^2B+(\delta D'-D'+1)D'^2B+\sqrt{nl}D'B\right)+2(D'+1)\sqrt{nl}D'B$

7

Results

One of the goals of this thesis is to compare the proposed protocols based on their run-time complexity and communication requirements. Therefore, in this chapter we evaluate and compare the run-time and communication costs of each protocol based on several parameters. Before the results sections, we first give details about and justification of the experiments in Section 7.1. Next, we provide a short description of our code and used libraries, together with some implementation details in Section 7.2. In Section 7.3 we describe some general results that hold for all the proposed protocols. We give the detailed results per protocol in Sections 7.4 to 7.6. In Section 7.7 we compare all protocols. We analyse the decision-node evaluations in Section 7.8 and lastly, we give some protocol modifications in Section 7.9. For the interested reader, in Appendix A.4, we give the results of using our protocols for evaluating the use-case tree in Figure 1.2.

7.1. Experimental description

In this section we shortly describe the experiments of which the results are presented in the next chapter. As a reminder, in Table 7.1, the parameters of the protocols and their meaning are given.

Table 7.1: The parameters of the protocols and the encryption schemes.

σ	The number of decision nodes of the tree
γ	The number of leaf nodes of the tree
D	The depth of the tree
$D'+1$	The closest prime bigger than D
δ	The bit length of the input variables
a_i	The number of input variables of user i
A	The total number of input variables $\sum_i a_i$
k	The number of users
q	The modulus of the encryption scheme
ℓ	$\lceil \log_2 q \rceil$ for the multiplicative protocols and $\lceil \log_{D'+1} q \rceil$ for the additive protocols that together with n defines the ciphertext size
n	The secret-key size and the dimension of the underlying LWE problem that together with ℓ defines the ciphertext size
λ, τ	Security parameters
m	Size of the public key given by $m = n\lceil \log_2 q \rceil + 2\tau$

As shown in the previous chapter, the number of homomorphic operations and the communication depends on the number of leaf nodes γ, the number of decision nodes σ and the bit-size of the inputs δ. We can also conclude that the depth of the tree D and the number of users k influence the complexity of the protocols. Namely, the higher these values, the higher q and therefore ℓ which impacts the size of each ciphertext. The bigger the ciphertexts, the higher the complexity of matrix multiplications or additions. The communication needed can be expressed in the number of rounds and the size of the communication.

A complete tree evaluation consists of (i) an evaluation for each decision node, (ii) optionally an extension or key-switching procedure per decision node, (iii) one homomorphic multiplication or addition for each node (minus the root and its children) during the path evaluation and (iv) an evaluation for each leaf node. The run-time complexity and communication sizes of these components are constant for a certain set of parameters D, k, δ and the corresponding parameter setting of q, n. Therefore, it is possible to define the run-time complexity of each of the above components separately. From this, the total run-time of evaluating complete trees according to the different protocols can be determined. We can assess if this leads to the correct run-time by evaluating a complete tree and comparing its run-time with the expected run-time. Since the evaluation time of a threshold decision node depends on the bits of the threshold value, we evaluate the run-time in the worst-case, with all threshold values equal to 0.

The parameters q, n for proper security need to be set according to the tables in Appendix A.2 given D, k, δ. These values are chosen such that both λ and τ are at least 110, which is a common reference according to [24]. The required values for n according to these tables are very high, which we were not able to run in practice for all parameter sets. Therefore, we choose to take $n = 5, 10, 25$ and 50 and deduce the dependency of the complexity on the parameter n. In this way we can derive the run-times for realistic and secure values of n for each protocol.

Next to determining the run-time complexity, we evaluate the communication sizes by determining the size of a ciphertext. The first experiments assume that all decision nodes are threshold nodes, where it is checked if $x > y$ with x the input variable and y a threshold value. To evaluate the difference in case the tree also has categorical decision nodes, where $x = y$ is evaluated, we test what the difference is between evaluating one categorical decision node or one threshold decision node for one specific set of parameters. To summarise, we run the following experiments:

- For every protocol and different setting of D, k, δ, we evaluate the run-time of all separate components of the complete decision-tree evaluation for $n = 5, 10, 25$ and 50,
- For every different setting of D, k, δ, we evaluate the ciphertext size to calculate the communication costs for each protocol and $n = 5, 10, 25$ and 50,
- For the threshold decision-node evaluation we evaluate the worst-case run-time which corresponds to a threshold value of 0,
- For one specific setting of D, k, δ, n, we evaluate the run-time of one categorical decision-node evaluation.

We vary the variables δ, k, D and determine the required modulus values based on Table 6.7. We choose $\delta \in \{4, 6, 8\}$, $k \in \{2, 3, 4\}$ and $D \in \{3, 7, 15\}$. For $\delta = 8$, per input variable, $2^8 = 256$ different values can be used. Also, a tree of depth 15 gives a maximum number of $2^{15} - 1$ decision nodes when it is assumed this tree is binary. The use-case in Figure 1.1 requires $k = 3$. Therefore, we believe these parameter sets give a realistic overview of possible decision trees and number of users. In the appendix in Chapter A.2, the parameters are given for all protocols based on our code [1] and the model of Albrecht et al. [3].

[1] Our code can be found at `https://github.com/Dieuwkevdende/Thesis-LWEparameters`.

7.2. Experimental setup

Our code is written in C++ and is executed on a server with 16 Intel Core Processors @ 2.4 GHz running 16.04.1-Ubuntu with 16 GB memory. Only one processor is used per run, since our code is not parallelized. For the implementation of big integers and matrices holding these big integers (our ciphertexts), the C++ Boost Multiprecision [74] and the Boost Basis Linear Algebra [111] libraries are used. Since all operations are done modulo q, all standard operations on the vectors or matrices are implemented by ourselves. In addition, we implemented all encryption schemes and other building blocks ourselves. The input decision tree is given by several mappings. The first mapping returns for each node the right and left child node (if these exist). The second mapping returns for every decision node the threshold/category with the corresponding input attribute name and for every leaf node the class label. The last mapping gives a boolean value that denotes if the respective node is a leaf node or not. The code of the protocols can be found at the below repositories. Next to this, in each repository a folder 'results' can be found which holds all result .txt files.

- `https://github.com/Dieuwkevdende/Thesis-Protocols1-2`,
- `https://github.com/Dieuwkevdende/Thesis-Protocols3-4`,
- `https://github.com/Dieuwkevdende/Thesis-Protocols5-6`,
- `https://github.com/Dieuwkevdende/Thesis-Protocols7-8`,
- `https://github.com/Dieuwkevdende/Thesis-Protocol9`,
- `https://github.com/Dieuwkevdende/Thesis-Protocol10`.

To conclude, the readers should note that our implementation is primarily used to compare the several protocols. The code is not yet optimised and therefore give an idea of the run-times of our protocols.

7.3. Common results

In this section we give some general results that hold for all our proposed protocols. In the next sections, specific results per protocol are given. As a reminder, a summary of all protocols can be found in Table 6.2.

7.3.1. Modulus

All the protocols' complexities are affected by the ciphertext sizes. One of the parameters that has an influence on the ciphertext size is ℓ, which is given by $\ell = \lceil \log_2 q \rceil$ for the multiplicative protocols and $\ell = \lceil \log_{D'+1} q \rceil$ for the additive protocols with q the modulus of the encryption scheme. Namely, the ciphertext size is $nk \times nkl$ for the first four protocols, with k the number of parties associated to the encryption. For the protocols that do not use Multi-Key FHE, the ciphertext size is always $n \times n\ell$. The modulus q is defined by the noise propagation formulas in Table 6.7 that depend on the parameters D, δ, k. The exact values of q are given in the tables in Appendix A.2. In these tables it can be seen that ℓ is hardly affected by the parameter δ. The parameters k and D do have a higher impact on the value of q and therefore on ℓ. This can be explained by noise propagation formulas in Table 6.7, since for the multiplicative protocols, q depends exponentially on k and D and only quadratically on δ. For the additive protocols, q depends on D' in a power of five. For the multiplicative protocols, the higher D or k, the higher the value of ℓ. For the additive protocols, ℓ increases for increasing k but decreases for higher D. This can be explained by the fact that $D'+1$ is the first prime higher than D and $\ell = \lceil \log_{D'+1} q \rceil$. In the run-time analysis for all protocols, we assume that this factor ℓ is known based on the parameters δ, D, k given in Appendix A.2.

7.3.2. Decryption

For all protocols, the run-time of the (partial) decryptions is very low. As an example, for $n = 25$ all partial decryptions (or full decryption for Protocols 5 to 10) for all values of δ, k, D run in less than 0.05 seconds. Since the (partial) decryption is an inner product of two n-length vectors and therefore runs in $\mathcal{O}(n)$ time, we expect that one partial decryption does not take more than $(0.05/25) \cdot n$ seconds. To justify this expectation, we simulated the partial decryption for Protocols 1 and 2 (that have the worst error propagation) with the parameter setting $\delta = 8, D = 15, k = 4$ and $n = 2250$ such that both $\lambda, \tau = 110$ according to Table A.3. The run-time of the partial decryption in this setting was 1.47 seconds which is indeed lower than $(0.05/25) \cdot 2250 = 4.5$ seconds. This is less than 0.0001% of the run-time of a homomorphic addition for these parameters. Therefore, the run-time is much less than the run-time of the key generation, encryption procedure or decision-tree evaluation. In this chapter the focus therefore lies on the run-time analysis of these three parts.

7.4. Multi-Key Fully Homomorphic Encryption protocols

We first give the results of the first four protocols, which require a Multi-Key FHE scheme and evaluate the tree either in the multiplicative way or the additive way. Because these protocols all make use of the Multi-Key FHE, they require a key extension procedure for the tree evaluation in order to combine the ciphertexts of the different users. Below, we show the run-times of the key generation, extension and the other parts of the complete tree evaluation. We also give the required communication costs per round. The most important results are summarised at the end of this section. Since the results of Protocols 3 and 4 are similar to that of Protocols 1 and 2, we mainly focus on the results of Protocols 1 and 2 in this section. We do address the aspects on which these protocols differ. The plots of Protocols 3 and 4 that are not given in this section, can be found in Appendix A.3.

7.4.1. Key generation

In these protocols, every user needs to generate public keys. Each user needs to generate three keys, of which the third key $\mathbf{D}$ is the biggest with size $nm\ell \times n^2\ell$. The complexity of the key generation is $\mathcal{O}(n^3 m\ell^2)$. Here, $m = n\lceil \log_2 q \rceil + 2\tau$. We performed the key generation for $n = 2, 3, 4$. The results can be seen in Figure 7.1 for Protocols 1 and 2. The differences between the plots can be explained by the effect of D, k and δ on ℓ which again affects the size of the public keys and therefore the generation time. Clearly, the key generation is very expensive, since for only a value of $n = 4$, the run-time can reach 8000 seconds already.

To give an idea of how the run-time is for even higher values of n, we take $n = 20$ such that $20^4/4^4 \approx 600$. This means that for $n = 20$ and $k = 2, D = 3, \delta = 4$, the run-time of the key generation is at least $600 \cdot 3000$ seconds which is equal to around 21 days. It is clear that for values of n above 1300 which is necessary for the security of the scheme, the key generation time gets too high.

In a similar way, we performed the key generation for $n = 2, 3, 4$ for Protocols 3 and 4. The result can be seen in the appendix in Figure A.5. Using the same approach as above, for $n = 20$ we can expect the key generation to take at least 90000 seconds (with $k = 2, D = 15, \delta = 4$). This is equal to 25 hours. The key generation for Protocols 3 and 4 is a bit lower than for the other two protocols, due to a lower error propagation and, therefore, a lower ℓ. But, the dependency on n to the power 4 makes the key generation quickly as high as for the other two protocols, even though the required value of n is now around 1000.

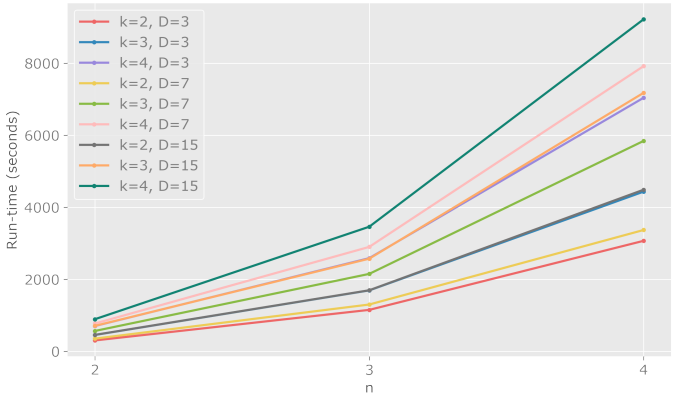


Figure 7.1: The run-time of the key generation for Protocols 1 and 2 plotted against n for different values of D, k with $\delta = 4$.

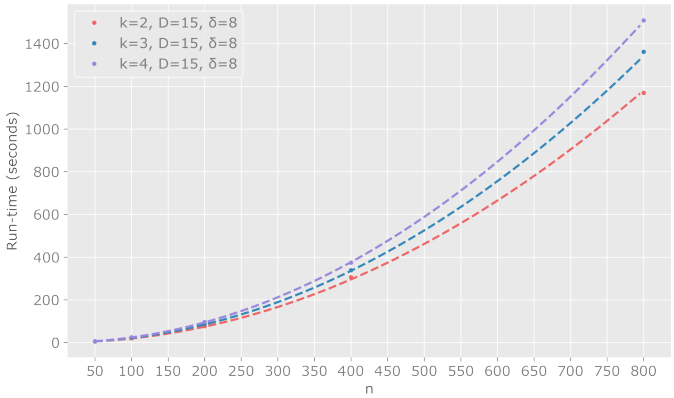


Figure 7.2: The run-time of an initial encryption of one bit for Protocols 1 and 2 plotted against the dimension n for a tree with $D = 15$, $\delta = 8$ and different k. The dashed functions are given by $2.53 \cdot 10^{-5} \cdot n^2 \ell$.

7.4.2. Encryption

As for the key generation, the parameters δ, D, k affect the run-time of the users' initial bit encryptions. This can again be explained by the difference in the factor ℓ that defines the generated ciphertext size $n \times n\ell$. As an example, in Figure A.2, the run-time one bit encryption is plotted against n for different values of D, δ for Protocols 1 and 2. A similar plot can be found for fixed δ and varying k. Clearly, the run-time is mostly dependent on the dimension n and less on the parameters of the tree and the number of users.

The generated ciphertext has $n^2\ell$ elements and, therefore, the expectation is that we can approximate the run-time for different values of n and ℓ. We also plotted a function $a \cdot n^2 \ell$ with $a = 2.53 \cdot 10^{-5}$, to clearly show the dependency of the results on n and ℓ. In Figure 7.2, the results can be seen for fixed $D = 15, \delta = 8$. For $n = 800$, the run-time of one bit encryption for a tree with $k = 2$ is around 1200 seconds according to Figure 7.2. Taking the appropriate value $n = 1700$ to have proper security as given in Appendix A.1, gives an expected run-time of $(1700/800)^2 \cdot 1200 \approx 5419$ seconds, which is equal to 90 minutes per bit encryption.

For Protocols 3 and 4, the same conclusions can be drawn regarding the complexity of the encryption procedure as for the first two protocols. The dependency on n and ℓ is similar to this dependency

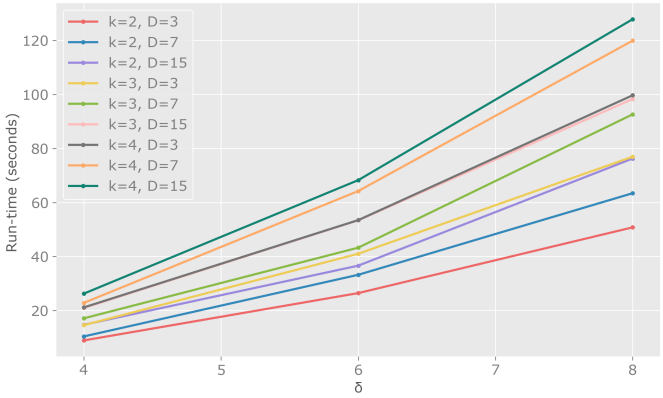


Figure 7.3: The run-time of one decision-node evaluation (worst-case with a threshold value of 0) for Protocols 1 and 2 plotted against δ for different values of D, k with $n = 10$.

for protocols 1 and 2. This is as expected, since the encryption only depends on the ciphertext size $n \times n\ell$. Again, the run-time of one bit encryption for these two protocols is lower, because of the lower value for ℓ. Taking the value for n for a security level of 110, for example $n = 1000$, $\ell = 20$ for $\delta = 4, k = 2$ and $D = 3$, gives an expected run-time of $(1000/800)^2 \cdot 320 \approx 500$ seconds, which is equal to around 8 minutes per bit encryption.

7.4.3. Extension

Per decision node, each ciphertext is extended $k-1$ times, which is part of the decision-tree evaluation. The most expensive part of this extension procedure is the calculation of matrix $\mathbf{Y}_2$ which is the result of a multiplication of a matrix of size $n \times nk'm\ell$ with a matrix of size $nk'm\ell \times n^2k'l$, where the k' is defined to be the number of parties connected to the ciphertext so far. The complexity of $k-1$ of these matrix multiplications is therefore dependent on n with a power of 5. Therefore, as for the key generation, the run-time of the extension procedure greatly increases in n, resulting in non-practical execution times of the extension for realistic values of n. For $k = 2, D = 3, \delta = 4$ the run-time of the extension of Protocols 1 and 2 is around 200 seconds already for $n = 4$, which can be found in the appendix in Figure A.1. Since $n = 20$ gives $20^5/4^5 \approx 3000$, the expectation is that the run-time of the extension for $n = 20$ and $k = 2, D = 3, \delta = 4$ is at least $3000 \cdot 200$ seconds. This is equal to almost 7 days. The same conclusion can be drawn regarding the extension procedure of Protocols 3 and 4. The results for these two protocols can be found in the appendix in Figure A.6.

7.4.4. Decision-tree evaluation

In the protocols, the server evaluates the decision tree using the encryptions of the users. The complete evaluation of the tree consists, next to the extension, out of the decision-node evaluations, the path evaluations and the leaf-node evaluations. We now discuss these three parts one by one.

Decision-node evaluations In Figure 7.3 (and in Figure A.10 in the appendix), it can be seen that the time complexity of one decision-node evaluation mostly depends on the parameter δ, compared to D and k. Since δ does not affect the parameter ℓ that much for a fixed D, k, the difference in complexity can not completely be explained by the factor ℓ. The reason for this dependency is that the number of homomorphic operations for a decision-node evaluation depends on the input bit length

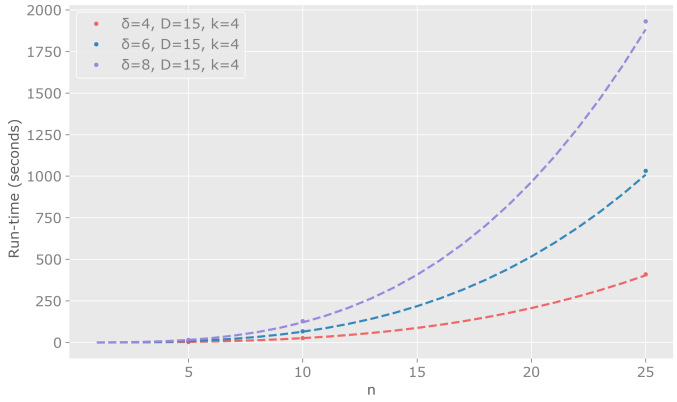


Figure 7.4: The run-time of one decision-node evaluation (worst-case with a threshold value of 0) for Protocols 1 and 2 plotted against the dimension n for different values of δ with $D = 15, k = 4$. The dashed functions are given by $2.49 \cdot 10^{-7} \cdot (\delta^2 - \delta) n^3 \ell^2$.

δ. Namely, the node evaluation with a threshold value of 0 consists of $\frac{1}{2}\delta(\delta-1)$ homomorphic multiplications, $\delta - 1$ homomorphic additions and $\delta - 1$ plaintext additions. This can be found in Table 6.5. The complexity of one homomorphic multiplication of two matrices of size $n \times n\ell$ and $nl \times n\ell$ is given by $\mathcal{O}(n^3 l^2)$ and the complexity of an addition, required for the homomorphic addition or plaintext addition, of matrices of size $n \times n\ell$ is given by $\mathcal{O}(n^2\ell)$. Therefore, the complexity of one worst-case decision-node evaluation depends on n, l and δ as input. Assuming the complexity regarding the matrix additions can be neglected compared to the matrix multiplications, we find that the complexity is $\mathcal{O}((\delta^2 - \delta) n^3 \ell^2)$. The result can be seen in Figure 7.4. Similar results are found for Protocols 3 and 4 and be found in Figure A.9 in the appendix.

Path evaluations The path evaluation for Protocols 1 and 2 consists of multiple homomorphic multiplications. Now, the ciphertext matrices are of size $nk \times nkl$ such that the complexity of a homomorphic multiplication is $\mathcal{O}(n^3 k^3 l^2)$ (a multiplication of a $nk \times nkl$ and $nkl \times nkl$ matrix). In Figure 7.5, it can be seen that the run-time of one path multiplication now greatly depends on the value of k given a fixed n, since now the ciphertext size also depends on k (and not only indirectly via the parameter ℓ). The path evaluation for Protocols 3 and 4 uses homomorphic additions instead of homomorphic multiplications. The complexity of one homomorphic addition is $\mathcal{O}(n^2 k^2 l)$ based on a ciphertext size of $nk \times nkl$. The run-time dependency on k can again be seen in the appendix in Figure 7.6. There, the results are plotted against n for different values of k. Clearly, the run-time of doing a homomorphic addition is much lower than that of a homomorphic multiplication.

Leaf-node evaluations Lastly, we look at the run-time of one leaf node evaluation. One leaf node evaluation of Protocols 1 and 2 consists of one homomorphic addition per label bit in the worst case, when all leaf label bits are equal to 1. This homomorphic addition has a complexity of $\mathcal{O}(n^2 k^2 l)$. Therefore, the expectation is that the run-time is similar to that found for one path addition in Figure 7.6. In Figure A.4 in the appendix it can be seen that this is indeed the case. The only difference is that now the run-times are slightly higher for the given parameters due to the higher value of ℓ for Protocols 1 and 2 compared to Protocols 3 and 4. One leaf node evaluation of Protocols 3 and 4 consists of one homomorphic plaintext multiplication and one plaintext addition per label bit with both a complexity of $\mathcal{O}(n^2 k^2 l)$. The result can be seen in Figure 7.7.

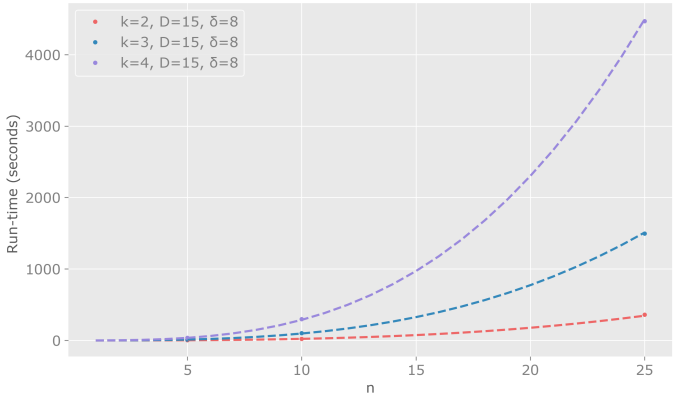


Figure 7.5: The run-time of one path multiplication for Protocols 1 and 2 plotted against the dimension n for different values of k with $D = 15, \delta = 8$. The dashed functions are given by $5.40 \cdot 10^{-7} n^3 k^3 l^2$.

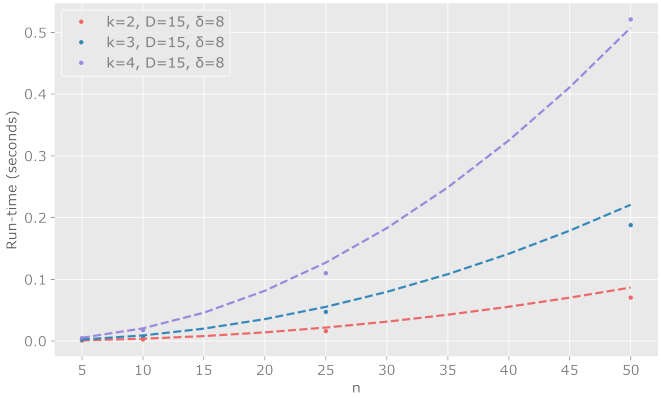


Figure 7.6: The run-time of one path addition for Protocols 3 and 4 plotted against the dimension n for different values of k with $D = 15, \delta = 8$. The dashed functions are given by $5.76 \cdot 10^{-7} n^2 k^2 l$.

7.4.5. Communication

Both multiplicative protocols consist of 3 rounds. In the first round, user i communicates with the server and sends $a_i \delta$ ciphertexts of his input and his public keys to the server. All these encryptions are ciphertexts of size $n \times n\ell$ that contain elements out of $\mathbb{Z}_q$. Every element in $\mathbb{Z}_q$ consists of 48 bytes due to the way these elements are stored by the C++ Boost Multiprecision library [74]. Every big integer in $\mathbb{Z}_q$ is represented by 8 integers with a maximal bit-length of 6, with a total space of $8 \cdot 6 = 48$ bytes. So a matrix of size $n \times n\ell$ is $48 \cdot n^2 \ell$ bytes. Therefore, in total user i sends $48 \cdot a_i \delta n^2 \ell$ bytes to the server in the first round. Next to this, each user sends his public keys to the server in the first round. These public keys are three matrices of sizes $m \times 1$, $n \times n^2 \ell$ and $nm\ell \times n^2 \ell$ of $48 \cdot m$, $48 \cdot n^3 l$ and $48 \cdot n^3 m \ell^2$ bytes respectively. These public keys only have to be sent once, since they can be re-used. In the second round, the server sends the result of the evaluation to each user. This result consists of $\lceil \log_2 \gamma \rceil$ encryptions; for every leaf label bit one encryption. Since for decryption only one column of the ciphertext is multiplied with the key, the result encryptions are of size $nk \times 1$. Therefore, this communication contains $48 \cdot \lceil \log_2 \gamma \rceil nk$ bytes. For Protocols 1 and 2, the third round is the users communicating the $\lceil \log_2 \gamma \rceil$ partial decryptions with each other or with the server, respectively. This communication is of size $48 \cdot \lceil \log_2 \gamma \rceil$ bytes.

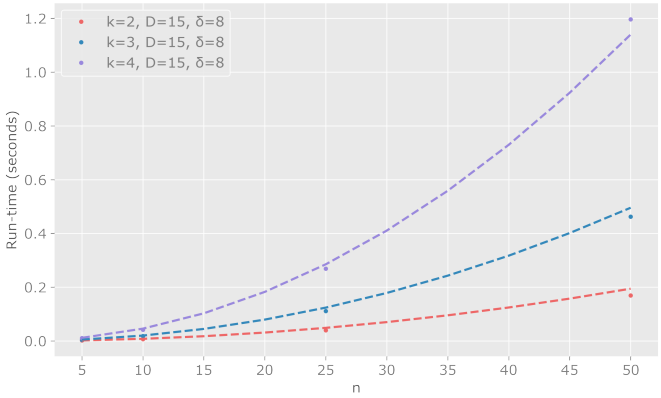


Figure 7.7: The run-time of one leaf bit evaluation for Protocols 3 and 4 plotted against the dimension n for different values of k with $D = 15, \delta = 8$. The dashed functions are given by $1.29 \cdot 10^{-6} n^2 k^2 l$.

The additive protocols have 3 and 4 rounds, respectively. The communication in the first round for these protocols is the same as for the multiplicative protocols. In the second round, now the result consists of $\gamma(\lceil \log_2 \gamma \rceil + 1)$ encryptions; for every leaf node the encrypted label bits and the encrypted cost. Again, only the column used for decryption needs to be communicated. Therefore, this communication contains $48 \cdot \gamma(\lceil \log_2 \gamma \rceil + 1) nk$ bytes. In the third round either $\gamma(\lceil \log_2 \gamma \rceil + 1)$ or γ partial decryptions are communicated that are each 48 bytes. The fourth round for Protocol 4 communicates $\lceil \log_2 \gamma \rceil$ elements that are also 48 bytes each.

Table 7.2: Communication sizes in bytes of the different rounds in Protocols 1 to 4. The communication between brackets in the first round only has to occur once and not every time a tree is evaluated.

	Round 1	**Round 2**	**Round 3**	**Round 4**
Protocol 1	$48 a_i \delta n^2 \ell + (48m + 48n^3 l + 48n^3 m\ell^2)$	$48\lceil \log_2 \gamma \rceil nk$	$48\lceil \log_2 \gamma \rceil$	-
Protocol 2	$48 a_i \delta n^2 \ell + (48m + 48n^3 l + 48n^3 m\ell^2)$	$48\lceil \log_2 \gamma \rceil nk$	$48\lceil \log_2 \gamma \rceil$	-
Protocol 3	$48 a_i \delta n^2 \ell + (48m + 48n^3 l + 48n^3 m\ell^2)$	$48\gamma(\lceil \log_2 \gamma \rceil + 1) nk$	$48\gamma(\lceil \log_2 \gamma \rceil + 1)$	-
Protocol 4	$48 a_i \delta n^2 \ell + (48m + 48n^3 l + 48n^3 m\ell^2)$	$48\gamma(\lceil \log_2 \gamma \rceil + 1) nk$	48γ	$48\lceil \log_2 \gamma \rceil$

7.4.6. Summary

The run-time complexities are dominated by the key generation and the extension procedure. We can give an example of this. By choosing the parameters $\delta = 4, D = 3, k = 2, \sigma = 2, \gamma = 3, a_i = 1$ and the corresponding parameters of $q = 2^{60}, \ell = 60$ with $n = 50$ for Protocols 1 and 2. For these parameters, the run-time for user i consists of $a_i \delta$ encryptions that cost $a_i \delta \cdot 3.49 = 13.96$ seconds. The key generation for these parameters is expected to cost around $50^4/4^4 \cdot 3000 \approx 20,000$ hours which is equal to around 2.3 years. For the server the evaluation without extension takes 5934 seconds. The extension is expected to take $50^5/4^5 \cdot 200 \approx 17000$ hours per decision node. Clearly, the run-times of the encryption and other evaluation parts can be neglected compared to the costs of the key generation for the user and the costs of the extension procedure. This also holds for Protocols 3 and 4. On top of this, for realistic values of n, given in Section A.2, for which the security is guaranteed, the run-times of the extension and key generation of our implementation on our used machine is expected to go beyond the order of years since the run-times increase with a power of 4 or a power of 5 in n, respectively.

Regarding the communication, Section 7.4.5 showed that the size of the third public key **D**, which is communicated in the first round, is $48 \cdot n^3 m\ell^2$ bytes. Using the parameters above, this results in a size of $6.9552 \cdot 10^{13}$ bytes or almost 70 terabytes for the multiplicative protocols, which is far from practical. Assuming a communication speed of around 15 megabytes per seconds, sharing this key will cost at least 70 days. This key only has to be sent once, since it can be re-used. For the additive protocols the corresponding parameters are $q = 5^{20}, \ell = 20$ such that the size of **D** for $n = 50$ with $m = 2570$ is 6 terabytes. Also for these sizes holds that they greatly depend on the dimension n and grow fast when n is chosen such that the protocols' security is guaranteed.

7.5. Semi-Trusted Third Party protocols

Protocols 5 to 8 make use of an STTP. This STTP is the only party that is required to do the key generation. The users use its public key to do the encryption. The error propagation and therefore the values of q and ℓ are therefore independent of the number of users. Only the parameter δ, D and n now affect the run-time complexity of these protocols. Again, the value of ℓ for the additive protocols is lower which automatically results in a lower run-time for similar procedures in comparison to the multiplicative protocols. Due to similar results, we neglect some results of Protocols 7 and 8. These plots can be found in Appendix A.3.

7.5.1. Key generation

The key generation for the STTP consists of generating both the secret key of size $n \times 1$ and the public key of size $n \times m$. In contrast to the key generation required for the Multi-Key FHE encryption scheme, the complexity is much lower. These protocols do not require the extension procedure such that the extension keys do not have to be shared. Now, for $n = 50$ the run-time of the key generation does not go above the 0.4 seconds. The expectation is therefore that for $n = 1550$ (which is required for a secure protocol for $\delta = 4$ and $D = 3$), the key generation only takes approximately $1550^2/50^2 \cdot 0.4 \approx 384$ seconds or 6 minutes.

7.5.2. Encryption

The encryption procedure for these protocols is the encryption under the public key of the STTP. This encryption is more costly than the encryption procedure that only uses the secret key, since now a matrix multiplication with a random binary matrix is required. The matrices that are being multiplied are the $n \times m$ public key matrix and the $m \times n\ell$ random binary matrix. Next to this, the random matrix needs to be generated. Therefore, the complexity of the encryption amounts to $\mathcal{O}(n^2 m\ell + nm\ell)$. Here, $m = n\lceil \log_2 q \rceil + 2\tau$. The first term accounts for the multiplication and the second term for the generation of the binary matrix. In Figure 7.8, the run-times of one bit encryption for Protocols 5 and 6 can be seen. Similar results can be found for Protocols 7 and 8 in Figure 7.9.

7.5.3. Decision-tree evaluation

The complete evaluation done by the server consist out of the decision-node evaluations, the path evaluations and the leaf-node evaluations. We now discuss these three parts one by one.

Decision-node evaluations The decision-node evaluations are similar to the previous protocols. Therefore, the complexity of these operations is again $\mathcal{O}((\delta^2 - \delta)n^3\ell^2)$. This can be seen in Figures A.12 and A.14 in the appendix.

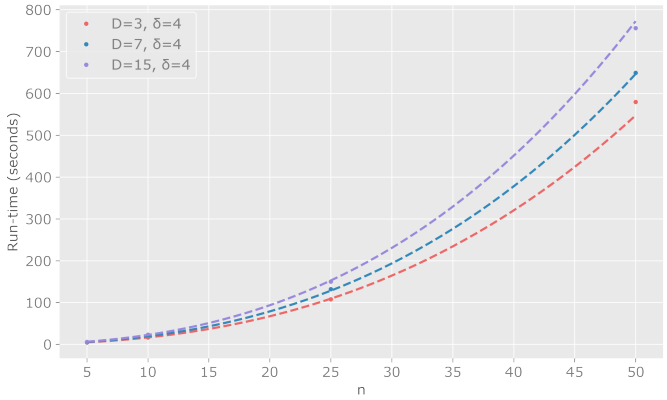


Figure 7.8: The run-time of an initial encryption of one bit for Protocols 5 and 6 plotted against the dimension n for a tree with $\delta = 4$ and different D. The dashed functions are given by $4.57 \cdot 10^{-7} \cdot n^2 m\ell + 2.29 \cdot 10^{-5} nm\ell$.

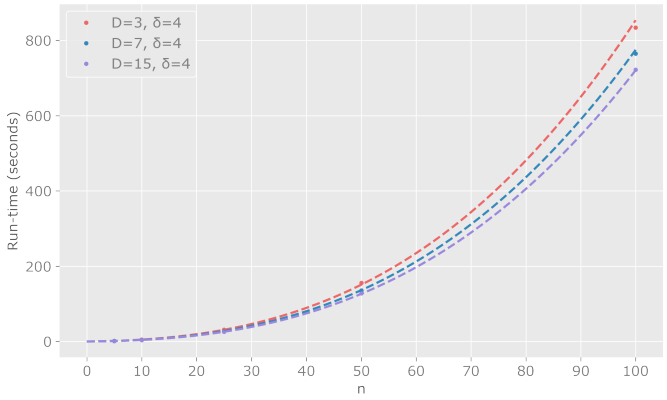


Figure 7.9: The run-time of an initial encryption of one bit for Protocols 7 and 8 plotted against the dimension n for a tree with $\delta = 4$ and different D. The dashed functions are given by $4.23 \cdot 10^{-7} \cdot n^2 m\ell + 2.38 \cdot 10^{-5} nm\ell$.

Path evaluations The path evaluation for Protocols 5 and 6 consists of multiple homomorphic multiplications, but now the ciphertexts are not extended. Therefore, they are of size $n \times n\ell$ such that the complexity of a homomorphic multiplication is $\mathcal{O}(n^3\ell^2)$ (a multiplication of a $n \times n\ell$ and $nl \times n\ell$ matrix). In Figure 7.10, the results of the run-time is plotted against n where δ is fixed to 4. The dependency on n and ℓ is similar to this dependency for the path multiplication run-time of Protocols 1 and 2. This is explained by the fact that it is exactly the same procedure for both protocols, but only the ciphertext sizes differ. The path evaluation for Protocols 7 and 8 consists of multiple homomorphic additions of matrices of size $n \times n\ell$ such that the complexity of a homomorphic addition is $\mathcal{O}(n^2\ell)$. The results can be seen in Figure 7.11. The dependency on n and ℓ is the same as for Protocols 3 and 4 in Figure 7.6, but now the value of k does not have an impact on the run-time.

Leaf-node evaluations The leaf node evaluation of Protocols 5 and 6 consists of one homomorphic addition per label bit in the worst case, when all leaf label bits are equal to 1. This homomorphic addition now has a complexity of $\mathcal{O}(n^2\ell)$. In Figure A.13 in the appendix, this dependency on n and ℓ is clearly visible for these two protocols. For Protocols 7 and 8 this operation consists of one homomorphic plaintext multiplications and one plaintext additions per label bit with both a complexity of $\mathcal{O}(n^2\ell)$. The result can be seen in Figure 7.12. We can estimate the total decision-tree evaluation

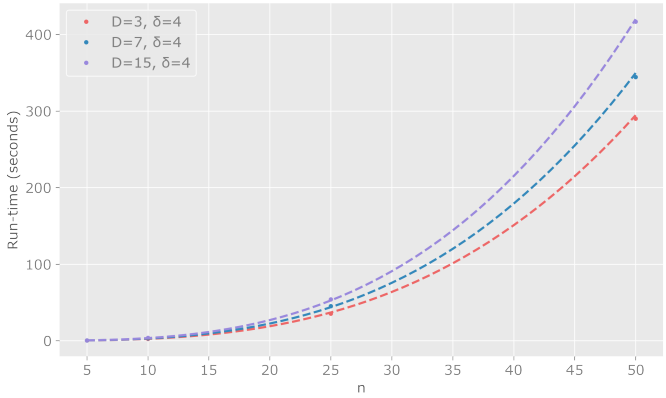


Figure 7.10: The run-time of one path multiplication for Protocols 5 and 6 plotted against the dimension n for different values of D with $\delta = 4$. The dashed functions are given by $5.25 \cdot 10^{-7} n^3 \ell^2$.

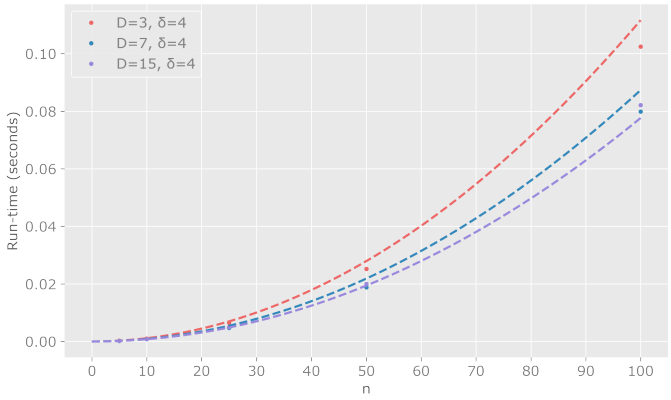


Figure 7.11: The run-time of one path addition for Protocols 7 and 8 plotted against the dimension n for different values of D with $\delta = 4$. The dashed functions are given by $4.85 \cdot 10^{-7} n^2 \ell$.

run-time for some input parameters, based on the analysis and the found dependencies above. First, all σ decision nodes are evaluated. Next, the path evaluation takes place that consists of $\sigma + \gamma - 3$ path multiplications or additions, depending on the approach. Then $\gamma \lceil \log_2 \gamma \rceil$ leaf node bit evaluations take place (if we neglect the γ multiplications of the path costs with a random number for the additive approach). To test this prediction, we evaluate a simple tree shown in Figure 7.13. Clearly, $\sigma = 2, \gamma = 3$. Taking $\delta = 4, D = 3$ and the corresponding parameters of q and ℓ, it is expected that for $n = 50$, the multiplicative approach evaluates the simple tree in $2 \cdot 1824 + 2 \cdot 290 + 6 \cdot 0.08 \approx 4228$ seconds. The exact run-time for this instance was 4185 seconds, around the same as expected. The additive evaluation run-time can be approximated by $2 \cdot 179 + 2 \cdot 0.025 + 6 \cdot 0.065 \approx 358$ seconds for $n = 50$. The exact run-time for this instance was 360.7 seconds. We can therefore conclude that by using our fond dependencies on all involved parameters, we can do a proper estimation of the total run-time of a specific instance.

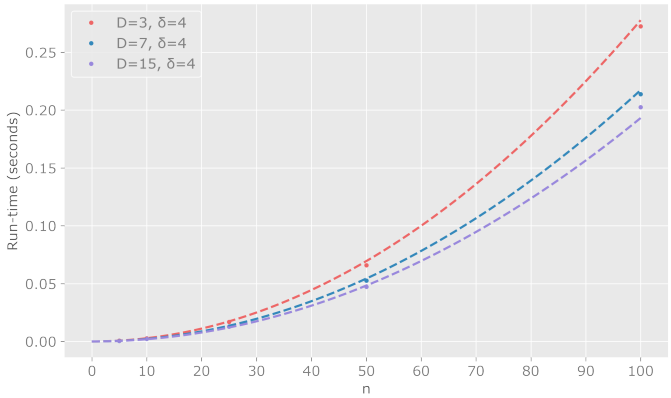


Figure 7.12: The run-time of one leaf bit evaluation for Protocols 7 and 8 plotted against the dimension n for different values of D with $\delta = 4$. The dashed functions are given by $1.21 \cdot 10^{-6} n^2 \ell$.

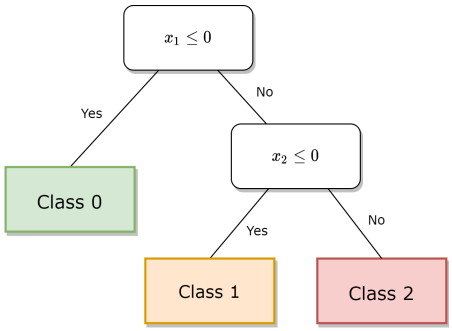


Figure 7.13: The decision tree used for testing the estimated total decision-tree evaluation run-time.

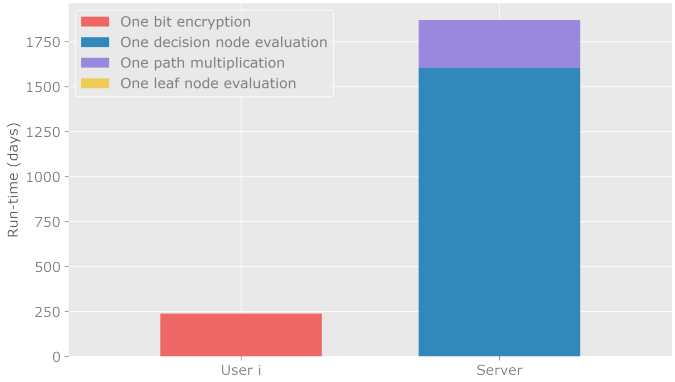


Figure 7.14: The expected run-times of the several parts executed by the users or server in Protocols 5 and 6 for $n = 1900, D = 15, \delta = 4$.

7.5.4. Communication

The protocols consist of 4 rounds. The communication sizes of the different rounds can be seen in Table 7.3. In the first round, the STTP communicates his public key to the users. This round only has to occur once, even when multiple trees are evaluated. The public keys of the STTP can be reused by the users to encrypt their input. The public key is a matrix of size $n \times m$ and is therefore of size $48 \cdot nm$ bytes. In the second round every user i send his $a_i\delta$ encryptions to the server with a total size of $48 \cdot a_i\delta n^2\ell$ bytes. In the third round, the server sends the result encryptions of either $48 \cdot \lceil \log_2 \gamma \rceil n$ or $48 \cdot \gamma(\lceil \log_2 \gamma \rceil + 1)n$ bytes to the STTP, depending one the approach. Namely, only one column per ciphertext needs to be communicated in order to do the decryption. The second amount is for the additive approach where for every leaf node the encrypted label bits and the encrypted cost have to be communicated. Additionally for Protocols 5 and 7, the server sends $\lceil \log_2 \gamma \rceil$ random bits to each user. In the fourth round, $\lceil \log_2 \gamma \rceil$ partial decryptions are communicated from the STTP to the decrypting party. This last communication is of size $48 \cdot \lceil \log_2 \gamma \rceil$ bytes.

Table 7.3: Communication sizes in bytes of the different rounds in Protocols 5 to 8. The communication between brackets in the first round only has to occur once and not every time a tree is evaluated.

	Round 1	**Round 2**	**Round 3**	**Round 4**
Protocol 5	$(48nm)$	$48a_i\delta n^2\ell$	$48\lceil \log_2 \gamma \rceil n + \lceil \log_2 \gamma \rceil$	$48\lceil \log_2 \gamma \rceil$
Protocol 6	$(48nm)$	$48a_i\delta n^2\ell$	$48\lceil \log_2 \gamma \rceil n$	$48\lceil \log_2 \gamma \rceil$
Protocol 7	$(48nm)$	$48a_i\delta n^2\ell$	$48\gamma(\lceil \log_2 \gamma \rceil + 1)n + \lceil \log_2 \gamma \rceil$	$48\lceil \log_2 \gamma \rceil$
Protocol 8	$(48nm)$	$48a_i\delta n^2\ell$	$48\gamma(\lceil \log_2 \gamma \rceil + 1)n$	$48\lceil \log_2 \gamma \rceil$

7.5.5. Summary

To give a better idea of the run-time complexity required for each party, in Figures 7.14 and 7.15, the distribution of the expected run-time of the separate parts for both the users and the server can be seen assuming the parameters $D = 15$ and $\delta = 4$. The run-time is again based on the times found for $n = 50$ and taking into account the found dependency on n. Here, we neglected the run-time required for the STTP, since this party only executes one key generation and some decryptions of which the run-times can be neglected compared to the run-times for the users and server. Here, n is chosen such that the security is guaranteed. Clearly, the run-time required for the leaf-node evaluations is very low compared to the decision-node and path evaluations. For the additive protocols, also the path

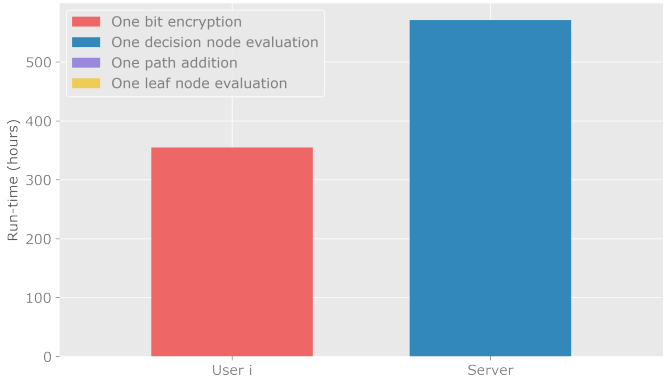


Figure 7.15: The expected run-times of the several parts executed by the users or server in Protocols 7 and 8 for $n = 1400, D = 15, \delta = 4$.

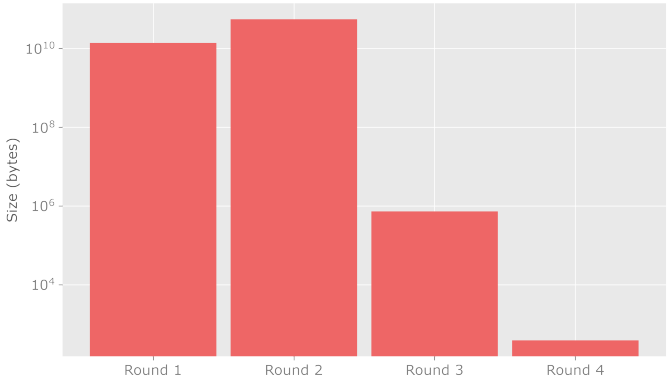


Figure 7.16: The communication sizes on a logarithmic scale of the different rounds in Protocols 5 and 6 for $n = 1900, D = 15, \delta = 4, a_i = 1$ for a tree with $\gamma = 256$.

evaluation (which are homomorphic additions) run-time is very low. The main bottlenecks regarding the run-time are the encryption time and the decision-node evaluation. Again, the reader should note here that the decision-node evaluation's run-time is worst-case here; it is assumed that all threshold values are equal to 0. In practice, this run-time can be lower. It should be noted here that the total encryption times increase linearly with the number of input variables a_i per user and the input bit-length δ. Also, for the users the total time for the decision-node evaluations is increasing linearly with the number of decision nodes and for the path multiplications linearly with both σ and γ.

For Protocols 5 and 6, One decision-node evaluation for $\delta = 4$ is expected to take 14600 hours for a tree with depth 3, which is around 1.5 years. For a tree with depth 15, this is even 38500 hours or 4.5 years. One path multiplication takes between 100 and 266 days. For one bit encryption executed by the users, we find run-times for these trees between 2200 and 5700 seconds or 92 and 237 days. For Protocols 7 and 8 one decision-node evaluation takes between 570 and 805 hours. One bit encryption between 233 and 594 hours. Clearly, the run-time complexity of the additive protocols is much lower.

We can also plot the communication costs per round. Here, we assume that the number of leaf nodes is equal to $\gamma = 256$ and that all users have one input variable, $a_i = 1$. The result can be seen in Figures 7.16 and 7.17. For these figures, we again took n such that both $\lambda, \tau = 110$ and the security

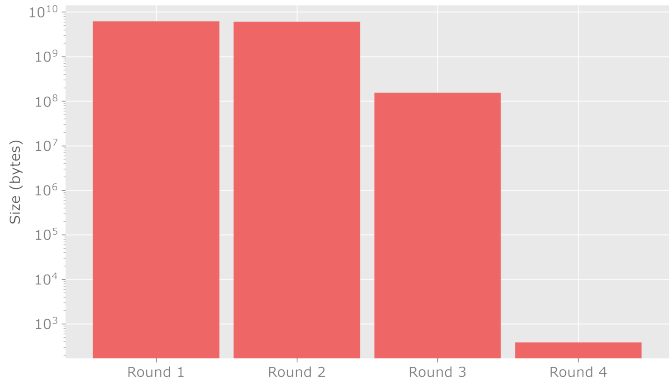


Figure 7.17: The communication sizes on a logarithmic scale of the different rounds in Protocols 7 and 8 for $n = 1400, D = 15, \delta = 4, a_i = 1$ for a tree with $\gamma = 256$.

is guaranteed. As expected, rounds 1 and 2 are significantly more costly than all other rounds. It is assumed that every user only has one input value ($a_i = 1$), thus the communication in round 2 is even bigger when the users have more inputs. For these parameters, the communication in the second round is around 55 or 6 gigabytes per input variable. The public key sizes in the first round are 14 and 6.2 gigabytes, respectively. For the third round, in Protocols 5 and 6, $\lceil \log_2 \gamma \rceil$ ciphertext columns of size n are communicated. In Protocols 7 and 8 this is $\gamma \left(\lceil \log_2 \gamma \rceil + 1 \right)$ columns. Therefore, the communication in this round for the last two protocols is slightly higher.

7.6. Semi-Trusted Third Party protocols using key-switching

The role of the STTP for Protocols 9 and 10 is only to generate multiple public keys and multiple switching keys. The public keys are used by the users to generate other switching keys. The server can then use these switching keys to switch all input ciphertexts to the same secret key (of the STTP). By doing this, the server is then able to combine the different decision-node evaluation results. Since the decision-node evaluations, the path evaluations and the leaf-node evaluations have similar results as Protocols 5 and 6 or 7 and 8, we do not show the result figures here. The plots can be found in Appendix A.3. Also, the encryption procedure is not mentioned in this section since it is the same method as for Protocols 1 to 4. Namely, now the encryption is done using the users' secret key. These results can also be seen in Appendix A.3.

7.6.1. Key generation

Now, the STTP needs to generate, next to his secret key, also one public key (an encryption of 0) for every user and, using the public keys of the users, one switching key per user. The users also need to generate one public key and one switching key. The generation time of the secret key and the switching keys can be neglected compared to the generation of the public keys. As an example, for $n = 50$ and $\delta = 8, D = 15$, the key generation times are $0.004, 3.867$ and 0.017 for one secret, public or switching key. We therefore neglect the computational cost of the secret key and switching key generation. The public key generation time can be estimated by the encryption run-time of one bit. The difference between the key generation of the STTP and the users, is that the server has to generate k public keys and the users only one public key.

7.6.2. Key-switching

The server switches the key of the input ciphertexts of the different users, such that these can be combined during the evaluation. The key-switching procedure switches the key of a ciphertext by homomorphically multiplying this ciphertext with the switching key. Therefore, the cost is similar to that of one path multiplication as can be seen in Figure 7.18. In total, the server has to do $\sigma + k\lceil \log_2 \gamma \rceil$ or $\sigma + k\gamma(\lceil \log_2 \gamma \rceil + 1)$ key-switches for Protocol 9 or 10 respectively. A similar result was found for Protocol 10 and can be seen in Figure A.21 in the appendix.

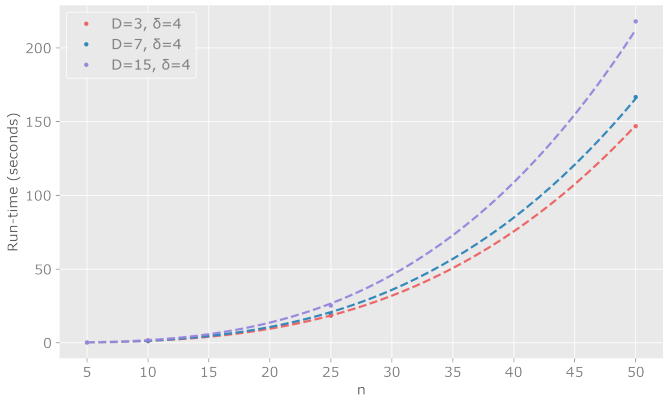


Figure 7.18: The run-time of one key-switching procedure for Protocol 9 plotted against the dimension n for different values of D with $\delta = 4$. The dashed functions are given by $4.71 \cdot 10^{-7} n^3 \ell^2$.

7.6.3. Communication

Again these protocola consist of 4 rounds, of which the first two are the rounds required to communicate the public keys and switching keys. These two rounds only have to occur once, since the public and switching keys can be re-used. In the first round, the STTP engages with every user to exchange their public key of size $48 \cdot n^2 \ell$ bytes (an encryption of 0). In the second round, the STTP sends k switching keys to the server with a total size of $48 \cdot kn^2 \ell$ bytes. After this, every user sends their $a_i \delta$ encryptions plus a switching key to the server. This together has size $48 \cdot (a_i \delta + 1) n^2 \ell$ bytes. In the fourth round, the server sends the result that consists of either $\lceil \log_2 \gamma \rceil$ ciphertext column for Protocol 9 or $\gamma(\lceil \log_2 \gamma \rceil + 1)$ encryptions for Protocol 10 to the user. Therefore, this communication contains $48 \cdot \lceil \log_2 \gamma \rceil n$ or $48 \cdot \gamma(\lceil \log_2 \gamma \rceil + 1) n$ bytes.

Table 7.4: Communication sizes in bytes of the different rounds in Protocols 9 and 10. The communication between brackets in the first two rounds only has to occur once and not every time a tree is evaluated.

	Round 1	**Round 2**	**Round 3**	**Round 4**
Protocol 9	$(48n^2\ell)$	$(48kn^2\ell)$	$48a_i\delta n^2\ell + 48n^2\ell$	$48\lceil \log_2 \gamma \rceil n$
Protocol 10	$(48n^2\ell)$	$(48kn^2\ell)$	$48a_i\delta n^2\ell + 48n^2\ell$	$48\gamma(\lceil \log_2 \gamma \rceil + 1)n$

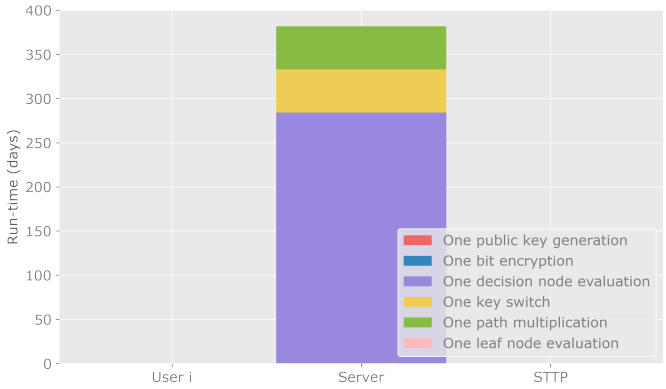


Figure 7.19: The expected run-time of the several parts executed by the users, server or STTP in Protocol 9 for $n = 1350, D = 15, \delta = 4$.

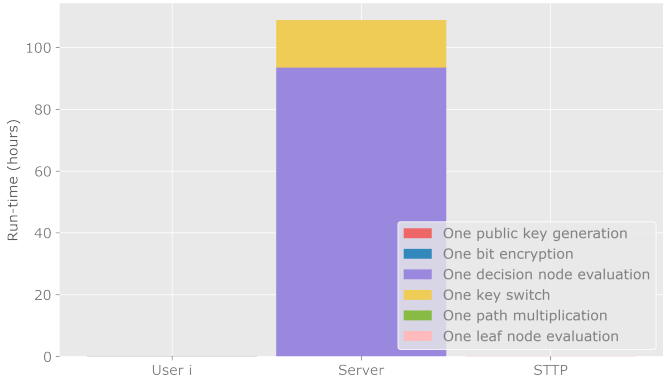


Figure 7.20: The expected run-time of the several parts executed by the users, server or STTP in Protocol 10 for $n = 950, D = 15, \delta = 4$.

7.6.4. Summary

In Figures 7.19 and 7.20, the distribution of the expected run-time of the separate parts for all parties can be seen for both protocols. These expectations are again based on the run-time found for the highest value of n of our runs, adapted for the found dependency on n. The parameters used here are the parameters for a secure protocol with $\lambda = 110$ and $\tau = 110$. The total contribution of the key-switches in the total run-time of the server depends linearly on k, since the result encryptions have to be switched to the key of all users. Clearly, the computation costs required by the users and the STTP are negligible compared to that of the server. The total contribution of the decision-node evaluations is linear in the number of decision nodes σ. Again, the run-time of a decision-node evaluation is for the worst-case situation (where all the threshold values are equal to 0). In practice, this run-time is lower. Protocol 9 requires $\sigma + k(\lceil \log_2 \gamma \rceil)$ key-switches and $\sigma + \gamma - 3$ path multiplications. Protocol 10 requires $\sigma + k\gamma(\lceil \log_2 \gamma \rceil + 1)$ key-switches. It can also be seen that the run-time of one decision-node evaluation of Protocol 9 parts is more than 75 times higher than for Protocol 10. The run-time of one key-switch for these parameters is 49 days and 15 hours for the two protocols respectively. This means that the run-time of one key-switch of Protocol 9 parts is also more than 75 times higher than for Protocol 10.

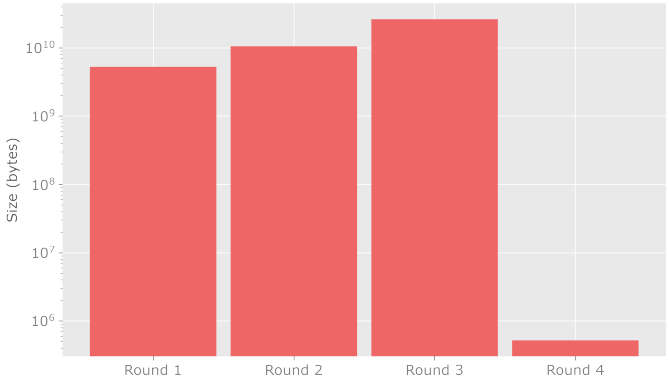


Figure 7.21: The communication sizes on a logarithmic scale of the different rounds in Protocol 9 for $n = 1350, D = 15, \delta = 4, a_i = 1$ and $k = 2$ for a tree with $\gamma = 256$.

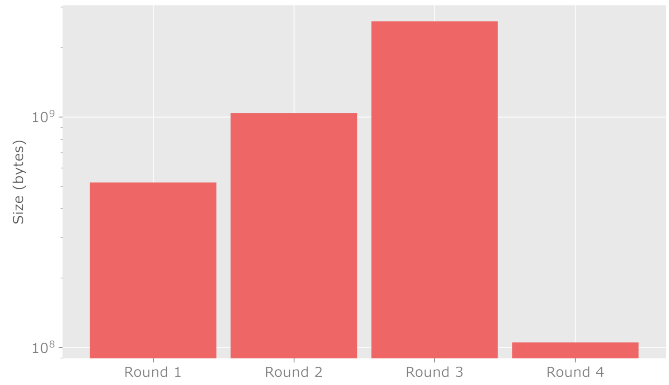


Figure 7.22: The communication sizes on a logarithmic scale of the different rounds in Protocol 10 for $n = 950, D = 15, \delta = 4, a_i = 1$ and $k = 2$ for a tree with $\gamma = 256$.

In the first three rounds, 1, k, $a_i\delta + 1$ ciphertexts are communicated respectively. The first two rounds only have to occur once. In the fourth round $\lceil \log_2 \gamma \rceil$ ciphertext columns of size n are shared. For protocol 10, this is $\gamma(\lceil \log_2 \gamma \rceil + 1)$ columns. This communication is very low compared to the communication in the other rounds. The first round stays constant in size. The other rounds depend on the parameter values of a_i, δ, k. An example of the communication for these two protocols can be seen in Figures 7.21 and 7.22 for a tree with $\gamma = 256$. The size of one ciphertext for Protocol 9 for $\delta = 4, D = 15$ is 5.3 gigabytes. For Protocol 10, this is 0.52 gigabytes, which is more than 10 times smaller. Only the communication in the fourth round is lower for Protocol 9 than for Protocol 10.

7.7. Comparison of the protocols

In this section we compare the different protocols on the aspects of key generation, encryption, decision-tree evaluation, and communication costs.

7.7.1. Key generation

The key generation of the protocols that make use of an STTP, namely Protocols 5 to 10, is very efficient compared to the key generation of the Multi-Key FHE protocols. For $n = 50$ and for all input parameters, the key generation for these protocols does not take more than 4 seconds. This is in great contrast to the key generation run-time of Protocols 1 to 4. For those protocols, the run-time can reach 8000 seconds already for only a value of $n = 4$. The key generation of these protocols is far from practical for realistic parameters.

7.7.2. Encryption

The encryption procedure of the STTP protocols without key-switching is more costly than for the other protocols. Namely, that encryption procedure encrypts under the public key instead of the secret key, which requires a costly multiplication of a $n \times m$ matrix and a $m \times n\ell$ matrix. The difference can be seen in Figure 7.23, where the encryption time is plotted against n for all protocols for $D = 3, \delta = 4$. Here, again we extrapolated our result for the highest value of n using the found dependencies of the previous sections.

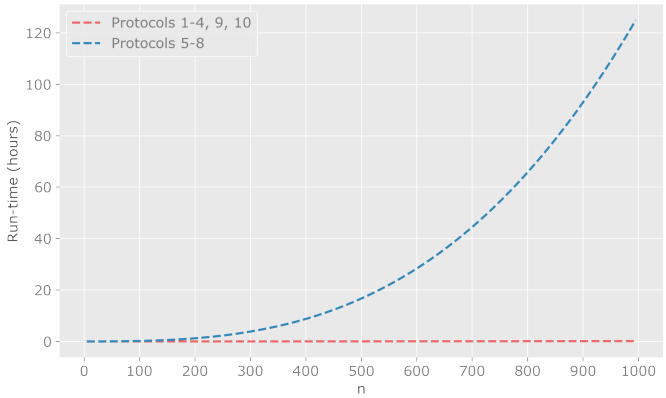


Figure 7.23: The expected encryption time of one bit (by a user) for the different protocols plotted against n for $D = 3, \delta = 4$.

7.7.3. Decision-tree evaluation

The decision-tree evaluation of the protocols that make use of Multi-Key FHE contains the extension procedure after every decision-node evaluation. The extension procedure depends on n with a power of 5 and therefore its run-time complexity greatly increases for higher n. Realistic values of n give non-practical decision-tree evaluation times. For these protocols, in comparison to the extension procedure, all other components of the evaluation can be neglected. As an example, for $k = 2, D = 3, \delta = 4$ the run-time of one complete extension procedure is around 200 seconds already for $n = 4$. For $n = 50$, the run-time will be in the order of years. In comparison to the other protocols that use an STTP, the decision-tree evaluation is much more costly.

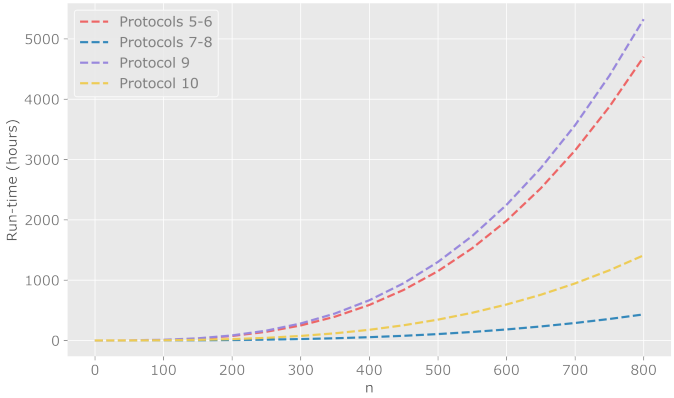


Figure 7.24: The expected decision-tree evaluation run-time for different protocols plotted against n for $D = 3, \delta = 4, \sigma = 2, \gamma = 3$ with $k = 8$.

To compare the additive decision-tree evaluation with the multiplicative evaluation, Figure 7.24 gives the expected total evaluation time of Protocols 5 to 10. Here, we took a tree with $\sigma = 2, \gamma = 3$ and $k = 6$. Clearly, the additive procedure is less expensive. This also holds for bigger trees. This can be explained by the fact that the paths are evaluated by less expensive additions instead of multiplications. Additionally, this causes the value of ℓ to be much lower, which makes the sizes of the ciphertexts smaller. Next to this, since q is lower due to lower error propagation, this also results in a lower necessary value of n for security. The key-switches do not cause a significant higher error propagation since each ciphertext is switched only twice: after the decision-node evaluation and before it is send back to a user (Table A.7 to A.11 in the appendix). Therefore, even Protocol 10, that needs to do more key-switches, namely a number that depends linearly on $k\gamma$ instead of only k, is more efficient than Protocol 9. This is due to the fact that every homomorphic multiplication of Protocol 9 is more expensive than of Protocol 10.

In Figure 7.25, the run-time of each evaluation component for the different protocols can be found. Here, n is fixed to 800. Clearly, the costs of the path evaluations are much smaller for Protocols 7, 8 and 10. The main cost for all protocols come from the decision-node evaluations and/or the key-switches. The difference between the run-times for the decision-node evaluations of the protocols can again be explained by the difference in ℓ and therefore in the size of each ciphertext. Protocol 10 requires many key-switches for this instance with $k = 8$, that, together, are more costly than the decision-node evaluations. Since the number of key-switches is linearly dependent on both σ, γ and k (and the decision-node evaluation only on σ), increasing either γ or k causes this protocol to be more expensive than Protocols 7 and 8. However, increasing γ always comes together with an increase in σ or vice versa. Therefore, of these parameters mainly the value of k impacts the difference in run-times between the additive protocols. But still, for a value of n that guarantees the security, which is higher for Protocols 7 and 8 than for Protocol 10, the total run-time is higher, even for $k = 8$. This can be seen in Figure 7.26.

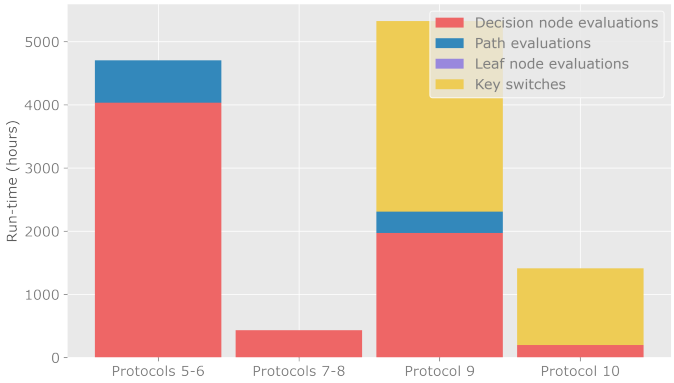


Figure 7.25: The expected decision-tree evaluation run-time for different protocols for $n = 800$ and $D = 3, \delta = 4, \sigma = 2, \gamma = 3$ with $k = 8$.

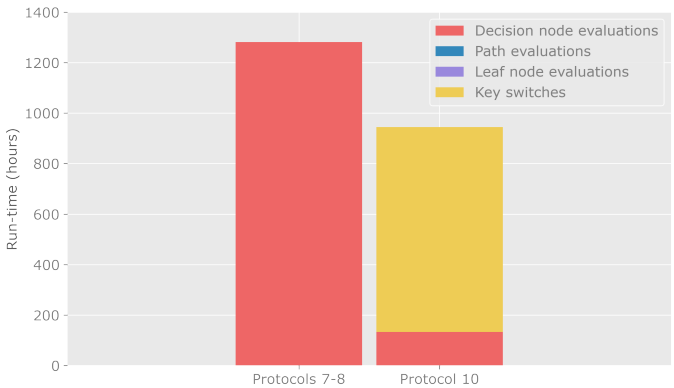


Figure 7.26: The expected decision-tree evaluation run-time for different protocols for $D = 3, \delta = 4, \sigma = 2, \gamma = 3$ with $k = 8$. The value of n is chosen such that security is guaranteed according to Tables A.8 and A.11.

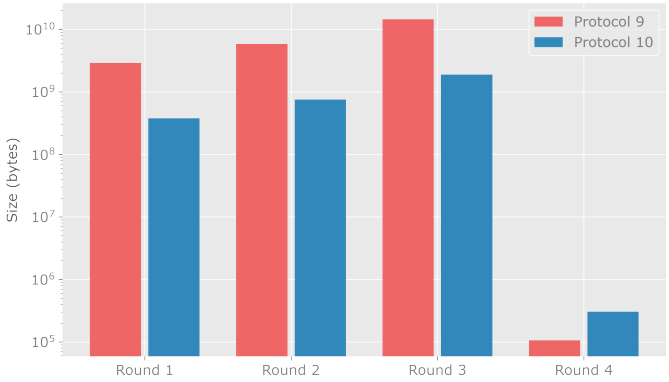


Figure 7.27: The required communication on a logarithmic scale per round for Protocols 9 and 10 for $D = 3, \delta = 4, \gamma = 3, a_i = 1, k = 2$. The value of n is chosen such that security is guaranteed according to Tables A.10 and A.11.

7.7.4. Communication

In this section we compare the communication costs of all protocols. Previously, we already concluded that the public key required for extension in the first four protocols is very big. The communication in the other rounds is very low compared to the communication of this big public key in the first round. To be specific, this public key is of size $48 \cdot n^3 m\ell^2$ which already reaches a size in the order of terabytes for $n = 50$ and the lowest possible tree parameters.

Protocols 5 to 10 do not require such a big key to be communicated. In order to compare these protocols on the communication costs in the different rounds, in Figure 7.27 these are plotted for the protocols that use an STTP in combination with key-switching. This is done for the parameters that guarantee security, namely such that $\lambda, \tau = 110$. It was already demonstrated above that the runtime of Protocol 10 is the lowest. In Figure 7.27, it can be seen that this protocol also requires less communication in the first three rounds. This is caused by the lower value for ℓ and n such that the ciphertext sizes are smaller. Only in the fourth round, for Protocol 10 for every leaf node some ciphertexts are communicated which results in a slightly higher communication size. In comparison to the first three rounds, the communication in the last round is that low, that the small difference does not have a big effect on the communication costs in total. The number of input variables per users is set to $a_i = 1$, which means that the communication in the third round will be higher if this is chosen to be a higher value (increases linearly in a_i).

The same conclusions can be drawn for Protocols 7 and 8 compared to Protocols 5 and 6. This can be seen in Figure 7.28. Now, the third round is the round where for every leaf node some ciphertexts are sent for the additive approach. Therefore, the communication in that round is a bit higher for Protocols 7 and 8. The first two rounds are again less expensive in terms of communication, due to a lower ℓ and n.

In Figure 7.29, the communication sizes of Protocols 7, 8 and 10 are shown together. Here, all key-exchange rounds are summed together. The input round is the round in which a user sends his input to the server. The result round is the round where the server communicates the result ciphertexts with another party. Clearly, the total communication required for the key exchange is lower for Protocol 10, even for the fixed value $k = 8$. The number of keys that needs to be communicated in Protocol 10 depends on the number of parties. But, these keys are equal in size to ciphertexts, such that they are much smaller than the public keys communicated in Protocols 7 and 8. The difference in the other rounds can be explained by the lower value of n and ℓ for Protocol 10.

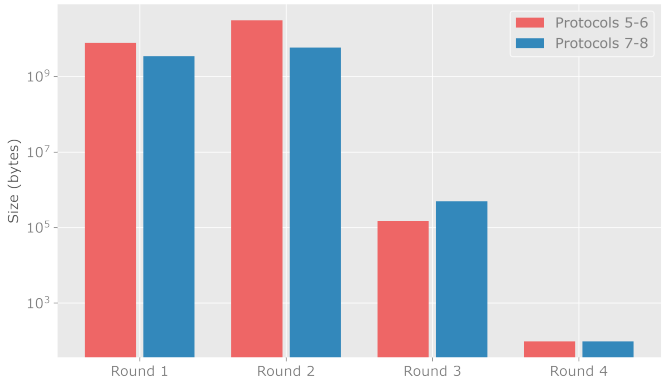


Figure 7.28: The required communication on a logarithmic scale per round for Protocols 5 to 8 for $D = 3, \delta = 4, \gamma = 3, a_i = 1$. The value of n is chosen such that security is guaranteed according to Tables A.7 and A.8.

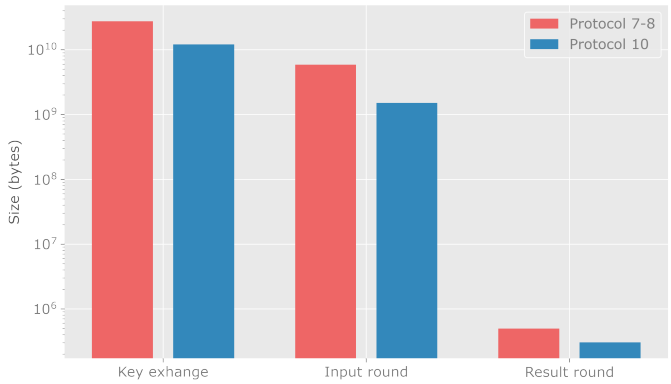


Figure 7.29: The required communication on a logarithmic scale per round for Protocols 7, 8 and 10 for $D = 3, \delta = 4, \gamma = 3, a_i = 1$ and $k = 8$. The value of n is chosen such that security is guaranteed according to Tables A.8 and A.11.

7.8. Decision-node evaluations

In the above analyses, we assumed the threshold values of all decision nodes to be equal to 0. This gives the worst-case run-time, since now for all bit positions multiple homomorphic multiplications (see Formula 5.1) need to be calculated. To give an idea of the run-time for other threshold values, in this section we evaluate the run-time of one decision-node evaluation while varying the threshold values. We fix the ciphertext dimension to $n = 100$ and choose $\ell = 16$ which is the value of ℓ for Protocol 10 given $\delta = 4, D = 3$. The result can be seen in Table 7.5. We also compute the number of homomorphic multiplication required for the decision-node evaluation given the threshold value. Clearly, the run-times agree with the number of homomorphic multiplications that are required for the specific threshold value.

Table 7.5: The run-time in seconds of one threshold or categorical decision-node evaluation for different threshold or category values (denoted in bits), given $n = 100, \ell = 16, \delta = 4$.

Threshold / Category	# Hom. multiplications comparison	Comparison protocol time (s)	Equality protocol time (s)
0000	6	689.7	333.7
0001	3	336.4	333.0
0010	4	447.3	332.7
0011	1	111.2	336.6
0100	5	577.5	334.9
0101	2	225.0	335.5
0110	3	339.1	349.1
0111	0	0.01145	338.7
1000	6	695.5	335.9
1001	3	336.9	332.3
1010	4	446.2	339.5
1011	1	110.5	332.1
1100	5	561.9	331.7
1101	2	224.0	346.9
1110	3	330.6	337.9

The comparison protocol sometimes homomorphically multiplies the same ciphertexts. Take for example the threshold value 0000, then

$$[[x > 0]]_2 = [[x_3]]_2 + [[x_0]]_2 \boxdot [[d_1]]_2 \boxdot [[d_2]]_2 \boxdot [[d_3]]_2 + [[x_1]]_2 \boxdot [[d_2]]_2 \boxdot [[d_3]]_2 + [[x_2]]_2 \boxdot [[d_3]]_2.$$

It can be seen that the result of $[[d_2]]_2 \boxdot [[d_3]]_2$ is used two times. This means that storing this value lowers the number of homomorphic multiplications by one. This also holds for all threshold value where the last two bits are 0, so the threshold values 0000, 0100, 1000 and 1100. Depending on the tree, this can slightly lower the run-time of some of the decision-node evaluations.

Next to threshold decision nodes, there can also be categorical decision nodes. In Section 5.6.2, we demonstrated how to evaluate these homomorphically. Every equality protocol requires $\delta - 1$ homomorphic multiplications. In Table 7.5, also the evaluation times of one categorical decision node for different category values are given. As expected, the run-times are similar to evaluating a threshold decision node that requires $\delta - 1 = 3$ homomorphic multiplications. Dependent on the threshold or category value, either the comparison protocol or equality protocol is more efficient.

7.9. Possible modifications

In this section we shortly describe two protocol modifications, that make some of the protocols slightly different and thereby impact the run-times or communication costs.

7.9.1. Encryption in STTP protocols without key-switching

Protocols 7 and 8 that use an STTP without key-switching have the most efficient decision-tree evaluation. The downside of these two protocols is that the users have to encrypt all their bits using the STTP's public key. This is a very expensive procedure, of which the run-time can be seen in Figure 7.8. Inspired by the key-switching procedure and the use of zero encryptions as public keys to produce the switching keys, we can change the encryption procedure for these protocols. The STTP can send to every user $a_i\delta$ encryptions of 0 which we again call public keys. The cost of generating these encryptions by the STTP are reasonable, since the STTP uses his private key for this encryption. Then, each user can use these 0 encryptions to encrypt their input bits under the STTP's secret key, namely applying the plaintext addition procedure with as plaintext the input bit. This procedure is less expensive than the initial encryption method. Also, the error propagation is reduced since the bit encryptions are less noisy, such that ℓ can be chosen as a lower value (Table A.9 in the appendix). The downside is that this communication has to take place each time the user sends input encryptions to the server. This can be solved by letting the STTP communicate more 0-encryptions such that multiple trees can be evaluated. As an example, in Figures 7.30 and 7.31, the expected run-times for the users and server and the communication sizes per round are given for Protocols 7 and 8 with the proposed adapted encryption procedure. In Table 7.6, we give the communication sizes of the different rounds for the adapted version of Protocols 7 and 8.

Table 7.6: Communication sizes in bytes of the different rounds in the adapted versions of Protocols 7 and 8.

	Round 1	**Round 2**	**Round 3**	**Round 4**
Protocol 7 - adapted	$48a_i\delta n^2\ell$	$48a_i\delta n^2\ell$	$48\gamma(\lceil\log_2\gamma\rceil+1)n+\lceil\log_2\gamma\rceil$	$48\lceil\log_2\gamma\rceil$
Protocol 8 - adapted	$48a_i\delta n^2\ell$	$48a_i\delta n^2\ell$	$48\gamma(\lceil\log_2\gamma\rceil+1)n$	$48\lceil\log_2\gamma\rceil$

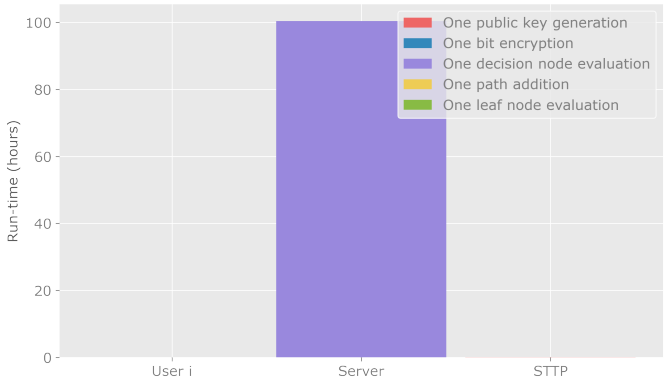


Figure 7.30: The expected run-time of the several parts executed by the users, server or STTP in Protocols 7 and 8 for $n = 950$, $D = 15$, $\delta = 4$.

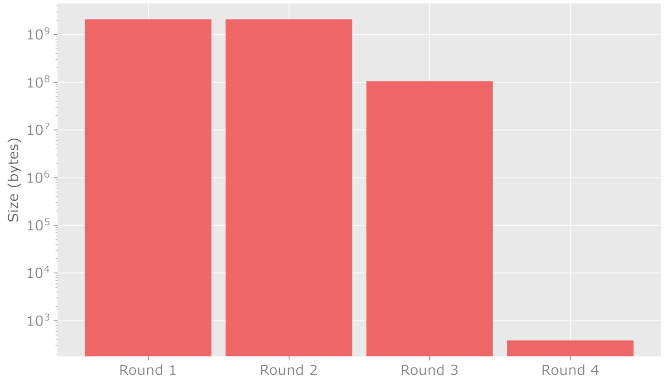


Figure 7.31: The communication sizes on a logarithmic scale of the different rounds in Protocols 7 and 8 with the adapted encryption procedure for $n = 950, D = 15, \delta = 4, a_i = 1$ for a tree with $\gamma = 256$.

7.9.2. Homomorphic operations on column ciphertexts

The protocols that follow the additive approach have the benefit that after the decision-node evaluations, no more homomorphic multiplications are executed. Only homomorphic additions, plaintext multiplications and plaintext additions are done next. Since for decryption only one column is needed, these operations can also be done on only this specific column. This results in the complexity of these operations to be $\mathcal{O}(n)$ instead of $\mathcal{O}(n^2\ell)$. This can slightly change the run-times of the protocols. Since the decision-node evaluations are still the most expensive procedures, this does not impact the overall execution times much. This is not possible for the protocols that make use of key-switching, since the key-switching procedure is a homomorphic multiplications which also needs to be executed at the end of the complete tree evaluation.

8

Discussion, conclusions and future work

This chapter starts with a short summary of our research. Next, we present the discussion and conclusions. Afterwards, we give recommendations for future research.

8.1. Summary

Decision-tree evaluation is a useful tool in many application areas. It is a promising method for determining access within a Risk-Adaptive Access Control (RAdAC) system where access-related information is provided by multiple entities, as considered as the exemplary use-case of this thesis. Decision trees can be used to make access decisions that are dynamic and flexible and can be adapted to the current context and exceptional or critical circumstances. Within healthcare, sharing EHRs of patients between medical providers is an example of where these dynamic-access decisions are essential. Next to this, decision-tree models are shown to be effective in classifying instances of fraud, malware or diseases. Especially when data of more than one party is combined for the evaluation, the results can be very fruitful.

However, collaborative decision-tree evaluation raises many privacy concerns. In order for the decision-tree classification to be successful, it needs access to sensitive data, like financial or health-related records. It is important to keep this data private. This is even more necessary when the decision-tree evaluation is done in a collaborative manner where more than one party provide sensitive input data. In addition, the decision-tree model can be a valuable asset to the holder and can not always be publicly known. As long as no solution exists for these privacy issues, parties are expected to be hesitant regarding working together using decision-tree evaluation in these application areas. Therefore, we investigated whether it is possible to evaluate a decision tree in a privacy-preserving way when the input originates from more than one party, while making use of homomorphic encryption. We proposed four different protocols in the access-control setting where the holder of the decision tree gains the evaluation result, and six different protocols in the setting where the users gain the result. Additionally, we thoroughly analysed and implemented all protocols.

8.2. Discussion

The first protocols we proposed for both settings make use of Multi-Key Fully Homomorphic Encryption (FHE). This makes it possible for the users to encrypt under their own key such that they do not have to put trust in another party. These characteristics make this method very promising. However, the implemented Multi-Key FHE is very inefficient due to the high run-time for the extension method, which extends a ciphertext to a new key. In addition, the method requires the users to generate big public keys which are needed by the server to perform the extension procedure. Our results show that the run-time complexity for both the users and the server reaches non-practical values in the order of years, even for very small trees. Namely, the run-times of the key generation and the extension procedure grow with a power of four or five in the order of the dimension n of the ciphertexts respectively, where n is a fixed value with a certain minimum to guarantee security of the protocols. Therefore, these protocols are currently not feasible. Nonetheless, since the exact complexity of these protocols depends on the underlying Multi-Key FHE, and given the progressing research within Multi-Key FHE, these protocols can become more and more efficient. Therefore, future application is not completely excluded.

All our proposed protocols differ in the type of path evaluation they perform: either they make use of the multiplicative approach where all decision bits per path are *multiplied*, or the additive approach where they are *added* per path. The additive approach has the advantage that $\sigma+\gamma-3$ fewer homomorphic multiplications are required in the complete protocol, where σ is the number of decision nodes and γ the number of leaf nodes. Homomorphic multiplications are a factor of $n\ell$ times less complex in terms of run-time than additions, such that the run-time of a homomorphic addition is negligible compared to a homomorphic multiplication. One homomorphic multiplication in our implementation takes at least 10 hours, depending on the tree parameters and the protocol. Replacing these path multiplications by additions can therefore result in a speed-up of at least $\sigma+\gamma-3$ times 10 hours. In addition, the amount of noise in the final result ciphertexts is lower which enables us to set the modulus q and therefore also the parameters n,ℓ to a lower value. Since this results in a lower ciphertext size, all operations can be done more efficiently. Therefore, the results show that the additive approach is less costly in terms of decision-tree evaluation run-time by the server.

For the additive approach, the communication sizes in the key-exchange round(s) and the input round are less costly. This can again be explained by the lower value of n an ℓ and therefore smaller ciphertext sizes. It is interesting to see that for higher D, the value of ℓ is lower, since D' is higher. Therefore, for fixed value of n, the higher D, the smaller the ciphertext sizes. The number of ciphertexts that the server needs to communicate in order to find the correct (randomised) evaluation, however, is γ times higher than for the multiplicative path evaluation approach. So, the communication cost of this round is higher than for the multiplicative protocols. However, in comparison to the other earlier rounds, this communication is at least 10000 times smaller. Therefore, this slightly higher communication does not have a big impact on the overall communication costs. Therefore, it can be concluded that both the overall communication costs and the run-time are lower for the protocols that make use of the additive approach.

The use of a Semi-Trusted Third Party (STTP) allows the users to encrypt under the public key provided by the STTP, such that the server is able to combine the different input ciphertexts. Therefore, no expensive key-extension method is required as for the protocols that make use of Multi-Key FHE. The protocols differ in the amount of trust that needs to be put into this additional party: either the STTP needs to generate keys *and* perform decryption; or it only participates in the key generation in which some additional keys are generated. These additional keys can then be used by the server to apply *key-switching* to switch the key under which a ciphertext is encrypted. This last method has the advantage that the users can encrypt under their own key (instead of the key of the STTP) and that no further communication is required with the STTP after the keys are exchanged. Unfortu-

nately, the method that uses an STTP in combination with key-switching, can not be applied within the access-control setting. Namely, the server would then able to decrypt the input ciphertexts of the users making use of the key-switching keys.

Due to randomisation, the STTP without key-switching does not gain any knowledge about the evaluation result, the tree, or the users' input in our protocols, except for the number of leaf nodes. Therefore, finding such an STTP is not a complex task. When key-switching is applied, this task is even easier since the STTP only joins the key generation. In case the STTP colludes with one of the users, both gain no advantage in our STTP protocols due to randomisation. When the server colludes with the STTP, the server can decrypt the input of the users. So, this should be prevented. For the protocol without key-switching and where the users receive the evaluation result, the evaluation result can also be determined by the server when the sever colludes with the STTP. In all protocols, the users do get to know the number of leaf nodes of the tree, but, sending fake additional ciphertexts can hide this value.

The protocol that makes use of an STTP without key-switching is suitable for access control, where the server receives the evaluation result. The protocol requires 4 rounds, of which the first round consists of the key-exchange and only needs to be executed once. After the tree evaluation done by the server, in which the additive approach is used, the randomised result is decrypted by the STTP and sent back to the server. The encryption procedure executed by the users, where encryption takes place using the public key of the STTP, is very expensive in comparison to encryption using a secret key. Concretely, the encryption time (of one bit) is in the order of days for all tree sizes. At the cost of more communication, we proposed an alternative; the STTP sends 0-encryptions to the users in the first round which can then be used by the users to encrypt their input bits. These 0-encryptions can be generated by the STTP in 150 to 300 seconds per encryption depending on the tree depth and the input bit length. Although this encryption time is quite high, the advantage is that these encryptions can be generated at any time and can be stored for later use. The run-time for the user to add their plaintext bits to the 0-encryptions is lower than 5 seconds per bit for any tree depth and input bit length. Therefore, this greatly reduces the load on the users. This is beneficial, since not all users have a high computational power. The communication size of one bit ciphertext varies between 376 MB and 606 MB depending on the tree depth and the input bit length. Assuming an internet speed of 15 MB per second, receiving/sending one ciphertext can take up to 40 seconds. The communication sizes in the last two rounds can be neglected compared to the communication in the other rounds.

For the setting where the users receive the classification result, and the decision tree acts as an external resource, two of our protocols are the most feasible options. One of those is the same as for the access-control setting, but now the last communication round takes place between the STTP and the users in order to communicate the result with the users instead of the server. This protocol requires 3 communication rounds every time a tree is evaluated, assuming that in the first key-exchange round enough 0-encryptions are sent for multiple tree evaluations. The other possibility is the method using key-switching in combination with an STTP. This method only requires 2 communication rounds for every new tree that is evaluated. Only once two key-exchange rounds have to take place between the server and both the users and the server. In the first round, the users and STTP interact to share 0-encryptions that again vary between 376 MB and 606 MB per encryption. Then some information is added by the server and the users, who then send the result to the server. The server receives $2k$ switching keys with the same size as the 0-encryptions. Clearly, in comparison to the other protocol, the advantage of using key-switching is that every evaluation only takes 2 rounds and that less communication is required in the key-exchange. The drawback of this method is that, since the server needs to switch the keys of the ciphertexts during the tree evaluation, more homomorphic multiplications are required. Therefore, the run-time complexity of the server is higher. The additional total run-time required is equal to the run-time of $\sigma + k\gamma\left(\lceil \log_2 \gamma \rceil + 1\right)$ homomorphic multiplications that each take at least 10 hours. Clearly, much more computational time is needed by the server for the protocols that makes use of key-switching. Depending on the preference, a decision can be made be-

tween the two protocols by determining if either a low number of communication rounds or a low computational overhead for the server is more important.

We now describe the computational complexity of the decision-tree evaluation in all STTP protocols, which is executed by the server. It is clear that the decision-node evaluations and the key-switching procedures are the most expensive. Namely, this is the only part of the whole evaluation where homomorphic multiplications need to take place. Neglecting the run-time of all other operations that are not homomorphic multiplications, the total evaluation requires σ decision-node evaluations, that each take in the worst-case $\frac{1}{2}\delta(\delta-1)$ multiplications. The protocol that uses key-switching requires in addition $\sigma + k\gamma\left(\lceil\log_2\gamma\rceil+1\right)$ homomorphic multiplications. Depending on the input bit length δ and the depth of the tree D, the run-time of one homomorphic multiplication varies between 10 hours and 20 hours.

Looking at the structure of a decision tree, it can be noticed that the decision-node evaluations can be done in parallel. In addition, the key-switches before and after the decision-node evaluations can be done in parallel. This means that the run-time can be optimised when having access to parallel processors. For the protocols without key-switching, the worst-case run-time is then defined by one decision-node evaluation. For the key-switching protocols, two homomorphic multiplications have to be added. Assuming an input bit length of $\delta = 4$, one decision-node evaluation requires in the worst case 6 homomorphic multiplications. This means that on a parallel machine that acts as our server with processors of the same computational power as ours, the protocols without key-switching take in the worst-case between 60 and 120 hours. The protocols that do use key-switching take in the worst-case between 80 and 160 hours. For an input bit length of 6 this is between 150 and 300 hours and between 170 and 340 hours, respectively.

Clearly, the best achievable run-time of the evaluation is in the order of days. Such a run-time is currently considered impractical for real-life application within access-control, since these systems should be able to make access decisions quickly. Still, we discuss how our protocols can be applied within access-control. Optimally, an access-control server does not learn anything about the input except for a bit-value that denotes the access decision. Our current approach evaluates the access request towards a risk level denoting the risk of granting access. This is then compared to an acceptable risk value based on the current context and the operational benefit of granting access. We wish to avoid that the access-control system gains insight into the access risk level. This can be achieved by letting the server privately evaluate two decision trees; one that gives the access risk and one that gives the acceptable risk. These risk values can homomorphically be compared such that the result is an encryption of 1 in case the access risk is lower than the acceptable risk. In two communication rounds with the STTP, the server can then receive this result, without the STTP gaining any knowledge.

Within an access-control system, there is a chance that some input variables are missing or not delivered on time. Since the server knows the decision tree, this can be solved by choosing the child node of the associated decision node that results in the highest risk value. Next to this, sometimes more complex evaluations of the inputs than comparison or equality tests are required. An example is that the sum of two input variables has to meet a certain condition. Since we make use of homomorphic encryption, this is easily solvable. The sum (or any other function regarding more than one input variable) can be homomorphically evaluated. However, this makes the decision-tree evaluation more costly, since the additional homomorphic operations that are required cause some additional overhead.

8.3. Conclusions

This thesis takes a step towards feasible solutions for collaboratively evaluating a decision tree in a privacy-preserving way. Our work is the first work to propose private decision-tree evaluation where the input originates from more than one user. Therefore, our protocols introduce a new line of research within private decision-tree evaluation. We focused on solutions that make use of homomorphic encryption. First, we gave ten different protocols that take place in a different setting; either the server that holds the decision tree receives the evaluation result or the users that send the input receive the evaluation result. The protocols make use Multi-Key FHE or normal FHE with an STTP. Additionally, we introduced a novel key-switching method within two of the STTP protocols such that the dependency on the STTP is greatly reduced.

All protocols are proven to be secure in the semi-honest model. Next to this, we did an elaborate correctness analysis of the FHE schemes and the noise propagation during all the operations in the protocols, which is essential for a correct implementation. We introduced some adaptations of the FHE schemes to make the schemes more intuitive and our protocols more efficient and provided several new operations or procedures that are essential for our protocols.

We gave a first implementation of the protocols in order to do an overall comparison of all protocols in terms of run-time complexity and communications costs. Due to the high computational overhead of the key generation and the extension method for the Multi-Key FHE schemes, the protocols that make use of these schemes are not yet feasible. Therefore, the protocols that use an STTP are the most promising. These protocols have a very low computational overhead for the users; per input bit, they need to execute a procedure that does not take more than 5 seconds. The protocols take 3 rounds, but requires an additional key-exchange round. The communication sizes of the first two rounds dominate the other rounds and vary between $376 - 606$ MB (size of one ciphertext) per input bit value. Assuming that the implementation can be parallelized, the protocols without key-switching take in the worst-case between 60 and 120 hours for an input bit length of 4. For the use-case where the decision tree is an external resource and the users receive the evaluation result, the key-switching method can be applied. This reduces the number of required communication rounds: only 2 rounds are needed every time a tree is evaluated. Only for initialisation, two rounds instead of one round is needed for the exchange of keys. The STTP only participates in the protocol for key generation. This reduces the amount of trust that needs to be put into the STTP. This method does have the drawback that the run-time of the total evaluation for the server is higher. Again assuming that the implementation can be parallelized, these protocols take in the worst-case between 80 and 160 hours for an input bit length of 4. The exact run-time depends on the depth of the tree.

Clearly, the best achievable run-time of the evaluation is in the order of days. Such a run-time is currently considered impractical for real-life application within risk-adaptive access-control systems, since these systems should be able to make access decisions quickly. In the other setting, when the decision tree is used as an external resource, the requirements on the total evaluation time are lower. Therefore, the application of our two proposed protocols in this setting is only feasible in case the server has enough computational power and possible optimisations regarding the implementation are applied. As a final remark, the research within (Multi-Key) FHE schemes and their efficiency is progressing. Since the efficiency of our protocols directly depends on the efficiency of these underlying schemes, it is expected that our protocols will become more efficient in the future.

8.4. Future work

Whilst doing this research, we gained some insights into different ways of improving or continuing our research. Below, we propose several interesting research directions for future work.

Size of the plaintext space First of all, for the additive path evaluation, we used the extended plaintext space $\{0,1,\ldots,D'\}$ where $D'+1$ is a prime number that is at least bigger than the depth D of the tree. One of the parameters that determines the size of the ciphertext is $\ell = \log_{D'+1} q$. Clearly, increasing D' results in a lower ℓ. Therefore, it is expected that by taking D' higher than required by the tree and q the same, our protocols will be more efficient, due to the decrease in ℓ. But, a higher D' also impacts the noise propagation and therefore increases the value of q and n. The expectation is that there is a certain optimal value for D' where the lower ℓ value does not compensate the increase in n anymore. Finding this optimal value of D' will give the most efficient protocol. A higher value of D' also has the advantage that the class labels can be encrypted in one ciphertext. For future research it would be interesting to look into this dependency.

Semi-honest model Our protocols take place in the semi-honest model in which it is assumed that all parties follow the protocol correctly. But, users can send wrong ciphertexts that are no encryptions of their inputs or execute the partial decryption incorrectly. Also the STTP can communicate wrong partial decryptions. An interesting research direction would be to find out how our our protocols can be extended such that they are secure in the malicious model.

Other main techniques Our research focused on using homomorphic encryption. However, other techniques, such as secret sharing, can be used for privately evaluating decision trees. Although the number of communication rounds will increase, the computational complexity for the server and the communication sizes will possibly decrease. Therefore, it is interesting to look at potential solutions that make use of a different technique than homomorphic encryption and make it possible to privately evaluate a decision tree in a collaborative manner.

Decision-node evaluation In our protocols we make use of a comparison protocol and an equality test for the decision-node evaluations that do not require any additional communication rounds but have a high run-time complexity. Therefore, the evaluation time is almost completely determined by the costly decision-node evaluations. There are many more private comparison protocols and equality tests that differ in terms of complexity and communication [34, 36, 51, 64, 82, 108]. Depending on the requirements of the use-case, research can be done into the effect of using a different way of performing the decision-node evaluation and the impact that this will have on the complexity and communication costs.

Different encryption schemes In [8], a threshold FHE scheme is introduced that was also briefly mentioned in Chapter 2. We decided to use the Multi-Key FHE scheme since this requires one communication round less in comparison to this threshold FHE scheme. Next to this, in [57, 84, 109] threshold additively-homomorphic encryption schemes are given. Using an additive encryption scheme requires us to change the decision-node evaluation method since this currently relies on homomorphic multiplications, which will incur more communication rounds. A direction for future work would be to investigate how these two schemes could be used for collaborative private decision-tree protocols.

Hardware and software optimisations The processor we used for running our code is 2.4 GHz and has a memory of 16 GB. We expect a significant performance upgrade by using a more advanced server with a better CPU and memory. Additionally, our code is not optimised. For example, we use an iterative algorithm for the matrix multiplication that does not do parallel calculations. Next to this, we did not implement any of the tree evaluation in parallel. It would be interesting to find out how much the performance of the server can be improved by optimising the code, changing the available computational power and using a parallelized implementation.

Novel FHE scheme Recently, as an improvement of the GSW FHE scheme, Chen et al. [29] propose a new scheme called TFHE which is a fully homomorphic encryption scheme over a mathematical Torus. They propose a novel way of doing homomorphic operations. This scheme is extended to a multi-key scheme in [24]. This extension evaluates a binary gate on multi-key ciphertexts followed by a bootstrapping procedure after every homomorphic operation. We decided to focus on the GSW FHE scheme and its multi-key variant, since this scheme is more intuitive to implement and does not require this expensive bootstrapping procedure. It would be interesting to investigate how the complexity of these new schemes compares to our schemes and what the impact will be of using these schemes in our protocols.

Single-hop FHE scheme Our used Multi-Key FHE scheme is multi-hop for keys, which means that additional keys can be added during the protocol, even when some homomorphic operations are done already. This has the advantage that the evaluation of a certain path can be done directly after all corresponding parties have communicated their input and public keys. There also exists a single-hop Multi-Key FHE scheme [80]. This scheme requires the ciphertexts to be extended to all users in the same procedure. Research can be done into how this scheme compares in terms of computational complexity and if this outweighs the reduction in flexibility of the system.

RAdAC using decision trees We are the first work that propose to use a decision tree as the RAdAC system. It would be interesting to investigate the feasibility of translating an access policy into a decision tree. Next to this, our approach can be compared with existing RAdAC systems in terms of model complexity, flexibility and ability to cope with complex and dynamic contexts. Regarding the privacy-preserving techniques within access-control, research can be done into how the other existing solutions compare to our solution. For example, anonymous credentials can be used to evaluate every decision node. Next to this, attributes can be disclosed based on the attribute sensitivity and the trust level of the access-control party. These solutions are easier to implement but also have their downsides; anonymous credentials still reveal the comparison result of each decision node and disclosing less attributes reduces the amount of input information given to the access-control system.

Acronyms

ABAC Attribute Based Access Control. 7

BDD Bounded Distance Decoding. 18, 19, 133

CRS Common Random String. 11, 51

EHC Emergency Health Care. 1

EHRs Electronic Health Records. 1, 111

FHE Fully Homomorphic Encryption. v, 5, 11–13, 20–22, 31–37, 40, 42, 43, 46–48, 50–63, 65–67, 74, 76, 78, 79, 81, 85, 86, 92, 102, 112, 115–117

GapSVP$_\gamma$ γ-Gap Shortest Vector Problem. 18, 19, 133

GP General Practitioner. 1, 3, 4

GSW Gentry, Sahai and Waters [54]. vii, viii, 11, 12, 21, 22, 31–34, 42, 46, 47, 50, 59–66, 74, 76, 117

IND-CPA Indistinguishable under Chosen Plaintext Attack. 16, 20, 21, 47, 48, 52, 77

LWE Learning with Errors. 11, 18, 19, 23, 34, 40, 47, 73, 83

MPC Multi-Party Computation. 7, 10–12, 51

PDTE Private Decision-Tree Evaluation. 4–7, 12, 13, 51

RAdAC Risk-Adaptive Access Control. 2–4, 6–8, 111, 117

RBAC Role Based Access Control. 7

STTP Semi-Trusted Third Party. v, ix, 5, 6, 50, 52, 56, 59–66, 74–77, 80, 92, 96, 98–100, 102, 105, 108, 112–116, 147

List of Symbols

$\mathcal{T}$	A decision tree defined by $\mathcal{T} = (\{\mathcal{D} \cup \mathcal{L}\}, E)$ where the set of vertices is split into a set of decision nodes $\mathcal{D}$ and leaf nodes $\mathcal{L}$.
$\mathcal{D}$	The set of decision nodes in a decision tree $\mathcal{T}$.
$\mathcal{L}$	The set of leaf nodes in a decision tree $\mathcal{T}$.
$\mathbb{Z}_q$	The set of integers modulo q.
$\mathcal{N}$	The normal (Gaussian) distribution.
$\mathcal{D}_{S,\mu,\sigma^2}$	The discrete normal probability distribution with mean μ and variance σ^2.
$\mathcal{D}_{S,\sigma^2}$	The discrete normal probability distribution with zero mean and variance σ^2.
$\mathbf{g}$	The gadget column vector.
$\mathbf{I}_n \otimes \mathbf{g}^{-1}$	The function from a matrix $\mathbf{M} \in \mathbb{Z}_q^{n \times m'}$ for any n, m' to a matrix in $\{0,1\}^{nl \times m'}$ with $l = \lceil \log_2 q \rceil$ or $l = \lceil \log_{D'+1} q \rceil$ such that $(\mathbf{I}_n \otimes \mathbf{g}^\intercal) \cdot (\mathbf{I}_n \otimes \mathbf{g}^{-1})[\mathbf{M}] = \mathbf{M}$.
$\mathrm{LWE}_{n,m,q,\chi}$	Learning with Errors problem with dimension n, m, modulus q and error distribution χ.
$\bar{\Psi}_q^\alpha$	The distribution on $\mathbb{Z}_q$ gained by sampling form a normal variable with mean 0 and standard deviation $\alpha/\sqrt{2\pi}$ and reducing modulo q.

Bibliography

[1] Mark Abspoel, Serge Fehr, Gabriele Spini, Thomas Attema, Jan de Gier, Thijs Veugen, Ronald Cramer, Maran van Heesch, and Daniël Worm. Secure Multi-Party Computation Survey - Theoretical and Practical Considerations. Technical report, TNO/CWI, 2018.

[2] Ruqayah R. Al-Dahhan, Qi Shi, Gyu Myoung Lee, and Kashif Kifayat. Survey on revocation in ciphertext-policy attribute-based encryption. *Sensors (Switzerland)*, 19(7):1–22, 2019. ISSN 14248220. doi: 10.3390/s19071695.

[3] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of Learning with Errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015. doi: 10.1515/jmc-2015-0016.

[4] Jacob Alperin-Sheriff and Chris Peikert. Faster Bootstrapping with Polynomial Error. In Juan Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014*, pages 297–314. Springer Berlin Heidelberg, 2014. ISBN 978-3-662-44371-2. doi: 10.1007/978-3-662-44371-2_17.

[5] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009*, pages 595–618, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-03356-8.

[6] Claudio A. Ardagna, Jan Camenisch, Markulf Kohlweiss, Ronald Leenes, Gregory Neven, Bart Priem, Pierangela Samarati, Dieter Sommer, and Mario Verdicchio. Exploiting cryptography for privacy-enhanced access control: A result of the PRIME Project. *Journal of Computer Security*, 18(1):123–160, 2010. ISSN 0926227X. doi: 10.3233/JCS-2010-0367.

[7] Claudio A. Ardagna, Sabrina De Capitani Di Vimercati, Gregory Neven, Stefano Paraboschi, Franz Stefan Preiss, Pierangela Samarati, and Mario Verdicchio. Enabling privacy-preserving credential-based access control with XACML and SAML. In *Proceedings - 10th IEEE International Conference on Computer and Information Technology, CIT-2010, 7th IEEE International Conference on Embedded Software and Systems, ICESS-2010, ScalCom-2010*, pages 1090–1095. IEEE, 2010. ISBN 9780769541082. doi: 10.1109/CIT.2010.199.

[8] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty Computation with Low Communication, Computation and Interaction via Threshold FHE. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012*, pages 483–501. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-29011-4. doi: 10.1007/978-3-642-29011-4_29.

[9] M Barni, P Failla, R Lazzeretti, A Sadeghi, and T Schneider. Privacy-Preserving ECG Classification With Branching Programs and Neural Networks. *IEEE Transactions on Information Forensics and Security*, 6(2):452–468, jun 2011. ISSN 1556-6021. doi: 10.1109/TIFS.2011.2108650.

[10] Sana Belguith, Nesrine Kaaniche, and Mohammad Hammoudeh. Analysis of attribute-based cryptographic techniques and their application to protect cloud services. *Transactions on Emerging Telecommunications Technologies*, May:1–19, 2019. ISSN 21613915. doi: 10.1002/ett.3667.

[11] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-Policy Attribute-Based Encryption. In *2007 IEEE Symposium on Security and Privacy (SP '07)*, pages 321–334, 2007. doi: 10.1109/SP.2007.11.

[12] Albert Bifet, Jiajin Zhang, Wei Fan, Cheng He, Jianfeng Zhang, Jianfeng Qian, Geoff Holmes, and Bernhard Pfahringer. Extremely Fast Decision Tree Mining for Evolving Data Streams. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 1733–1742, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450348874. doi: 10.1145/3097983.3098139.

[13] Raphael Bost, Raluca Ada Popa, Stephen Tu, and Shafi Goldwasser. Machine Learning Classification over Encrypted Data. Cryptology ePrint Archive, Report 2014/331, 2014.

[14] Zvika Brakerski. Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology - CRYPTO 2012*, pages 868–886. Springer Berlin Heidelberg, 2012. doi: 10.1007/978-3-642-32009-5_50.

[15] Zvika Brakerski and Renen Perlman. Lattice-Based Fully Dynamic Multi-key FHE with Short Ciphertexts. In Matthew Obshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016*, pages 190–213. Springer Berlin Heidelberg, 2016. ISBN 978-3-662-53018-4. doi: 10.1007/978-3-662-53018-4.

[16] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) Fully Homomorphic Encryption without Bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (ITCS '12)*, pages 309–325. Association for Computing Machinery, 2012. doi: 10.1145/2090236.2090262.

[17] S.A. Brands. *Rethinking public key infrastructures and digital certificates building in privacy*. PhD thesis, Technische Universiteit Eindhoven, 1999.

[18] Justin Brickell, Donald E. Porter, Vitaly Shmatikov, and Emmett Witchel. Privacy-Preserving Remote Diagnostics. In *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS '07)*, pages 498–507, Alexandria, Virginia, USA, 2007. Association for Computing Machinery. ISBN 9781595937032. doi: 10.1145/1315245.1315307.

[19] David W. Britton and Ian A. Brown. Security Risk Measurement for the RAdAC Model. Master's thesis, Naval Postgraduate School, 3 2007.

[20] Jan Camenisch and Anna Lysyanskaya. An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In Birgit Pfitzmann, editor, *Advances in Cryptology - EUROCRYPT 2001*, pages 93–118, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. ISBN 978-3-540-44987-4.

[21] Jan Camenisch, Sebastian Mödersheim, Gregory Neven, Franz Stefan Preiss, and Dieter Sommer. A card requirements language enabling privacy-preserving access control. In *Proceedings of ACM Symposium on Access Control Models and Technologies, SACMAT*, pages 119–128, 2010. ISBN 9781450300490. doi: 10.1145/1809842.1809863.

[22] Sanjit Chatterjee, Neal Koblitz, Alfred Menezes, and Palash Sarkar. Another look at tightness II: Practical issues in cryptography. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10311 LNCS(August):21–55, 2017. ISSN 16113349. doi: 10.1007/978-3-319-61273-7_3.

[23] David Chaum. Security Without Identification: Transaction Systems To Make Big Brother Obsolete. *Communications of the ACM*, 28(10), 1985.

[24] Hao Chen, Ilaria Chillotti, and Yongsoo Song. Multi-Key Homomorphic Encryption from TFHE. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019*, pages 446–472. Springer International Publishing, 2019. ISBN 978-3-030-34621-8. doi: 10.1007/978-3-030-34621-8_16.

[25] Hao Chen, Miran Kim, Wei Dai, and Yongsoo Song. Efficient Multi-Key Homomorphic Encryption with Packed Ciphertexts with Application to Oblivious Neural Network Inference. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS '19)*, pages 395–412. Association for Computing Machinery, 2019. ISBN 9781450367479. doi: 10.1145/3319535.3363207. URL `https://dl.acm.org/doi/10.1145/3319535.3363207`.

[26] Long Chen, Zhenfeng Zhang, and Xueqing Wang. Batched Multi-hop Multi-key FHE from Ring-LWE with Compact Ciphertext Extension. In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography*, pages 597–627. Springer International Publishing, 2017. ISBN 978-3-319-70503-3. doi: 10.1007/978-3-319-70503-3_20.

[27] Pau Chen Cheng, Pankaj Rohatgi, Claudia Keser, Paul A. Karger, Grant M. Wagner, and Angela Schuett Reninger. Fuzzy Multi-Level Security: An experiment on quantified risk-adaptive access control. *Proceedings - IEEE Symposium on Security and Privacy*, pages 222–227, 2007. ISSN 10816011. doi: 10.1109/SP.2007.21.

[28] Jung Hee Cheon, Miran Kim, and Myungsun Kim. Optimized Search-and-Compute Circuits and Their Application to Query Evaluation on Encrypted Data. *IEEE Transactions on Information Forensics and Security*, 11(1):188–199, 2016. ISSN 15566013. doi: 10.1109/TIFS.2015.2483486.

[29] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: Fast Fully Homomorphic Encryption Over the Torus. *Journal of Cryptology*, 33(1):34–91, 2020. ISSN 14321378. doi: 10.1007/s00145-019-09319-x.

[30] Donghee Choi, Dohoon Kim, and Seog Park. A Framework for Context Sensitive Risk-Based Access Control in Medical Information Systems. *Computational and Mathematical Methods in Medicine*, 2015, 2015. ISSN 17486718. doi: 10.1155/2015/265132.

[31] Michael Clear and Ciarán McGoldrick. Multi-identity and Multi-key Leveled FHE from Learning with Errors. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015*, pages 630–656, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. ISBN 978-3-662-48000-7. doi: 10.1007/978-3-662-48000-7_31.

[32] Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. *Integer Programming*. Springer International Publishing, Cham, 2014. ISBN 978-3-319-11007-3. doi: 10.1007/978-3-319-11008-0.

[33] William J. Cook, William H. Cunningham, William R. Pulleyblank, and Alexander Schrijver. *Combinatorial Optimization*. John Wiley & Sons, Inc, USA, 1998. ISBN 047155894X.

[34] Geoffroy Couteau. New Protocols for Secure Equality Test and Comparison. In Bart Preneel and Frederik Vercauteren, editors, *Applied Cryptography and Network Security*, pages 303–320, Cham, 2018. Springer International Publishing. ISBN 978-3-319-93387-0.

[35] Ronald Cramer and Ivan Damgård. *Multiparty Computation, an Introduction*, pages 41–87. Birkhäuser Basel, Basel, 2005. ISBN 978-3-7643-7394-8. doi: 10.1007/3-7643-7394-6_2.

[36] Ivan Damgård, Martin Geisler, and Mikkel Krøigård. Homomorphic encryption and secure comparison. *International Journal of Applied Cryptography*, 1(1):22–31, 2008. ISSN 17530571. doi: 10.1504/IJACT.2008.017048.

[37] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. Multiparty Computation from Somewhat Homomorphic Encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology - CRYPTO 2012*, pages 643–662. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-32009-5.

[38] Ivan Damgård, Daniel Escudero, Tore Frederiksen, Marcel Keller, Peter Scholl, and Nikolaj Volgushev. New Primitives for Actively-Secure MPC over Rings with Applications to Private Machine Learning. In *IEEE Symposium on Security and Privacy (SP '19)*, pages 1102–1120, may 2019. doi: 10.1109/SP.2019.00078.

[39] Ernesto Damiani, Sabrina De Capitani Di Vimercati, and Pierangela Samarati. New paradigms for access control in open environments. In *Proceedings of the Fifth IEEE International Symposium on Signal Processing and Information Technology*, pages 540–545. IEEE, 2005. ISBN 0780393147. doi: 10.1109/ISSPIT.2005.1577155.

[40] Martine De Cock, Rafael Dowsley, Caleb Horst, Raj Katti, Anderson C.A. Nascimento, Wing Sea Poon, and Stacey Truex. Efficient and Private Scoring of Decision Trees, Support Vector Machines and Logistic Regression Models Based on Pre-Computation. *IEEE Transactions on Dependable and Secure Computing*, 16(2):217–230, 2019. ISSN 19410018. doi: 10.1109/TDSC.2017.2679189.

[41] Jay L. Devore and Kenneth N. Berk. *Modern Mathematical Statistics with Applications*. Springer International Publishing, New York, NY, USA, second edition, 2018. doi: 10.1007/978-1-4614-0391-3.

[42] N N Diep, L X Hung, Y Zhung, S Lee, Y Lee, and H Lee. Enforcing Access Control Using Risk Assessment. In *Fourth European Conference on Universal Multiservice Networks (ECUMN'07)*, pages 419–424, feb 2007. doi: 10.1109/ECUMN.2007.19.

[43] Daniel Ricardo dos Santos, Roberto Marinho, Gustavo Roecker Schmitt, Carla Merkle Westphall, and Carlos Becker Westphall. A framework and risk assessment approaches for risk-based access control in the cloud. *Journal of Network and Computer Applications*, 74:86–97, 2016. ISSN 10958592. doi: 10.1016/j.jnca.2016.08.013.

[44] Wenliang Du, Wenliang Du, Zhijun Zhan, and Zhijun Zhan. Building decision tree classifier on private data. In Chris Clifton and Vladimir Estivill-Castro, editors, *Proceedings of the IEEE international conference on Privacy, security and data mining-Volume 14*, pages 1–8, Maebashi City, Japan, 2002. ISBN 0-909-92592-5. doi: 10.5555/850782.850784.

[45] Dheeru Dua and Casey Graff. UCI Machine Learning Repository, 2017. URL `http://archive.ics.uci.edu/ml`.

[46] Ali Esmaeeli and Hamid Reza Shahriari. Privacy Protection of Grid Service Requesters through Distributed Attribute Based Access Control Model. In Paolo Bellavista, Ruay-Shiung Chang, Han-Chieh Chao, Shin-Feng Lin, and Peter M. A. Sloot, editors, *Advances in Grid and Pervasive Computing*, pages 573–582. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-13067-0.

[47] Kai Fan, Huiyue Xu, Longxiang Gao, Hui Li, and Yintang Yang. Efficient and privacy preserving access control scheme for fog-enabled IoT. *Future Generation Computer Systems*, 99:134–142, 2019. ISSN 0167739X. doi: 10.1016/j.future.2019.04.003.

[48] Bassam Farroha and Deborah Farroha. Challenges of 'operationalizing' dynamic system access control: Transitioning from ABAC to RAdAC. In *SysCon 2012 - 2012 IEEE International Systems Conference, Proceedings*, pages 1–7. IEEE, 2012. ISBN 9781467307499. doi: 10.1109/SysCon.2012.6189525.

[49] Marc Fischlin. A cost-effective pay-per-multiplication comparison method for millionaires. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2020:457–471, 2001. ISSN 16113349. doi: 10.1007/3-540-45353-9_33.

[50] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model Inversion Attacks That Exploit Confidence Information and Basic Countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS '15)*, pages 1322–1333, Denver, Colorado, USA, 2015. Association for Computing Machinery. ISBN 9781450338325. doi: 10.1145/2810103.2813677.

[51] Juan Garay, Berry Schoenmakers, and José Villegas. Practical and Secure Solutions for Integer Comparison. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *Public Key Cryptography - PKC 2007*, pages 330–342. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-71677-8. doi: 10.1007/978-3-540-71677-8_22.

[52] Luca Gasparini. Risk-Aware Access Control and XACML. Technical report, Università degli Studi di Padova, 2013.

[53] Craig Gentry. Fully Homomorphic Encryption Using Ideal Lattices. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing (STOC '09)*, pages 169–178, Bethesda, MD, USA, 2009. Association for Computing Machinery. ISBN 9781605585062. doi: 10.1145/1536414.1536440.

[54] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013*, pages 75–92. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-40041-4. doi: 10.1007/978-3-642-40041-4_5.

[55] Daniel Gibert, Carles Mateu, and Jordi Planes. The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. *Journal of Network and Computer Applications*, 153(November 2019):102526, 2020. ISSN 10958592. doi: 10.1016/j.jnca.2019.102526.

[56] Hossein Hassani, Xu Huang, Emmanuel S Silva, and Mansi Ghodsi. A Review of Data Mining Applications in Crime. *Statistical Analysis and Data Mining*, 9(3):139–154, jun 2016. ISSN 1932-1864. doi: 10.1002/sam.11312.

[57] Carmit Hazay, Gert Læssøe Mikkelsen, Tal Rabin, Tomas Toft, and Angelo Agatino Nicolosi. Efficient RSA Key Generation and Threshold Paillier in the Two-Party Setting. *Journal of Cryptology*, 32(2):265–323, 2019. ISSN 14321378. doi: 10.1007/s00145-017-9275-7.

[58] Vincent C. Hu, David F. Ferraiolo, Rick Kuhn, Adam Schnitzer, Kenneth Sandlin, Robert Miller, and Karen Scarfone. Guide to Attribute Based Access Control (ABAC) Definition and Considerations. Technical report, National Institute of Standards and Technology, 2014.

[59] Marc Joye and Fariborz Salehi. Private yet Efficient Decision Tree Evaluation. In Florian Kerschbaum and Stefano Paraboschi, editors, *Data and Applications Security and Privacy XXXII*, pages 243–259, Cham, 2018. Springer International Publishing. ISBN 978-3-319-95729-6. doi: 10.1007/978-3-319-95729-6_16.

[60] Savith Kandala, Ravi Sandhu, and Venkata Bhamidipati. An Attribute Based Framework for Risk-Adaptive Access Control Models. In *2011 Sixth International Conference on Availability, Reliability and Security*, pages 236–241. IEEE, 2011. ISBN 9780769544854. doi: 10.1109/ARES.2011.41.

[61] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. CRC Press, Taylor and Francis Group, 2015. ISBN 978-1-4665-7026-9.

[62] Hemanth Khambhammettu, Sofiene Boulares, Kamel Adi, and Luigi Logrippo. A framework for risk assessment in access control systems. *Computers and Security*, 39(PARTA):86–103, 2013. ISSN 01674048. doi: 10.1016/j.cose.2013.03.010.

[63] Markulf Kohlweiss and Alfredo Rial. Optimally private access control. *Proceedings of the ACM Conference on Computer and Communications Security*, pages 37–48, 2013. ISSN 15437221. doi: 10.1145/2517840.2517857.

[64] Vladimir Kolesnikov, Ahmad-Reza Sadeghi, and Thomas Schneider. Improved Garbled Circuit Building Blocks and Applications to Auctions and Computing Minima. In Juan A Garay, Atsuko Miyaji, and Akira Otsuka, editors, *Cryptology and Network Security*, pages 1–20, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-10433-6.

[65] Vladimir Kolesnikov, Ahmad Reza Sadeghi, and Thomas Schneider. Improved garbled circuit building blocks and applications to auctions and computing minima. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5888 LNCS:1–20, 2009. ISSN 03029743. doi: 10.1007/978-3-642-10433-6_1.

[66] Jan Kolter, Rolf Schillinger, and Günther Pernul. A privacy-enhanced attribute-based access control system. In *Proceedings of the 21st Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, pages 129–143. Springer-Verlag, 2007. ISBN 9783540735335. doi: 10.1007/978-3-540-73538-0_11.

[67] Junzuo Lai, Robert H. Deng, and Yingjiu Li. Expressive CP-ABE with partially hidden access structures. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security (ASIACCS '12)*, pages 18–19. Association for Computing Machinery, 2012. ISBN 9781450313032. doi: 10.1145/2414456.2414465.

[68] Cheng Chi Lee, Pei Shan Chung, and Min Shiang Hwang. A survey on attribute-based encryption schemes of access control in cloud environments. *International Journal of Network Security*, 15(4):231–240, 2013. ISSN 1816353X. doi: 10.6633/IJNS.201307.15(4).01.

[69] Don Stephen Lemons. *An Introduction to Stochasitc Processes in Physics*, volume 7. The Johns Hopkins University Press, Baltimore, Maryland, 2002. ISBN 0801868661.

[70] Li Lin, Ting Ting Liu, Shuang Li, Chathura M. Sarathchandra Magurawalage, and Shan Shan Tu. PriGuarder: A Privacy-Aware Access Control Approach Based on Attribute Fuzzy Grouping in Cloud Environments. *IEEE Access*, 6:1882–1893, 2018. ISSN 21693536. doi: 10.1109/ACCESS.2017.2780763.

[71] Yehuda Lindell. How To Simulate It – A Tutorial on the Simulation Proof Technique. In *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*, pages 277–346. Springer International Publishing, Cham, 2017. ISBN 978-3-319-57048-8. doi: 10.1007/978-3-319-57048-8_6.

[72] Yehuda Lindell and Benny Pinkas. Privacy Preserving Data Mining. In Mihir Bellare, editor, *Advances in Cryptology - CRYPTO 2000*, pages 36–54, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. ISBN 978-3-540-44598-2. doi: 10.1007/3-540-44598-6_3.

[73] Qingkai Ma and Ping Deng. Secure Multi-party Protocols for Privacy Preserving Data Mining. In Yingshu Li, Dung T Huynh, Sajal K Das, and Ding-Zhu Du, editors, *Wireless Algorithms, Systems, and Applications*, pages 526–537, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-88582-5. doi: 10.1007/978-3-540-88582-5_49.

[74] John Maddock and Christopher Kormanyos. Boost Multiprecision Library. URL `https://www.boost.org/doc/libs/1_73_0/libs/multiprecision/doc/html/index.html`.

[75] Robert W. McGraw. Risk-Adaptable Access Control (RAdAC). Technical report, Information Assurance Architecture and Systems Security Engineering Group, National Security Agency, 2009.

[76] Daniele Micciancio and Petros Mol. Pseudorandom Knapsacks and the Sample Complexity of LWE Search-to-Decision Reductions. In Philip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011*, pages 465–484. Springer Berlin Heidelberg, 2011. doi: 10.1007/978-3-642-22792-9_26.

[77] Daniele Micciancio and Chris Peikert. Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012*, pages 700–718, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-29011-4.

[78] Ministerie van Volksgezondheid, Sport en Welzijn. Belangrijkste huisartsinformatie tijdelijk te raadplegen, apr 2020. URL `https://www.rijksoverheid.nl/ministeries/ministerie-van-volksgezondheid-welzijn-en-sport/nieuws/2020/04/08/belangrijkste-huisartsinformatie-tijdelijk-te-raadplegen`.

[79] Ian Molloy, Luke Dickens, Charles Morisset, Pau-Chen Cheng, Jorge Lobo, and Alessandra Russo. Risk-Based Security Decisions under Uncertainty. In *Proceedings of the Second ACM Conference on Data and Application Security and Privacy (CODASPY '12)*, pages 157–168, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450310918. doi: 10.1145/2133601.2133622.

[80] Pratyay Mukherjee and Daniel Wichs. Two Round Multiparty Computation via Multi-key FHE. In Marc Fischlin and Jean Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016*, pages 735–763. Springer Berlin Heidelberg, 2016. ISBN 978-3-662-49896-5. doi: 10.1007/978-3-662-49896-5.

[81] Madhura Mulimani and Rashmi Rachh. Analysis of Access Control Methods in Cloud Computing. *International Journal of Education and Management Engineering*, 7(3):15–24, 2017. ISSN 23053623. doi: 10.5815/ijeme.2017.03.02.

[82] Majid Nateghizad, Thijs Veugen, Zekeriya Erkin, and Reginald L. Lagendijk. Secure equality testing protocols in the two-party setting. In *Proceedings of the 13th International Conference on Availability, Reliability and Security (ARES 2018)*, pages 1–10. Association for Computing Machinery, 2018. ISBN 9781450364485. doi: 10.1145/3230833.3230866.

[83] Qun Ni, Elisa Bertino, and Jorge Lobo. Risk-Based Access Control Systems Built on Fuzzy Inferences. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security (ASIACCS '10)*, pages 250–260, Beijing, China, 2010. Association for Computing Machinery. ISBN 9781605589367. doi: 10.1145/1755688.1755719.

[84] Takashi Nishide and Kouichi Sakurai. Distributed Paillier cryptosystem without trusted dealer. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6513 LNCS:44–60, 2011. ISSN 16113349. doi: 10.1007/978-3-642-17955-6_4.

[85] Takashi Nishide, Kazuki Yoneyama, and Kazuo Ohta. Attribute-Based encryption with partially hidden encryptor-specified access structures. In Steven M. Bellovin, Rosario Gennaro, Angelos Keromytis, and Moti Yung, editors, *Applied Cryptography and Network Security*, pages 111–129. Springer Berlin Heidelberg, 2008.

[86] Sang M. Park and Soon M. Chung. Privacy-preserving attribute-based access control for grid computing. *International Journal of Grid and Utility Computing*, 5(4):286–296, 2014. ISSN 17418488. doi: 10.1504/IJGUC.2014.065372.

[87] Chris Peikert. A decade of lattice cryptography. *Foundations and Trends in Theoretical Computer Science*, 10(4):283–424, 2016. ISSN 15513068. doi: 10.1561/0400000074.

[88] Chris Peikert and Sina Shiehian. Multi-key FHE from LWE, Revisited. In Martin Hirt and Adam Smith, editors, *Theory of Cryptography*, pages 217–238. Springer Berlin Heidelberg, 2016. ISBN 978-3-662-53644-5. doi: 10.1007/978-3-662-53644-5_9.

[89] Herma Pieterman. Geef huisarts spilfunctie in epd. *Medisch Contact 75*, pages 34 – 36, dec 2020.

[90] Rachel Player. *Parameter selection in lattice-based cryptography*. PhD thesis, University of London, 2017.

[91] Inese Polaka, Igor Tom, and Arkady Borisov. Decision Tree Classifiers in Bioinformatics. *Scientific Journal of Riga Technical University*, 42(1):118–123, 2011. ISSN 1407-7493. doi: 10.2478/v10143-010-0052-4.

[92] Ivens Portugal, Paulo Alencar, and Donald Cowan. The use of machine learning algorithms in recommender systems: A systematic review. *Expert Systems with Applications*, 97:205–227, 2018. ISSN 09574174. doi: 10.1016/j.eswa.2017.12.020.

[93] Andreas Put and Bart De Decker. Attribute-Based Privacy-Friendly Access Control with Context. *Communications in Computer and Information Science*, 764:291–315, 2017. ISSN 18650929. doi: 10.1007/978-3-319-67876-4_14.

[94] Sriram Ramgopal, Christopher M Horvat, Naveena Yanamala, and Elizabeth R Alpern. Machine Learning to Predict Serious Bacterial Infections in Young Febrile Infants. *Pediatrics*, 146(3), 2020. doi: 10.1542/peds.2019-4096.

[95] Oded Regev. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. *Journal of the Association for Computing Machinery*, 56(6):1–40, 2009. ISSN 00045411. doi: 10.1145/1568318.1568324.

[96] Oded Regev. The Learning with Errors Problem. Technical report, Blavatnik School of Computer Science, Tel Aviv, 2010. URL `https://cims.nyu.edu/~regev/papers/lwesurvey.pdf`.

[97] Alex Sangers, Maran van Heesch, Thomas Attema, Thijs Veugen, Mark Wiggerman, Jan Veldsink, Oscar Bloemen, and Daniël Worm. Secure Multiparty PageRank Algorithm for Collaborative Fraud Detection. In Ian Goldberg and Tyler Moore, editors, *Financial Cryptography and Data Security*, pages 605–623, Cham, 2019. Springer International Publishing. ISBN 978-3-030-32101-7.

[98] Igor Santos, Felix Brezo, Xabier Ugarte-Pedrero, and Pablo G. Bringas. Opcode sequences as representation of executables for data-mining-based unknown malware detection. *Information Sciences*, 231:64–82, 2013. ISSN 00200255. doi: 10.1016/j.ins.2011.08.020.

[99] Daniel Servos and Sylvia L. Osborn. Current research and open problems in attribute-based access control. *ACM Computing Surveys*, 49(4), 2017. ISSN 15577341. doi: 10.1145/3007204.

[100] Riaz Ahmed Shaikh, Kamel Adi, and Luigi Logrippo. Dynamic risk-based decision methods for access control systems. *Computers and Security*, 31(4):447–464, 2012. ISSN 01674048. doi: 10.1016/j.cose.2012.02.006.

[101] Adi Shamir. How to Share a Secret. *Communications of the ACM*, 22(11):612–613, 1979. ISSN 15577317. doi: 10.1145/359168.359176.

[102] Mina Sheikhalishahi, Gamze Tillem, Zekeriya Erkin, and Nicola Zannone. Privacy-Preserving Multi-Party Access Control. In *Proceedings of the 18th ACM Workshop on Privacy in the Electronic Society*, pages 1–13, London, United Kingdom, 2019. Association for Computing Machinery. ISBN 9781450368308. doi: 10.1145/3338498.3358643.

[103] Herman Stil. Topman Philips: privacyangst houdt uitwisseling coronagegevens tegen, sep 2020. URL `https://www.parool.nl/nederland/topman-philips-privacyangst-houdt-uitwisseling-coronagegevens-tegen~b96f69e6`.

[104] Raymond K. H. Tai, Jack P. K. Ma, Yongjun Zhao, and Sherman S. M. Chow. Privacy-Preserving Decision Trees Evaluation via Linear Functions. In Simon N Foley, Dieter Gollmann, and Einar Snekkenes, editors, *Computer Security (ESORICS '17)*, pages 494–512, Cham, 2017. Springer International Publishing. ISBN 978-3-319-66399-9. doi: 10.1007/978-3-319-66399-9_27.

[105] Jayashamani Tamibmaniam, Narwani Hussin, Wee Kooi Cheah, Kee Sing Ng, and Prema Muninathan. Proposal of a Clinical Decision Tree Algorithm Using Factors Associated with Severe Dengue Infection. *PLOS ONE*, 11(8):1–10, 2016. doi: 10.1371/journal.pone.0161696.

[106] Anselme Tueno, Florian Kerschbaum, and Stefan Katzenbeisser. Private Evaluation of Decision Trees using Sublinear Cost. *Proceedings on Privacy Enhancing Technologies*, 2019(1):266–286, 2019. doi: 10.2478/popets-2019-0015.

[107] Anselme Tueno, Yordan Boev, and Florian Kerschbaum. Non-interactive Private Decision Tree Evaluation. In Anoop Singhal and Jaideep Vaidya, editors, *Data and Applications Security and Privacy XXXIV*, pages 174–194, Cham, 2020. Springer International Publishing. ISBN 978-3-030-49669-2. doi: 10.1007/978-3-030-49669-2_10.

[108] Thijs Veugen. Improving the DGK comparison protocol. *WIFS 2012 - Proceedings of the 2012 IEEE International Workshop on Information Forensics and Security*, pages 49–54, 2012. doi: 10.1109/WIFS.2012.6412624.

[109] Thijs Veugen, Thomas Attema, and Gabriele Spini. An implementation of the Paillier crypto system with threshold decryption without a trusted dealer, 2019. URL `https://eprint.iacr.org/2019/1136`.

[110] Volgjezorg. Corona opt-in en de effecten op de gegevensuitwisseling in de zorg, may 2020. URL `https://www.volgjezorg.nl/nieuws/corona-opt-en-de-effecten-op-de-gegevensuitwisseling-de-zorg`.

[111] Joerg Walter, Mathias Koch, Gunter Winkler, and David Bellot. Basic Linear Algebra Library. URL `https://www.boost.org/doc/libs/1_65_1/libs/numeric/ublas/doc/index.html`.

[112] Guojun Wang, Qin Liu, Jie Wu, and Minyi Guo. Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers. *Computers and Security*, 30(5):320–331, 2011. ISSN 01674048. doi: 10.1016/j.cose.2011.05.006.

[113] Qihua Wang and Hongxia Jin. Quantified Risk-Adaptive Access Control for Patient Privacy Protection in Health Information Systems. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security (ASIACCS '11)*, pages 406–410, Hong Kong, China, 2011. Association for Computing Machinery. ISBN 9781450305648. doi: 10.1145/1966913.1966969.

[114] David J. Wu, Tony Feng, Michael Naehrig, and Kristin Lauter. Privately Evaluating Decision Trees and Random Forests. *Proceedings on Privacy Enhancing Technologies*, 2016(4):335–355, 2016. doi: 10.1515/popets-2016-0043.

[115] Yang Xu, Quanrun Zeng, Guojun Wang, Cheng Zhang, Ju Ren, and Yaoxue Zhang. A Privacy-Preserving Attribute-Based Access Control Scheme. In Guojun Wang, Jinjun Chen, and Laurence T Yang, editors, *Security, Privacy, and Anonymity in Computation, Communication, and Storage*, pages 361–370. Springer International Publishing, 2018. ISBN 978-3-030-05345-1.

[116] Yang Xu, Quanrun Zeng, Guojun Wang, Cheng Zhang, Ju Ren, and Yaoxue Zhang. An efficient privacy-enhanced attribute-based access control mechanism. *Concurrency Computation*, 32 (5):1–10, 2020. ISSN 15320634. doi: 10.1002/cpe.5556.

[117] Kan Yang, Qi Han, Hui Li, Kan Zheng, Zhou Su, and Xuemin Shen. An Efficient and Fine-Grained Big Data Access Control Scheme with Privacy-Preserving Policy. *IEEE Internet of Things Journal*, 4(2):563–571, 2017. ISSN 23274662. doi: 10.1109/JIOT.2016.2571718.

[118] A. C. Yao. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pages 162–167, 1986. ISBN 0818607408. doi: 10.1109/SFCS.1986.25.

[119] Guoping Zhang, Jing Liu, and Jianbo Liu. Protecting sensitive attributes in attribute based access control. In Aditya Ghose, Huibiao Zhu, Qi Yu, Alex Delis, Quang Z. Sheng, Olivier Perrin, Jianmin Wang, and Yan Wang, editors, *Service-Oriented Computing - ICSOC 2012 Workshops*, pages 294–305. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-37804-1. doi: 10.1007/978-3-642-37804-1_30.

[120] Yinghui Zhang, Dong Zheng, and Robert H. Deng. Security and Privacy in Smart Health: Efficient Policy-Hiding Attribute-Based Access Control. *IEEE Internet of Things Journal*, 5(3):2130–2145, 2018. ISSN 23274662. doi: 10.1109/JIOT.2018.2825289.

A

Appendix

A.1. Proof description of Theorem 3.17

We will briefly describe the idea of the proof of Theorem 3.17. It combines two important statements which Regev proves in his work, namely [95]:

- When having access to a Search $\text{LWE}_{n,m,q,\bar{\Psi}_\alpha}$ oracle, the BDD problem for a lattice Λ^* and distance $\alpha q/\sqrt{2}r$ for some r can be *clasisally* solved given a polynomial number of samples from a discrete Gaussian distribution $\mathscr{D}_{\Lambda,\sigma^2}$ with $\sigma^2 = \frac{r^2}{2\pi}$.
- For any $d > 0$, there is a efficient *quantum* reduction from sampling from $\mathscr{D}_{\Lambda,\sigma^2}$ with $\sigma^2 = \frac{n}{2\pi d^2}$ to solving the BDD problem on Λ^* to within distance d.

Here Λ^* is the *dual* of the lattice $\Lambda \in \mathbb{R}^m$ which is defined as $\{\mathbf{y} \in \mathbb{R}^m : \mathbf{x}\cdot\mathbf{y} \in \mathbb{Z}, \forall \mathbf{x} \in \Lambda\}$. The dual of a lattice Λ with basis $\mathbf{B}$ can be shown to be a lattice with basis $\mathbf{B}(\mathbf{B}^\intercal\mathbf{B})^{-1}$ [77].

The intuition behind the first statement is that, given the LWE oracle, the unknown closest lattice point can be found by solving for several LWE samples. These LWE samples, of which the secret vector is equal to this unknown vector, can be generated by the discrete Gaussian distribution given in the statement.

Assuming $\alpha q > 2\sqrt{n}$ and starting from the fact that for a sufficiently large r, sampling from $\mathscr{D}_{\Lambda,\sigma^2}$ with $\sigma^2 = \frac{r^2}{2\pi}$ can be done efficiently [96], a solution to the BDD problem can be obtained within distance $\sqrt{2n}/r$. Given the second statement, now we can create samples from $\mathscr{D}_{\Lambda,\sigma'^2}$ with $\sigma'^2 = \frac{(r/\sqrt{2})^2}{2\pi}$. Going back and forth between the two statements, each time we can sample from a Gaussian distribution over Λ with a lower variance. At some point, the variance is low enough such that the narrow samples can be used to solve the GapSVP_γ problem [95].

A.2. Parameters

A.2.1. Protocols 1 and 2

Table A.1: Parameter setting for Protocols 1 and 2 for different δ, D based on Table 6.7 and [3] for $k = 2$ and $m = n\lceil \log_2 q \rceil + 2\tau$ such that $\lambda, \tau = 110$.

	$D = 3$	$D = 7$	$D = 15$
$\delta = 4$	$q = 2^{60}, \alpha = 10^{-16}, n = 1350$	$q = 2^{63}, \alpha = 10^{-17}, n = 1450$	$q = 2^{70}, \alpha = 10^{-19}, n = 1600$
$\delta = 6$	$q = 2^{63}, \alpha = 10^{-17}, n = 1450$	$q = 2^{67}, \alpha = 1^{-18}, n = 1500$	$q = 2^{70}, \alpha = 10^{-19}, n = 1600$
$\delta = 8$	$q = 2^{63}, \alpha = 10^{-17}, n = 1450$	$q = 2^{67}, \alpha = 1^{-18}, n = 1500$	$q = 2^{73}, \alpha = 10^{-20}, n = 1700$

Table A.2: Parameter setting for Protocols 1 and 2 for different δ, D based on Table 6.7 and [3] for $k = 3$ and $m = n\lceil \log_2 q \rceil + 2\tau$ such that $\lambda, \tau = 110$.

	$D = 3$	$D = 7$	$D = 15$
$\delta = 4$	$q = 2^{70}, \alpha = 10^{-19}, n = 1600$	$q = 2^{77}, \alpha = 1^{-21}, n = 1800$	$q = 2^{83}, \alpha = 10^{-23}, n = 2000$
$\delta = 6$	$q = 2^{73}, \alpha = 10^{-20}, n = 1700$	$q = 2^{77}, \alpha = 1^{-21}, n = 1800$	$q = 2^{83}, \alpha = 10^{-23}, n = 2000$
$\delta = 8$	$q = 2^{73}, \alpha = 10^{-20}, n = 1700$	$q = 2^{80}, \alpha = 1^{-22}, n = 1900$	$q = 2^{83}, \alpha = 10^{-23}, n = 2000$

Table A.3: Parameter setting for Protocols 1 and 2 for different δ, D based on Table 6.7 and [3] for $k = 4$ and $m = n\lceil \log_2 q \rceil + 2\tau$ such that $\lambda, \tau = 110$.

	$D = 3$	$D = 7$	$D = 15$
$\delta = 4$	$q = 2^{83}, \alpha = 10^{-23}, n = 2000$	$q = 2^{87}, \alpha = 10^{-24}, n = 2050$	$q = 2^{93}, \alpha = 10^{-26}, n = 2250$
$\delta = 6$	$q = 2^{83}, \alpha = 10^{-23}, n = 2000$	$q = 2^{90}, \alpha = 10^{-25}, n = 2150$	$q = 2^{93}, \alpha = 10^{-26}, n = 2250$
$\delta = 8$	$q = 2^{83}, \alpha = 10^{-23}, n = 2000$	$q = 2^{90}, \alpha = 10^{-25}, n = 2150$	$q = 2^{93}, \alpha = 10^{-26}, n = 2250$

A.2.2. Protocols 3 and 4

Table A.4: Parameter setting for Protocols 3 and 4 for different δ, D based on Table 6.7 and [3] for $k = 2$ and $m = n\lceil \log_2 q \rceil + 2\tau$ such that $\lambda, \tau = 110$.

	$D = 3$	$D = 7$	$D = 15$
$\delta = 4$	$q = 5^{20}, \alpha = 10^{-12}, n = 1000$	$q = 11^{17}, \alpha = 10^{-15}, n = 1200$	$q = 17^{14}, \alpha = 10^{-15}, n = 1250$
$\delta = 6$	$q = 5^{22}, \alpha = 10^{-13}, n = 1050$	$q = 11^{17}, \alpha = 10^{-15}, n = 1200$	$q = 17^{15}, \alpha = 10^{-16}, n = 1300$
$\delta = 8$	$q = 5^{22}, \alpha = 10^{-13}, n = 1050$	$q = 11^{17}, \alpha = 10^{-15}, n = 1200$	$q = 17^{15}, \alpha = 10^{-16}, n = 1300$

Table A.5: Parameter setting for Protocols 3 and 4 for different δ, D based on Table 6.7 and [3] for $k = 3$ and $m = n\lceil \log_2 q \rceil + 2\tau$ such that $\lambda, \tau = 110$.

	$D = 3$	$D = 7$	$D = 15$
$\delta = 4$	$q = 5^{26}, \alpha = 10^{-16}, n = 1350$	$q = 11^{21}, \alpha = 10^{-19}, n = 1550$	$q = 17^{17}, \alpha = 10^{-19}, n = 1650$
$\delta = 6$	$q = 5^{26}, \alpha = 10^{-16}, n = 1350$	$q = 11^{21}, \alpha = 10^{-19}, n = 1550$	$q = 17^{17}, \alpha = 10^{-19}, n = 1650$
$\delta = 8$	$q = 5^{26}, \alpha = 10^{-16}, n = 1350$	$q = 11^{22}, \alpha = 10^{-20}, n = 1800$	$q = 17^{17}, \alpha = 10^{-19}, n = 1650$

Table A.6: Parameter setting for Protocols 3 and 4 for different δ, D based on Table 6.7 and [3] for $k = 4$ and $m = n\lceil \log_2 q \rceil + 2\tau$ such that $\lambda, \tau = 110$.

	$D = 3$	$D = 7$	$D = 15$
$\delta = 4$	$q = 5^{30}, \alpha = 10^{-19}, n = 1600$	$q = 11^{23}, \alpha = 1^{-22}, n = 1900$	$q = 17^{21}, \alpha = 10^{-23}, n = 1900$
$\delta = 6$	$q = 5^{30}, \alpha = 10^{-19}, n = 1600$	$q = 11^{23}, \alpha = 1^{-22}, n = 1900$	$q = 17^{22}, \alpha = 10^{-25}, n = 2100$
$\delta = 8$	$q = 5^{30}, \alpha = 10^{-19}, n = 1600$	$q = 11^{23}, \alpha = 1^{-22}, n = 1900$	$q = 17^{22}, \alpha = 10^{-25}, n = 2100$

A.2.3. Protocols 5 and 6

Table A.7: Parameter setting for Protocols 5 and 6 for different δ, D based on Table 6.7 and [3] for $k = 2, 3, 4$ and $m = n\lceil \log_2 q \rceil + 2\tau$ such that $\lambda, \tau = 110$.

	$D = 3$	$D = 7$	$D = 15$
$\delta = 4$	$q = 2^{67}, \alpha = 1^{-18}, n = 1550$	$q = 2^{73}, \alpha = 10^{-20}, n = 1700$	$q = 2^{80}, \alpha = 1^{-22}, n = 1900$
$\delta = 6$	$q = 2^{70}, \alpha = 10^{-19}, n = 1600$	$q = 2^{73}, \alpha = 10^{-20}, n = 1700$	$q = 2^{80}, \alpha = 1^{-22}, n = 1900$
$\delta = 8$	$q = 2^{70}, \alpha = 10^{-19}, n = 1600$	$q = 2^{77}, \alpha = 1^{-21}, n = 1800$	$q = 2^{80}, \alpha = 1^{-22}, n = 1900$

A.2.4. Protocols 7 and 8

Table A.8: Parameter setting for Protocols 7 and 8 for different δ, D based on Table 6.7 and [3] for $k = 2, 3, 4$ and $m = n\lceil \log_2 q \rceil + 2\tau$ such that $\lambda, \tau = 110$.

	$D = 3$	$D = 7$	$D = 15$
$\delta = 4$	$q = 5^{23}, \alpha = 10^{-14}, n = 1150$	$q = 11^{18}, \alpha = 10^{-16}, n = 1300$	$q = 17^{16}, \alpha = 10^{-17}, n = 1400$
$\delta = 6$	$q = 5^{25}, \alpha = 10^{-15}, n = 1250$	$q = 11^{18}, \alpha = 10^{-16}, n = 1300$	$q = 17^{16}, \alpha = 10^{-17}, n = 1400$
$\delta = 8$	$q = 5^{25}, \alpha = 10^{-15}, n = 1250$	$q = 11^{19}, \alpha = 10^{-17}, n = 1400$	$q = 17^{17}, \alpha = 10^{-19}, n = 1600$

Table A.9: Parameter setting for Protocols 7 and 8 with adapted encryption procedure for different δ, D based on Table 6.7 and [3] for $k = 2, 3, 4$ such that $\lambda, \tau = 110$.

	$D = 3$	$D = 7$	$D = 15$
$\delta = 4$	$q = 5^{16}, \alpha = 1^{-9}, n = 700$	$q = 11^{13}, \alpha = 1^{-11}, n = 900$	$q = 17^{12}, \alpha = 10^{-12}, n = 950$
$\delta = 6$	$q = 5^{16}, \alpha = 1^{-9}, n = 700$	$q = 11^{13}, \alpha = 1^{-11}, n = 900$	$q = 17^{12}, \alpha = 10^{-12}, n = 950$
$\delta = 8$	$q = 5^{17}, \alpha = 1^{-10}, n = 850$	$q = 11^{14}, \alpha = 10^{-12}, n = 950$	$q = 17^{12}, \alpha = 10^{-12}, n = 950$

A.2.5. Protocol 9

Table A.10: Parameter setting for Protocol 9 for different δ, D based on Table 6.7 and [3] for $k = 2, 3, 4$ such that $\lambda, \tau = 110$.

	$D = 3$	$D = 7$	$D = 15$
$\delta = 4$	$q = 2^{50}, \alpha = 10^{-13}, n = 1100$	$q = 2^{53}, \alpha = 10^{-14}, n = 1200$	$q = 2^{60}, \alpha = 10^{-16}, n = 1350$
$\delta = 6$	$q = 2^{50}, \alpha = 10^{-13}, n = 1100$	$q = 2^{57}, \alpha = 10^{-15}, n = 1250$	$q = 2^{60}, \alpha = 10^{-16}, n = 1350$
$\delta = 8$	$q = 2^{53}, \alpha = 10^{-14}, n = 1200$	$q = 2^{57}, \alpha = 10^{-15}, n = 1250$	$q = 2^{63}, \alpha = 10^{-17}, n = 1450$

A.2.6. Protocol 10

Table A.11: Parameter setting for Protocol 10 for different δ, D based on Table 6.7 and [3] for $k = 2,3,4$ such that $\lambda, \tau = 110$.

	$D = 3$	$D = 7$	$D = 15$
$\delta = 4$	$q = 5^{16}, \alpha = 1^{-9}, n = 700$	$q = 11^{13}, \alpha = 1^{-11}, n = 900$	$q = 17^{12}, \alpha = 10^{-12}, n = 950$
$\delta = 6$	$q = 5^{16}, \alpha = 1^{-9}, n = 700$	$q = 11^{13}, \alpha = 1^{-11}, n = 900$	$q = 17^{12}, \alpha = 10^{-12}, n = 950$
$\delta = 8$	$q = 5^{17}, \alpha = 1^{-10}, n = 850$	$q = 11^{14}, \alpha = 10^{-12}, n = 950$	$q = 17^{12}, \alpha = 10^{-12}, n = 950$

A.3. Additional results

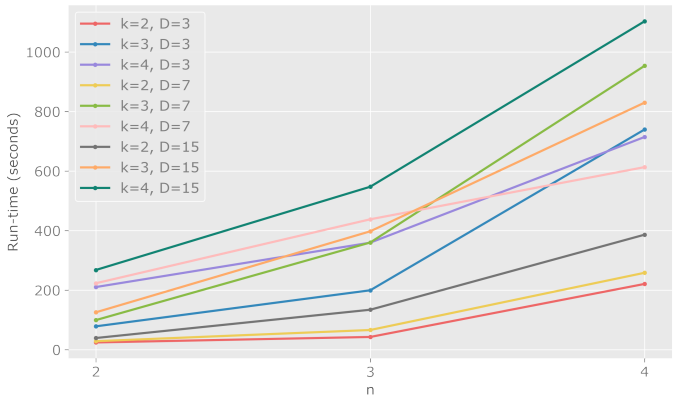


Figure A.1: The run-time of the extension procedure for Protocols 1 and 2 plotted against n for different values of D, k with $\delta = 4$.

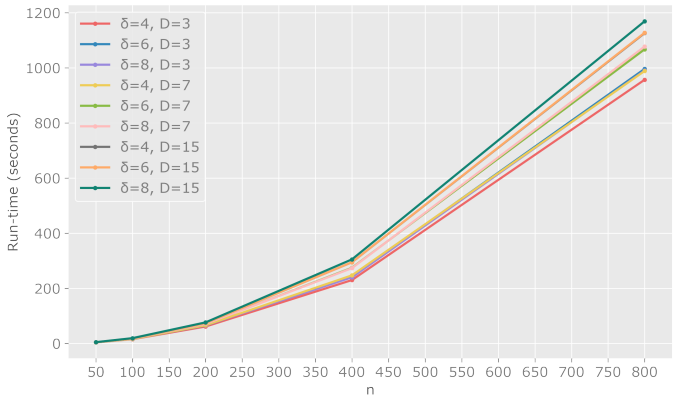


Figure A.2: The run-time of an initial encryption of one bit for Protocols 1 and 2 plotted against the dimension n for different values of D, δ with $k = 2$.

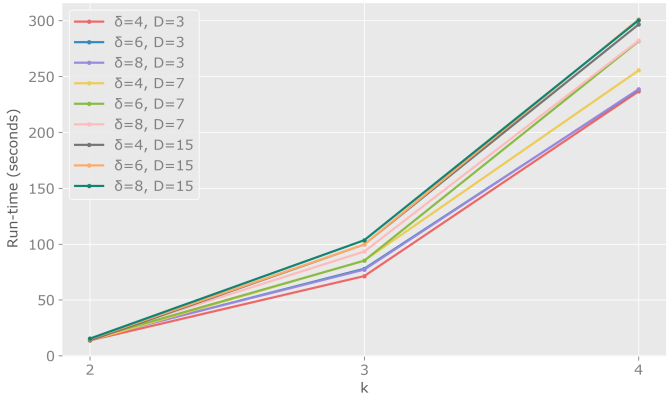


Figure A.3: The run-time of one path multiplication for Protocols 1 and 2 plotted against the number of users k for different values of D, δ with $n = 10$.

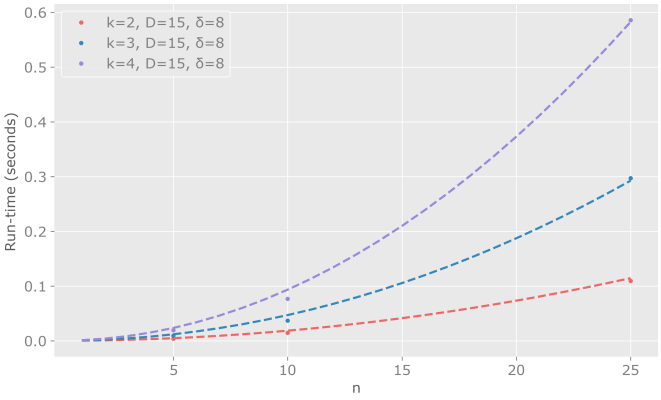


Figure A.4: The run-time of one leaf-bit evaluation for Protocols 1 and 2 plotted against the dimension n for different values of k with $D = 15, \delta = 8$. The dashed functions are given by $6.26 \cdot 10^{-7} n^2 k^2 l$.

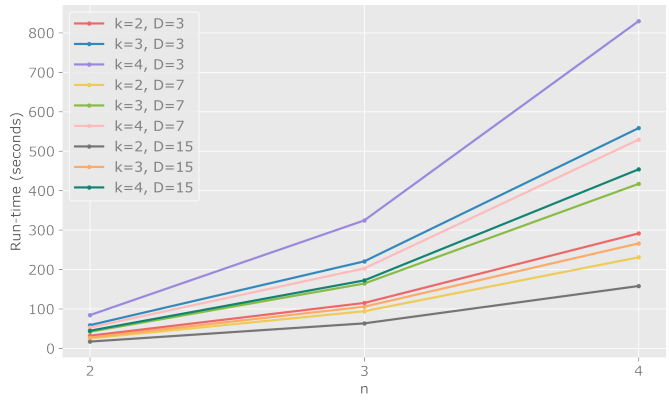


Figure A.5: The run-time of the key generation for Protocols 3 and 4 plotted against n for different values of D, k with $\delta = 4$.

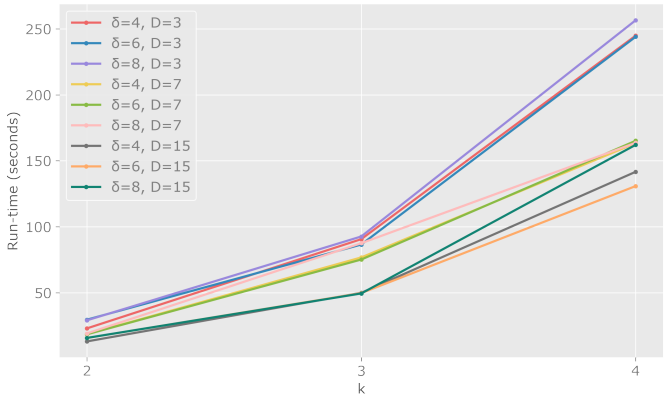


Figure A.6: The run-time of the extension for Protocols 3 and 4 plotted against k for different values of D, δ with $n = 4$.

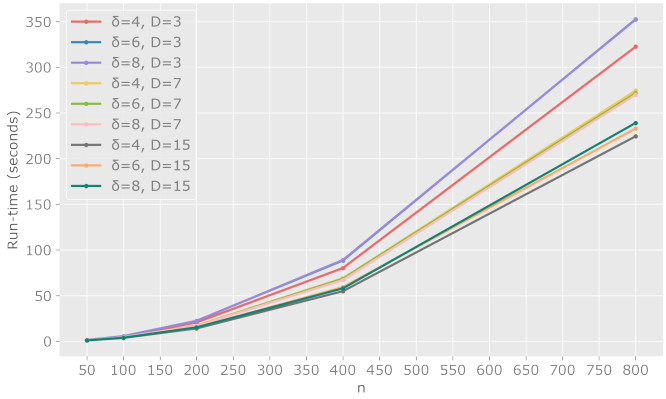


Figure A.7: The run-time of an initial encryption of one bit for Protocols 3 and 4 plotted against the dimension n for different values of D, δ with $k = 2$.

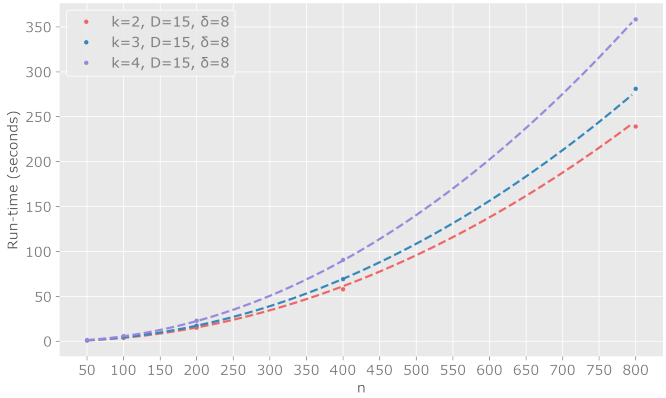


Figure A.8: The run-time of an initial encryption of one bit for Protocols 3 and 4 plotted against the dimension n for a tree with $D = 15$, $\delta = 8$ and different k. The dashed functions are given by $2.55 \cdot 10^{-5} \cdot n^2 l$.

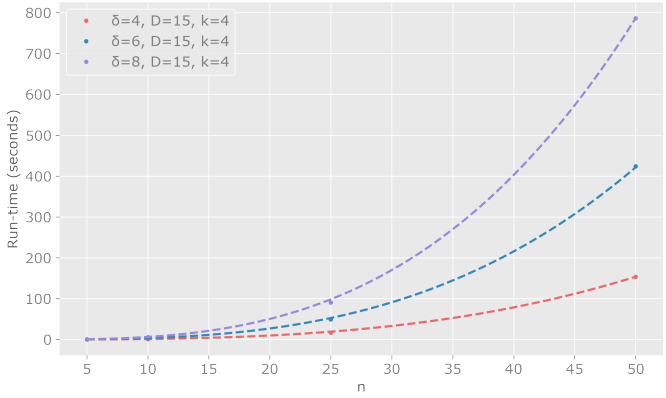


Figure A.9: The run-time of one decision-node evaluation (worst case with a threshold value of 0) for Protocols 3 and 4 plotted against the dimension n for different values of δ with $D = 15, k = 4$. The dashed functions are given by $2.32 \cdot 10^{-7} \cdot (\delta^2 - \delta) n^3 \ell^2$.

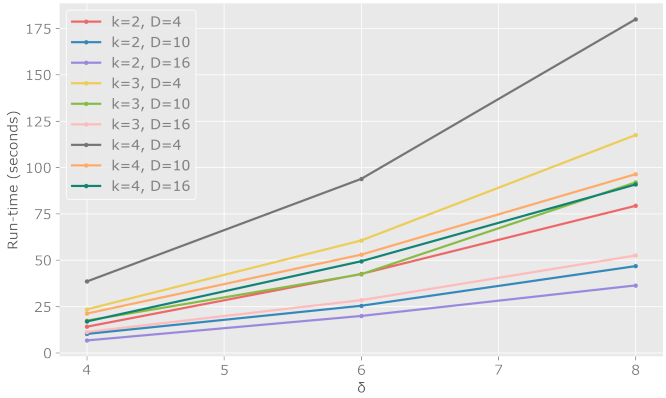


Figure A.10: The run-time of one decision-node evaluation (worst case with a threshold value of 0) for Protocols 3 and 4 plotted against δ for different values of D, k with $n = 25$.

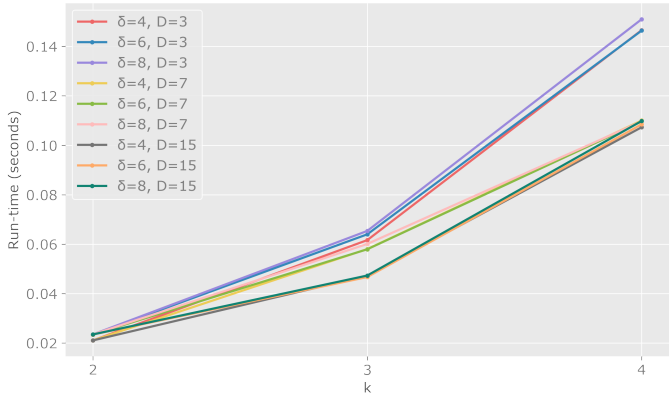


Figure A.11: The run-time of one path addition for Protocols 3 and 4 plotted against the number of users k for different values of D, δ with $n = 25$.

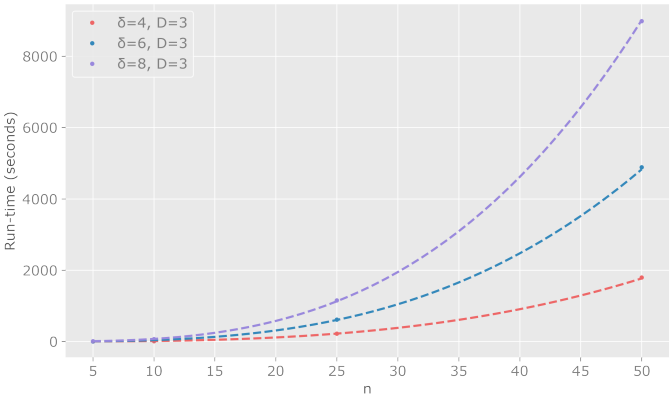


Figure A.12: The run-time of one decision-node evaluation (worst case with a threshold value of 0) for Protocols 5 and 6 plotted against the dimension n for different values of δ with $D = 3$. The dashed functions are given by $2.63 \cdot 10^{-7} \cdot (\delta^2 - \delta) n^3 \ell^2$.

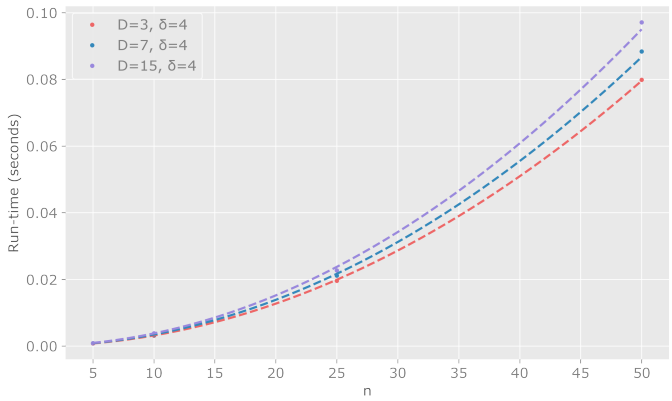


Figure A.13: The run-time of one leaf-bit evaluation for Protocols 5 and 6 plotted against the dimension n for different values of D with $\delta = 4$. The dashed functions are given by $4.75 \cdot 10^{-7} n^2 \ell$.

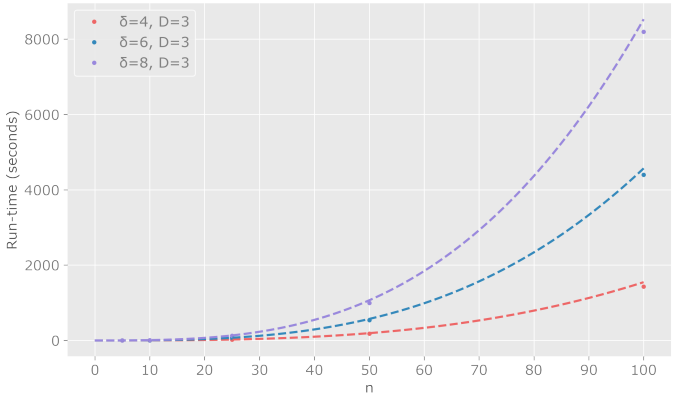


Figure A.14: The run-time of one decision-node evaluation (worst case with a threshold value of 0) for Protocols 7 and 8 plotted against the dimension n for different values of δ with $D = 3$. The dashed functions are given by $2.44 \cdot 10^{-7} \cdot (\delta^2 - \delta) n^3 \ell^2$.

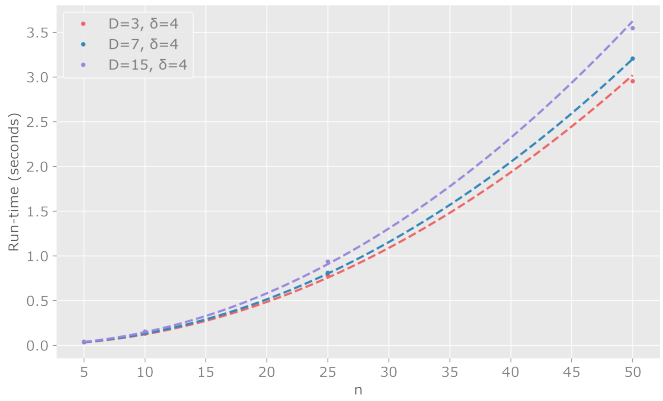


Figure A.15: The run-time of an initial encryption of one bit for Protocol 9 plotted against the dimension n for a tree with $\delta = 4$ and different D. The dashed functions are given by $2.42 \cdot 10^{-5} \cdot n^2 \ell$.

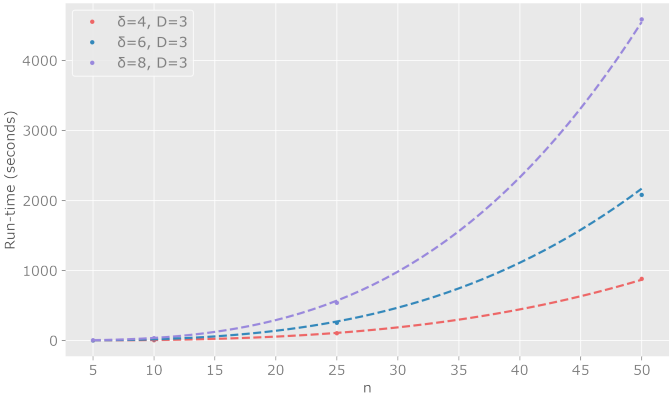


Figure A.16: The run-time of one decision-node evaluation (worst case with a threshold value of 0) for Protocol 9 plotted against the dimension n for different values of δ with $D = 3$. The dashed functions are given by $2.31 \cdot 10^{-7} \cdot (\delta^2 - \delta) n^3 \ell^2$.

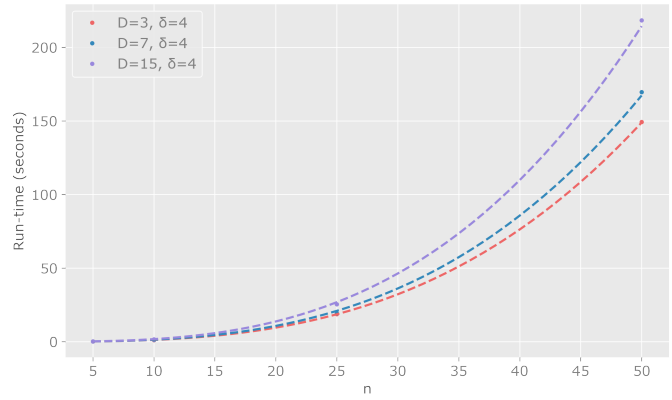


Figure A.17: The run-time of one path multiplication for Protocol 9 plotted against the dimension n for different values of D with $\delta = 4$. The dashed functions are given by $4.76 \cdot 10^{-7} n^3 \ell^2$.

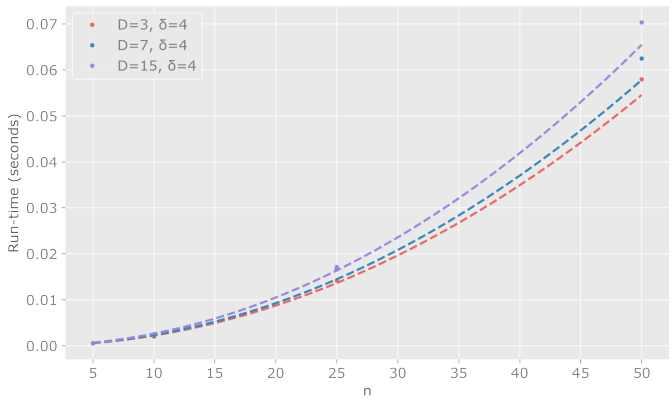


Figure A.18: The run-time of one leaf-bit evaluation for Protocol 9 plotted against the dimension n for different values of D with $\delta = 4$. The dashed functions are given by $4.36 \cdot 10^{-7} n^2 \ell$.

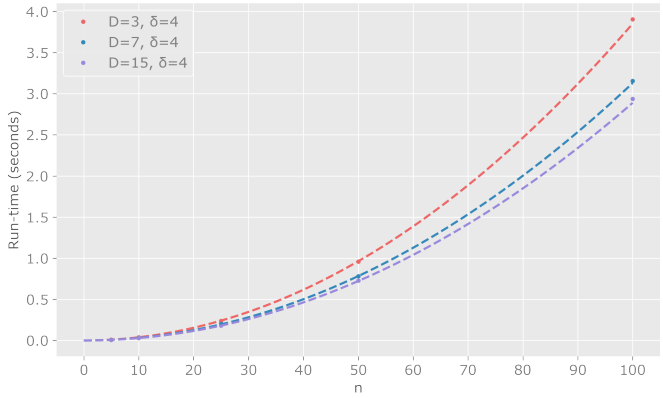


Figure A.19: The run-time of an initial encryption of one bit for Protocol 10 plotted against the dimension n for a tree with $\delta = 4$ and different D. The dashed functions are given by $2.41 \cdot 10^{-5} \cdot n^2 \ell$.

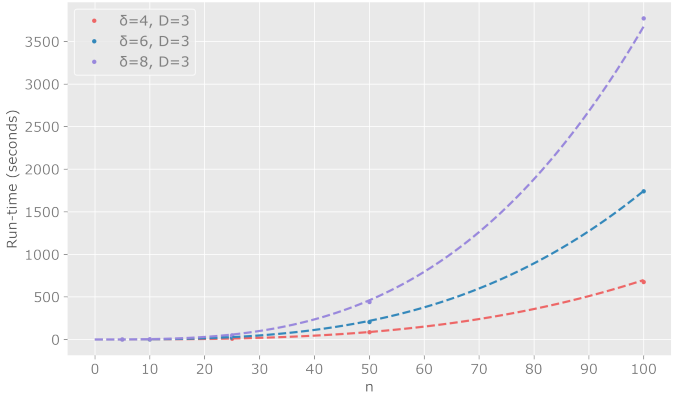


Figure A.20: The run-time of one decision-node evaluation (worst case with a threshold value of 0) for Protocol 10 plotted against the dimension n for different values of δ with $D = 3$. The dashed functions are given by $2.27 \cdot 10^{-7} \cdot (\delta^2 - \delta) n^3 \ell^2$.

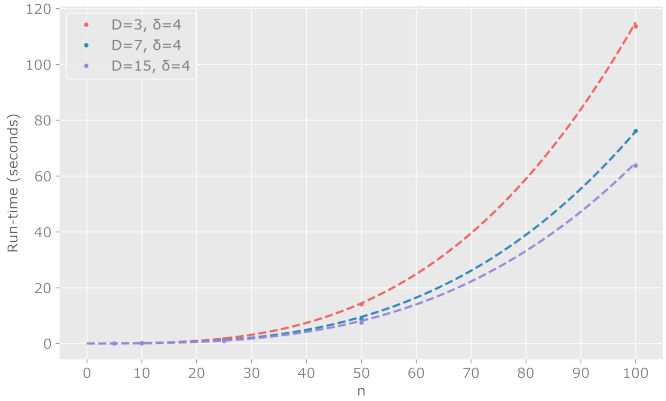


Figure A.21: The run-time of one key-switching procedure for Protocol 10 plotted against the dimension n for different values of D with $\delta = 4$. The dashed functions are given by $4.50 \cdot 10^{-7} n^3 \ell^2$.

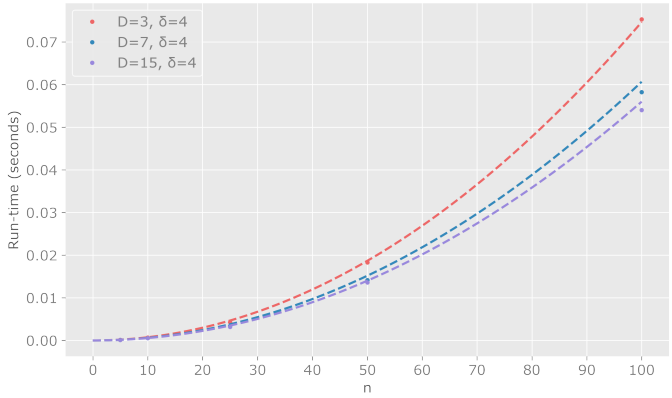


Figure A.22: The run-time of one path addition for Protocol 10 plotted against the dimension n for different values of D with $\delta = 4$. The dashed functions are given by $4.66 \cdot 10^{-7} n^2 \ell$.

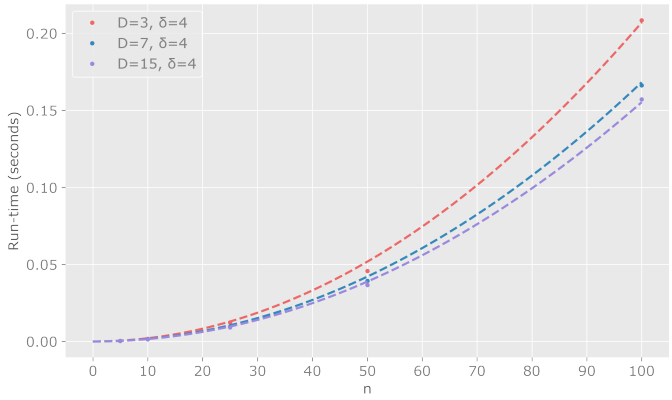


Figure A.23: The run-time of one leaf-bit evaluation for Protocol 10 plotted against the dimension n for different values of D with $\delta = 4$. The dashed functions are given by $1.29 \cdot 10^{-6} n^2 \ell$.

A.4. Example use-case

In Figure 1.2 in the introduction, we gave an example use-case tree which demonstrates how a decision tree can be used within access control. In this section we show the results of evaluating this example tree using our protocols, to give an idea of how our protocols will behave for such a tree. We focus on Protocols 7, 8 and 10, since we concluded that these are the most promising ones. Of these protocols, only Protocol 8 can be applied within access control. The other two protocols can only be applied when the users receive the evaluation result. Still, it is interesting to see how these protocols will behave for this example tree.

In order to evaluate the tree, the input values and the threshold and categorical values need to be numerical. These translations to numerical values can be seen in Table A.12. We set the input bit length to $\delta = 6$ such that at most 16 different values per input variable can exist. Using these sets, we can translate the tree to a numeric tree which can be seen in Figure A.24. For this use-case we have $k = 3$, since three parties send their input to the sever, namely the party that tries to gain access, the hospital of the requestor and the emergency centre where the location of the emergency situation and region is known.

Table A.12: The numerical sets of the different input variables of the example decision tree in Figure 1.2.

Employment years	$0,1,2,\ldots,15$
Function	$0,1,2,3,4$:= Nurse, co-assistant, ANIOS, AIOS, specialist
Hospital region	$0,1,2,\ldots,11$:= Friesland, Groningen, Drenthe, Overijssel, Gelderland, Flevoland, Noord-Brabant, Utrecht, Limburg, Noord-Holland, Zuid-Holland, Zeeland
Department	$0,1,2,\ldots,7$:= Cardiology, EHC, IC, Surgery, Gynecology, Neurology, Radiology, Pediatrics
Owner device	$0,1,2$:= Private, Public, Hospital
Owner network	$0,1,2$:= Private, Public, Hospital
Location	$0,1,2,\ldots,15$

It can be seen that for this tree it holds that the number of decision nodes $\sigma = 12$ and the number of leaf nodes $\gamma = 13$. The depth of the tree is 5. Looking at Table A.8 in the appendix, the correct parameters for Protocols 7 or 8 are $q = 11^{18}$ such that $\ell = 18$ with $n = 1300$ to guarantee security. For Protocol 10, we find $q = 11^{13}$ such that $\ell = 13$ and $n = 900$. The decision-tree evaluation run-times can be seen in Table A.13, for fixed $n = 100$ and $n = 200$. The numbering of each decision node is in the order of breadth-first search as given in Figure A.24.

As expected, the decision-node evaluation run-times of nodes $0,1,2,6,8,10$ and 11 are similar, since these are all categorical decision nodes. Nodes 3 and 5 have the same threshold value given by 0010. This requires 4 homomorphic multiplications and is therefore more costly than the equality protocol that requires 3 homomorphic multiplications for $\delta = 4$. Nodes $4,7$ and 9 have the threshold value 0001 which requires the same number of homomorphic multiplications as the equality protocol. This agrees with the results in Table A.13.

Comparing the two different protocols, it can be seen that the total evaluation time of Protocol 10 is higher, although the time per decision-node evaluation is lower due to the smaller value of ℓ. The additional time is caused by the $\sigma + k\gamma(\lceil \log_2 \gamma \rceil + 1)$ key switches that have to take place, that each require around 73 or 613 seconds for $n = 100$ and $n = 200$ respectively. The result labels are stored in $\lceil \log_2 5 \rceil = 3$ ciphertexts (5 different class labels). Therefore, the total cost of all key switches for Protocol 10 for $n = 100$ is around $73 \cdot (\sigma + 4k\gamma) = 12264$ seconds. For $n = 200$, $613 \cdot (\sigma + 4k\gamma) = 102984$ seconds.

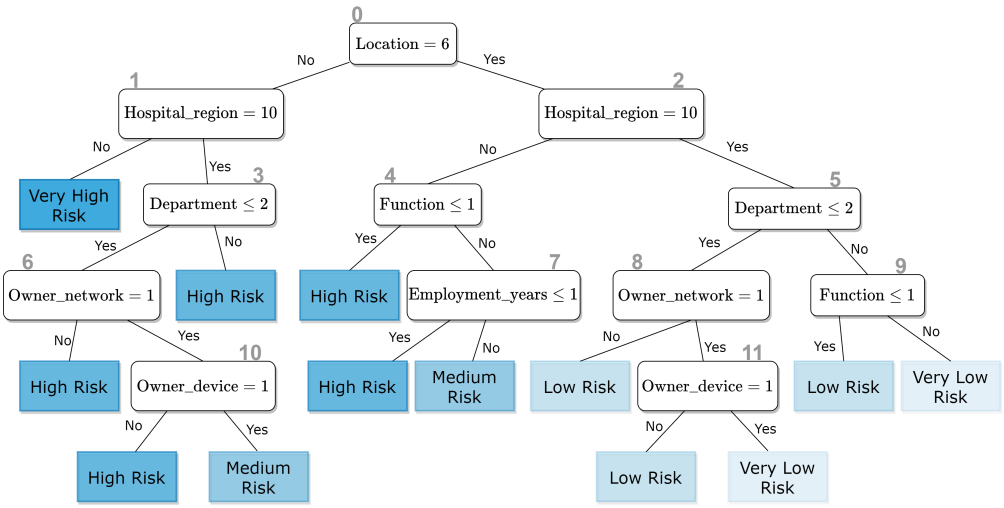


Figure A.24: The decision tree of the use-case in Figure 1.2 translated to numeric characters and in the same breadth-first order as the implementation.

Table A.13: The run-time in seconds of the several parts of the server's evaluation of the use-case tree in Figure A.24 for Protocols 7, 8 and 10, for fixed $n = 100$ and $n = 200$.

Part	$n = 100$		$n = 200$	
	Protocols 7 and 8	Protocol 10	Protocols 7 and 8	Protocol 10
Total evaluation	5573.5	15119.8	43109.9	123450.0
Decision-node evaluation 0	445.2	212.5	3384.5	1839.3
Decision-node evaluation 1	451.0	211.1	3373.9	1781.1
Decision-node evaluation 2	443.3	213.3	3387.6	1825.5
Decision-node evaluation 3	582.8	291.9	4541.0	2422.0
Decision-node evaluation 4	435.1	216.1	3382.3	1790.6
Decision-node evaluation 5	585.6	285.0	4534.2	2396.0
Decision-node evaluation 6	440.8	221.6	3413.8	1787.8
Decision-node evaluation 7	436.3	217.0	3403.8	1793.4
Decision-node evaluation 8	434.6	216.9	3387.1	1809.0
Decision-node evaluation 9	431.2	213.2	3395.0	1797.5
Decision-node evaluation 10	435.3	218.3	3433.8	1822.2
Decision-node evaluation 11	437.4	218.5	3403.0	1809.4
Key switch	-	72.78	-	613.0
Path addition	0.08273	0.06150	0.4267	0.2477
Leaf node evaluation	0.2081	0.1505	0.9688	0.6362

For the same security level, the value for n in Protocols 7 and 8 should be 1300, while in Protocol 10 this is 900. We can give an expectation of the total run-times for these values assuming the protocols only consist of homomorphic multiplications. The complexity of a homomorphic multiplications depends on n with a power of 3, which can also be seen by the difference in the results between $n = 100$ and $n = 200$. Therefore, for Protocol 10, it is expected that the total evaluation time is $900^3/200^3 \cdot 123450.0 \approx 130$ days. For Protocols 7 and 8 this expectation is given by $1300^3/200^3 \cdot 43109.9 \approx 137$ days.

Protocols 7 and 8 require the users to do an expensive encryption procedure which, according to the results in Figure 7.9, results in encryption times of around 12 days per input bit for the above parameters. Luckily, in Section 7.9.1 we proposed a modification to Protocols 7 and 8, in which for every input bit a 0-encryption is sent from the STTP to the user. Since the value of ℓ for this adapted version is the same as the value of ℓ for Protocol 10, the run-time of the total evaluation is the same as for Protocol 10 minus the costs for the key switches. The expected run-time for this adapted version is then $900^3/200^3 \cdot (123450.0 - 102984) \approx 22$ days. Please note that all of the above run-times do not take into account the possibility of a parallelized implementation.

In Figure A.25, the communication sizes are given for the first one or two key-exchange rounds and the rounds in which the three users send their input to the server. The communication in the key exchange round(s) is the sum of all communication required in those rounds between the users and the STTP. The key-exchange communication of the adapted version of Protocols 7 and 8 needs to occur every time a decision tree gets evaluated. This is not the case for the other protocols. In red, it can be seen that the communication for Protocols 7 and 8 is higher than for the adapted version. Clearly, the public key is bigger than the $a_i\delta$ 0-encryptions that are communicated to the users. Also, the ciphertext sizes are bigger due to higher n and ℓ. Since $k = 3$, the number of switching-keys is lower than the number of 0-encryptions that are communicated. Therefore, the communication in the key-exchange rounds of Protocol 10 is lower than in the adapted version of Protocols 7 and 8. The differences in communication between the input rounds can be explained by the fact that user 1 and 3 have two input variables and user 2 three input variables.

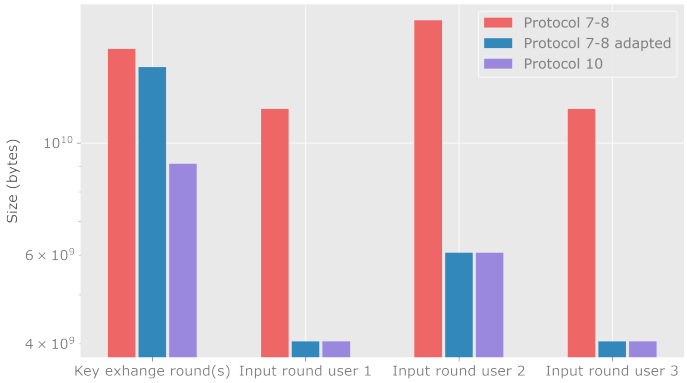


Figure A.25: The communication sizes on a logarithmic scale of the different rounds in (adapted) Protocols 7, 8 and 10 of the use-case in Figure A.24. The value of n is chosen such that security is guaranteed according to Tables A.8, A.9 and A.11.