PRIVACY-PRESERVING CROSS-DEVICE TRACKING

Alexandru BĂBEANU
(4881133)

to obtain the degree of Master of Science in Computer Science
Software Technology Track
with specialisation in Cyber Security
to be defended publicly on the 25th of November 2021

DELFT UNIVERSITY OF TECHNOLOGY
Faculty of Electrical Engineering, Mathematics & Computer Science
Department of Intelligent Systems
Cyber Security Group

Supervisor: Dr. Zekeriya Erkin

Thesis Committee: Dr. Zekeriya Erkin
Dr. Stjepan Picek
Dr. Jérémie Decouchant

ABSTRACT

Online advertisement is a multi-billion dollar industry that constitutes a primary source of income for most publishers offering free content on the Web. Online behavioural advertisement refers to the practice of serving targeted ads to online users based on their potential interests. In order to infer these interests, online advertisers aggregate and process vast amounts of behavioural data collected from the browsing activities of Internet users. Until recently, online advertisers have been using a device-centric approach to studying user behaviour and delivering targeted ads. With the development and widespread adoption of smart devices, it has become commonplace for a person to own and browse the Web from multiple devices, and online advertisers have been quick to adapt.

Cross-device tracking (CDT) constitutes the practice of identifying and tracking the user of a device rather than the device itself. This is usually achieved by grouping devices based on the likelihood that they belong to the same user. By employing CDT, advertisers are able to build more comprehensive user profiles based on a user's overall online behaviour, and serve advertisement tailored to a user's interests on all of their devices.

While these online advertising practices greatly benefit both advertisers and their clients, the privacy of online users is compromised by the amount and nature of the data collected in the process. Various solutions have been proposed over the years for performing online behavioural advertisement in a privacy-preserving manner, while no such privacy-conscious technological alternatives have been designed to address CDT practices, even though the data collected for the purpose of CDT has considerable overlap with the data collected for serving targeted ads.

In this thesis, we aim to offer a technological solution to the privacy-issues posed by the collection and processing of customer data for the purpose of cross-device tracking. To this end we design PCDT, a protocol that uses fully-homomorphic encryption and keyed hash functions to implement both deterministic and probabilistic CDT techniques that operate on encrypted data.

PCDT operates in a two server setting, where both servers engage in privacy-preserving computations to perform CDT, while simultaneously concealing the secret device data from each other through cryptographic means. The security of PCDT is based on the semi-honest security model, where parties attempt to learn as much as possible from the information presented to them, but do not deviate from the protocol. PCDT performs deterministic CDT in a privacy-preserving manner by concealing the relevant device data using a keyed hash function. The deterministic hashes allow for the fast association of devices through deterministic CDT, while the secrecy of the hashing key ensures the secrecy of the plaintext data. To perform privacy-preserving probabilistic CDT, PCDT uses fully-homomorphic encryption to train and evaluate gradient boosting decision tree models on encrypted device data. To the best of our knowledge, PCDT is the first protocol designed to perform privacy-preserving CDT.

CONTENTS

List of Figures

List of Tables

List of Pseudocodes

1

INTRODUCTION

Online advertisement is multi-billion dollar industry that focuses on conducting advertisement campaigns through the means of the Internet. The major advantages of online advertising over traditional means of advertising (e.g. television) come from allowing advertisers to tailor which ads are displayed for which Internet users based on their individual preferences, as well as measure the success of individual ad campaigns by counting how many users click on an ad (click-through), or how many users have actually spent money as a result of the ad (conversion rate) [1]. However, while these online advertising practices greatly benefit advertisers, they come at the detriment of privacy for Internet users.

The practice of targeted advertisement requires vast amounts of user data, which are collected by and shared between dozens of parties within the advertisement ecosystem, often without the user's knowledge or consent [2, 3]. This data contains all sorts of sensitive information, such as a user's Internet browsing behaviour, location, products purchased, religious or political beliefs, etc., and constitutes a serious threat to the privacy of Internet consumers [4].

In recent years, following the proliferation of *smart* devices, such as smartphones, tablets, smart TVs, etc., it has become commonplace for people to browse the Internet using multiple devices. This has caused a shift in the ad-targeting paradigm from a device-centric approach to a user-centric one [18]. As a result, online advertisers engage in the process of *cross-device tracking*, which involves various techniques of identifying which devices belong to the same user. The knowledge that a group of devices represents a single potential customer allows advertisers to build more comprehensive user profiles based on their overall online behaviour, and serve advertisement tailored to a user's interests on all their devices.

Various solutions try to address the privacy issues posed by online advertisement, ranging from outright blocking ads from being displayed on user devices [5], to privacy-preserving alternatives of existing practices [7, 10–12], while regulators are primarily concerned with security and transparency [55, 56]. With regards to cross-device tracking, ad blockers do have functionality that prevents tracking to some extent [6], and regula-

tors are taking notice of the practice [20], but no privacy-preserving solutions exist in literature as of the writing of this thesis. This work aims to address that gap and provide the first privacy-preserving protocol for performing cross-device tracking.

1.1. CROSS-DEVICE TRACKING

Cross-device tracking (CDT) refers to the process of tracking a user across multiple devices in order to construct a cohesive user profile that is linked to all devices owned by said user. The core building block of CDT is the device association graph [14], in which nodes representing individual devices are linked based on the likelihood that said devices belong to the same user. Using this graph, devices can be clustered into user profiles based on various techniques. Multiple technologies are employed in performing CDT, such as tracking cookies, data mining, machine learning, and even ultrasound signals [15, 16, 18, 19, 69, 70]. Based on the techniques involved, CDT is divided into two categories, deterministic CDT and probabilistic CDT [20].

Determinitic CDT is primarily performed by having a user log into an online account from multiple devices. When a user logs into a webservice account, e.g. email or social media account, on both their smartphone and their desktop computer, it informs the service provider that both devices belong to the same user. Alternatively, websites can track users even without them signing in, by matching the personally identifiable information (PII) they input when signing up for an account or a subscription, shopping online, ordering food, or filling in a questionnaire. This information may include email addresses, home addresses, phone numbers, credit card numbers etc.

Probabilistic CDT is performed by using statistical analysis to analyse data that has not been specifically inputted by the user, but is instead inferred from the device or communication channel used to access the webservice. This data includes IP addresses, location data, browsing history, temporal information such as location or IP addresses over time, and terminal information such as browser version, window dimensions, operating system, device model, screen dimensions, etc. IP addresses are given particular importance, as two devices sharing an IP address is the primary probabilistic indicator that they might belong to the same user [64, 65, 69]. As none of the information used in probabilistic CDT is unique, a more involved analysis is required than in the case of deterministic CDT, usually in the form of machine learning models [69, 70].

While deterministic CDT is the more precise and reliable of the two, probabilistic CDT is more covert, and can supplement a deterministic approach by tracking users even when they do not log in or provide PII [15].

The main incentive for the development of CDT technologies lays in their utility in the field of online advertisement, namely enabling advertisers to monitor user behaviour and deliver targeted advertisement across devices. However, online advertisement is not the only use case for CDT. The fields of fraud detection and account security can also benefit from the use of CDT [20]. For example, tracking which devices belong to a user would enable companies to detect when a consumer is using a new device and perform a more thorough authentication procedure than in the case of known devices.

1.2. PRIVACY ISSUES

The downsides of CDT practices involve lack of transparency, privacy breaches, and the collection and aggregation of private user information [20].

The lack of transparency stems from the fact that CDT is not obvious from a user perspective, especially in the case of probabilistic CDT, and webservice providers rarely inform their users about the practice [13]. Furthermore, users are largely unaware of the scope and capabilities of CDT practices [21]. To address this issue, the FTC suggests that organisations disclose their use of CDT technologies in their privacy statements, be transparent with the types of data being collected for the procedure, and even give their users the option to opt out.

A major privacy concern regarding CDT comes from the interaction between CDT and online behavioural advertising, where advertisement tailored based on online activities conducted on private devices can appear on public or shared devices (e.g. home desktop, work laptop), disclosing private information [22, 23]. In regards to this issue, FTC recommends that online advertisers refrain from engaging in CDT on sensitive topics such as health, financial status, or children information.

Finally, in a similar vein as online behavioural advertisement, CDT practices cause privacy concerns simply from the collection and aggregation of private user information that makes CDT possible. This data can be quite sensitive, involving names, phone numbers, email addresses, home addresses, location data, Internet browsing history, etc. This data is collected, aggregated and stored by various parties within the online advertisement ecosystem, which poses serious security risks. In this case, FTC simply requests companies to only keep data that is necessary for business purposes, and to properly secure the data they do keep.

1.3. RESEARCH GOAL AND MOTIVATION

Most works that approach CDT from a privacy-conscious perspective are concerned with detecting it [16, 18], preventing it [6, 16], or measuring its impact on user privacy [13]. In this paper, we address the third privacy issue posed by cross-device tracking, namely the collection of user data required to perform CDT, from a technical perspective. Inspired by the works in privacy-preserving data-mining [29, 30, 32], and especially by Helsloot's work on privacy-preserving behavioural advertising [7], we aim to design a system where user data is collected in encrypted form, such that CDT can still be performed without compromising user privacy. The main goal of our research is to answer the following question:

> *How can a system become aware of which devices that access an online service are used by the same user, such that no party has access to a user's data besides the user who owns it ?*

Based on the main research question, we formulate the following sub-questions:

1. How can user data be made available for the purpose of CDT, without disclosing the actual meaning of the data ?

2. How can deterministic CDT be performed without knowledge of a user's personally identifiable information ?

3. How can probabilistic CDT be performed without knowledge of a user's inferred data ?

4. How efficient, and therefore practical, is privacy-preserving CDT ?

The primary motivation behind our work is to limit the impact of CDT on the privacy of online users. By providing a privacy-preserving technological alternative to existing CDT practices, we aim to remove the need of online advertisers and service providers to collect and store sensitive user information for the purpose of cross-device tracking.

The impact of data leaks would also be mitigated as a result of user data being stored in a format that preserves its secrecy even from the organisations storing it. Additionally, our design would allow organisations to conduct their CDT-related activities while complying with data collection regulations.

Furthermore, the existence of practical privacy-preserving technologies for both online behavioural advertising and CDT can give policy makers a better case when pushing for more privacy-conscious practices from online marketers and service providers.

1.4. OUR CONTRIBUTION

In this thesis we present PCDT, a protocol for performing privacy-preserving cross-device tracking. PCDT operates in a two-server setting, where a privacy service (PS) assists a tracking service (TS) with privacy-preserving computations in order to perform CDT over homomorphically encrypted data. The security of PCDT is based on the semi-honest security model, where parties attempt to learn as much as possible from the information presented to them, but do not deviate from the protocol.

The main cryptographic primitives employed by PCDT are the BGV fully-homomorphic scheme [34], which allows for the computation of arbitrary binary circuits over encrypted data, and the keyed hash function AES-CMAC [46], which we modify to compute hashes over homomorphic ciphertext space.

The protocol is comprised of three main algorithms: an algorithm that allows devices to update the encrypted data used to perform CDT, an algorithm that updates a device association graph based on encrypted data received from users, and an algorithm used to train a prediction model on encrypted data. The prediction model trained by PCDT is used to perform probabilistic CDT when updating the association graph. The device association graph produced by the protocol is constructed using both deterministic and probabilistic CDT.

Deterministic CDT is performed by hashing each PII item supplied by a device and using simple table-lookup operations to match hashes between devices instead of the actual PII. This construction is inspired by the searchable encryption scheme proposed by Bellare et al. [37], which uses hashes of the search keys to allow for sublinear search times, but does not share the same vulnerability to table-lookup attacks, since TS does not have the key under which the hashes are computed. To keep the plaintext data hidden from both servers and safe from table lookup attacks, the hashes are computed using

a modified version of AES-CMAC, which takes as input a BGV-encrypted string s and a key k, and outputs the homomorphic ciphertext encrypting the output of AES-CMAC given s and k. As a result, PS learns neither the initial strings nor the resulted hashes, while TS obtains the deterministic values which can be searched for in constant time, but cannot compute hashes of its own to perform a table lookup attack as it does not know the hash key.

Probabilistic CDT is performed by evaluating a gradient-boosting decision tree model on homomorphically encrypted data. The choice of machine learning algorithm was made based on existing research [69, 70] and winning solutions for CDT-focused data mining competitions [64–68]. The evaluation and training algorithms are based on existing constructions for evaluating [73] and training [72] decision trees over homomorphically-encrypted data, modified to fit the setting of PCDT, and use the BGV encryption scheme and the comparison algorithm proposed by Cheon et al. in [47]. The use of homomorphic encryption keeps the user data hidden from PS, while a symmetric encryption scheme is used to hide both the data and the machine learning model from TS.

One downside of our approach to privacy-preserving probabilistic CDT is that TS cannot evaluate the output of models that were not produced by the training algorithm of PCDT, nor can TS learn the plaintext of the trained model. Both design decisions were made to prevent TS from learning information about the plaintext user data from the model output, by tracing the decision path in reverse and finding out how values in the input vector compare against the threshold values in the tree nodes. This is a consequence of the fact that decision trees are white-box models, so there is a simple, observable path through the model that connects each input to its respective output.

Overall, this thesis offers three contributions to the field of privacy-preserving cross-device tracking:

1. PCDT, a protocol for constructing device-association graphs in a privacy-preserving manner, using both deterministic and probabilistic CDT.

2. An algorithm for evaluating the AES-CMAC function over BGV-encrypted data, which is used to perform fast searches on encrypted data in both deterministic and probabilistic privacy-preserving CDT.

3. Privacy-preserving algorithms for training and evaluating gradient-boosting regression-tree models on encrypted data using BGV fully-homomorphic encryption, for the purpose of privacy-preserving probabilistic CDT.

Separate from our contributions to privacy-preserving CDT, our research has also led us to provide a contribution to the field of searchable encryption in the form of MUSE. MUSE is a multi-user searchable encryption scheme that uses the homomorphic evaluation of AES-CMAC to offer sub-linear complexity for searching over encrypted data. As MUSE is unrelated to our work on privacy-preserving CDT, we offer the details of this scheme in a separate paper, which we append to this thesis in Appendix A.

1.5. THESIS OUTLINE

Section 2 gives an overview of security-related works in the field of cross-device tracking, as well as relevant works in the field of privacy-preserving technologies, on the topics of privacy-preserving behavioural advertisement, searchable encryption, and privacy-preserving data-mining. In section 3, we describe the building blocks used in the construction of the PCDT protocol, namely the underlying cryptographic primitives, the techniques used for cross-device tracking, and the machine learning algorithms, as well as their privacy-preserving counterparts. Section 4 gives a detailed description of the PCDT protocol, the setting in which the protocol operates, and the algorithms that comprise it. In section 5, we analyse the security and performance of our construction, with section 5.1 containing a security analysis of PCDT, and section 5.2 containing both a theoretical performance analysis of the algorithms that comprise PCDT, as well as an empirical analysis based on a C++ implementation of PCDT procedures, constructed using HElib [52]. Finally, section 6 contains a discussion on the properties and capabilities of PCDT, as well as possible future improvements.

In addition to the main body of the thesis, which focuses of privacy-preserving CDT, a paper on a multi-user searchable encryption scheme that makes use of the homomorphic evaluation of AES-CMAC is presented in Appendix A.

2

Related Work

The existing literature that approaches CDT from a privacy-conscious perspective focuses on observing the effects of cross-device tracking on user privacy [13, 16, 20], measure the extent to which CDT technologies are employed on the Internet [13, 16–18], and evaluate the effectiveness of available countermeasures against online trackers [6, 16].

To the best of our knowledge, this thesis constitutes the first study into offering privacy-preserving technological alternatives in order to reduce the impact of CDT practices on the privacy of Internet users. Since no existing prior work explores this specific subject, we draw inspiration from research into other privacy-preserving technologies, primarily from the field of privacy-preserving online advertisement, a field we consider to be closely related to privacy-preserving CDT.

Section 2.1 offers an overview of existing works that study CDT practices and their impact on the privacy of online users, while section 2.2 describes various works on privacy-preserving solutions, specifically in fields of privacy-preserving online advertisement, searchable encryption, and secure machine learning.

2.1. Detecting and preventing CDT

In 2017, the U.S. Federal Trade Commission (FTC) published a report [20] that describes the practice of online cross-device tracking, outlining both the benefits and security concerns posed by the practice, and offering suggestions on how to mitigate the impact of CDT on the privacy of online users. The report offers a broad overview of the techniques involved in both deterministic and probabilistic CDT, and lists the main use cases of CDT in the fields of online targeted advertisement, fraud detection, and account security. The authors also offer insight into the potential security issues posed by CDT practices, particularly the threats to user privacy that emerge from the use of CDT in the field of online behaviour advertisement.

One such issue comes in the form of privacy leaks from online targeted ads appearing on shared devices (e.g. work laptop, family desktop), when the ads target online behaviour conducted on private devices. Such events constitute a significant threat to

user privacy as they can leak sensitive information about a person's financial status, ideological views, sexuality, or medical information [22, 23]. In regards to this issue, FTC recommends that online advertisers refrain from engaging in CDT on sensitive topics such as health, financial status, or children information.

The report also mentions the specific issue this thesis aims to address, namely the large amounts of user data collected and stored by the organisations engaging in CDT practices. As this data contains various types of sensitive information, such as home addresses, location data, and Internet browsing history, it infringes on the privacy of users simply by being available to the organisations that collect it, who may be willing to sell it or share it with additional parties [24]. Additionally, these large collections of user data are also at risk of being leaked and becoming public or falling into the hands of malicious parties [25–28]. To address this issue, FTC recommends that companies only keep data that is necessary for business purposes, and take proper security measures regarding the storage and handling of the data they do keep.

A third security issue identified in the report constitutes the lack of transparency for CDT practices. The authors remark that CDT, and especially probabilistic CDT, is not obvious from a user perspective, and that organisation that engage in CDT practices rarely make this fact apparent to their users. Furthermore, the controls available to users for opting out of being tracked across devices are limited, do not provide full protection, or rely on self-regulating practices. The report bases its claims on a paper published by FTC staff, in which Brookman et al. [13] conduct a study of cross-device tracking practices within the online advertisement ecosystem, and provide insight into the prevalence of these techniques, as well as their transparency from a user perspective.

The study evaluates 100 popular websites for activities that could facilitate cross-device tracking, such as the collection of behavioural data or personally identifiable information, the use of third-party cookies, or known third-party trackers. The paper is concerned with both deterministic and probabilistic CDT, as the study accounts for login sessions, the amount of personally identifiable information collected by each website, as well as device-identifying cookies or device fingerprinting techniques used in probabilistic CDT. The data involved in the study is collected by using two virtual devices to simulate the behaviour of a user browsing multiple websites on two distinct devices. The data is collected over multiple runs in which the virtual devices are used to sign up for various email and social media accounts and simulate both authenticated and unauthenticated users. The results of the study show significant usage of third-party trackers, which coordinate with each other through the use of third-party cookies. The authors identify six organisations that enable user login on their platforms, while also collecting large quantities of data from third-parties across multiple devices. Out of the 100 websites considered by the study, 34 were found to share data used for probabilistic CDT with third-parties, and 16 were found to share personally identifiable information with third parties, which is used in deterministic CDT.

The paper also explores how transparent organisations that engage in CDT are about their practices. By reviewing the privacy policies of the 100 websites involved in the study, the authors find that only three websites specifically mention third-party cross device tracking, and 73 websites reserved broader rights for the usage and sharing of non-personally identifiable information, which is likely used primarily for online be-

havioural advertisement practices, but can also be used for probabilistic CDT. Additionally, 67 websites provided information on how to limit the use of behavioural data for ad targeting, but few mention controls for limiting CDT. Out of 100 websites, only 50 mention the "Do Not Track" header, out of which 22 simply state that they do not honor the setting, while another 26 state that the standard is still being worked out by the industry.

Roesner et al. [16] perform an empirical study of CDT practices by observing the behaviour of online trackers on a sample of 1000 websites. The authors develop a classification framework that categorizes tracker behaviour based on the type of interactions between a user and a tracking service. Based on this framework, the authors design a Firefox add-on called TrackingTracker that automatically detects and classifies trackers on websites visited by the user.

Out of the 500 most popular websites considered for the study, the authors find that 445 of them embed at least one cross-site tracker, with an average of 7 trackers per website. Furthermore, the study counts a total of 524 unique trackers, the most popular of which being Google Analytics, which appears on almost 300 of the 500 domains, and Doubleclick (also owned by Google), which can track users across almost 40% of these 500 most popular websites.

The paper also explores the utility of various controls that users can employ to attempt to limit cross-device tracking. While none of the controls considered in the exploration are able to completely prevent CDT, third-party cookie blocking and disabling JavaScript were found to be the most effective means of thwarting tracking behaviour. Third-party cookie blocking is obviously effective against third-party trackers, but does not protect users against trackers visited directly, i.e. when the tracker service is also the website owner. While disabling JavaScript is effective against trackers that use API calls to set and read cookies, it is circumvented by trackers that control cookies via HTTP headers, and comes with the major downside of rendering a lot of modern websites unusable.

In the same paper, Roesner et al. also introduce a new defense against CDT in the form of a Firefox add-on called ShareMeNot. This add-on combines the various defense techniques considered by the study with a blacklist of known tracker domains to provide an effective measure for countering online tracking.

In [17], Acar et al. introduce FPDetective, a framework for detecting web-based fingerprinting. Device fingerprinting is a fundamental technique in the field of cross-device tracking, since in order to track a user across multiple devices, a tracker must first be able to recognize said device across multiple website visits.

The main forms of fingerprinting studied in the paper are JavaScript-based fingerprinting and plugin-based fingerprinting. JavaScript based fingerprinting uses JavaScript to capture various properties about a device, such as device model, OS, browser version, screen dimensions, available fonts, etc.; which in combination provide sufficient information for a tracker to distinguish between devices. Plugin-based fingerprinting operates in a similar way to its JavaScript counterpart, except that the information used for fingerprinting is collected through plugin applications such as Adobe Flash or Java.

Using FPDetective, Acar et al. perform a crawl of one million popular websites and find a lower bound of 404 websites that use JavaScript-based font-probing to fingerprint devices. Furthermore, the authors identify a total of 13 tracking services that provide the

scripts used to perform this type of device fingerprinting.

In [18], Solomos et al. introduce Talon, a framework for detecting and measuring probabilistic CDT. The detection method involves the use of artificial online personas that mimic user behaviour on various devices, followed by the collection and processing of targeted advertisement for the purpose of detecting CDT.

The artificial users are constructed based on the online behaviour of real-life users and categorized into various experimental setups based on characteristics such as browsing interests, the duration of browsing, or focused vs diverse browsing behaviour.

When performing an experiment, the framework makes use of three devices, out of which two use of the same public IP address to facilitate probabilistic CDT, while the third device acts as control by using a different public IP and mimicking the behaviour of one of the other devices to establish a baseline set of ads which is then compared against the ads received by the paired devices. Various machine learning techniques, such as random forests and linear regression, are used to evaluate the collected advertisement data and infer whether or not the two test devices were linked using probabilistic CDT.

The experiments performed by Solomos et al. measure the effect of multiple factors on the performance of probabilistic CDT. From the results of these experiments, the authors conclude that shared IP addresses play an important role in probabilistic CDT, while other factors such as browsing interests and even the time and duration of online activity also influence CDT. Furthermore, the experiments reveal that browsing in incognito mode reduces the effects of CDT, but does not completely remove them. One specific online activity, namely the browsing of online vendors, was found to significantly increase the likelihood of the two test devices to be linked through CDT, as those organisations are among the most interested in tracking their users, mainly for the purpose of online targeted advertisement.

Merzdovnik et al. [6] conduct a study into the extent to which CDT technologies are employed on the Internet, as well as the effectiveness of existing tools for preventing online tracking. The authors analyse over 100.000 popular websites and more than 10.000 Android applications to examine the techniques employed for CDT, as well as their reach and limitations. The authors note that a small number of companies are responsible for tracking users across the majority of the websites involved in the study. This is a direct consequence of most websites relying on third-party services to perform CDT, usually the same parties involved in serving online advertisement. Merzdovnik et al. also conduct a study into the effectiveness of various tools for preventing CDT, primarily network-based blocking and adblocker browser extensions.

Network-based blocking is performed by blocking devices on a network from accessing certain (sub)domains involved in performing CDT. This method offers the advantage of blocking CDT regardless of device or application, i.e. not just web browsers, but is limited in its degree of finesse. More precisely, if the tracking service uses the same (sub)domain as a target webservice, network-based techniques cannot block the tracker without also denying the user access to the webservice.

Following their analysis into the effectiveness of various browser extensions with track-blocking functionality, Merzdovnik et al. conclude that a small number of such extensions are capable of blocking the majority of tracking services, but none offer complete protection. The authors also note a lack of available tools for preventing mobile

tracking, i.e. tracking users through smartphone applications.

2.2. PRIVACY-PRESERVING SOLUTIONS

In this section we give an overview of related works in the field of privacy-preserving solutions, and focus specifically on the fields of privacy-preserving online advertisement, searchable encryption, and secure machine learning, as they relate to our study.

2.2.1. PRIVACY-PRESERVING ONLINE BEHAVIOURAL ADVERTISEMENT

Toubiana et al. [12] offer a browser-based solution to the privacy concerns involving online behavioural advertisement. The authors introduce Adnostic, a browser extension that runs the behavioral targeting algorithm on the user's browser by processing the browser's history. When a user visits a web page containing an ad slot, the browser extension contacts an advertising network to request a list of multiple advertisements based on the page content. The extension then selects the most relevant advertisement from the downloaded list by comparing the topics associated with each advertisement against the locally-constructed user profile. To report which advertisement was viewed or clicked on by a user, Adnostic uses a voting system based on homomorphic encryption. This way, user-ad interactions are reported to the advertisement network without revealing user interests. While Adnostic reduces the amount of data advertisement networks collect about their users, it still reveals a portion of the users' browsing behaviour by allowing the advertisement network to know the web pages on which ads were requested.

Guha et al. [11] design PrivAd, a privacy-preserving protocol for online behavioural advertisement which uses a semi-honest anonymizing proxy that interfaces between users and the advertisement network. The client software of PrivAd makes advertisement requests to the advertisement network in the form of subscriptions that cover broad interest keywords and broad non-sensitive demographics. The advertisement network replies with sets of advertisements that match the keywords and demographics received from the client. These ads are filtered and stored by the client-side software, and the ones selected as relevant are shown to the user. Communication between the client and the advertisement network is anonymized by the proxy, whereas the contents of the communication are hidden from the proxy by means of public-key cryptography. If a user has multiple interest categories, these are separately reported in a manner that prevents linking interests to the same user.

In [10], Backes et al. introduce ObliviAd, a privacy-preserving protocol for online behavioural advertisement that uses secure hardware-based private information retrieval. ObliviAd makes use of local user profiles that consist of collections of keywords which indicate the categories of ads the user may be interested in. The protocol uses an untrusted broker server, which acts as a matchmaker between multiple advertisers and Web publishers, and uses a secure coprocessor to match ads with the encrypted keywords received from the users. To prevent the broker from knowing which advertisement(s) were retrieved by which user, which would leak information about the user profiles, ObliviAd uses an oblivious RAM protocol to hide the access pattern of the secure coprocessor on the advertisement storage.

To report advertisement feedback, such as views and clicks, ObliviAd attaches an encrypted token signed by the secure coprocessor to each ad. These tokens contain information about their respective ads along with a timestamp. When an user interacts with an ad, it sends the token back to the broker, who uses the secure coprocessor to decrypt the ad and, by mixing it with tokens received from other users, obtains a list of advertisement interactions without knowing which users interacted with which ad. The authors note that a malicious broker could derive information about a user's profile by only allowing the tokens received from that user to reach the secure coprocessor. While anonymous channels would solve this issue, Backes et al. do not recommend their usage due to their associated computational costs and network delays. Instead, ObliviAd assumes that brokers behave *rationally*, and prioritise the monetary gains of processing advertisements over the possibility of associating ads to specific users.

Leon et al. [7–9] offer two protocols for performing privacy-preserving online behavioural advertisement, AHEad and BAdASS, that involve the training and evaluation of a logistic regression machine learning model on encrypted user data.

The first protocol, AHEad, which serves as inspiration for the PCDT protocol presented in this paper, makes use of a semi-trusted non-colluding third party called a Privacy Service Provider (PSP), which assists the Web publishers in performing privacy-preserving computations. The protocol uses a two-party threshold variant of the additively-homomorphic Paillier encryption scheme to encrypt the user data in a way that allows the evaluation and training of the logistic regression model.

Within the setting of the protocol, ad publishers use the regression model to predict a user's response to various ad campaigns, and rely on PSP to compute which advertisement will generate the most income that among the ones associated with a positive response. The advertisement itself is encrypted under the public key of the user in order to keep it private from the PSP.

The logistic regression model is trained on the actual responses made by a user on various ads, labeled as 1 in the case of a click, or 0 in the case of no click. In order to keep the user-ad interactions secret from PSP, the protocol relies on the client-side software to compute the gradient of the loss function used to update the model. Since the gradient values computed by users have to be kept private from PSP in order to protect the secrecy of user data, the model update protocol comes at a significant communication cost on the side of the users.

The second protocol, BAdASS, makes use of additively-homomorphic secret sharing to securely split user information between advertiser brokers. The protocol allows advertisers to collaboratively evaluate and train a logistic regression model used to predict user responses to various ad campaigns. BAdASS also makes use of a hierarchical secure auction protocol to perform advertisement bidding and select the winning ad that is served to the user. By using threshold secret sharing, BAdASS distributes trust, such that collusion between any number of parties smaller than a predefined threshold does not reveal any sensitive information.

BAdASS is shown to be considerably faster than AHEad in terms of ad delivery, and manages to achieve a practical response time in the order of milliseconds. Although model updates are expensive in both protocols, the model update phase of BAdASS is more than 20 times faster than that of AHEad. In terms of communication, both pro-

tocols have considerable bandwidth requirements due to the transmission of complete user profiles and update gradients, but BAdASS requires approximately 20% of the bandwidth usage of AHEad.

2.2.2. SEARCHABLE ENCRYPTION

Searchable encryption (SE) [40, 41] allows a server to search in encrypted data on behalf of a client without learning information about the plaintext data. Some schemes encrypt the data in a manner that allows searches to be performed directly on the ciphertext, while others require the client to generate one or more encrypted indices associated with the data. Most SE schemes require for each document stored on the server to be associated with a set of keywords, which can be used in search queries to retrieve matching documents.

Based on the number of parties that are allowed to store encrypted data on a server, SE schemes are divided into single-writer and multi-writer schemes. Additionally, depending on the amount of parties that can query and retrieve the stored data, SE schemes are divided into single-reader and multi-reader schemes. A multi-writer/multi-reader SE scheme, also called a multi-user SE scheme, matches the functionality of a generic database, since it allows multiple parties to have store and search capabilities.

Most multi-user schemes offer search complexity linear in the total number of stored documents, except for [37]. Most of them use a third-party for user authentication, and sometimes for interactive encryption protocols [40].

Dong et al. [42] propose a protocol which uses an El-Gamal proxy re-encryption scheme in combination with a collision-resistant hash function, in a three-party setting. The protocol employs a fully trusted key-management server, which is separate from the data-storage server, and generates the key pairs used by users to encrypt the search keywords. The key-management server also acts as an access manager that grants and revokes the users' permission to query the data-storage server. The keyword ciphertexts are initially encrypted under each user's secret key, and are then re-encrypted by the server so that they can be matched with queries produced by any user. The protocol offers semantic security under the assumption that the data-storage server is not an authenticated user and cannot generate queries on its own. In terms of efficiency, the storage operation is linear in the amount of keywords associated with a document, while the search operation is linear in the total amount of keyword-document pairs.

Bao et al. [43] propose a scheme which uses bilinear maps to allows users with different secret keys to generate the same search index for a given keyword. This scheme introduces a trusted third-party to manage user credentials, which allows for authenticated search queries and the revocation of a user's permission to query the database. The scheme offers query privacy as well as query unforgeability, i.e. only registered users can query the database and users cannot impersonate each other. In terms of efficiency, the storage operation is linear in the amount of keywords associated with a document, while the search operation is linear in the total amount of documents in the database.

Wang et al. [44] onstruct a protocol which allows users to perform conjunctive search queries on the encrypted data, i.e. a document is retrieved only if it matches all the keywords in the search query. The protocol uses a dynamic accumulator for user authentication and a combinatorial accumulator to build a search index from a padded list of

hashed keywords. It also makes use of a semi-honest third-party which assists the users in decrypting the data retrieved via search queries. The protocol offers semantic security against chosen-keyword attacks. With regards to efficiency, the storage operation is linear in the maximum amount of keywords associated with a document, and the search operation is linear in the total amount of documents.

Bellare et al. [37] propose a multi-writer/multi-reader SE scheme, which makes the encrypted keywords searchable by appending a hash of the keyword to its ciphertext. This makes both the storage and the search operations very efficient, i.e. constant in the total amount of keywords and documents, but lacks semantic security, and the server can derive the keywords from their hashes through reverse lookup table attacks.

2.2.3. PRIVACY-PRESERVING MACHINE LEARNING

In [32], Mohassel and Zhang provide a number of efficient protocols for training linear regression, logistic regression, and neural network models in a privacy-preserving fashion. The authors experiment with a wide range of cryptographic primitives for performing privacy-preserving computations, such as oblivious transfer, garbled circuits, linearly-homomorphic encryption. The presented protocols operate in the two-server model, where data owners distribute the training data between two semi-honest, non-colluding servers who engage in secure two-party computations to train various models on the joint data.

In [30], Agrawal and Srikant present a protocol that allows an untrusted server party to train a decision tree classifier on perturbed data aggregated from multiple sources. First, the data owners apply a randomizing function to perturb the values in their data sets, e.g. Gaussian or Uniform perturbation, such that they cannot be estimated with sufficient precision, and will therefore be kept secret from the training server. The server aggregates perturbed data from multiple sources and uses Bayesian statistical analysis to approximate the distribution of the original data. The perturbed training set is corrected such that its distribution matches that of the original data, and the corrected values, which are not same as the original values, are then used to train the classifier. Through empirical analysis, the authors conclude that the models trained on perturbed data perform similarly to the models trained on the original data, reporting a loss of accuracy bellow 15%.

Lindell and Pinkas [29] design a privacy-preserving decision tree learning protocol that allows two semi-honest parties to train a decision tree model on the union of their data sets without revealing data to each other. The protocol has both parties run the training algorithm locally, on their their respective data sets, and obtain a set of intermediary values which are then combined using oblivious transfer such that both parties obtain the complete model.

Akavia et al. [72] construct a CPA-secure protocol for approximate evaluation and training of decision tree models on homomorphically encrypted data. The protocol uses the CKKS fully-homomorphic encryption scheme for approximate arithmetic, and substitute the comparison operation with a polynomial approximation of a step function.

3

PRELIMINARIES

In this section, we describe the building blocks of our protocol. First, we offer some insight into the technicalities of CDT, mainly on how to construct a device association graph using both deterministic and probabilistic approaches. Then, we describe the machine learning techniques used by PCDT, i.e. gradient boosting decision trees, as well as how they are used on homomorphically encrypted data. Finally, we cover the cryptographic preliminaries used in our construction, namely the BGV fully-homomorphic scheme, the homomorphic computations of AES and AES-CMAC, and several algorithms for performing integer arithmetic on BGV-encrypted data.

3.1. CROSS-DEVICE TRACKING

In this section we give an overview of the techniques used in performing CDT, primarily how to construct a device association graph using both the deterministic and the probabilistic approaches. Additionally, we discuss how these techniques are adapted by our protocol in order to operate on encrypted data.

3.1.1. DEVICE ASSOCIATION GRAPH

In order to track users across devices, an organisation must first identify which devices belong to which users. Since most organisations engaging in CDT only interface with users indirectly, through the devices of said users, the problem CDT techniques aim to address can be more accurately formulated as follows:

> *Given a set of devices, each with an associated collection of data, divide the set into groups such that the devices in each group belong to the same user.*

To this end, organisation engaging in CDT organise the collected device data into a *device association graph* [14, 15]. A device association graph is a undirected graph $G = (ID, E)$, where ID is a set of device identifiers, and E is the set of edges, such that two devices with identifiers id_1 and id_2 respectively, are considered *linked* if and only if $\{id_1, id_2\} \in E$.

Two *linked* devices imply the existence of at least one user who has used both devices at some point in the past. Further analysis is necessary to determine if the devices are personal devices owned by the same user or shared devices used by multiple users (e.g. household computer, library computers, etc.). Based on this, organisations may choose to either ignore shared devices, try to differentiate between the multiple users of a shared device based on behavioural cues, or simply build less specific, shared user profiles which treat a group of users as a singular person.

Two approaches are considered when building device association graphs, namely deterministic and probabilistic CDT. The techniques, as well as our privacy-preserving adaptations, are described in detail in sections 3.1.2 and 3.1.3.

3.1.2. DETERMINISTIC CDT

Deterministic CDT is primarily performed by monitoring the devices used by a client to access their online accounts. When a user logs into a webservice account, e.g. email or social media account, on both their smartphone and their desktop computer, it informs the service provider that both devices belong to the same user. Alternatively, websites can track users even without them signing in, by matching the personally identifiable information they input when signing up for an account or a subscription, shopping online, ordering food, or filling in a questionnaire. This information may include email addresses, home addresses, phone numbers, credit card numbers etc.

To measure the association between two devices based on the personally identifiable information collected for each one, a webservice provider simply has to count the data points shared by both devices. It is worth noting that some matches may be more significant than others, e.g. matching email addresses are more relevant than matching physical addresses, since multiple people can live at the same address. To address this, one can divide the data in multiple categories based on relevance, count the matches in each category, then multiply it by some index of significance to obtain the final association score. The resulting score can then be compared against a threshold value to decide whether or not to add the pair $(id,\ id')$ to the set of edges in the device association graph. A small value is usually chosen for this threshold, as a single match in the right category, e.g. email addresses, is enough to assume that both devices are used by the same user, but a higher value can be chosen to increase precision.

Let Γ_d denote a set of data categories with different degrees of importance, where each category is described by a mapping μ, such that $\mu(id)$ represents the set of data in the category collected from the device with identifier id, and a value γ_d representing the significance of a match in the category, then Equation 3.1 describes how to calculate the association score between two devices with identifiers id and id' using deterministic CDT.

$$s_d(id, id') = \sum_{(\mu_d,\ \gamma_d)\in\Gamma_d} \gamma_d \cdot |\mu_d(id) \cap \mu_d(id')| \tag{3.1}$$

For the sake of simplicity, our protocol only considers the case where $|\Gamma_d| = 1$, i.e. all data matches are assumed to be equally important. However, a protocol for the general case can be easily derived from PCDT.

Two approaches were considered for computing the number of matches in a privacy-preserving way: private set intersection [38, 39] and multi-user searchable encryption [40, 41]. The main issue with either approach lays in the computational complexity that arises from having to match each new encrypted data point against all data points provided by all other devices. As the underlying encryption is assumed to be probabilistic, the time complexity of this matching process is bound to be at least linear in the amount of data points collected for deterministic CDT. Given the large amount of devices serviced by the top organisations engaging in CDT [13], any approach with linear complexity would be considered too slow for large-scale practical use.

To address this issue, following the work of Bellare et al. [37], we adjust our security requirements to allow the use of deterministic encryption, which in turn allows for constant-time search operations.

In [37], Bellare et al. present a multi-user searchable encryption scheme with sub-linear search complexity, which computes the search indices of encrypted documents by hashing the searchable keywords. However, as a result of using deterministic encryption, the scheme lacks semantic security, and is therefore vulnerable to table-lookup attacks. Bellare et al. argue for the security of their scheme, as well as the security of deterministic encryption in general, by introducing the notion of PRIV security, which states that deterministic encryption can be considered secure provided the plaintext space is sufficiently large.

Given the nature of the data used to perform deterministic CDT, it is obvious that the requirement of a sufficiently large plaintext space cannot be met. For example, it is computationally feasible for a server that stores hashed phone numbers or email addresses to match the stored hashes against hashes of large lists of known values and compromise the secrecy of the data. As a result, our construction makes use of a two-server setting and an algorithm for computing a keyed hash function over homomorphically encrypted data, to hide the device data behind hashes computed using a key unknown to the storage server. Since the hash key is kept secret from the storage server, it cannot compute hashes of values of its choice, and therefore cannot conduct a table-lookup attack. This approach allows our protocol to perform privacy-preserving deterministic CDT in constant time, with respect to the amount of collected device data.

In order to conceal the plaintext data from the server computing the hashes, we construct an algorithm for the homomorphic computation of AES-CMAC, and give a pseudocode representation in Pseudocode 4, section 4.3.1. This algorithm is employed when storing device data in preparation of performing deterministic CDT, see Pseudocode 1 in section 4.2.2. Finally, procedure TS:UPDATE-DET (Pseudocode 2, section 4.2.3) shows how these hashes are used in PCDT to update a device association graph by performing privacy-preserving deterministic CDT.

3.1.3. PROBABILISTIC CDT

Probabilistic CDT is performed by using statistical analysis on data inferred from the device or communication channel used to access the webservice. This data includes IP addresses, location data, browsing history, temporal information such as location or IP addresses over time, and terminal information such as browser version, window dimensions, operating system, device model, screen dimensions, etc. IP addresses are given

particular importance, as two devices sharing an IP address is the primary probabilistic indicator that they might belong to the same user [64, 65, 69].

In order to perform probabilistic CDT, a webservice provider must first construct a model for predicting the likelihood of two devices belonging to the same user. This model is often constructed using machine learning techniques [70]. The machine learning algorithm selected for our protocol is gradient-boosting decision trees (GBDT), chosen based on existing research [69, 70] and its performance in data-analysis competitions focusing on CDT [64–68].

The main limitation of using GBDT to perform cross-device tracking stems from the difficulty of constructing reliable training data sets. Since GBDT is a supervised learning algorithm, the training algorithm used for model construction requires labeled data, where each training data point has an associated target value which indicates whether or not the data point corresponds to a pair devices that share a user. Unlike the case of online behavioural advertisement, where users provide feedback for the machine learning models in the form of *clicks* [7], CDT techniques receive no confirmation on whether or not a device was linked to the appropriate user profile, which makes the construction of training data sets particularly challenging.

To address this issue, organisations engaging in CDT rely on the performance of deterministic techniques, which can be used to provide feedback for probabilistic techniques [70]. More precisely, when training a machine learning model to perform probabilistic CDT, the training set can be constructed by using the output of deterministic CDT techniques to label the data. This is exactly the approach used in PCDT, where a GBDT regression model is trained on data labeled with the scores obtained from performing deterministic CDT.

Another challenge posed by probabilistic CDT comes from the computational complexity involved in the task. Since the machine learning model is designed to evaluate one pair of devices at a time, the resulting complexity of both constructing a device association graph and training a model becomes quadratic in the number of devices involved in the analysis. To address this issue, PCDT uses a down-sampling method inspired from [66–68], where a pair is only considered if the two devices share an IP address among those used withing a given time-frame. This way, PCDT only considers device pairs that are already likely to share a user, both when performing probabilistic CDT or constructing the GBDT model.

In order to perform probabilistic CDT in a privacy-preserving manner, our protocol uses homomorphic encryption to conceal the device data, while allowing the computations necessary for training and evaluating GBDT models to be performed in ciphertext space. As a result, PCDT uses special algorithms for training and evaluating GBDT models on homomorphically encrypted data, see Pseudocode 13 in section 4.3.10, and Pseudocode 14 in section 4.3.11. Pseudocode 1 in section 4.2.2 details the steps taken in the collection of device data, such that the plaintext data is concealed from the tracking services without impeding the performance of probabilistic CDT. Procedure TS:UPDATE-PROB of Pseudocode 2, section 4.2.3, shows how probabilistic CDT is performed on encrypted data, while Pseudocode 3 in section 4.2.4 shows how encrypted device data is used to train a GBDT model to perform probabilistic CDT.

3.2. MACHINE LEARNING ALGORITHMS

In this section we describe the machine learning algorithms employed by our protocol, namely the training and evaluation of gradient boosting decision trees. We also provide insight into how these algorithms can be adapted to operate on encrypted data through the use of fully-homomorphic encryption.

3.2.1. GRADIENT BOOSTING

Gradient boosting [57] is a machine learning technique used for regression and classification. The prediction model produced by this method takes the form of an initial prediction value p_0, a learning rate γ, and an ensemble of n weaker prediction models $T_{i\in\{1,\ ...,\ n\}}$ trained to predict residual values. A residual, as shown in Equation 3.4 is the difference between the target value and the current prediction. The main idea behind gradient boosting is to combine a group of weaker models so that each predictor makes a small additive contribution to the overall prediction. This way, the overall model becomes more powerful as the number of weak predictors increases.

Let x be a vector of m feature vectors $x_{j\in\{1,\ ...,\ m\}}$, and y be a vector of target values $y_{j\in\{1,\ ...,\ m\}}$, such that y_j is the target value associated with x_j for all $j \in \{1,\ ...,\ m\}$. When training a gradient boosting decision tree (GBDT) model, one must first compute the initial prediction value, which is equal to the average of the target values, as shown in Equation 3.2. The decision trees are then trained iteratively, where each tree T_i, with $i \in \{1,\ ...,\ n\}$, is trained on a data set composed of feature vectors $x_{j\in\{1,\ ...,\ m\}}$ that uses as target values the residuals $r_{i-1,j\in\{1,\ ...,\ m\}}$ produced by the previous trees. The residual values are computed based on the predictions made by the model so far, as shown in Equation 3.4, while the prediction themselves are computed in accordance with Equation 3.3, where $T_i(x_j)$ represents the vector of predictions made by the decision tree T_i on input vector x_j, and γ is the learning rate. The learning rate γ is a constant used to control the speed at which the model predictions approach the target values, in order to reduce the likelihood of overfitting.

In order to evaluate the prediction y of a GBDT model $(p_0,\ \gamma,\ T_{i\in\{1,\ ...,\ n\}})$ on input x, one must simply evaluate $p_{i,j}$ as given in Equation 3.3, with $x_j = x$, and $p_{0,j} = p_0$.

$$p_{0,j} = \frac{\sum_{k=1}^{m} y_k}{m},\ \forall j \in \{1,\ ...,\ m\} \tag{3.2}$$

$$p_{i,j} = p_{i-1,j} + \gamma \cdot T_i(x_j) \tag{3.3}$$

$$r_{i,j} = y_j - p_{i,j} \tag{3.4}$$

The PCDT protocol provides an algorithm for training a GBDT model on homomorphically encrypted data. This algorithm is described in detail in section 4.3.11, and a pseudocode representation is given in Pseudocode 14. Since the target values are public, the average value is computed in plaintext, while the homomorphic computations only concern the private training of the underlying decision tree models. Similarly, the model evaluation is performed by homomorphically computing the predictions of the decision tree ensemble, which are decrypted and used to compute the prediction of the

overall model. The evaluation algorithm is described in detail in section 4.3.10, and the pseudocode representation is given in Pseudocode 13.

3.2.2. DECISION TREES

A decision tree is a predictive model used in the fields of statistics, data mining, and machine learning, to perform regression and classification [59–61]. A decision tree is composed of a series of nodes arranged in the structure of a complete binary tree, where each node t has two child nodes $t.left$ and $t.right$, except for leaf nodes. Each non-leaf node is also associated with a feature index $t.feature$ and a threshold value $t.\theta$, used to partition the input space along one dimension.

When evaluating a decision tree on a feature vector x, the evaluation path starts with the root node, and each non-leaf node t in the path decides which of its two children is the next node in the path based on a comparison between $x[t.feature]$ and $t.\theta$.

Leaf nodes have no children and are described by a value $t.value$. The evaluation of a decision tree model ends when the evaluation path reaches a leaf node, and the result of the evaluation is the value stored in said leaf node. This value can be a class label if the model is used for classification, or a value from a continuous range of numbers if the model performs regression.

Let x be a vector of m feature vectors $x_{j\in\{1,\ ...,\ m\}}$, and y be a vector of target values $y_{j\in\{1,\ ...,\ m\}}$, such that y_j is the target value associated with x_j for all $j \in \{1,\ ...,\ m\}$. The training algorithm for decision tree models aims to produce a model T such that its evaluation on input x_j is equal to y_j for as many training data points as possible. Ideally, the algorithm would output a model such that $\forall\, j \in \{1,\ ...,\ m\}\ T(x_j) = y_j$, but this task is known to be NP-complete [62], and even if an efficient algorithm that produces the ideal model would exist, the model would suffer from overfitting [63]. Instead, the training algorithm uses a greedy approach, which optimizes the local quality of each node, such that the chosen pair of feature index and threshold partitions the data as *cleanly* as possible. For example, when training a classifier on a training set with two classes A and B, an ideal node splits the data such that one partition contains all the points labeled as A, and the other partition contains all the nodes labeled as B.

Different approaches are used when measuring the quality of a node for either regression or classification. When training a classifier, metrics like Gini impurity, goodness, or entropy [60], are used to measure the quality of non-leaf nodes based on the class distributions they produce. When training a regression tree, the quality of nodes is decided based on the mean deviation of each partition [60]. More specifically, regression trees are trained to minimise the mean squared error (MSE) between the target values and the mean target value in both partitions produced by each node. Given a vector of m target values $y_{j\in\{1,\ ...,\ m\}}$, the formula for computing the MSE used by the training algorithm is given by Equation 3.5. When training a node, the algorithm iterates over a set of combinations of feature indices and threshold values, and selects the pair which produces the minimum MSE.

$$\texttt{MSE} = \frac{1}{m}\sum_{j=1}^{m}(y_j - \overline{y})^2 \quad , \ \texttt{where} \ \ \overline{y} = \frac{1}{m}\sum_{j=1}^{m} y_j \tag{3.5}$$

$$cmp(x,y) = \begin{cases} 1 & x \leq y \\ 0 & x > y \end{cases} \tag{3.6}$$

$$t(x) = \begin{cases} t.value & t \texttt{ is a leaf node} \\ cmp(x[t.feature], t.theta) \cdot t.left(x) & t \texttt{ is not a leaf node} \\ \quad + (1 - cmp(x[t.feature], t.theta)) \cdot t.right(x) & \end{cases} \tag{3.7}$$

Private evaluation of decision trees [71–73] makes use of the evaluation function given in Equation 3.7. A fully-homomorphic encryption scheme that allows for private comparison, multiplication, and addition operations can be used to evaluate a decision tree over encrypted data. In [71, 72], approximate evaluation algorithms are used for private decision tree evaluation. These algorithms make use of CKKS, a fully-homomorphic encryption scheme for approximate arithmetic, and substitute the comparison operation with a polynomial approximation of a step function. Our evaluation algorithm is inspired from these works, but uses the BGV fully-homomorphic encryption scheme and the private comparison function proposed by Cheon et al. [47] to obtain exact results. The algorithm used by PCDT to homomorphically evaluate decision trees is described in detail in section 4.3.12, and a pseudocode representation is given in Pseudocode 15.

Note that the comparison function used to divide the data between left and right child-nodes is arbitrary, and either operation from the set $\{<, \leq, >, \geq\}$ can be used, provided that it is used consistently across nodes and in both the training and the evaluation procedures. To simplify the homomorphic circuit, our construction uses the less-than operation in the homomorphic evaluation and training of decision trees.

Akavia et al. [72] offer an algorithm for training decision tree classification models on homomorphically encrypted data. Same as with their evaluation algorithm, the training algorithm uses the CKKS fully-homomorphic encryption scheme and an approximate step-function for comparison. Again, our protocol uses the BGV fully-homomorphic scheme instead, as well as an exact comparison algorithm. Since we want our algorithm to perform regression instead of classification, we also change the function used to measure the quality of each node, from the Gini impurity to MSE.

Another important aspect of the original algorithm, which we preserve in our construction, is the use of a set of thresholds S_θ supplied as input, from which threshold values are selected during the training procedure. In plaintext decision tree training, threshold values are selected for each feature from the range of values in the training set. When training on encrypted data, the range in the training set is unknown, so threshold values have to be selected from the entire range of feature values. To address this issue, Akavia et al. limit the range of feature values to the interval $[-1, 1]$, and prepare the threshold set $S_\theta = \{0.05 \cdot i \mid i \in \mathbb{Z} \texttt{ and } -20 < i < 20\}$. Our algorithm uses a similar, albeit more generic approach, adapted to accommodate integer values. The algorithm used by PCDT to homomorphically evaluate decision trees is described in detail in section 4.3.13, and a pseudocode representation is given in Pseudocode 16.

3.3. CRYPTOGRAPHIC PRIMITIVES

In this section we cover the cryptographic primitives used by PCDT, namely the BGV fully-homomorphic scheme, the homomorphic computations of AES and AES-CMAC, and several algorithms for performing integer arithmetic on BGV-encrypted data.

3.3.1. BGV HOMOMORPHIC ENCRYPTION

BGV [34] is an asymmetric non-deterministic fully-homomorphic encryption scheme based on the ring learning with errors problem (RLWE). Both ciphertexts and secret keys are vectors over a polynomial ring $R = \mathbb{Z}[X]/\Phi_m(X)$, which represents the ring of integers over the m-th cyclotomic number field. The plaintext space is the space of polynomials $R_p = R/pR = \mathbb{Z}[X]/(\Phi_m(X), p)$, for some fixed $p \geq 2$, which represents the set of integer polynomial of degree up to $\phi(m) - 1$, reduced modulo p. The plaintext space of binary polynomials R_2 is of particular interest, as it allows for the evaluation of binary circuits.

BGV is fully homomorphic, allowing the computation of both addition and multiplication on encrypted data. Given two ciphertexts $[x]$ and $[y]$, encoding ring elements x and y, a party can compute both $[x+y]$ and $[xy]$ without knowing x or y. At any point in the homomorphic evaluation, there is a current secret key s under which the ciphertext is valid, and a current modulus q, both of which change as the homomorphic evaluation progresses. The polynomial $[\langle c, s\rangle \mod \Phi_m(X)]_q$, obtained from computing the inner product over R_q between the ciphertext vector c and the current secret key s, where q is the current modulus, represents the noise of the ciphertext c. A ciphertext c is *valid* with respect to key s and modulus q if the magnitude of its noise is sufficiently small relative to q, so that it does not wrap around q when performing homomorphic operations.

Homomorphic addition has no effect on the current modulus or key, and the noise magnitude is at most the sum of noises of the arguments. Homomorphic multiplication does not change the current modulus, but it does change the key. If the two input ciphertexts are valid under an n-dimension key s, then the output ciphertext is valid under the n^2-dimension key equal to the tensor product of s with itself. The scheme offers a *key switching* operation, which is used to reduce the size of the key after a multiplication. With respect to noise, the multiplication operation will produce a ciphertext with a noise magnitude equal to the product of the noises of the arguments. Since this may cause the noise to grow too large, the *modulus switching* operation is used before each multiplication. The scheme is instantiated with a list of L moduli, $q_0 < q_1 < ... < q_{L-1}$, where L is the depth of the homomorphic circuit to be evaluated. After a ciphertext reaches the last modulus in the list, it can no longer be used for homomorphic computations, except by using bootstrapping.

Smart and Vercauteren show in [35] that the structure of the BGV plaintext space allows for the evaluation of single instructions, multiple data (SIMD) operations. By setting m odd, the plaintext space R_2 is isomorphic to the direct sum of l copies of $\mathrm{GF}(2^d)$, where $l = \phi(m)/d$. This way, plaintexts can be treated as l-vectors of elements of $\mathrm{GF}(2^d)$, where arithmetic operations over plaintexts corresponds to element-wise operations over l-vectors. The elements of these l-vectors are called slots. In [36], Gentry et al. show how to permute the contents of these slots through the use of automorphisms over R_2, as well as how to raise slot contents to powers of 2 through the use of Frobenius automorphisms.

3.3.2. ADVANCED ENCRYPTION STANDARD (AES)

The AES [45] block cipher is a substitution-permutation network which symmetrically encrypts 128-bit messages. AES-128 is a specification of the AES cipher which uses 128-bit keys. The encryption operation of AES-128 consists of 10 rounds operating on 128-bit blocks of data organised as a 4×4 matrix of bytes. The bytes are viewed as elements of $\mathrm{GF}(2^8)$ defined by the polynomial $x^8+x^4+x^3+x+1$. Each round consists of the following operations, except for the last round which does not use MixColumns:

1. **AddRoundKey**: XOR the data with the 128-bit round key derived from the secret key.
2. **SubBytes**: compute the multiplicative inverse of each byte over $\mathrm{GF}(2^8)$ and apply an invertible affine transformation over $\mathrm{GF}(2)$.
3. **ShiftRows**: using the matrix representation of the data, right-shift the elements of each row i by i positions.
4. **MixColumns**: left-multiply the state matrix by a fixed matrix of the same dimensions.

Since all operations performed by the encryption function are reversible, to decrypt an AES-128 ciphertext, simply perform the inverse operations in the reverse order as in the encryption.

In [50], Gentry et al. describe how to modify the operations of AES-128 in order to evaluate the cipher over BGV-encrypted data. Their approach uses packed ciphertexts, which encode one or more 128-bit blocks of data and are operated on using automorphisms and SIMD operations.

The scheme is initialized such that the plaintext space allows operations over the finite field $\mathbb{F}_{2^8}$. The polynomial Φ_m is chosen so that it factors modulo 2 into at least 16 irreducible polynomials of degree d, with d divisible by 8. This results in each slot holding one byte of the data, and the number of slots being large enough to store an entire block. Additionally, m is chosen so that there exists an element $g \in \mathbb{Z}_m^*$ that has order 16 in both $\mathbb{Z}_m^*$ and the quotient group $\mathbb{Z}_m^*/\langle 2 \rangle$. This way, if the 16 bytes of the state block are placed in slots $t, t \cdot g, ..., t \cdot g^{15}$, for some $t \in \mathbb{Z}_m^*$, then the conjugation operation $X \to X^g$ implements a cyclic right-shift over these sixteen bytes. The 4x4 state matrix is encoded by placing the 16 bytes into slots in column-ordering, i.e. for each row i and column j, byte α_{ij} is stored in slot $t \cdot g^{4 \cdot j+i}$.

The **AddRoundKey** operation is performed through a homomorphic addition between the state and key ciphertexts.

The **SubBytes** operation is implemented using Frobenius automorphisms, $X \to X^{2^j}$. For a power of two $k = 2^j$, the transformation $\kappa_k(a(X)) = (a(X^k) \mod \Phi_m(X))$ is applied separately to each slot, and can be used to transform the byte vector $(\alpha_i)_{i=1}^l$ into $(\alpha_i^k)_{i=1}^l$. The inversion over $\mathbb{F}_{2^8}$, i.e. $\alpha^{-1} = \alpha^{254}$, is computed using three Frobenius automorphisms and four multiplications arranged in a depth-3 circuit. The affine transformation T_{AES} over $\mathbb{F}_2$ is converted to an affine transformation over $\mathbb{F}_{2^8}$ by computing the constants $\gamma_0, \gamma_1, ..., \gamma_7, \delta \in \mathbb{F}_{2^8}$, such that $T_{AES}(\alpha^{-1}) = \delta + \sum_{j=0}^{7} \gamma_j \cdot (\alpha^{-1})^{2^j}$ over $\mathbb{F}_{2^8}$.

The affine transformation is applied using Frobenius automorphisms to compute the ciphertexts encrypting the polynomials κ_{2^j} for $j = 0, ..., 7$, and applying the appropriate linear combination with coefficients γ_j for $j = 0, ..., 7$, to get the encryption of the vector $(T_{AES}(\alpha^{-1}))_{i=1}^{l}$.

The **ShiftRows** and **MixColumns** operations are combined into a single linear transformation over vectors from $(\mathbb{F}_{2^8})^{16}$. Due to the choice of parameter m and the placement of state bytes into plaintext slots, the operation $X \to X^{g^i}$ can be used to right-rotate row i of the AES state matrix by i positions. By combining these rotation operations with select operations, four ciphertexts are computed, each encoding the appropriate permutation of each row. These four ciphertexts are then combined via a linear operation with coefficients X, $(1 + X)$, and 1, to obtain the final state of the round.

Gentry et al. estimate the depth of the homomorphic AES circuit to be of 4 levels per round, for a total depth of 40 levels for the homomorphic computation of AES-128.

3.3.3. AES-CMAC

The Cipher-based Message Authentication Code (CMAC) algorithm is a keyed hash function based on a symmetric-key block cipher. The AES-CMAC [46] algorithm is a specification of CMAC that uses AES as its underlying block cipher. AES-CMAC takes as input a 128-bit key K and an arbitrary-length message m. Using K, two derived keys K_1 and K_2 are computed. The message m is divided into n blocks of 128 bits, denoted as m_i, where $i \in \{1, ..., n\}$. If necessary, 10^*-padding is used so that the bit-length of m is a multiple of 128. Let $\texttt{AES}(x, K)$ be the output of AES-128 on block x and key K, and let $K_x = K_2$ if the message m was padded, and $K_x = K_1$ otherwise, then the 128-bit output of AES-CMAC is computed as shown in Equation 3.8, where $h_{i \in \{1, ..., n-1\}}$ represent the intermediary hashes computed in accordance with Equations 3.9 and 3.10.

$$\texttt{AES-CMAC}(m,\ K,\ K_x) = \texttt{AES}(h_{n-1} \oplus m_n \oplus K_x,\ K) \tag{3.8}$$

$$h_1 = \texttt{AES}(m_1,\ K) \tag{3.9}$$

$$h_i = \texttt{AES}(h_{i-1} \oplus m_i,\ K) \tag{3.10}$$

Building on the homomorphic evaluation of AES-128, we implement the homomorphic evaluation of AES-CMAC and give a detailed description of the algorithm and its pseudocode in section 4.3.1.

3.3.4. INTEGER OPERATIONS OVER HOMOMORPHIC CIPHERTEXT SPACE

The BGV scheme allows the computation of modular addition and multiplication operations on encrypted integers. However, PCDT involves the training and evaluation of gradient-boosting decision tree models, which require more complex operations over ciphertext space, such as comparison and division. To compute these operations, we restrict the plaintext space to R_2, which allows us to use boolean circuits to compute arbitrary functions on bit-wise encrypted integers. More precisely, using R_2 as plaintext space allows us to compute boolean operations over ciphertext data as follows:

- homomorphic addition is equivalent to logical **XOR**
- homomorphic multiplication is equivalent to logical **AND**
- homomorphic addition by 1 is equivalent to logical **NOT**

Integer representation. The algorithms used in this paper to perform homomorphic arithmetic operations over integers follow the intuition of Cheon et al. [47], who represent integers using packed ciphertexts over binary plaintext and operate on them using SIMD operations and automorphisms, mainly rotations over the array of plaintext slots. In this representation, each plaintext slot encodes one bit of the binary representation of an integer, with one ciphertext encrypting all bits of an integer.

Let N_s be the number of plaintext slots of an initialized BGV scheme, and L_b be the desired bit-length of an integer encoding. Since the integer arithmetic algorithms are at least linear in L_b, it quickly becomes unfeasible for N_s and L_b to be equal, as N_s increases with the security level of the scheme. As a result, we expect L_b to be considerably smaller than N_s, which causes rotations over plaintext slots to no longer coincide with rotations over integer bits. To solve this, the scheme is initialized such that $L_b | N_s$, and the plaintext slots are arranged in an $L_b \times (N_s/L_b)$-matrix using the 2-dimensional representation [52]. This way, a ciphertext can encode multiple integers, one in each row, to allow parallel computation, and rotations over the 1^{st} dimension correspond to rotations over the bits of each integer.

Integers are encoded in binary notation using two's complement to allow arithmetic operations over signed numbers, with the bits arranged from left to right starting with the most significant bit.

Homomorphic Less-than Operation. To evaluate decision trees over homomorphically encrypted data, the PCDT protocol requires an algorithm that can evaluate a comparison function over the homomorphic ciphertext space. In [47], Cheon et al. propose an algorithm for computing the *less-than* function over bit-wise homomorphically encrypted integers.

Let $x = (x_{L_b}, \ldots, x_1)$ and $y = (y_{L_b}, \ldots, y_1)$, be two integers and their binary representations, then the function $lt(x, y)$, as given in Equation 3.11 evaluates to 1 if $x < y$, and 0 otherwise.

$$lt(x,y) = x_{L_b} + y_{L_b} + (1 + x_{L_b}) \cdot y_{L_b} + \sum_{i=1}^{L_b - 1} ((1 + x_i) \cdot y_i \cdot \prod_{j=i+1}^{L_b} (1 + x_j + y_j)) \pmod 2 \tag{3.11}$$

The main insight of Cheon et al. is the use of SIMD operations and rotations to evaluate lt by performing only $2L_b - 2$ homomorphic multiplications and using a homomorphic circuit of depth $\log(L_b)$. For example, $(1 + x_i) \cdot y_i$ can be computed for all i using a single homomorphic addition and a single multiplication, and $\prod_{j=i+1}^{L_b}(1 + x_j + y_j)$ can be computed for all i through $L_b - 1$ rotations and $2L_b - 4$ multiplications. A pseudocode representation of the algorithm is given in Pseudocode 8 in section 4.3.5.

Homomorphic Addition Operation. In [47], Cheon et al. also propose an algorithm that performs integer addition over homomorphically encrypted bit-strings. The addition algorithm is similar to the comparison one, using packed ciphertexts and leveraging SIMD operations and automorphisms to speed up computation.

Given two integers in binary encoding, $x = (x_{L_b}, ..., x_1)$ and $y = (y_{L_b}, ..., y_1)$, the bits of the sum value $s = (s_{L_b}, ..., s_1)$ can be calculated as shown in Equation 3.12.

$$s_i = x_i + y_i + \sum_{j=0}^{i-1} (x_j \cdot y_j \cdot \prod_{k=j+1}^{i-1} (x_k + y_j)) \ (\mathtt{mod} 2) \tag{3.12}$$

The original algorithm given by Cheon et al. can compute s by performing $3L_b - 5$ homomorphic multiplications, using a circuit of depth $\log(L_b - 2) + 1$. Hou et al. [48] propose an improved algorithm which computes s with only $3L_b/2 - 4$ homomorphic multiplications while maintaining logarithmic circuit depth. A pseudocode of this algorithm is given in section 4.3.6, Pseudocode 9.

Based on this addition algorithm, a homomorphic subtraction algorithm (Pseudocode 10 in section 4.3.7) and a multiplication algorithm (Pseudocode 11 in section 4.3.8) can be easily derived.

Homomorphic Division Operation. An integer division algorithm that operates on encrypted data is required to calculate averages in the process of training decision trees on encrypted data.

In [49], Chen et al. adapt the non-restoring division algorithm to work on BGV-encrypted data. Since their algorithm uses a bit-sliced integer representation, where bits are encrypted in separate ciphertexts, we adapt it to use packed ciphertexts in order to use it in our construction. Pseudocode 12 in section 4.3.9 offers a pseudocode representation of this division algorithm that uses the same integer encoding as the other algorithms mentioned in this section.

4

PRIVACY-PRESERVING CROSS-DEVICE TRACKING (PCDT)

In this section, the PCDT protocol for performing cross-device tracking in a privacy-preserving manner is presented in detail. The main goal of PCDT is to produce a device association graph that links devices based on the likelihood that they belong to the same user, while keeping the user data collected for this task secret from all parties involved in the protocol. To this end, the protocol involves the privacy-preserving aggregation of data from various devices, the training of a GBDT model for performing probabilistic CDT on encrypted data, and the construction of a device association graph in a privacy-preserving manner, using both deterministic and probabilistic techniques.

Section 4.1 offers an overview of the parties involved in the protocol, including the assumptions made about their behaviours and interactions. Section 4.2 gives a detailed description of the main algorithms that comprise the PCDT protocol, while section 4.3 describes the auxiliary algorithms used for various privacy-preserving operations.

The following tables present the meaning of various symbols and notations used when describing the algorithms of PCDT. Table 4.1 gives a list of notations used to represent cryptographic operations, such as encryption, decryption, and homomorphic computation, as well as the list of keys used by the various cryptographic schemes employed by the protocol. Table 4.2 offers a list of mathematical notations used throughout the section, along with various constants and important variables used by the protocol. Finally, Table 4.3 describes the constants and symbols used by the machine learning algorithms employed by PCDT.

Table 4.1: Cryptographic symbols and notations

Notation	Meaning
$[x]_{pk}$	The encryption of x using the encryption key pk under the corresponding scheme; element-wise if x is a vector or a set.
$Dec([x]_{pk}, sk)$	The decryption of $[x]_{pk}$ using the decryption key sk, returns the plaintext value x.
$pk_{BGV-AES}$	The public key of the BGV scheme used in the homomorphic evaluation of AES.
$sk_{BGV-AES}$	The secret key of the BGV scheme used in the homomorphic evaluation of AES.
pk_{BGV-IA}	The public key of the BGV scheme used for homomorphic integer arithmetic.
sk_{BGV-IA}	The secret key of the BGV scheme used for homomorphic integer arithmetic.
K_{AES}	The secret hash key used by PS.
K_{CMAC1}	The *no-pad* key derived from K_{AES}, used to compute AES-CMAC hashes.
K_{CMAC2}	The *with-pad* key derived from K_{AES},used to compute AES-CMAC hashes.
ψ_n	Automorphism performing a right-rotation by n slots over each segment of L_b slots in a packed BGV-IA ciphertext.
$[x \oplus y]_{pk_{BGV-IA}}$	Homomorphic addition of ciphertexts encrypting bitstrings x and y under the BGV-IA scheme.
$[x \wedge y]_{pk_{BGV-IA}}$	Homomorphic multiplication of ciphertexts encrypting bitstrings x and y under the BGV-IA scheme.
$[x + y]_{pk_{BGV-IA}}$	Homomorphic integer addition, equivalent to computing HE-ADD($[x]_{pk_{BGV-IA}}$, $[y]_{pk_{BGV-IA}}$) (Pseudocode 9, section 4.3.6).
$[x - y]_{pk_{BGV-IA}}$	Homomorphic integer subtraction, equivalent to computing HE-SUBTRACT($[x]_{pk_{BGV-IA}}$, $[y]_{pk_{BGV-IA}}$) (Pseudocode 10, section 4.3.7).
$[x \cdot y]_{pk_{BGV-IA}}$	Homomorphic integer multiplication, equivalent to computing HE-MULTIPLY($[x]_{pk_{BGV-IA}}$, $[y]_{pk_{BGV-IA}}$) (Pseudocode 11, section 4.3.8).
HE-AES $([m]_{pk_{BGV-AES}}, K_{AES})$	Homomorphic computation of AES as described in section 3.3.2, on input m and key K_{AES}, equivalent to $[\text{AES}(m, K_{AES})]_{pk_{BGV-AES}}$.
AES-CMAC $(m, K_{AES}, K_{CMAC1}, K_{CMAC2})$	Plaintext computation of AES-CMAC, as described in section 3.3.3.

Table 4.2: Important symbols, constants and mathematical notations

Notation	Meaning
b^x	The bit string consisting of x repetitions of $b \in \{0, 1\}$.
$\|x\|$	The number of elements in x, where x is a vector, set, or string.
$\|\|$	Concatenation operator for strings and vectors.
$pad10^*(x, L)$	$x\|\|10^{L-\|x\|-1}$.
()	Empty vector.
$(x_i)_{i=1}^n$	A vector of n elements $(x_1, ..., x_n)$.
x_d	A set of strings used for deterministic CDT.
x_p	A feature vector used for probabilistic CDT.
L_d	The length of the strings in x_d after padding.
L_p	The number of elements in x_p.
L_b	The bit-length of integers used for homomorphic integer arithmetic.
ϵ	The chosen precision for simulating rational number arithmetic using integer arithmetic.
ID	The set of device identifiers.
E	The set of edges in the device association graph.
G	The device association graph (ID, E).
θ_{IP}	The minimum amount of time after which an IP address is no longer relevant for a device.
θ_E	The minimum deterministic or probabilistic association score for a pair of devices to be connected in the device association graph.
τ	Current time as integer value.
μ_d	Maps each device identifier to the set of all hashes used for deterministic CDT received from the associated device.
μ_d^{-1}	The inverse of μ_d, mapping each hash to the set of devices which share the hash.
μ_p	Maps each device identifier to the latest encrypted x_p received from the associated device.
μ_{IP}	Maps each device identifier to the set of all hashed IP addresses from which the corresponding device has sent updates.
μ_{IP}^{-1}	The inverse of μ_{IP}, mapping each IP address hash to the set of devices which sent updates from that address.
σ_{IP}	Maps each pair of device identifier and IP address hash to the latest time when the corresponding device has sent an update from that address.

Table 4.3: Machine-learning symbols and notations

Notation	Meaning
θ_p	The bound on the values in x_p, i.e. if $x_p = (x_{p_i})_{i=1}^{L_p}$, then $\forall i \in \{1, ..., L_p\}: x_{p_i} \in [-\theta_p, \theta_p]$.
Δ_p	The difference between two consecutive thresholds selected from $[-\theta_p, \theta_p]$.
S_θ	The set of thresholds used to train decision tree models, $S_\theta = \{i \cdot \Delta_p \mid -\frac{\theta_p}{\Delta_p} < i < \frac{\theta_p}{\Delta_p}\}$
A	Average value used by the GBDT model.
n	The number of decision trees in the GBDT model.
d_{max}	The depth of each decision tree in the GBDT model.
γ	The learning rate of the GBDT model.
$t.feature$	Index of the feature checked by tree node t.
$t.\theta$	Threshold value used by tree node t.
$t.right$	Right child node of tree node t.
$t.left$	Left child node of tree node t.
$t.value$	Value stored in leaf node t.

4.1. SETTING

The setting in which PCDT operates contains two parties of interest, the tracking service (**TS**) and the privacy service (**PS**), as well as any amount of devices ($\mathbf{D_{id}}$) to be tracked, identified by their unique ids.

$\mathbf{D_{id}}$ provides encrypted data to TS, so that TS can link $\mathtt{D}_{id}$ to other devices and construct a device-association graph. The devices do not interact with each other in any stage of the protocol, nor does any operation involving a device require other devices to be online.

TS engages with PS in interactive privacy-preserving computations in order to construct a device association graph and train a GBDT model on the encrypted information collected from the devices.

PS acts as a middleman between the device and the TS, to allow the device to conceal its IP, and assists TS in performing data analysis on encrypted device data. The purpose of PS is to ensure that no party besides the original owner of the data, not even PS itself, gains knowledge about the plain-text device data.

These parties are assumed to be "honest-but-curious", meaning that they follow the protocol but will derive as much knowledge as possible from the data presented to them. All communication is assumed to be performed over secure channels.

The reason for using a two-server setting stems from the need of privacy-preserving computations involving data aggregated from multiple sources. In the case of a single source of data, homomorphic encryption is sufficient to allow a server to perform privacy-preserving computations and allow the data owner to learn and potentially share the result. In the case of our protocol however, the result is dependent on data from multiple sources, and other techniques have to be employed. One such technique would be

the use of an interactive algorithm in which all data owners participate in the computation [9, 33]. We consider this approach infeasible, as it would require all participating devices to be online every time TS updates the device association graph, or the GBDT model. Instead, our protocol uses a similar approach as [8, 32], and divides the privacy-preserving computations between two non-colluding server parties, TS and PS. This way, the algorithm for updating the profile of a device is the only algorithm in which devices are involved, and only one at a time.

4.2. PROTOCOL ALGORITHMS

This section describes the four main algorithms that comprise the PCDT protocol:

1. **Setup**: initializes the underlying cryptographic schemes.
2. **Update device profile**: allows a device to update the data used by TS to perform CDT.
3. **Update device association graph**: allows TS to update the device association graph based on new device data.
4. **Update model**: allows TS to train a GBDT model to perform probabilistic CDT on encrypted data.

4.2.1. SETUP

TS initializes two BGV cryptosystems. The first cryptosystem is initialized as described in section 3.3.2 so that it uses plaintext space $\mathbb{F}_{2^8}$ to facilitate the homomorphic computation of AES-CMAC hashes. Its corresponding secret and public keys are denoted by $sk_{BGV-AES}$ and $pk_{BGV-AES}$ respectively. The second cryptosystem is initialized as described in section 3.3.4 so that it uses plaintext space R_2 and allows the homomorphic evaluation of arithmetic operations over binary encoded integers. Its corresponding secret and public keys are denoted by sk_{BGV-IA} and pk_{BGV-IA} respectively. TS sends the encryption keys $pk_{BGV-AES}$ and pk_{BGV-IA} to PS, who will share them with any device that participates in the protocol.

PS initializes a symmetric encryption scheme, e.g. AES-GCM or any other scheme with IND-CCA security, and generates the symmetric key K_{SE}. PS also generates the secret key K_{AES} used for the homomorphic computation of AES-128, and the two keys K_{CMAC1} and K_{CMAC2} derived from K_{AES} using the AES-CMAC key derivation algorithm.

A new device joins the setting by making a request to PS, which generates a unique identifier id for the device, shares it with TS, and replies with the information needed by the device to participate in the protocol, namely the vector $(id, pk_{BGV-AES}, pk_{BGV-IA})$.

4.2.2. UPDATE DEVICE PROFILE

For each device, TS maintains a profile consisting of a collection of data used to identify the user associated with said device. In this stage of the protocol, a device sends its latest data to TS in order to update the corresponding device profile. This data can be split into two categories, data explicitly sent by the device and data inferred from the interaction between parties.

The **explicit data** is composed of a device identifier id, a set x_d of strings that constitute user-identifiable information, which is used to perform deterministic CDT, and a feature vector x_p, containing data used for probabilistic CDT. The vector x_d contains user identifiable information which can be used for deterministic CDT. This information may include email addresses, home addresses, phone numbers, or cookies from current or previous login sessions. The feature vector x_p represents data that is used by the TS to perform probabilistic cross-device tracking. This includes location data, device information such as browser, operating system, device model, etc., and a set of cookies used to track the device browsing history. To lower the communication and computational complexity, feature hashing is used to transform this data into a feature vector of fixed length L_p. The values in x_p are also normalised to the range $[-\theta_p, \theta_p]$. This range restriction allows for a simpler process of generating threshold values when training decision tree models on the data in x_p. To simplify the process of updating device data, each $\mathtt{D}_{id}$ is assumed to maintain a local profile containing all the data needed by TS.

The **inferred data** consists of the IP address of the device and time-related data. The device IP address is a very important piece of information for probabilistic CDT [13, 68, 69], since most devices owned by a user connect to the Internet using the same external IP, the user's home IP address. Because of this, the protocol assumes that the devices are not concealing their IP addresses from the parties they interact with, i.e. through anonymity networks. On the other hand, IP addresses are considered personal information [55], and therefore must be concealed from TS. Time-related data is also very useful in cross-device tracking, since it allows TS to asses the relevance of various events, such as distinguishing a public WiFi network from a home one based on location and time. Furthermore, timestamps are used to assess the relevance of device IP matching, since IP changes with time, as well as differentiating between users who share one or multiple devices. Time data is included in the device local profile, e.g. location data over time or browsing history over time, and is also inferred by the other parties participating in the protocol from the time of an interaction.

Pseudocode 1 shows the steps performed during the profile update phase. The device $\mathtt{D}_{id}$ calls the UPDATE procedure and provides as input the device identifier id, the set of strings x_d used for deterministic CDT, and the feature vector x_p used for probabilistic CDT. Each element in x_d is 10^*-padded to length L_d if necessary, and a padding flag is paired with each string to aid in the homomorphic computation of AES-CMAC hashes. The padding flag is set to 1^{128} if the string was padded, and to 0^{128} if it was not. This produces a set of string pairs denoted by x'_d. Finally, $\mathtt{D}_{id}$ encrypts both x'_d and x_p element-wise, under the encryption keys $pk_{BGV-AES}$ and pk_{BGV-IA} respectively, and sends the tuple $(id, [x'_d]_{pk_{BGV-AES}}, [x_p]_{pk_{BGV-IA}})$ to PS.

PS infers the IP address of $\mathtt{D}_{id}$, and computes the AES-CMAC hash of it, denoted as h_{IP}. PS also uses the homomorphic computation of AES-CMAC to compute $[h_d]_{pk_{BGV-AES}}$, the set of BGV-encrypted hashes of the strings in x_d. PS also encrypts the BGV-encrypted vector $[x_p]_{pk_{BGV-IA}}$ under its secret key K_{SE}, using the symmetric encryption scheme. Finally, PS sends the update data $(id, h_{IP}, [h_d]_{pk_{BGV-AES}}, [[x_p]_{pk_{BGV-IA}}]_{K_{SE}})$ to TS.

Upon receiving the update data, TS decrypts the hashes in h_d, and stores the relevant device data using the following mappings:

1. μ_d maps each device id to the set of all hashes representing data relevant for de-

terministic CDT received from the device

2. μ_d^{-1} is the inverse of μ_d, mapping each hash to the set of ids of devices to which it corresponds
3. μ_p maps each device id to the latest encrypted vector $[[x_p]_{pk_{BGV-IA}}]_{K_{SE}}$ received from that device
4. μ_{IP} maps each device id to the set of all hashed IP addresses from which that device has sent updates
5. μ_{IP}^{-1} is the inverse of μ_{IP}, mapping each IP address hash to the set of ids which sent updates from that address
6. σ_{IP} maps each pair of device id and IP address hash to the latest time when the corresponding device has sent an update from that address

TS updates each mapping accordingly, including updating $\sigma_{IP}(id,\ h_{IP})$ to the current time τ.

4.2.3. UPDATE DEVICE ASSOCIATION GRAPH

In order to associate devices with user profiles, TS has to first group devices based on the likelihood that they belong to the same user, and then associate each group with a user profile. This grouping is performed based on association scores assigned to each pair of devices through both deterministic and probabilistic CDT. TS maintains a device association graph $G = (ID, E)$, in which two devices with identifiers $id_1, id_2 \in ID$ are connected, i.e. $\{id_1, id_2\} \in E$, if either the deterministic or probabilistic association scores corresponding to the pair are above a certain threshold θ_E. As devices update their profiles, TS must also update G according to the current information. Pseudocode 2 offers a pseudocode representation of this process, where the association graph G is updated based on new information provided by some device $\mathtt{D}_{id}$. This process can be initiated by TS each time a device updates its profile, or for a batch of devices that have updated their profiles after the last time the association graph was updated.

The procedure TS:UPDATE-DET updates G based on association scores computed using deterministic CDT. Using μ_d and μ_d^{-1}, the procedure maps each pair of distinct identifiers $(id,\ id')$ to an association score equal to the number of hashed user-identifiable information pieces submitted by both devices, i.e. $|\mu_d(id) \cap \mu_d(id')|$.

The procedure TS:UPDATE-PROB updates G by performing probabilistic CDT over encrypted data. TS maintains a GBDT model which performs regression to produce an association score, which is later compared to θ_E do decide whether two devices belong to the same user or not. Since it may not be feasible to run the model for every pair, the data is down-sampled based on shared IP addresses, following a method inspired from [66–68]. This method only runs the model on pairs of devices that have sent updates from the same IP address within a relevant time-frame. Since most devices owned by a user connect to the Internet using the same external IP address, the devices paired this way are more likely to belong to the same user. TS performs this down-sampling by only running the model on device pairs that share *relevant* IP address hashes in their respective μ_{IP} mappings. An IP address hash $ipHash$ is considered *relevant* for device $\mathtt{D}_{id}$ if

Pseudocode 1 Update device profile, PPCDT protocol

procedure $\mathtt{D}_{id}$:UPDATE(id, x_d, x_p)
  Let $x'_d = \emptyset$.
  for all $x \in x_d$ **do**
    if $|x| < L_d$ **then**
      $x'_d \leftarrow x'_d \cup \{(pad10^*(x, L_d),\ 1^{128})\}$
    else if $|x| = L_d$ **then**
      $x'_d \leftarrow x'_d \cup \{(x,\ 0^{128})\}$
    else
      return $\perp$
    end if
  end for
  PS:UPDATE(id, $[x'_d]_{pk_{BGV-AES}}$, $[x_p]_{pk_{BGV-IA}}$)
end procedure

procedure PS:UPDATE(id, $[x'_d]_{pk_{BGV-AES}}$, $[x_p]_{pk_{BGV-IA}}$)
  $IP \leftarrow$ Infer the device IP address.
  $h_{IP} \leftarrow$ AES-CMAC(IP, K_{AES}, K_{CMAC1}, K_{CMAC2})
  $[h_d]_{pk_{BGV-AES}} \leftarrow \emptyset$
  for all $[(x,\ p)]_{pk_{BGV-AES}} \in [x'_d]_{pk_{BGV-AES}}$ **do**
    $[h]_{pk_{BGV-AES}} \leftarrow$ HE-AES-CMAC($[x]_{pk_{BGV-AES}}$, $[p]_{pk_{BGV-AES}}$, K_{AES}, K_{CMAC1}, K_{CMAC2})
    $[h_d]_{pk_{BGV-AES}} \leftarrow [h_d \cup \{h\}]_{pk_{BGV-AES}}$
  end for
  TS:UPDATE(id, h_{IP}, $[h_d]_{pk_{BGV-AES}}$, $[[x_p]_{pk_{BGV-IA}}]_{K_{SE}}$)
end procedure

procedure TS:UPDATE(id, h_{IP}, $[h_d]_{pk_{BGV-AES}}$, $[[x_p]_{pk_{BGV-IA}}]_{K_{SE}}$)
  $h_d \leftarrow Dec([h_d]_{pk_{BGV-AES}}, sk_{BGV-AES})$
  $\mu_d(id) \leftarrow \mu_d(id) \cup h_d$
  for all $h \in h_d$ **do**
    $\mu_d^{-1}(h) \leftarrow \mu_d^{-1}(h) \cup \{id\}$
  end for
  $\mu_p(id) \leftarrow [[x_p]_{pk_{BGV-IA}}]_{K_{SE}}$
  $\mu_{IP}(id) \leftarrow \mu_{IP}(id) \cup \{h_{IP}\}$
  $\mu_{IP}^{-1}(h_{IP}) \leftarrow \mu_{IP}^{-1}(h_{IP}) \cup \{id\}$
  $\sigma_{IP}(id,\ h_{IP}) \leftarrow \tau$
end procedure

the difference between the current time and the time of the last update made from that address, i.e. $\sigma_{IP}(id,\ ipHash)$, is lower than a threshold value θ_{IP}. The data presented to the model for a pair of devices $(\mathtt{D}_{id},\ \mathtt{D}_{id'})$ consists of the concatenation of the encrypted feature vectors $\mu_p(id)$ and $\mu_p(id')$. The pseudocode of the model evaluation procedure GBDT-EVAL is given in Pseudocode 13 in section 4.3.10.

Pseudocode 2 Update device association graph, PPCDT protocol

```
1:  procedure TS:UPDATE-ASSOCIATIONS(id)
2:      TS:UPDATE-DET(id)
3:      TS:UPDATE-PROB(id)
4:  end procedure

5:  procedure TS:UPDATE-DET(id)
6:      for all h ∈ μ_d(id) do
7:          for all id' ∈ μ_d^{-1}(h) do
8:              s_d ← |μ_d(id) ∩ μ_d(id')|
9:              if s_d ≥ θ_E then
10:                 E ← E ∪ {{id, id'}}
11:             end if
12:         end for
13:     end for
14: end procedure

15: procedure TS:UPDATE-PROB(id)
16:     for all h_IP ∈ μ_IP(id) do
17:         if τ − σ_IP(id, h_IP) < θ_IP then
18:             for all id' ∈ μ_IP^{-1}(h_IP) do
19:                 if {id, id'} ∉ E and τ − σ_IP(id', h_IP) < θ_IP then
20:                     [[x_p]_{pk_BGV-IA}]_{K_SE} ← μ_p(id)||μ_p(id')
21:                     s_p ← TS:GBDT-EVAL([[[x_p]_{pk_BGV-IA}]_{K_SE}])
22:                     if s_p ≥ θ_E then
23:                         E ← E ∪ {{id, id'}}
24:                     end if
25:                 end if
26:             end for
27:         end if
28:     end for
29: end procedure
```

4.2.4. UPDATE MODEL

The machine learning model used to perform probabilistic CDT is trained on the encrypted data to predict association scores. A data point represents the concatenation of $\mu_p(id)$ and $\mu_p(id')$ for a pair of devices $(\mathtt{D}_{id},\ \mathtt{D}_{id'})$. Since there is no feedback on whether a device was correctly linked to a user profile, the training process relies on the assump-

tion that deterministic CDT is sufficiently reliable, and uses the association scores produced deterministically, i.e. $|\mu_d(id) \cap \mu_d(id')|$, as the target labels for the data.

Pseudocode 3 shows the steps involved in this procedure. First, TS down-samples the training data based on shared IP addresses from which updates were received within a time-frame no larger than θ_{IP}. The down-sampled data is then passed to the procedure TS:GLDB-TRAIN, which implements the GBDT training algorithm in a way that allows training over homomorphically encrypted data. The pseudocode for this procedure is given in Pseudocode 14 in section 4.3.11.

Pseudocode 3 Update model, PPCDT protocol

1: **procedure** TS:UPDATE-MODEL
2: $D \leftarrow ()$
3: **for all** $(id_1,\ id_2) \in ID \times ID$, s.t. $id_1 < id_2$ **do**
4: $S_{IP} \leftarrow \mu_{IP}(id_1) \cap \mu_{IP}(id_2)$
5: **if** $\exists ip \in S_{IP}$ s.t. $|\sigma_{IP}(id_1,\ ip) - \sigma_{IP}(id_2,\ ip)| < \theta_{IP}$ **then**
6: $y \leftarrow |\mu_d(id_1) \cap \mu_d(id_2)|$
7: $[[x_p]_{pk_{BGV-IA}}]_{K_{SE}} \leftarrow \mu_p(id_1)||\mu_p(id_2)$
8: $D \leftarrow D||((([[x_p]_{pk_{BGV-IA}}]_{K_{SE}},\ y))$
9: **end if**
10: **end for**
11: TS:GBDT-TRAIN(D)
12: **end procedure**

4.3. AUXILIARY ALGORITHMS

This section describes various auxiliary algorithms used by the PCDT protocol to perform more involved operations on homomorphic ciphertexts.

4.3.1. HOMOMORPHIC COMPUTATION OF AES-CMAC

Pseudocode 4 shows the steps involved in evaluating the AES-CMAC function over BGV-encrypted strings. The HE-AES-CMAC procedure makes use of the homomorphic evaluation of AES-128 to homomorphically compute the AES-CMAC hash of the BGV-encrypted input value m. It is necessary for m to be 10^*-padded to a size divisible by 128 prior to calling the HE-AES-CMAC procedure, and the padding flag p to be set to 1^{128} if m was padded or 0^{128} otherwise. The padding flag is also encrypted to hide the information from the computing party.

The input string is divided into 128-bit blocks $m_{i \in \{1,\ ...,\ n\}}$. Through homomorphic addition and multiplication, the appropriate key is selected based on the padding flag, such that $K^* = K_1$ if $p = 0^{128}$, and $K^* = K_2$ if $p = 1^{128}$. The output hash h is initialized to the homomorphically computed AES encryption of the first message block m_1. For each subsequent block m_i, with $i = 1,\ ...,\ n-1$, the current value of h is XORed with m_i using homomorphic addition, and the AES cipher is homomorphically applied to this value to produce the next value of h. On the last step of the algorithm, h is XORed with m_n and K^*, and the AES cipher is homomorphically applied one last time to produce

the final result. Note that this final result, as well as all intermediate values of h, are BGV-encrypted strings.

Pseudocode 4 Homomorphic AES-CMAC algorithm

procedure HE-AES-CMAC($[m]_{pk_{BGV-AES}}$, $[p]_{pk_{BGV-AES}}$, K_{AES}, K_1, K_2)
  Let $[m]_{pk_{BGV-AES}} = [m_1||...||m_n]_{pk_{BGV-AES}}$, s.t. $|m_1| = ... = |m_n| = 128$.
  $[K^*]_{pk_{BGV-AES}} \leftarrow [((1^{128} \oplus p) \wedge K_1) \oplus (p \wedge K_2)]_{pk_{BGV-AES}}$
  $[h]_{pk_{BGV-AES}} \leftarrow$ HE-AES($[m_1]_{pk_{BGV-AES}}$, K_{AES})
  for $i \leftarrow 2, ..., (n-1)$ **do**
    $[h]_{pk_{BGV-AES}} \leftarrow$ HE-AES($[h \oplus m_i]_{pk_{BGV-AES}}$, K_{AES})
  end for
  return HE-AES($[h \oplus m_n \oplus K^*]_{pk_{BGV-AES}}$, K_{AES})
end procedure

4.3.2. AGGREGATE OPERATION

Pseudocode 5 presents an aggregation algorithm that takes as parameters an associative binary operand $\boxplus$ and a vector $x = (x_i)_{i=1}^{|x|}$, and uses a circuit of depth $\log(|x|)$ to compute the value of $x_1 \boxplus x_2 \boxplus .... \boxplus x_{|x|}$. This algorithm is used to reduce the multiplicative depth of other homomorphic operations.

Pseudocode 5 Aggregation algorithm

procedure AGGREGATE($\boxplus$, x)
  Let $x = (x_1, x_2, ..., x_{|x|})$.
  $\Delta \leftarrow |x|$
  while $\Delta > 1$ **do**
    $\chi \leftarrow \Delta \mod 2$
    $\Delta \leftarrow \lfloor \Delta/2 \rfloor$
    for $i \leftarrow 1, ..., \Delta$ **do**
      $x_i \leftarrow x_i \boxplus x_{i+\Delta}$
    end for
    if $\chi = 1$ **then**
      $x_{\Delta+1} \leftarrow x_{2 \cdot \Delta+1}$
      $\Delta \leftarrow \Delta + 1$
    end if
  end while
  return x_1
end procedure

4.3.3. HOMOMORPHIC BITSHIFT OPERATION

Procedure HE-BITSHIFT presented in Pseudocode 6 performs the bitshift operation over the homomorphically encrypted input integer x, by moving the contents of its bits to the right by n places, or to the left by $-n$ places if $n < 0$. This operation differs from a bit rotation as performed by ψ_n in that the bits that would “loop around" in a rotation are

instead replaced by the input value b. The algorithm simply applies a mask, i.e. a homomorphic multiplication by a constant, to zero out the bits that would be replaced, applies the rotation ψ_n, then fills in the value b at the right locations through a homomorphic addition.

Pseudocode 6 Homomorphic bitshift algorithm

1: **procedure** HE-BITSHIFT($[x]_{pk_{BGV-IA}}$, n, b)
2: **if** $|n| \geq L_b$ **then**
3: **return** $[b^{L_b}]_{pk_{BGV-IA}}$
4: **end if**
5: **if** $n < 0$ **then**
6: **return** $[\psi_n(x \wedge (0^{-n}||1^{L_b+n})) \oplus (0^{L_b+n}||b^{-n})]_{pk_{BGV-IA}}$
7: **end if**
8: **return** $[\psi_n(x \wedge (1^{L_b-n}||0^n)) \oplus (b^n||0^{L_b-n})]_{pk_{BGV-IA}}$
9: **end procedure**

4.3.4. HOMOMORPHIC BIT-AGGREGATE OPERATION

Pseudocode 7 describes the steps of procedure HE-BIT-AGGREGATE, which takes as input an operand $\boxplus \in \{\oplus, \wedge\}$ and a ciphertext encrypting an integer $x = x_{L_b}||...||x_1$, and outputs the encrypted integer b^{L_b}, where $b = x_{L_b} \boxplus ... \boxplus x_1$. The result is computed over $\log(L_b)$ iterations, where on each iteration the value of x is updated to $x \boxplus x_\Delta$, where x_Δ is the current value of x bit-rotated by $\Delta \in \{1, 2, 4, ..., L_b/2\}$. This way, the result is computed using a circuit of depth $\log(L_b)$. Note that the algorithm only works if $L_b = 2^k$ for some $k \in \mathbb{N}$.

Pseudocode 7 Homomorphic bit-aggregate algorithm

1: **procedure** HE-BIT-AGGREGATE($\boxplus$, $[x]_{pk_{BGV-IA}}$)
2: $\Delta \leftarrow 1$
3: **while** $\Delta < L_b$ **do**
4: $[x_\Delta]_{pk_{BGV-IA}} \leftarrow \psi_\Delta([x]_{pk_{BGV-IA}})$
5: $[x]_{pk_{BGV-IA}} \leftarrow [x \boxplus x_\Delta]_{pk_{BGV-IA}}$
6: $\Delta \leftarrow 2 \cdot \Delta$
7: **end while**
8: **return** $[x]_{pk_{BGV-IA}}$
9: **end procedure**

4.3.5. HOMOMORPHIC INTEGER COMPARISON

Pseudocode 8 presents the steps of the homomorphic comparison algorithm described by Cheon et al. in [47]. Procedure HE-LESS-THAN take as parameters two ciphertexts encrypting two integer values $x = x_{L_b}||...||x_1$ and $y = y_{L_b}||...||y_1$. On the first step, homomorphic multiplication is used to select the bits x_{L_b} and y_{L_b}, which are then used to compute lt, having the first bit $lt_{L_b} = x_{L_b} \oplus y_{L_b} \oplus ((1 \oplus x_{L_b}) \wedge y_{L_b})$ and the rest $lt_i = (1 \oplus x_i) \wedge y_i$

for all $i \in \{1, ..., L_b - 1\}$. The values $d_{i \in \{0, ..., L_b - 1\}}$ are mapped to i-rotations of $1^{L_b} \oplus x \oplus y$, such that when multiplying them together, the result $d_\wedge$ holds in each bit i the product $\prod_{j=i+1}^{L_b} 1 \oplus x_j \oplus y_j$. Finally, the lt and $d_\wedge$ are multiplied together, and the procedure HE-BIT-AGGREGATE is used to sum up the resulted bits to obtain the final result bit b, as given by Equation 3.11, section 3.3.4. The algorithm outputs a ciphertext encrypting the integer b^{L_b}, where $b = 1$ if $x < y$, and $b = 0$ otherwise.

Pseudocode 8 Homomorphic less-than algorithm

```
1: procedure HE-LESS-THAN([x]_pk_BGV-IA, [y]_pk_BGV-IA)
2:     [lt]_pk_BGV-IA ← [((1^Lb ⊕ x) ∧ y) ⊕ ((1||0^(Lb−1)) ∧ (x ⊕ y))]_pk_BGV-IA
3:     [d_0]_pk_BGV-IA ← [1^Lb ⊕ x ⊕ y]_pk_BGV-IA
4:     for i ← 1, ..., Lb − 1 do
5:         [d_i]_pk_BGV-IA ← HE-BITSHIFT([d_0]_pk_BGV-IA, i, 1)
6:     end for
7:     [d_∧]_pk_BGV-IA ← AGGREGATE(∧, [(d_1, ..., d_Lb−1)]_pk_BGV-IA)
8:     return HE-BIT-AGGREGATE(⊕, [lt ∧ d_∧]_pk_BGV-IA)
9: end procedure
```

4.3.6. HOMOMORPHIC INTEGER ADDITION

Pseudocode 9 presents the steps of the improved homomorphic addition algorithm described by Hou et al. in [48]. Procedure HE-ADD take as parameters two ciphertexts encrypting two integer values $x = x_{L_b}||...||x_1$ and $y = y_{L_b}||...||y_1$.

The intermediary values $s_{i \in \{0, ..., (L_b/2)-1\}}$ are computed by starting with $s_0 = x \oplus y$, and s_1 being equal to s_0 with its least-significant-bit zeroed out and then bit-shifted to the left by 1 bit. The subsequent values are computed iteratively according to the following rules:

1. if $i = 2^{(k+1)}$ for some integer k, then $s_i = s_{2^k} \wedge s'$, where s' is equal to s_{2^k} bit-shifted to the left by k;
2. if $i = 2^k + j$, with $j < 2^k$, then $s_i = s_{2^k - 1} \wedge s'$, where s' is equal to s_{i+1-2^k} bit-shifted to the left by $k - 1$.

The intermediary values $c_{i \in \{0, ..., L_b - 1\}}$ are computed by starting with $c_0 = x \wedge y$, and c_1 being equal to c_0 bit-shifted to the left by 1 bit. The subsequent values are computed iteratively according to the following rules:

1. if $i = 2^{(k+1)}$ for some integer k, then $c_i = s_{2^k} \wedge c'$, where c' is equal to c_{2^k} bit-shifted to the left by k;
2. if $i = 2^k + j$, with $j < 2^k$, then $c_i = s_{2^k - 1} \wedge c'$, where c' is equal to c_{i+1-2^k} bit-shifted to the left by $k - 1$.

Using these aforementioned intermediate values, all the bits in $((\sum_{j=0}^{i-1}(x_j \cdot y_j \cdot \prod_{k=j+1}^{i-1}(x_k + y_j))) \mod 2)_{i=1}^{L_b}$ can be calculated together by homomorphically computing $\oplus_{i=1}^{L_b - 1} c_i$. Finally, the values s_0, c_1, ..., $c_{L_b - 1}$ are homomorphically

summed to evaluate the formula given in Equation 3.12 of section 3.3.4, for all bits $i \in \{1, ..., L_b\}$.

Pseudocode 9 Homomorphic addition algorithm

procedure HE-ADD($[x]_{pk_{BGV-IA}}$, $[y]_{pk_{BGV-IA}}$)
  $[s_0]_{pk_{BGV-IA}} \leftarrow [x \oplus y]_{pk_{BGV-IA}}$
  $[s_1]_{pk_{BGV-IA}} \leftarrow [\psi_{-1}(s_0 \wedge (0||1^{L_b-2}||0))]_{pk_{BGV-IA}}$
  $pow = 1$
  for $i \leftarrow 2, ..., L_b/2-1$ **do**
    if $i = 2 \cdot pow$ **then**
      $[s']_{pk_{BGV-IA}} \leftarrow$ HE-BITSHIFT($[s_{pow}]_{pk_{BGV-IA}}, -pow, 0$)
      $[s_i]_{pk_{BGV-IA}} \leftarrow [s_{pow} \wedge s']_{pk_{BGV-IA}}$
      $pow \leftarrow i$
    else
      $[s']_{pk_{BGV-IA}} \leftarrow$ HE-BITSHIFT($[s_{i-pow+1}]_{pk_{BGV-IA}}, 1-pow, 0$)
      $[s_i]_{pk_{BGV-IA}} \leftarrow [s_{pow-1} \wedge s']_{pk_{BGV-IA}}$
    end if
  end for
  $[c_0]_{pk_{BGV-IA}} \leftarrow [x \wedge y]_{pk_{BGV-IA}}$
  $[c_1]_{pk_{BGV-IA}} \leftarrow$ HE-BITSHIFT($[c_0]_{pk_{BGV-IA}}, -1, 0$)
  $pow = 1$
  for $i \leftarrow 2, ..., L_b-1$ **do**
    if $i = 2 \cdot pow$ **then**
      $[c']_{pk_{BGV-IA}} \leftarrow$ HE-BITSHIFT($[c_{pow}]_{pk_{BGV-IA}}, -pow, 0$)
      $[c_i]_{pk_{BGV-IA}} \leftarrow [s_{pow} \wedge c']_{pk_{BGV-IA}}$
      $pow \leftarrow i$
    else
      $[c']_{pk_{BGV-IA}} \leftarrow$ HE-BITSHIFT($[c_{i-pow+1}]_{pk_{BGV-IA}}, 1-pow, 0$)
      $[c_i]_{pk_{BGV-IA}} \leftarrow [s_{pow-1} \wedge c']_{pk_{BGV-IA}}$
    end if
  end for
  return AGGREGATE($\oplus$, $[(s_0, c_1, ..., c_{L_b-1})]_{pk_{BGV-IA}}$)
end procedure

4.3.7. HOMOMORPHIC INTEGER SUBTRACTION

Pseudocode 10 describes the homomorphic subtraction procedure HE-SUBTRACT. Since integers are encoded using two's complement, addition can be used to carry out subtraction as follows. First convert the subtrahend y to $-y$ by applying two's complement (XOR with 1^{L_b} and add 1), then add $-y$ to the minuend.

4.3.8. HOMOMORPHIC INTEGER MULTIPLICATION

Pseudocode 11 describes the homomorphic multiplication procedure HE-MULTIPLY. The procedure takes as input two ciphertexts encrypting two integer values $x = x_{L_b}||...||x_1$ and $y = y_{L_b}||...||y_1$, and outputs one ciphertext encrypting the product $x \cdot y$. For each bit

Pseudocode 10 Homomorphic subtraction algorithm

1: **procedure** HE-SUBTRACT($[x]_{pk_{BGV-IA}}$, $[y]_{pk_{BGV-IA}}$)
2: **return** $[(y \oplus 1^{L_b}) + (0^{L_b-1}||1) + x]_{pk_{BGV-IA}}$
3: **end procedure**

y_i, the algorithm computes a corresponding value $t_i = x \wedge (y_i)^{L_b}$. The intermediary value $(y_i)^{L_b}$ is computed using a homomorphic multiplication to zero-out all the bits in y except y_i, and then using the HE-BIT-AGGREGATE procedure to copy it to all other slots. The final result is obtained by computing $\sum_{i=1}^{L_b} t_i$.

Pseudocode 11 Homomorphic multiplication algorithm

1: **procedure** HE-MULTIPLY($[x]_{pk_{BGV-IA}}$, $[y]_{pk_{BGV-IA}}$)
2: **for** $i \leftarrow 1, ..., L_b$ **do**
3: $[t_i]_{pk_{BGV-IA}} \leftarrow$ HE-BIT-AGGREGATE($\oplus$, $[y \wedge (0^{L_b-i}||1||0^{i-1})]_{pk_{BGV-IA}}$)
4: $[t_i]_{pk_{BGV-IA}} \leftarrow$ HE-BITSHIFT($[t_i \wedge x]_{pk_{BGV-IA}}$, $1-i$, 0)
5: **end for**
6: **return** AGGREGATE(+, $[(t_i)_{i=1}^{L_b}]_{pk_{BGV-IA}}$)
7: **end procedure**

4.3.9. HOMOMORPHIC INTEGER DIVISION

Pseudocode 12 presents the steps of the non-restoring division algorithm adapted to operate on homomorphically encrypted integers. The procedure HE-DIVIDE takes as input two ciphertexts encrypting two integer values $x = x_{L_b}||...||x_1$ and $y = y_{L_b}||...||y_1$, and outputs one ciphertext encrypting the quotient $q = \lfloor x/y \rfloor$.

The algorithm first computes the intermediary value $r = (x_{L_b}||0^{L_b-1}) - y$ and the initial quotient value $q = (1 \oplus r_{L_b})||0^{L_b-1}$. The algorithm then iterates over the bits of q from q_{L_b-1} to q_1, computing each bit q_i as follows:

1. compute $c = (r_{L_b})^{L_b}$;
2. if $c = 1^{L_b}$, then $y' = y$, otherwise y' equals the complement of y, plus one;
3. bitshift r by 1 to the left;
4. compute $r' = 0^{L_b-1}||x_i$;
5. XOR r and r' to set the last bit of r to x_i, then add y' using the homomorphic integer addition algorithm and assign the result to r;
6. compute $q' = 0^{L_b-i}||(1 \oplus r_{L_b})||0^{i-1}$, where r_{L_b} is the most significant bit of r;
7. set the bit q_i to the correct value by XORing q and q'.

Pseudocode 12 Homomorphic division algorithm

procedure HE-DIVIDE($[x]_{pk_{BGV-IA}}$, $[y]_{pk_{BGV-IA}}$)
  $[r]_{pk_{BGV-IA}} \leftarrow [(1||0^{L_b-1}) \wedge x]_{pk_{BGV-IA}}$
  $[r]_{pk_{BGV-IA}} \leftarrow [r-y]_{pk_{BGV-IA}}$
  $[q]_{pk_{BGV-IA}} \leftarrow [(1||0^{L_b-1}) \wedge (r \oplus (1||0^{L_b-1}))]_{pk_{BGV-IA}}$
  for $i \leftarrow L_b - 1, \ldots, 1$ **do**
    $[c]_{pk_{BGV-IA}} \leftarrow$ HE-BIT-AGGREGATE($\oplus$, $[r \wedge (1||0^{L_b-1})]_{pk_{BGV-IA}}$)
    $[y']_{pk_{BGV-IA}} \leftarrow [(c \wedge y) \oplus ((1^{L_b} \oplus c) \wedge ((y \oplus 1^{L_b}) + (0^{L_b-1}||1)))]_{pk_{BGV-IA}}$
    $[r]_{pk_{BGV-IA}} \leftarrow$ HE-BITSHIFT($[r]_{pk_{BGV-IA}}, -1, 0$)
    $[r']_{pk_{BGV-IA}} \leftarrow [\psi_{i-1}(x \wedge (0^{L_b-i}||1||0^{i-1}))]_{pk_{BGV-IA}}$
    $[r]_{pk_{BGV-IA}} \leftarrow [(r \oplus r') + y']_{pk_{BGV-IA}}$
    $[q']_{pk_{BGV-IA}} \leftarrow [\psi_{L_b-i}((1||0^{L_b-1}) \oplus (r \wedge (1||0^{L_b-1})))]_{pk_{BGV-IA}}$
    $[q]_{pk_{BGV-IA}} \leftarrow [q \oplus q']_{pk_{BGV-IA}}$
  end for
  return $[q]_{pk_{BGV-IA}}$
end procedure

4.3.10. HOMOMORPHIC EVALUATION OF GRADIENT BOOSTING DECISION TREES

Pseudocode 13 describes the evaluation procedure of evaluating a GBDT model on the homomorphically encrypted feature vector x_p. This vector is doubly-encrypted, first under the BGV-IA encryption scheme and then under the symmetric scheme used by PS.

The BGDT model maintained by TS consists of an average value A and n trees of depth d_{max}. Each tree of index i is described by a tree node t_i representing the root of the tree. Each non-leaf tree node t contains a threshold value $t.\theta \in S_\theta$, a feature index $t.feature \in \{1,\ldots,2L_p\}$, and two children nodes, $t.right$ and $t.left$. Each leaf tree node t contains an integer value $t.value$. The threshold and leaf values are also doubly encrypted to keep them secret from both TS and PS. This is done to prevent either party from learning information about the input data from the model predictions.

TS first sends the tree ensemble T and the feature vector x_p to PS, and receives the BGV-encrypted vector y containing the n predictions made by the trees in T when presented with the data in x_p. TS decrypts y and calculates the sum of its values. Since the decision trees are trained on target values scaled up by a factor of 10^ϵ to simulate rational number arithmetic using integers, TS will divide the sum of predictions by 10^ϵ to bring it to the correct scale. This value is then multiplied by the learning rate γ and added to the average value A to obtain the final prediction of the model. Note that only the encrypted values are integers. When computing in plaintext space, floating-point arithmetic can be used for increased precision.

The procedure PS:GBDT-EVAL details the steps taken by PS to evaluate the trees in ensemble T on the feature vector x_p. PS first decrypts x_p, as well as the threshold and leaf values of the trees in the ensemble, using its symmetric key K_{SE}. As these values are now encrypted only under the BGV-IA homomorphic scheme, the prediction of each tree can be homomorphically evaluated using the procedure PS:TREE-EVAL. The pseudocode of

this procedure is given in Pseudocode 15 of section 4.3.12. After obtaining the list of predictions y of each tree, PS sends y to TS.

Pseudocode 13 Homomorphic evaluation of gradient boosting decision trees

1: **procedure** TS:GBDT-EVAL($[[x_p]_{pk_{BGV-IA}}]_{K_{SE}}$)
2: $[y]_{pk_{BGV-IA}} \leftarrow$ PS:GBDT-EVAL(T, $[[x_p]_{pk_{BGV-IA}}]_{K_{SE}}$)
3: $y \leftarrow Dec([y]_{pk_{BGV-IA}}, sk_{BGV-IA})$
4: Let $y = (y_i)_{i=1}^{n}$.
5: **return** $A + \gamma \cdot \frac{\sum_{i=1}^{n} y_i}{10^{\epsilon}}$
6: **end procedure**

7: **procedure** PS:GBDT-EVAL(T, $[[x_p]_{pk_{BGV-IA}}]_{K_{SE}}$)
8: $[x_p]_{pk_{BGV-IA}} \leftarrow Dec([[x_p]_{pk_{BGV-IA}}]_{K_{SE}}, K_{SE})$
9: Let $T = (t_i)_{i=0}^{|T|}$.
10: $T' \leftarrow$ (PS:DECRYPT-TREE(t_i) $)_{i=0}^{|T|}$
11: Let $T' = (t'_i)_{i=0}^{|T'|}$.
12: $[y]_{pk_{BGV-IA}} \leftarrow$ (PS:TREE-EVAL(t'_i, $[x_p]_{pk_{BGV-IA}}$) $)_{i=0}^{|T'|}$
13: **return** $[y]_{pk_{BGV-IA}}$
14: **end procedure**

15: **procedure** PS:DECRYPT-TREE(t)
16: **if** t is a leaf node **then**
17: $[t.value]_{pk_{BGV-IA}} \leftarrow Dec([[t.value]_{pk_{BGV-IA}}]_{K_{SE}}, K_{SE})$
18: **return** t
19: **end if**
20: $[t.\theta]_{pk_{BGV-IA}} \leftarrow Dec([[t.\theta]_{pk_{BGV-IA}}]_{K_{SE}}, K_{SE})$
21: $t.right \leftarrow$ PS:DECRYPT-TREE($t.right$)
22: $t.left \leftarrow$ PS:DECRYPT-TREE($t.left$)
23: **return** t
24: **end procedure**

4.3.11. HOMOMORPHIC TRAINING OF GRADIENT BOOSTING DECISION TREES

Pseudocode 4.3.11 describes the steps taken to train a GBDT model on encrypted data. The procedure TS:GLDB-TRAIN takes as input a vector D of pairs $(x_{p_i},\ y_i)_{i \in \{1,\ ...,\ |D|\}}$, where x_{p_i} is the feature vector of a training data point, and y_i is its corresponding target prediction. The feature vectors are provided in doubly-encrypted form, first under the public key pk_{BGV-IA}, and then under the symmetric key K_{SE}, while the target values are provided in plaintext form. It is worth mentioning that the target values are unencrypted because they are already known to TS, and not because the operations involved necessitate it. An algorithm that operates on encrypted target values could easily be derived.

First, the average value A of the GBDT model is computed in plaintext space. A new value A' is derived from A by multiplying it by 10^{ϵ}. This value is then encrypted under the BGV-IA scheme to hide it from PS. The data vector D' is derived from D by multiplying

each target value by 10^{ϵ} and then encrypting them under BGV-IA. The multiplications with 10^{ϵ} allow us to use integer arithmetic to simulate rational number arithmetic up to ϵ decimal digits of precision when performing computations on homomorphic ciphertexts. θ_p and Δ_p are parameters required to compute the threshold set S_θ, describing the range of values in the feature vector and the difference between consecutive threshold values respectively. γ, n, and d_{max} are training parameters selected by TS, corresponding to the learning rate, the number of decision trees in the GBDT model, and the depth of each tree respectively. Lastly, TS sends the data tuple $(D', [A']_{pk_{BGV-IA}}, \theta_p, \Delta_p, \gamma, n, d_{max})$ to PS and receives a vector T' of n decision tree models trained on the data in D'. The threshold and leaf values of the decision trees are doubly-encrypted under both BGV-IA and the symmetric scheme employed by PS, in order to hide them from TS. The feature indices are only encrypted under the BGV-IA homomorphic scheme, which allows TS to decrypt them. Having the feature indices in plaintext form allows for a faster evaluation of model predictions.

The procedure PS:GLDB-TRAIN executed by PS takes as input the data sent by TS and outputs a vector of decision trees homomorphically trained on this data. The set of threshold values S_θ is constructed based on the two constants θ_p and Δ_p. For security reasons, the procedure only accepts parameters that will result in at least two threshold values, and terminates if $|S_\theta| \leq 1$. PS then derives a vector D_w from D' by decrypting the symmetrically encrypted feature vectors x_{p_i}, homomorphically subtracting A' from each target value y'_i, and attaching a third value to each data point pair, which represents the path flag used for the homomorphic decision trees training algorithm, and is set to 1^{L_b} for all data points. Note that each value in D_w is still homomorphically encrypted and therefore hidden from PS. PS uses the procedure PS:TREE-TRAIN to train n trees on the the data in D_w, effectively training each tree to predict the residual between the final target value y'_i and the prediction of the model so far. The pseudocode for this procedure is given in Pseudocode 16 of section 4.3.13. After training each tree, its predictions are multiplied by the learning rate and subtracted from the corresponding target values to update D_w in preparation for training the next tree. After training n trees, PS returns the tree ensemble to TS.

4.3.12. HOMOMORPHIC EVALUATION OF A DECISION TREE

Pseudocode 15 shows the steps taken by PS to evaluate a decision tree model on BGV-encrypted input. Procedure PS:TREE-EVAL takes as input a tree node t and a BGV-encrypted feature vector x_p. If t is a leaf node, then the procedure returns the encrypted value of the node. If t is not a leaf node, then the output values of its left and right subtrees are recursively computed and denoted by v_L and v_R respectively. The comparison value lt is also homomorphically computed using the procedure HE-LESS-THAN (Pseudocode 8 in section 4.3.5). This value is set to a BGV-encryption of 1^{L_b} if the element of x_p with index $t.feature$ is less than the threshold value of the node $t.\theta$, and to an encryption of 0^{L_b} otherwise. The correct return value is selected between lv and rv through homomorphic multiplications and additions as shown on line 8 of the algorithm.

Pseudocode 14 Homomorphic training of gradient boosting decision trees

procedure TS:GBDT-TRAIN(D)
  Let $D = (([[x_{p_i}]_{pk_{BGV-IA}}]_{K_{SE}}, y_i))_{i=1}^{|D|}$.
  $A \leftarrow \frac{\sum_{i=1}^{|D|} y_i}{|D|}$
  $A' \leftarrow A \cdot 10^{\epsilon}$
  $D' \leftarrow (([[x_{p_i}]_{pk_{BGV-IA}}]_{K_{SE}}, [y_i \cdot 10^{\epsilon}]_{pk_{BGV-IA}}))_{i=1}^{|D|}$
  $T' \leftarrow$ PS:GBDT-TRAIN(D', $[A']_{pk_{BGV-IA}}, \theta_p, \Delta_p, \gamma, n, d_{max}$)
  Let $T' = (t_i)_{i=1}^{n}$.
  $T \leftarrow (\text{ TS:DECRYPT-FEATURES}(t_i)\ )_{i=1}^{n}$
end procedure

procedure PS:GBDT-TRAIN(D', $[A']_{pk_{BGV-IA}}, \theta_p, \Delta_p, \gamma, n, d_{max}$)
  $S_\theta \leftarrow \{i \cdot \Delta_p \mid -\frac{\theta_p}{\Delta_p} < i < \frac{\theta_p}{\Delta_p}\}$
  if $|S_\theta| \leq 1$ **then**
    return ()
  end if
  Let $D' = (([[x'_{p_i}]_{pk_{BGV-IA}}]_{K_{SE}}, [y'_i]_{pk_{BGV-IA}}))_{i=1}^{|D'|}$.
  $[D_w]_{pk_{BGV-IA}} \leftarrow$
    $((Dec([[x'_{p_i}]_{pk_{BGV-IA}}]_{K_{SE}}, K_{SE}), [y'_i - A]_{pk_{BGV-IA}}, [1^{L_b}]_{pk_{BGV-IA}}))_{i=1}^{|D'|}$
  Let $T = (t_i)_{i=1}^{n}$.
  for $i \leftarrow 1, .., n$ **do**
    $(t_i, [P]_{pk_{BGV-IA}}) \leftarrow$
      PS:TREE-TRAIN($[D_w]_{pk_{BGV-IA}}, [S_\theta]_{pk_{BGV-IA}}, 1, d_{max}$)
    if $i = n$ **then**
      return T
    end if
    Let $D_w = ((x'_{p_j}, y'_j, w_j))_{j=1}^{|D_w|}$.
    Let $P = (p_j)_{j=1}^{|P|}$.
    $[D_w]_{pk_{BGV-IA}} \leftarrow [((x'_{p_j}, y'_j - (\gamma \cdot p_j), w_j))_{j=1}^{|D_w|}]_{pk_{BGV-IA}}$
  end for
end procedure

procedure TS:DECRYPT-FEATURES(t)
  if t is a leaf node **then**
    return t
  end if
  $t.feature \leftarrow Dec([t.feature]_{pk_{BGV-IA}}, sk_{BGV-IA})$
  $t.right \leftarrow$ TS:DECRYPT-FEATURES($t.right$)
  $t.left \leftarrow$ TS:DECRYPT-FEATURES($t.left$)
  return t
end procedure

Pseudocode 15 Homomorphic evaluation of a decision tree

procedure PS:TREE-EVAL(t, $[x_p]_{pk_{BGV-IA}}$)
  if t is a leaf node **then**
    return $[t.value]_{pk_{BGV-IA}}$
  end if
  $[v_L]_{pk_{BGV-IA}} \leftarrow$ PS:TREE-EVAL($t.left$, $[x_p]_{pk_{BGV-IA}}$)
  $[v_R]_{pk_{BGV-IA}} \leftarrow$ PS:TREE-EVAL($t.right$, $[x_p]_{pk_{BGV-IA}}$)
  Let $x_p = (x_{p_i})_{i=1}^{2L_p}$.
  $[lt]_{pk_{BGV-IA}} \leftarrow$ HE-LESS-THAN($[x_{t.feature}]_{pk_{BGV-IA}}$, $[t.\theta]_{pk_{BGV-IA}}$)
  return $[(lt \wedge v_L) \oplus ((lt \oplus 1^{L_b}) \wedge v_R)]_{pk_{BGV-IA}}$
end procedure

4.3.13. HOMOMORPHIC TRAINING OF A DECISION TREE MODEL

Pseudocode 16 presents the steps taken by PS to train a decision tree model on an encrypted training set. The procedure PS:TREE-TRAIN takes as input a collection of encrypted training data D_w, a set of encrypted threshold values S_θ, the current depth d of the tree, and the maximum depth d_{max}, and recursively builds a decision tree by selecting for each non-leaf node the combination of feature index and threshold value that minimise the mean squared error between the target values present in a child branch and the mean of the target values present in said branch.

The collection D_w is composed of tuples $(x_i,\ y_i,\ w_i)_{i \in \{1,\ ...,\ |D_w|\}}$, where each x_i is a feature vector of length $2L_p$ representing a data point in the training set, y_i is its associated target value, and w_i is a flag that marks whether or not the data point is relevant for the current node. When $d = 1$, $w_i = 1^{L_b}$ for all $i \in \{1,\ ...,\ |D_w|\}$, but as the data is split by the tree nodes based on features and thresold values, a data point with index i that is no longer in the path has $w_i = 0^{L_b}$.

All the values in D_w and S_θ are encrypted under the BGV-IA homomorphic scheme. The procedure returns a tuple (t, P), where t is a tree node and P is the prediction made by the subtree t on the set of feature vectors x_i present in the path, i.e $w_i = 1^{L_b}$.

If $d = d_{max}$, then PS:TREE-TRAIN returns a leaf node t and the recursion ends. The leaf value $t.value$ is set to the mean of the y_i values of data points still present in the path. The leaf value is also encrypted using the symmetric key K_{SE} to hide it from TS. When computing the mean of the target values, each y_i value is multiplied by its corresponding flag value w_i and their sum, $ySum$, is computed homomorphically. The number of data points still present in the path, $wSum$, is computed by converting each value $w_i = b^{L_b}$, with $b \in \{0,\ 1\}$, to a new value $(0^{L_b - 1}||b)$ through homomorphic multiplication, and summing them up. Finally the mean is computed using the homomorphic division procedure HE-DIVIDE (Pseudocode 12 in section 4.3.9). The prediction vector P is constructed such that it's value at index i is equal to the leaf value if $w_i = 1^{L_b}$, and equal to 0^{L_b} otherwise.

To account for the possibility of zero data points being present in the path, in which case the leaf value would be the result of a division by 0, $wSum$ is homomorphically compared with 1 using the procedure HE-LESS-THAN (Pseudocode 8 in section 4.3.5), and the flag $div0$ is set to 1^{L_b} if $wSum = 0$, and to 0^{L_b} otherwise. Based on the value of

$div0$, through homomorphic multiplications and additions, the leaf value is set to 0 if a division by 0 occurred, to remove the contribution of the tree to the overall GBDT model prediction for data points that reach that leaf.

If $d < d_{max}$, then PS:TREE-TRAIN returns a non-leaf node, characterised by a threshold value $t.\theta \in S_\theta$, and a feature index $t.feature \in \{1, ..., 2L_p\}$. The algorithm iterates over all possible combinations of feature and threshold, selecting the pair that minimizes the mean squared error between the predictions and target values. To do so, for each pair feature-threshold pair (f, θ), the algorithm uses the HE-LESS-THAN procedure to obtain a vector $lt_{f,\theta}$, where the value at index i is equal to 1^{L_b} if $x_{i,f} < \theta$, and 0^{L_b} otherwise, where $x_{i,f}$ is the element at index f in the i-th feature vector in D_w. Using $lt_{f,\theta}$ and the w_i values in D_w, two vectors of flag values are computed, w_L and w_R, marking the data points in the path of each subtree. In a similar fashion to the way leaf values are computed, the average target values are computed for both the left and right subtree, denoted by y_L and y_R respectively. For each data point with index i, its corresponding error is equal to the difference between its target value y_i and the average value $\overline{y} \in \{y_L, y_R\}$ of the path it is in. The sums of the squared errors are computed for each subtree and denoted by e_L and e_R, and the mean squared error e is computed by dividing $(e_L + e_R)$ by the total number of data points in the path.

If no data points are present in the path or in the path of either subtree, the error obtained is a result of one or more divisions by 0. To account for this, two flag values $div0_L$ and $div0_R$ are computed in the same fashion as in the leaf-node case, and their values are combined into a single flag $div0$. Based on $div0$, the error produced by a feature-threshold pair is set to $(0||1^{L_b-1})$, i.e. the highest integer allowed by the encoding, if any divisions by 0 were involved in its computation. This way, a useless node that does not cause a split in the training data is only produced if no better alternative exists.

The iteration produces an error value for each possible feature-threshold pair. These values are stored in a vector of tuples $V_e = ((f_i, \theta_i, lt_{f_i,\theta_i}, e_i))_{i=1}^{|V_e|}$, where f_i denotes the feature index, θ_i denotes the threshold value, lt_{f_i,θ_i} is the comparison vector produced by the feature-threshold pair, and e_i is the error value. The procedure PS:BEST-NODE returns the tuple $(f^*, \theta^*, lt_{f^*,\theta^*}, e^*)$ in V_e that contains the minimum error, which it computes using a circuit with depth $\log(|V_e|)$ of homomorphic comparisons, additions, and multiplications. The tree node returned by the procedure has its feature index set to f^* and its threshold value set to θ^*. Both values are BGV-encrypted as a result of homomorphic operations, and the threshold value is additionally encrypted by PS using the symmetric key K_{SE} to keep it secret from TS. The comparison vector lt_{f^*,θ^*} is used to compute the path flags for each subtree of the node, and the procedure recursively computes the two children of the current node.

Along with the child nodes, the recursive calls also produce two prediction vectors, one for each subtree. Since each data point reaches a single leaf node, given a data point with index i, only one of the two prediction vectors can have a non-zero value at index i. Because of this, an element-wise homomorphic addition is sufficient to combine the prediction vectors from the left and right subtrees.

Pseudocode 16 Homomorphic training of a decision tree

procedure PS:TREE-TRAIN($[D_w]_{pk_{BGV-IA}}$, $[S_\theta]_{pk_{BGV-IA}}$, d, d_{max})
  Let $D_w = ((x_i,\ y_i,\ w_i))_{i=1}^{|D_w|}$.
  Let $x_i = (x_{i,j})_{j=1}^{2L_p}$ for all $i \in \{1,\ ...,\ |D_w|\}$.
  if $d = d_{max}$ **then**
    $t \leftarrow$ leaf node
    $[ySum]_{pk_{BGV-IA}} \leftarrow$ AGGREGATE(+, $[(w_i \wedge y_i)_{i=1}^{|D_w|}]_{pk_{BGV-IA}}$)
    $[wSum]_{pk_{BGV-IA}} \leftarrow$ AGGREGATE(+, $[(w_i \wedge (0^{L_b-1}||1))_{i=1}^{|D_w|}]_{pk_{BGV-IA}}$)
    $[val]_{pk_{BGV-IA}} \leftarrow$ HE-DIVIDE($[ySum]_{pk_{BGV-IA}}$, $[wSum]_{pk_{BGV-IA}}$)
    $[div0]_{pk_{BGV-IA}} \leftarrow$ HE-LESS-THAN($[wSum]_{pk_{BGV-IA}}$, $[0^{L_b-1}||1]_{pk_{BGV-IA}}$)
    $[val]_{pk_{BGV-IA}} \leftarrow [(1^{L_b} \oplus div0) \wedge val]_{pk_{BGV-IA}}$
    $[[t.value]_{pk_{BGV-IA}}]_{K_{SE}} \leftarrow [[val]_{pk_{BGV-IA}}]_{K_{SE}}$
    $[P]_{pk_{BGV-IA}} \leftarrow [(w_i \wedge val)_{i=1}^{|D_w|}]_{pk_{BGV-IA}}$
    return $(t,\ [P]_{pk_{BGV-IA}})$
  end if
  $[V_e]_{pk_{BGV-IA}} \leftarrow ()$
  for all $(f,\ [\theta]_{pk_{BGV-IA}}) \in \{1,...,2L_p\} \times [S_\theta]_{pk_{BGV-IA}}$ **do**
    $[lt_{f,\theta}]_{pk_{BGV-IA}} \leftarrow (\text{ HE-LESS-THAN}([x_{i,f}]_{pk_{BGV-IA}}, [\theta]_{pk_{BGV-IA}}))_{i=1}^{|D_w|}$
    Let $lt_{f,\theta} = (lt_{f,\theta,i})_{i=1}^{|D_w|}$.
    $[w_L]_{pk_{BGV-IA}} \leftarrow [(lt_{f,\theta,i} \wedge w_i)_{i=1}^{|D_w|}]_{pk_{BGV-IA}}$
    Let $w_L = (w_{L_i})_{i=1}^{|D_w|}$.
    $[wSum_L]_{pk_{BGV-IA}} \leftarrow$ AGGREGATE(+, $[(w_{L_i} \wedge (0^{L_b-1}||1))_{i=1}^{|D_w|}]_{pk_{BGV-IA}}$)
    $[ySum_L]_{pk_{BGV-IA}} \leftarrow$ AGGREGATE(+, $[(w_{L_i} \wedge y_i)_{i=1}^{|D_w|}]_{pk_{BGV-IA}}$)
    $[y_L]_{pk_{BGV-IA}} \leftarrow$ HE-DIVIDE($[ySum_L]_{pk_{BGV-IA}}$, $[wSum_L]_{pk_{BGV-IA}}$)
    $[e_L]_{pk_{BGV-IA}} \leftarrow$ AGGREGATE(+, $[(w_{L_i} \wedge (y_i - y_L)^2)_{i=1}^{|D_w|}]_{pk_{BGV-IA}}$)
    $[w_R]_{pk_{BGV-IA}} \leftarrow [((1^{L_b} \oplus lt_{f,\theta,i}) \wedge w_i)_{i=1}^{|D_w|}]_{pk_{BGV-IA}}$
    Let $w_R = (w_{R_i})_{i=1}^{|D_w|}$.
    $[wSum_R]_{pk_{BGV-IA}} \leftarrow$ AGGREGATE(+, $[(w_{R_i} \wedge (0^{L_b-1}||1))_{i=1}^{|D_w|}]_{pk_{BGV-IA}}$)
    $[ySum_R]_{pk_{BGV-IA}} \leftarrow$ AGGREGATE(+, $[(w_{R_i} \wedge y_i)_{i=1}^{|D_w|}]_{pk_{BGV-IA}}$)
    $[y_R]_{pk_{BGV-IA}} \leftarrow$ HE-DIVIDE($[ySum_R]_{pk_{BGV-IA}}$, $[wSum_R]_{pk_{BGV-IA}}$)
    $[e_R]_{pk_{BGV-IA}} \leftarrow$ AGGREGATE(+, $[(w_{R_i} \wedge (y_i - y_R)^2)_{i=1}^{|D_w|}]_{pk_{BGV-IA}}$)
    $[wSum]_{pk_{BGV-IA}} \leftarrow [wSum_L + wSum_R]_{pk_{BGV-IA}}$
    $[e]_{pk_{BGV-IA}} \leftarrow$ HE-DIVIDE($[e_L + e_R]_{pk_{BGV-IA}}$, $[wSum]_{pk_{BGV-IA}}$)
    $[div0_L]_{pk_{BGV-IA}} \leftarrow$
      HE-LESS-THAN($[wSum_L]_{pk_{BGV-IA}}$, $[0^{L_b-1}||1]_{pk_{BGV-IA}}$)
    $[div0_R]_{pk_{BGV-IA}} \leftarrow$
      HE-LESS-THAN($[wSum_R]_{pk_{BGV-IA}}$, $[0^{L_b-1}||1]_{pk_{BGV-IA}}$)
    $[div0]_{pk_{BGV-IA}} \leftarrow [div0_L \oplus div0_R \oplus (div0_L \wedge div0_R)]_{pk_{BGV-IA}}$
    $[e]_{pk_{BGV-IA}} \leftarrow [(div0 \wedge (0||1^{L_b-1})) \oplus ((div0 \oplus 1^{L_b}) \wedge e)]_{pk_{BGV-IA}}$
    $[V_e]_{pk_{BGV-IA}} \leftarrow [V_e||((f,\ \theta,\ lt_{f,\theta},\ e))]_{pk_{BGV-IA}}$
  end for

$[(f^*, \theta^*, lt_{f^*,\theta^*}, e^*)]_{pk_{BGV-IA}} \leftarrow$ PS:BEST-NODE($[V_e]_{pk_{BGV-IA}}$)
Let $lt_{f^*,\theta^*} = (lt_{f^*,\theta^*,i})_{i=1}^{|D_w|}$.
$t \leftarrow$ non-leaf node
$[t.feature]_{pk_{BGV-IA}} \leftarrow [f^*]_{pk_{BGV-IA}}$
$[[t.\theta]_{pk_{BGV-IA}}]_{K_{SE}} \leftarrow [[\theta^*]_{pk_{BGV-IA}}]_{K_{SE}}$
$[D_L]_{pk_{BGV-IA}} \leftarrow [(x_i, y_i, lt_{f^*,\theta^*,i} \wedge w_i)_{i=1}^{|D_w|}]_{pk_{BGV-IA}}$
$(t.left, [P_L]_{pk_{BGV-IA}}) \leftarrow$
PS:TREE-TRAIN($[D_L]_{pk_{BGV-IA}}, [S_\theta]_{pk_{BGV-IA}}, d+1, d_{max}$)
$[D_R]_{pk_{BGV-IA}} \leftarrow [(x_i, y_i, (1^{L_b} \oplus lt_{f^*,\theta^*,i}) \wedge w_i)_{i=1}^{|D_w|}]_{pk_{BGV-IA}}$
$(t.right, [P_R]_{pk_{BGV-IA}}) \leftarrow$
PS:TREE-TRAIN($[D_R]_{pk_{BGV-IA}}, [S_\theta]_{pk_{BGV-IA}}, d+1, d_{max}$)
$[P]_{pk_{BGV-IA}} \leftarrow [P_L \oplus P_R]_{pk_{BGV-IA}}$
return $(t, [P]_{pk_{BGV-IA}})$
end procedure

procedure PS:BEST-NODE($[V_e]_{pk_{BGV-IA}}$)
Let $V_e = ((f_i, \theta_i, lt_{f_i,\theta_i}, e_i))_{i=1}^{|V_e|}$.
if $|V_e| = 1$ **then**
return $[(f_1, \theta_1, lt_{f_1,\theta_1}, e_1)]_{pk_{BGV-IA}}$
end if
$[(f_L, \theta_L, lt_{f_L,\theta_L}, e_L)]_{pk_{BGV-IA}} \leftarrow$
PS:BEST-NODE($[((f_i, \theta_i, lt_{f_i,\theta_i}, e_i))_{i=1}^{\lfloor |V_e|/2 \rfloor}]_{pk_{BGV-IA}}$)
$[(f_R, \theta_R, lt_{f_R,\theta_R}, e_R)]_{pk_{BGV-IA}} \leftarrow$
PS:BEST-NODE($[((f_i, \theta_i, lt_{f_i,\theta_i}, e_i))_{i=\lfloor |V_e|/2 \rfloor + 1}^{|V_e|}]_{pk_{BGV-IA}}$)
$[lt_e]_{pk_{BGV-IA}} \leftarrow$ HE-LESS-THAN($[e_L]_{pk_{BGV-IA}}, [e_R]_{pk_{BGV-IA}}$)
$[gt_e]_{pk_{BGV-IA}} \leftarrow [lt_e \oplus 1^{L_b}]_{pk_{BGV-IA}}$
$[f^*]_{pk_{BGV-IA}} \leftarrow [(lt_e \wedge f_L) \oplus (gt_e \wedge f_R)]_{pk_{BGV-IA}}$
$[\theta^*]_{pk_{BGV-IA}} \leftarrow [(lt_e \wedge \theta_L) \oplus (gt_e \wedge \theta_R)]_{pk_{BGV-IA}}$
$[lt_{f^*,\theta^*}]_{pk_{BGV-IA}} \leftarrow [(lt_e \wedge lt_{f_L,\theta_L}) \oplus (gt_e \wedge lt_{f_R,\theta_R})]_{pk_{BGV-IA}}$
$[e^*]_{pk_{BGV-IA}} \leftarrow [(lt_e \wedge e_L) \oplus (gt_e \wedge e_R)]_{pk_{BGV-IA}}$
return $[(i^*, \theta^*, lt_{f^*,\theta^*}, e^*)]_{pk_{BGV-IA}}$
end procedure

5

ANALYSES

5.1. SECURITY EVALUATION

In this section we evaluate the security of our protocol. We first offer an overview of the threat model and assumptions under which we conduct our analysis. We then construct an argument based on the security properties of the underlying primitives, to support our claim that PCDT preserves the secrecy of the collected device data. Finally, we analyse the impact of collusion attacks against PCDT, since they often pose a significant threat to multi-server systems.

We divide the device data into two categories, based on whether it is used when performing deterministic or probabilistic CDT. The device data used for deterministic CDT consists of the string sets x_d provided by devices as shown in Pseudocode 1, section 4.2.2, while the data used for probabilistic CDT consists of the feature vectors x_p and the IP addresses collected by PS. We use the symbol X_d to denote the collection of all data shared by devices for the purpose of deterministic CDT. Similarly, we use the symbol X_p to denote the set of all x_p vectors shared by all devices, and the symbol Π_p to denote the set of all decision tree nodes trained on the data in X_p.

As PCDT employs multiple security schemes to preserve the secrecy of each category of device data from each server party, in the following sections we analyse the security of PCDT for each type of device data and each adversary $P \in \{\text{PS, TS}\}$.

5.1.1. THREAT MODEL

We base our analysis on a *semi-honest* security model, where the parties involved try to infer any information they can from the data presented to them, but they do not deviate from the protocol, nor do they collude with each other. Additionally, we assume that all communication takes place over secure channels and direct our analysis towards the privacy-preserving properties of PCDT. We treat PS and TS as adaptive, computationally-bound adversaries, whose goal is to obtain plaintext device data from the encrypted data shared by a device.

5.1.2. SECURITY OF DETERMINISTIC CDT

All the strings in X_d are hidden from PS using the BGV fully-homomorphic encryption scheme. The data derived from these strings, mainly the associated padding flags and AES-CMAC hashes, is also encrypted under BGV. Since the AES-CMAC hashes are computed homomorphically, as shown in Pseudocode 4, section 4.3.1, PS never learns the plaintext hashes of the strings provided by the device.

As PS only has access to BGV ciphertexts related to X_d, it follows that PCDT offers the same level of security as BGV for the data in X_d stored on PS. Since the BGV encryption scheme is proven to be IND-CPA secure under the assumption that the "General Learning with Errors" (GLWE) problem is hard (Theorem 5 in [34]), we conclude that PCDT offers IND-CPA security for the device data that is stored on PS and used in deterministic CDT, under the GLWE assumption.

The only information related to X_d received by TS consists of the collection of AES-CMAC hashes computed homomorphically by PS. The choice of using a keyed hash function as opposed to a non-keyed one results from the security consideration of addressing reverse lookup-table attacks. Since PCDT employs deterministic encryption to allow constant-time search operations, it does not offer semantic security for data in X_d stored on TS. As a result, if TS were able to compute hashes of values of is choice, the privacy of X_d would be compromised in two ways. First, for any bit-string m, TS would be able to find whether or not $m \in X_d$ by computing its hash and comparing it against the hashes received from PS. Second, by iteratively exploring the plaintext space of bit-strings in X_d, TS would be able to construct a table of hash values which, when matched against any hash received from PS, would allows TS to obtain the plaintext device data of any hashes received in the future.

Bellare et al. [37] offer an argument for the security of deterministic encryption under the assumption that the plaintext space has sufficiently large min-entropy. We note that the strings used for deterministic CDT do not meet this requirement, as they belong to specific domains, e.g. phone numbers, email addresses, etc.; which restrict the plaintext space considerably. To address this issue, we use the AES-CMAC function as a keyed hash, which allows us to extend the size of the plaintext space of deterministic CDT by a factor of 2^{128}, i.e. the size of the keyspace of AES-CMAC. This way, TS cannot hash values of its choice without knowing the key K_{AES}, which is kept secret by PS.

We base the security of data in X_d that is stored on TS on the security of AES-CMAC, which is in turn based on the security of AES-128. We say that a scheme has passive security, if there is no strategy better than brute-force for an adversary to obtain the plaintext M or the encryption key K from a ciphertext C obtained by encrypting M with K. We conclude that PCDT offers passive security for the data that is used in determinitic CDT and stored on TS, under the assumption that AES-128 offers passive security.

5.1.3. SECURITY OF PROBABILISTIC CDT

Since PS is the party with which the devices communicate, PCDT offers no security for IP addresses stored on PS. This is an unavoidable consequence of Internet communication, and we leave it to the device users to use VPNs or anonymizing networks if they wish to keep their IP addresses secret.

Since TS only receives the AES-CMAC hashes of the IP addresses collected by PS,

PCDT offers the same level of security for the device IP data stored on TS as it does for the data used in deterministic CDT, namely passive security.

All the data related to X_p and the GBDT models, including the training data used for the PS:GBDT-TRAIN procedure, are kept secret from PS using the BGV encryption scheme. Since both the evaluation and training algorithms operate on homomorphic ciphertexts, PS does not learn information about the device data from the machine learning operations used for probabilistic CDT. In similar fashion with the security of X_d on PS, we conclude that PCDT offers IND-CPA security for the device data that is stored on PS and used in probabilistic CDT, under the GLWE assumption.

Any BGV ciphertext of device data $x_p \in X_p$ is encrypted by PS using a symmetric encryption scheme before being sent to TS. The BGV ciphertexts of all threshold or leaf values of decision-tree nodes trained by PS are also encrypted using the same symmetric scheme. The decision to encrypt the threshold and leaf values of the GBDT model in order to keep them secret from TS was made to prevent TS from deriving information about elements of X_p from the results of evaluating the model.

Consider the case of a decision tree of depth 2, consisting of a non-leaf node t with threshold value $t.\theta$ and feature index $t.feature$, and two leaf nodes $t.left$ and $t.right$, with values $t.left.val = \alpha$ and $t.right.val = \beta$. If TS knows these values, then given an encrypted feature vector x_p and the plaintext result y of evaluating t on x_p, then TS learns that $x.p[t.feature] < t.\theta$ if $y = \alpha$, or $x.p[t.feature] \geq t.\theta$ if $y = \beta$.

Since both X_p and Π_p are stored on TS while encrypted under the symmetric key K_{SE}, which is kept secret by PS, it follows that PCDT offers the same level of security for the data in X_p stored on TS, as the symmetric encryption scheme used by PS. We conclude that PCDT offers IND-CCA security for the data in X_p stored on TS, if the symmetric scheme employed by PS is IND-CCA secure.

5.1.4. EFFECTS OF COLLUSION ON THE SECURITY OF PCDT

The security of the PCDT protocol relies on the fact that both server parties employ encryption schemes to keep the device data secret from each other. As a result, if PS and TS collude with one another, then they will both have the information necessary to decrypt the collected device data in its entirety.

By colluding with a device, TS can gain the ability to compute AES-CMAC hashes on chosen values, which can be leveraged to conduct a chosen-plaintext attack against the privacy of all device data used for deterministic CDT, as well as all device IP addresses. However, this type of collusion would be easily detectable by any monitoring or auditing party. Since PS acts as a proxy for TS, any communication between TS and any other party besides PS would fall under suspicion.

By colluding with a device, PS cannot learn anything about the data of other devices, as PS gains no additional capabilities from the collusion.

5.2. PERFORMANCE ANALYSIS

In this section we analyse the performance of PCDT with respect to both computational and communication complexities. Section 5.2.1 offers an evaluation of the depth of homomorphic circuits employed by PCDT procedures, while sections 5.2.2 and 5.2.3 contain theoretical analyses into the computational and communication complexities of PCDT procedures. In section 5.2.4, we examine the performance of deterministic and probabilistic privacy-preserving CDT as performed by an implementation of our construction.

Throughout this section, we divide the procedure TS:UPDATE-ASSOCIATIONS into its constituent methods, namely TS:UPDATE-DET and TS:UPDATE-PROB, in order to distinguish between the deterministic and probabilistic CDT functionalities offered by PCDT.

The symbols and notations used in this section follow those presented in Tables 4.1, 4.2, and 4.3, from section 4, with a few additions shown in Table 5.1.

Table 5.1: Performance analysis notations

Notation	Meaning
x	Input vector for the AGGREGATE procedure.
$\boxplus$	Binary operation passed as argument to procedures AGGREGATE and HE-BIT-AGGREGATE.
μ_d^*	Largest set of collected PII hashes.
μ_{IP}^*	Largest set of collected IP hashes.
ID_d	Set of device identifiers that share a PII string with the device considered by the procedure.
ID_{ip}	Set of device identifiers that share a relevant IP address with the device considered by the procedure procedure.
D	Data-set used to train the GBDT model.

5.2.1. HOMOMORPHIC CIRCUIT DEPTH

In this section we evaluate the depth of the homomorphic circuits used by the various algorithms employed by PCDT. The depth of a homomorphic circuit refers to the number of modulus switching operations performed on a ciphertext in order to maintain its validity, i.e. keep its noise within bounds. We denote by $\mathcal{L}(P)$ the circuit depth of a procedure P, and say that the evaluation of P consumes $\mathcal{L}(P)$ levels. Different homomorphic operations have different impacts on the noise magnitude of ciphertexts, which translates to different requirements for modulus switching [53]. We list the homomorphic operations employed by PCDT and their associated levels:

1. Homomorphic addition has a small impact on the noise magnitude, and does not increase the circuit depth.

2. Rotation automorphisms have a small impact on the noise magnitude, and do not increase the circuit depth.

3. Homomorphic multiplication of ciphertexts has a significant impact on the noise magnitude, and increases the circuit depth by 1.

4. Homomorphic multiplication between a ciphertext and a plaintext value has a moderate impact on the noise magnitude, and increases the circuit depth by 0.5.

Following the aforementioned rules, we evaluate the circuit depths of the auxiliary procedures employed by PCDT and show the results in Table 5.3. Using these values, we calculate the circuit depths of the PCDT main procedures, namely D_{id}:UPDATE, TS:UPDATE-DET, TS:UPDATE-PROB, and TS:UPDATE-MODEL, and show the results in Table 5.2.

The only homomorphic computations performed by the D_{id}:UPDATE procedure consist of evaluating the AES-CMAC hashes over homomorphic ciphertext space. As these hashes are computed independently, the circuit depth of D_{id}:UPDATE is equal to the depth of procedure HE-AES-CMAC, which is calculated as $\mathcal{L}$(HE-AES-CMAC) = $0.5 + \frac{5}{16}L_d$, based on the depth of HE-AES evaluated by Gentry et al. [50] as $\mathcal{L}$(HE-AES) = 40.

Procedure TS:UPDATE-DET has no homomorphic circuit, as all the homomorphic computations required for deterministic CDT are performed by D_{id}:UPDATE.

Procedure TS:UPDATE-PROB evaluates the GBDT model on each pair of device feature vectors independently. As a result, TS:UPDATE-PROB has the same circuit depth as procedure PS:GBDT-EVAL, which in turn has the same depth as PS:TREE-EVAL, since the trees in the ensemble are evaluated using independent circuits.

Finally, TS:UPDATE-MODEL delegates all homomorphic computations to PS:GBDT-TRAIN, so the two procedures have the same circuit depth. Procedure PS:GBDT-TRAIN trains each tree T_i on the residuals computed from the previous $i-1$ trees. As a result, the circuit will require n subsequent evaluations of the circuit of procedure PS:TREE-TRAIN. One HE-SUBTRACT operation is used to compute the first residual vector, while the next $n-1$ residual vectors are computed using one HE-SUBTRACT and one HE-MULTIPLY operation. Note that the elements of the residual vectors are computed independently from each other. The circuit depth of procedure PS:TREE-TRAIN is given by the depth of the circuit for training a single branch, which is comprised of $d_{max}-1$ circuits for training a non-leaf node and one circuit for training a leaf node. Training a non-leaf node involves the independent computation of errors for each feature-threshold combination, followed by running procedure PS:BEST-NODE to find the minimum error. Procedure PS:BEST-NODE uses a $\log(|D|)$-depth circuit that performs an HE-LESS-THAN operation and a homomoprhic multiplication at each level, therefore $\mathcal{L}$(PS:BEST-NODE) = $\log(|D|) \cdot (3.5 + \log(L_b))$. The circuit for training a leaf node involves three homomorphic multiplications, one use of the HE-DIVIDE procedure, and a $\log(|D|)$-depth circuit that uses HE-ADD at each level.

Based on these results, we compute the length of the modulus chain required by each BGV scheme involved in PCDT, such that the protocol can be executed without bootstrapping. We use $L_{BGV-AES}$ and L_{BGV-IA} to denote the number of required moduli for the BGV-AES and BGV-IA schemes respectively, and give their evaluations in Equations 5.1 and 5.2. Note that, since the GBDT model contains BGV ciphertexts resulted from the

Table 5.2: PCDT - Depth of homomorphic circuits (main procedures)

Procedure	Levels
D_{id}:UPDATE	$0.5 + \frac{5}{16} L_d$
TS:UPDATE-DET	0
TS:UPDATE-PROB	$\mathcal{L}$(PS:GBDT-EVAL)
TS:UPDATE-MODEL	$\mathcal{L}$(PS:GBDT-TRAIN)

Table 5.3: PCDT - Depth of homomorphic circuits (auxiliary procedures)

Procedure	Levels
HE-AES-CMAC	$0.5 + \frac{5}{16} L_d$
AGGREGATE	$\log(\lvert x \rvert) \cdot \mathcal{L}(\boxplus)$
HE-BITSHIFT	0.5
HE-BIT-AGGREGATE	$\log(L_b) \cdot \mathcal{L}(\boxplus)$
HE-LESS-THAN	$2.5 + \log(L_b)$
HE-ADD	$1.5 \log(L_b)$
HE-SUBTRACT	$3 \log(L_b)$
HE-MULTIPLY	$1.5 \log^2(L_b) + 1.5$
HE-DIVIDE	$1 + 3 \log(L_b) + 1.5 L_b$
PS:TREE-EVAL	$d_{max}(3.5 + \log(L_b))$
PS:TREE-TRAIN	$3 + 1.5 \log(\lvert D \rvert) \cdot \log(L_b) + (d_{max} - 1) \cdot (11 + 11.5 \log(L_b) + 3.5 \log(\lvert D \rvert) + 4 \log(\lvert D \rvert) \cdot \log(L_b) + 1.5 \log^2(L_b) + 3 L_b)$
PS:GBDT-EVAL	$d_{max}(3.5 + \log(L_b))$
PS:GBDT-TRAIN	$\mathcal{L}$(HE-SUBTRACT) + $(n-1) \cdot (\mathcal{L}$(HE-SUBTRACT) + $\mathcal{L}$(HE-MULTIPLY)) + $n \cdot \mathcal{L}$(PS:TREE-TRAIN)

evaluation of procedure PS:GBDT-TRAIN, the total amount of levels required for BGV-IA is $L_{BGV-IA} = \mathcal{L}$(TS:UPDATE-PROB) + $\mathcal{L}$(TS:UPDATE-MODEL).

$$L_{BGV-AES} = 0.5 + \frac{5}{16} L_d \tag{5.1}$$

$$\begin{aligned} L_{BGV-IA} = n \cdot ((d_{max} - 1) \cdot (11 + 11.5 \log(L_b) + 3.5 \log(|D|) + 4 \log(|D|) \cdot \log(L_b) \\ + 1.5 \log^2(L_b) + 3 L_b) + 4.5 + 1.5 \log(|D|) \cdot \log(L_b) + 3 \log(L_b) \\ + 1.5 \log^2(L_b)) - 1.5 \cdot (\log^2(L_b) + 1) + d_{max}(3.5 + \log(L_b)) \end{aligned} \tag{5.2}$$

Due to the slow scaling of the circuit depth involving the BGV-AES scheme, it is possible to employ BGV-AES without using bootstrapping for some small but sufficient values of L_d. On the other hand, it is clear that the BGV-IA scheme, which is used for probabilistic CDT, requires too many levels to evaluate all procedures without the use of bootstrapping, for any practical parameter values.

5.2.2. COMPUTATIONAL COMPLEXITY ANALYSIS

The BGV scheme [34] offers two initialization methods that affect the per-gate computational complexity in terms of the circuit depth L and the security parameter λ. Depending on the choice of BGV scheme, we can have the per-gate computational complexity be $\mathcal{O}(\lambda \cdot L^3)$ or $\mathcal{O}(\lambda^2)$, where the latter complexity is achieved by using bootstraping. In the implementation of HElib, a C++ library that implements the BGV and CKKS fully-homomorphic encryption schemes, Halevi et al. [54] use a construction in which a ciphertext can be used until a small amount L of levels have been consumed, at which point the ciphertext can be refreshed using bootstrapping to allow L more levels, and so on. Since this bootstrapping operation has computational complexity $\mathcal{O}(\lambda^2)$, the resulting per-gate complexity is still an amortised $\mathcal{O}(\lambda^2)$.

Let $Comp(P)$ denote the computational complexity of a procedure P. We denote by $\mathcal{O}_{BGV}$ the complexity class of homomorphic computations, such that $\mathcal{O}_{BGV}(1) \in \{\mathcal{O}(\lambda \cdot L^3), \mathcal{O}(\lambda^2)\}$.

Table 5.5 shows the computational complexities of the auxiliary procedures employed by PCDT. These results are used to evaluate the complexities of the four main procedures shown in Table 5.4.

Procedure D_{id}:UPDATE involves iterating over all elements in x_d and x_p and encrypting them under the corresponding BGV scheme. Additionally, the elements of x_d are separated into $\frac{1}{128}L_d$ blocks of 128 bits before encryption, which are then iterated over by the HE-AES-CMAC procedure. Overall, the computational complexity of D_{id}:UPDATE is $\mathcal{O}_{BGV}(|x_d| \cdot L_d + L_p)$.

Procedure TS:UPDATE-DET computes the deterministic association scores between a device D_{id} and all devices that share a hash with it, by counting the number of common hashes. As a result, $Comp$(TS:UPDATE-DET) $= \mathcal{O}(|\mu_d^*| \cdot |ID_d|)$.

Procedure TS:UPDATE-PROB performs an IP-based down-sampling technique by iterating over the sets of collected IP addresses with $\mathcal{O}(\mu_{IP}^* \cdot |ID_{ip}|)$ computational complexity, and invokes TS:GBDT-EVAL for each device pair selected by the sampling procedure. Procedure TS:GBDT-EVAL invokes PS:GBDT-EVAL, and then performs n decryption operations. Procedure PS:GBDT-EVAL first decrypts the feature vector x_p, then iterates over all n trees in the ensemble, evaluating each one using the PS:TREE-EVAL procedure. In turn, procedure PS:TREE-EVAL evaluates each node using a constant amount of homomorphic operations, including an HE-LESS-THAN operation.

Procedure TS:UPDATE-MODEL iterates over all device pairs and their IP addresses in order to construct the training data set which is passed to procedure TS:GBDT-TRAIN. The computational complexity of TS:GBDT-TRAIN is equal to that of procedure PS:GBDT-TRAIN, which iterates over all n trees in the ensemble, training each one using the PS:TREE-TRAIN procedure. When training a non-leaf node, procedure PS:TREE-TRAIN iterates over all $2L_p \cdot |S_\theta|$ combinations of feature index and threshold, and calculates the error corresponding to each pair. The computation of one error involves an amount linear in $|D|$ of homomorphic multiplications, HE-LESS-THAN, HE-ADD, HE-SUBTRACT, and HE-MULTIPLY operations, as well as a constant amount of HE-DIVIDE and AGGREGATE operations, where $\boxplus$ = HE-ADD, and $|x| = |D|$. The procedure PS:BEST-NODE iterates over the $2L_p \cdot |S_\theta$ elements of V_e, performing one HE-LESS-THAN operation and a constant amount of homomorphic additions and multiplications at each step. The data

partitions D_L and D_R are computed using $2|D|$ homomorphic multiplications, while the prediction vector P is computed using an additional $|D|$ homomorphic additions. The complexity class of training a leaf node is lower than that of computing a single error for a non-leaf node.

Table 5.4: PCDT - Computational complexities (main procedures)

Procedure	Complexity
D_{id}:UPDATE	$\mathcal{O}_{BGV}(\|x_d\| \cdot L_d + L_p)$
TS:UPDATE-DET	$\mathcal{O}(\|\mu_d^*\| \cdot \|ID_d\|)$
TS:UPDATE-PROB	$\mathcal{O}(\mu_{IP}^* \cdot \|ID_{ip}\| \cdot L_p) + \mathcal{O}_{BGV}(\mu_{IP}^* \cdot \|ID_{ip}\| \cdot n \cdot L_b \cdot 2^{d_{max}})$
TS:UPDATE-MODEL	$\mathcal{O}(\|ID\|^2 \cdot \|\mu_{IP}^*\|) + \mathcal{O}_{BGV}(n \cdot 2^{d_{max}} \cdot \|D\| \cdot \|S_\theta\| \cdot L_p \cdot L_b^2)$

Table 5.5: PCDT - Computational complexities (auxiliary procedures)

Procedure	Complexity
HE-AES-CMAC	$\mathcal{O}_{BGV}(L_d)$
AGGREGATE	$\mathcal{O}(\|x\|) \cdot Comp(\boxplus)$
HE-BITSHIFT	$\mathcal{O}_{BGV}(1)$
HE-BIT-AGGREGATE	$\mathcal{O}_{BGV}(\log(L_b))$
HE-LESS-THAN	$\mathcal{O}_{BGV}(L_b)$
HE-ADD	$\mathcal{O}_{BGV}(L_b)$
HE-SUBTRACT	$\mathcal{O}_{BGV}(L_b)$
HE-MULTIPLY	$\mathcal{O}_{BGV}(L_b^2)$
HE-DIVIDE	$\mathcal{O}_{BGV}(L_b \cdot \log(L_b))$
PS:TREE-EVAL	$\mathcal{O}_{BGV}(L_b \cdot 2^{d_{max}})$
PS:TREE-TRAIN	$\mathcal{O}_{BGV}(2^{d_{max}} \cdot \|D\| \cdot \|S_\theta\| \cdot L_p \cdot L_b^2)$
TS:GBDT-EVAL	$\mathcal{O}(L_p) + \mathcal{O}_{BGV}(n \cdot L_b \cdot 2^{d_{max}})$
PS:GBDT-EVAL	$\mathcal{O}(L_p) + \mathcal{O}_{BGV}(n \cdot L_b \cdot 2^{d_{max}})$
TS:GBDT-TRAIN	$\mathcal{O}_{BGV}(n \cdot 2^{d_{max}} \cdot \|D\| \cdot \|S_\theta\| \cdot L_p \cdot L_b^2)$
PS:GBDT-TRAIN	$\mathcal{O}_{BGV}(n \cdot 2^{d_{max}} \cdot \|D\| \cdot \|S_\theta\| \cdot L_p \cdot L_b^2)$

5.2.3. COMMUNICATION COMPLEXITY ANALYSIS

In this section we use $\mathcal{O}_{BGV}$ to denote the communication complexity related to transferring BGV ciphertexts over the communication channels, where $\mathcal{O}_{BGV}(1)$ represents the size of one ciphertext.

Table 5.6 shows the communication complexities of all PCDT procedures that transfer data between parties.

Procedure D_{id}:UPDATE sends the BGV-encrypted set x_d and vector x_p to PS, which then forwards $|x_d|$ BGV-encrypted AES-CMAC hashes and the doubly-encrypted x_p to TS. Since the number of ciphertexts required to encrypt x_d depends on L_d, and $|x_p| = L_p$, the overall communication complexity of D_{id}:UPDATE is $\mathcal{O}_{BGV}(|x_d| \cdot L_d + L_p)$. It is worth noting that the bit-length L_b used to represent the integer values in x_p does not factor

into the communication complexity, since all the bits of an integer are encrypted within a single ciphertext.

Procedure TS:UPDATE-PROB invokes procedure TS:GBDT-EVAL for every device found to share a relevant IP address with D_{id}, where id is he parameter passed when invoking TS:UPDATE-PROB. Procedure TS:GBDT-EVAL sends a doubly-encrypted feature vector of length $2L_p$ to PS, along with the tree ensemble to be evaluated. PS returns a single ciphertext, which is inconsequential.

Note that an efficient implementation of TS:UPDATE-PROB would have TS only send the ensemble once, together with all the sampled feature vectors. The pseudocode however is left as is for readability purposes.

TS:UPDATE-MODEL invokes procedure TS:GBDT-TRAIN, which sends the data set D to PS, each element consisting of $2L_p+1$ BGV ciphertexts. PS replies with an encrypted decision tree ensemble, which is comprised of $1.5n \cdot 2^{d_{max}}$ ciphertexts.

Table 5.6: PCDT - Communication complexities

Procedure	Complexity
D_{id}:UPDATE	$\mathcal{O}_{BGV}(\|x_d\| \cdot L_d + L_p)$
TS:UPDATE-PROB	$\mathcal{O}_{BGV}(\|ID_{ip}\| \cdot (L_p + n \cdot 2^{d_{max}}))$
TS:UPDATE-MODEL	$\mathcal{O}_{BGV}(\|D\| \cdot L_p + n \cdot 2^{d_{max}}))$
TS:GBDT-EVAL	$\mathcal{O}_{BGV}(L_p + n \cdot 2^{d_{max}})$
TS:GBDT-TRAIN	$\mathcal{O}_{BGV}(\|D\| \cdot L_p + n \cdot 2^{d_{max}}))$

5.2.4. EXPERIMENTAL ANALYSIS

The most computationally-intensive procedures of PCDT are those that involve homomorphic computations, such as HE-AES-CMAC, PS:TREE-EVAL, and PS:TREE-TRAIN. To test the performance of our protocol, we build a C++ implementation of these procedures [74] and use it to analyse their run-time through a series of experiments. Our implementation makes use of the implementation of the BGV homomorphic encryption scheme offered by the HElib [52] open-source library.

The experiments described in this section were performed on the TU Delft HPC cluster, which uses the CentOS Linux operating system, allocating one CPU core, model Intel® Xeon™ E5-2620 v4 2.10GHz, and 16 GB of memory to each experiment. We run each experiment 10 times and average the results.

Table 5.7 shows the parameters used to initialize each BGV scheme involved in our experiments. Since all operations are performed over binary values, the prime modulus is set to 2 for every scheme. The parameter m represents the order of the cyclothomic polynomial used by each scheme, and the lists denoted as $gens$ and $ords$ represent the generators used to decide the dimensionality of the ciphertext hypercube, and their respective orders. The number of generators decides the number of dimensions of the ciphertext hypercube, while the order of each generator decides the number of slots in each dimension. The generators have to be specifically selected such that the product of their orders is equal to the totient of m divided by the multiplicative order of 2 modulo m. Finally, $\log(q)$ denotes the bit-length of the modulus chain.

For each scheme, we are interested in a set of parameters that offer a specific dimensionality of the ciphertext hypercube, a modulus chain that allows for a practical number of subsequent homomorphic computations without bootstrapping, and a security level above 80, which is considered sufficient by existing research [51].

The HE-AES-CMAC procedure requires that the underlying BGV scheme is initialized such that one dimension of the hypercube is a multiple of 16. The scheme is also initialized with a sufficiently long modulus chain to evaluate up to three AES blocks without bootstrapping. For the case of deterministic CDT, we argue that a length of 32 or 48 bytes would be sufficient for any PII string, and therefore the evaluation of HE-AES-CMAC does not need to employ bootstrapping.

For the PS:TREE-EVAL and PS:TREE-TRAIN procedures, we are interested in parameters that produce ciphertexts in which one dimension of the hypercube holds a number of slots equal to the number of bits L_b chosen for our integer representation. The length of the modulus chain is chosen to obtain the required security level, while the homomorphic circuit depth is decided by the bootstrapping operation.

Table 5.7: BGV parameters

Scheme	m	$gens$	$ords$	$\log(q)$	Security level
BGV-AES	65281 $= 97 \cdot 673$	{43073, 22214}	{96, −14}	1800	83
BGV-IA (8-bit integers)	32317 $= 17 \cdot 1901$	{3}	{8}	600	118
BGV-IA (16-bit integers)	35377 $= 17 \cdot 2081$	{3, 725}	{2, 16}	600	139
BGV-IA (32-bit integers)	58803 $= 3 \cdot 17 \cdot 1153$	{11, 5}	{4, 32}	600	166

Figure 5.1 shows the average run-time of the HE-AES-CMAC procedure on input strings of one, two, and three 128-bit blocks. Based on these results, we argue that a device profile update would, in practice, take between 10 and 15 minutes per PII string, depending on the choice of L_d, and with PS performing most of the computations. As these hashes are computed independently, PS could leverage parallelism to speed up the process.

To study the performance of the PS:TREE-TRAIN procedure, we measure its average run-time in relation to the depth of the tree and the selected integer bit-length. The tree-depth values used in our experiments are chosen based on existing implementations of GBDT [58], which tend to use fairly short trees, with depth lower than 10, in order to prevent overfitting. PCDT requires for the integer bit-length L_b to be a power of two. While modern computers use 32 or 64 bits to represent integer values, we consider a 16-bit representation sufficient for most use cases of our protocol. In addition to the 16-bit in-

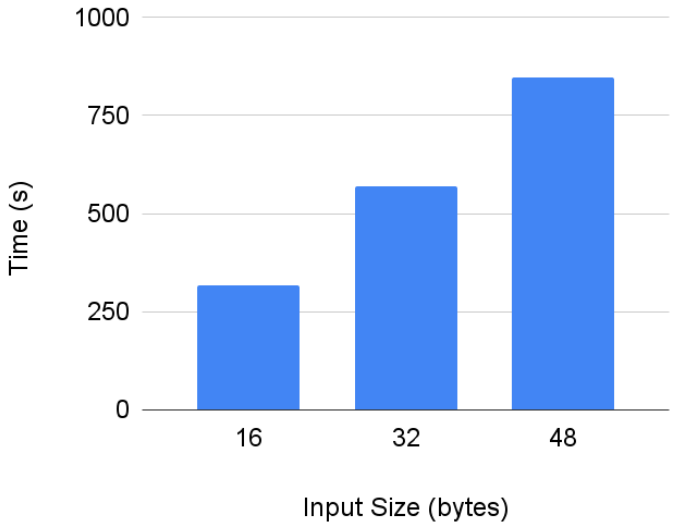

Figure 5.1: Run-time analysis of the HE-AES-CMAC operation based input size.

teger representation, we also explore the 8-bit representation, which provides the highest speed while still being potentially useful in practice, and the 32-bit representation, which may be used in applications that involve large numbers or require high-precision rational-number computations.

The experimental results shown in Figure 5.2 indicate that PCDT can homomorphically evaluate a decision tree with depth 6 or lower in under 40 minutes, for any integer representation, and as fast as 10 minutes for the 8-bit representation. For depth values above 6, the exponential scaling in the number of nodes creates a substantial difference in run-time based on the integer representation, from 40 minutes for 8 bits, to 75 minutes for 16 bits, and 160 minutes for 32 bits, in the case of a tree with depth 8.

Due to unresolved technical issues, our implementation of PS:TREE-EVAL does not employ bootstrapping, and instead mimics it by re-encrypting the ciphertexts when necessary. As a result, the values offered in Figure 5.2 serve as a lower-bound on the actual execution times of a method that employs bootstrapping.

In their implementation of BGV, Halevi and Shoup [54] offer a fast bootstrapping operation named *thin bootstrapping*, optimised for schemes initialised with prime modulus 2. The experimental results shown in their paper indicate that this operation can re-encrypt homomorphic ciphertexts larger than the ones used in our experiments in less than 5 minutes.

Through our experiments, we evaluate an average number of 2, 3, and 4, bootstrapping operations per tree level required by the 8-bit, 16-bit, and 32-bit setups respectively. Given these results, we estimate that the run-time of PS:TREE-EVAL would increase by at most 10, 15, and 20 minutes per tree-level, for each respective setup.

In order to perform probabilistic CDT, PCDT employs the PS:TREE-EVAL procedure

to evaluate an ensemble of trees. Existing implementations indicate that the usual number of trees in a GBDT ensemble exceeds 100 [58]. By assuming an ensemble containing 100 predictors, we estimate that the computational time required to evaluate the probabilistic association score of a pair of devices ranges from three days to three weeks, depending on other parameters. This could however be sped up trough the use of parallel computations, since the trees can be evaluated independently.

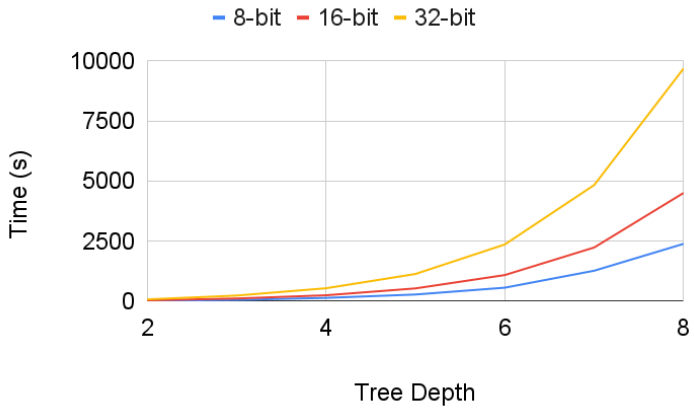

Figure 5.2: Run-time analysis of the PS:TREE-EVAL operation based tree depth.

Unfortunately, a functioning implementation of PS:TREE-TRAIN could not be completed in time for its performance to be analysed experimentally, and as a result, we cannot offer an estimate of the time required for PCDT to train a GBDT model.

6

Discussion and Future Work

In this section we give an overview of our contribution and how it relates to the problem at hand. To this end, we return to the research questions formulated in the introduction section and offer answers based on the results of our exploration. We also discuss the limitations of our work and give a list of possible improvements that can be explored in future research. Finally, this section ends with a few concluding remarks which reiterate the contributions presented in this thesis in relation to the motivations behind our study.

6.1. Discussion

We first restate our research question and its four sub-questions.

Research question: *How can a system become aware of which devices that access an online service are used by the same user, such that no party has access to a user's data besides the user who owns it ?*

Sub-questions:

1. How can user data be made available for the purpose of CDT, without disclosing the actual meaning of the data ?
2. How can deterministic CDT be performed without knowledge of a user's personally identifiable information ?
3. How can probabilistic CDT be performed without knowledge of a user's inferred data ?
4. How efficient, and therefore practical, is privacy-preserving CDT ?

The overall design of PCDT answers the main research question by demonstrating a practical method of performing cross-device tracking in a privacy-preserving manner. Our protocol uses a two-server setting, in the semi-honest security model, and employs cryptographic techniques such as fully-homomorphic encryption, symmetric encryption, and keyed hashing, in order to construct a device association graph through both

deterministic and probabilistic CDT without either server learning the plaintext device data.

To address the first sub-question, our protocol makes use of three cryptographic schemes to maintain the secrecy of device data while allowing the server parties to perform CDT-related computations. First, fully-homomorphic encryption is used to hide the plaintext data from PS, while allowing the evaluation of machine-learning algorithms and the AES-CMAC function over homomorphic ciphertext space. Furthermore, to keep the device data secret from TS, PS will homomorphically compute the AES-CMAC hashes of the data used for deterministic CDT, and use a symmetric encryption scheme to doubly-encrypt the ciphertexts of the data used for probabilistic CDT. AES-CMAC hashes are also used to conceal the device IP addresses from TS.

The use of a keyed hash function protects the secrecy of device data and IP addresses from reverse lookup table attacks, while allowing TS to perform both deterministic CDT and the IP-related down-sampling method used in probabilistic CDT at the same speed as if operating over plaintext values. The two-fold encryption used on data collected for probabilistic CDT conceals the data from both servers and allows PS to evaluate and train decision tree models using encrypted values, and TS to retrieve the plaintext results of model evaluations, which are used to perform probabilistic CDT.

PCDT performs deterministic CDT by matching hashes of the device data instead of plaintext strings. The hash values are computed using the keyed hash function AES-CMAC, in order to protect the data against reverse table lookup attacks. If the values were simply hashed without a secret key, TS would be able to compute hashes of chosen values and, by comparing them against the collected hashes, compromise the secrecy of the device data. In order to conceal the plaintext strings from PS, they are first encrypted using the BGV fully-homomorphic encryption scheme. By operating on these homomorphic ciphertexts, PS is able to obtain the BGV-encrypted AES-CMAC hashes, which are then decrypted by TS to obtain the deterministic hashes. Since these hashes are computed deterministically, and considering that the chance of a hash collision is negligible for AES-CMAC, two devices sharing the same hash indicates the existence of a common data point used for deterministic CDT.

The protocol computes the deterministic association score between two devices as the number of hashes they have in common. This score is then compared against a threshold value, to decide whether or not the two devices should be linked in the device association graph. In practical applications, the importance of a single match may depend on the type of data, e.g. a shared email address may be more conclusive than a shared home address. While PCDT assumes that all strings collected for deterministic CDT are equally important, the protocol can be easily extended to the general case by using separate hash sets for each category of data, and multiplying the number of matches in each category by a weight factor representing the relevance of the category.

In terms of performance, the use of deterministic hashes allows PCDT to perform deterministic CDT just as fast as if it was operating on plaintext values. The only increase in computation related to deterministic CDT comes from the evaluation of AES-CMAC over homomorphic ciphertext space, an operation which takes place during the data collection phase, i.e. the update device profile operation described in section 4.2.2. As shown in section 5.2.4, one evaluation of the hash function takes approximately 10 minutes for

a practical input length of 32 characters. We consider this to be sufficiently fast for most CDT applications.

PCDT performs probabilistic CDT by training and evaluating a gradient-boosting decision tree model on BGV-encrypted data. Unlike the case of online behavioural advertisement, the users do not provide any feedback with which to measure the accuracy of CDT techniques. As a result, the model used to perform probabilistic CDT is trained to predict the association scores obtained through deterministic CDT, a decision based on the assumption that deterministic CDT is the more reliable of the two.

The protocol computes the probabilistic association score of a pair of devices as the result of the GBDT model when evaluated on the concatenated feature vectors supplied by both devices. This score is then compared against a threshold value to decide whether or not to link the two devices in the device association graph. Because we expect a very large number of device pairs, we consider it unfeasible to evaluate the model on all device pairs. As a result, when constructing a set of device-pair data for either model training or evaluation, we apply an IP-based down-sampling rule that limits the amount of device pairs considered in the data set. According to this rule, a pair of devices is further considered for probabilistic CDT only if there is an IP address from which both devices have connected to PS within a relevant time-frame.

Since the devices communicate with PS directly, the device IP addresses are known to PS, while being hidden from TS in the form of AES-CMAC hashes. In a similar fashion as when performing deterministic CDT, the deterministic hashes of the IP addresses allow TS to quickly down-sample the device data in accordance to the aforementioned IP-based rule.

In order to prevent TS from deriving information about the device data from the GBDT model or the results of its evaluations, the model produced by PCDT is encrypted by PS using a symmetric encryption scheme. As a consequence, TS cannot evaluate models that were not produced by the training algorithm of PCDT, or learn the plaintext of the trained model, perform model optimization techniques, such as pruning. This poses substantial limitations on the practical uses of PCDT with respect to privacy-preserving probabilistic CDT.

In terms of performance, the results shown in section 5.2.4 indicate that a large number of trees in the GBDT ensemble, which exceeds 100 in most practical applications, would render the probabilistic CDT performed by our protocol infeasible without significant use of parallelism.

The evaluation procedure stands to benefit considerably from parallel computation, as the trees in the ensemble can be evalauted independently from each other. Furthermore, the evaluation of each individual decision tree model can be sped up through parallelism since each child subtree of a non-leaf node can be evaluated independently from its sibling.

While the decision trees in the ensemble have to be trained sequentially, as each new tree model is trained on the residuals produced by the already models, the training algorithm can still benefit from parallelism in three ways. First, the child sub-trees of a non-leaf node are trained independently from each other, and can therefore be trained in parallel. Second, the errors associated with each feature-threshold pair can be computed in parallel, as they do not depend on each other. Third, computing the minimum

error and its associated feature index and threshold values is similar to the procedure of evaluating a decision tree, and can therefore benefit from parallelism in a similar fashion.

6.2. FUTURE WORK

To the best of our knowledge, PCDT is the first protocol designed to perform cross-device tracking while preserving the privacy of the collected device data. However, it explores only a narrow approach to the issue of privacy-preserving CDT, resulted from our choices concerning the balance between security and performance. As a result, a wide variety of improvements and alternative approaches can be explored in future research, several of which are described in this section.

Identifying users on shared devices. One aspect of CDT which is not accounted for in our protocol refers to the ability of tracking services to differentiate between multiple users on the same device. In practice, this is usually performed based on behavioural changes in the browsing activity of a device at different time instances. PCDT assumes that all data collected from a device corresponds to a single user, and does not offer any procedures for analysing the encrypted behaviour data to detect multiple users. One possible extension that would allow PCDT to account for shared devices is to perform the user differentiation locally, i.e. on the device-side software, then maintain multiple device profiles, one for each separate user. However, further exploration into this issue is required to verify that this extension is sufficient for identifying users on shared devices.

Multi-server setting. PCDT relies on the assumption that TS and PS do not collude with one another. If PS and TS were to collude, the secrecy of all collected device data would be compromised. One way of addressing this issue, and making privacy-preserving CDT more resilient to collusion attacks, would be to use a multi-server setting where a group of n servers engage in secure multi-party computation to perform CDT in a federated manner. This scheme could make use of secret sharing, oblivious transfer, or multi-key homomorphic encryption to perform CDT while preserving the secrecy of the data in a k-out-of-n model of security, i.e. a collusion attack will not compromise the security of the device data if the number of colluding servers is less than k.

Use plaintext ML models. A major downside of PCDT comes from the fact that the machine learning evaluation procedure only works on models produced by the training procedure of the protocol. Additionally, these models are encrypted and kept secret from the tracking service, so that the service cannot derive information about the device data from the results of the model evaluation. As a consequence, the tracking service cannot use models trained by other means, nor can it perform model optimization techniques, such as pruning, on the encrypted model. In order to overcome these limitations, one must first address the issue of information leakage associated with the model evaluation procedure. To prevent information leakage, a privacy-preserving CDT protocol would have to either obfuscate the connection between an input value and its result, or change the evaluation procedure such that it outputs an encrypted result, which can be used to link devices in the device association graph without revealing its plaintext value. Once the model evaluation procedure has been modified to evaluate plaintext models without revealing information about the input data, the training procedure would no longer need to encrypt the output model to keep it secret from TS, and TS would be able to provide

arbitrary models to the model evaluation procedure.

Increase performance by encoding multiple integer values into one ciphertext. In our construction, each homomorphic ciphertext used to perform probabilistic CDT encodes a single integer value. While the privacy-preserving machine learning algorithms employed by PCDT make use of SIMD operations to speed up homomorphic integer operations, such as addition and comparison, further speedup can be achieved by encoding multiple integer values within a single homomorphic ciphertext. While not trivial, it is possible to adapt the algorithms used to perform homomorphic integer operations such that they perform vector-wise operations within the same computational complexity as their element-wise counterparts. By packing a vector of integers within a single ciphertext, and leveraging SIMD operations, the resulting scheme would offer significantly reduced execution times for privacy-preserving probabilistic CDT, especially in the case of the model training procedure.

Use a more secure searchable encryption scheme for deterministic CDT. Deterministic encryption allows PCDT to perform deterministic CDT over encrypted data at a speed similar to performing it over plaintext data. On the other hand, deterministic encryption does not offer security against chosen-plaintext attacks, and therefore, the secrecy of device data used for deterministic CDT relies on the assumption that TS is unable to encrypt values of its choosing. Should TS gain the ability to compute AES-CMAC hashes under the secret key of PS, either by colluding with a device or by impersonating a device, the secrecy of the PII used for deterministic CDT would be compromised. Searchable encryption schemes that use probabilistic encryption can be employed to address this issue and offer a more secure protocol for privacy-preserving deterministic CDT, at the expense of performance. When using probabilistic encryption, the computational complexity of finding the set of devices that share a common data point is at least linear in the number of devices considered in the search, which is expected to be very large. Further research would have to be conducted into the possibility of using such schemes in practice, as well as methods of reducing their computational complexity. One such method that could be explored in future works consists of reducing the search space based on additional device information. For example, devices could be grouped based on geographical proximity using their location data, and only devices within the same geographical region would be considered when searching for common data points.

6.3. CONCLUDING REMARKS

The objective of our research is to reduce the impact of cross-device tracking on the privacy of online users by offering a privacy-preserving technological alternative to standard CDT practices. Through our solution, we seek to offer an argument against the necessity of collecting and storing plaintext sensitive user information for the purpose of cross-device tracking. To this end we design PCDT, a protocol for constructing device association graphs using both deterministic and probabilistic techniques, that operates on encrypted data.

PCDT employs fully-homomorphic encryption and the AES-CMAC keyed hash function to perform deterministic CDT in a privacy-preserving manner. The use of deterministic hashes allows privacy-preserving deterministic CDT to be performed at a speed

comparable to its non-privacy-preserving counterpart, but offers very limited security, and the secrecy of device data relies on the secrecy of the key used to compute the AES-CMAC hashes.

Probabilistic CDT is performed by training and evaluating machine learning models on device data encrypted using fully-homomorphic encryption. Based on existing research in the field of probabilistic CDT, we have chosen gradient-boosting decision trees as the machine learning algorithm employed by PCDT. In the case of probabilistic CDT, the increase in user privacy comes with a significant decrease in speed, to the point where it would require significant use of parallelism to be considered practical.

Finally, we implement two of the most computationally-intensive PCDT procedures in the C++ programming language, using the BGV implementation provided by the HElib library, and evaluate their performance through experimental analysis.

In addition to performing deterministic CDT, we also use the homomorphic evaluation of AES-CMAC to construct a multi-user searchable encryption scheme, which is described in detail in the paper presented in Appendix A.

REFERENCES

[1] Paul W. Farris, Neil Bendle, Phillip E. Pfeifer, and David Reibstein. “Marketing metrics: The definitive guide to measuring marketing performance". Pearson Education, 2010.

[2] M. A. Bashir, S. Arshad, W. Robertson, and C. Wilson. “Tracing Information Flows Between Ad Exchanges Using Retargeted Ads". In Proceedings of the 25th USENIX Security Symposium. Aug. 2016, pp. 481–496.

[3] L. Olejnik, T. Minh-Dung, and C. Castelluccia. “Selling Off Privacy at Auction". In Network and Distributed System Security Symposium. NDSS ’14. Feb. 2014.

[4] J. Estrada-Jiménez, J. Parra-Arnau, A. Rodríguez-Hoyos, and J. Forné. “Online Advertising: Analysis of Privacy Threats and Protection Approaches". In: Computer Communications 100 (Mar. 2017), pp. 32–51.

[5] PageFair. “2021 PageFair Adblock Report". 3 May 2021. https://blockthrough.com/blog/2021-adblock-report/

[6] Georg Merzdovnik, Markus Huber, Damjan Buhov, Nick Nikiforakis, Sebastian Neuner, Martin Schmiedecker, and Edgar Weippl. “Block me if you can: A large-scale study of tracker-blocking tools." In 2017 IEEE European Symposium on Security and Privacy (EuroS&P), pp. 319-333. IEEE, 2017.

[7] Leon J. Helsloot. “Preserving Privacy through Cryptography in Online Behavioural Advertising". 2017.

[8] Leon J. Helsloot, Gamze Tillem, and Zekeriya Erkin. “AHEad: privacy-preserving online behavioural advertising using homomorphic encryption." In 2017 IEEE Workshop on Information Forensics and Security (WIFS), pp. 1-6. IEEE, 2017.

[9] Leon J. Helsloot, Gamze Tillem, and Zekeriya Erkin. “Badass: Preserving privacy in behavioural advertising with applied secret sharing." In International Conference on Provable Security, pp. 397-405. Springer, Cham, 2018.

[10] Michael Backes, Aniket Kate, Matteo Maffei, and Kim Pecina. “Obliviad: Provably secure and practical online behavioral advertising." In 2012 IEEE Symposium on Security and Privacy, pp. 257-271. IEEE, 2012.

[11] Saikat Guha, Bin Cheng, and Paul Francis. “Privad: Practical privacy in online advertising." In USENIX conference on Networked systems design and implementation, pp. 169-182. 2011.

[12] Vincent Toubiana, Arvind Narayanan, Dan Boneh, Helen Nissenbaum, and Solon Barocas. “Adnostic: Privacy preserving targeted advertising." In Proceedings Network and Distributed System Symposium. 2010.

[13] Justin Brookman, Phoebe Rouge, Aaron Alva, and Christina Yeung. “Cross-device tracking: Measurement and disclosures." Proceedings on Privacy Enhancing Technologies 2017, no. 2 (2017): 133-148.

[14] Andres Corrada and David Tannenbaum. “System and method for creating a scored device association graph." U.S. Patent Application 14/492,642, filed March 26, 2015.

[15] Adbrain. “Demystifying Cross-Device". https://www.iabuk.com/sites/default/files/white-paper-docs/Adbrain-Demystifying-Cross-Device.pdf

[16] Franziska Roesner, Tadayoshi Kohno, and David Wetherall. “Detecting and defending against third-party tracking on the web." In Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12), pp. 155-168. 2012.

[17] Gunes Acar, Marc Juarez, Nick Nikiforakis, Claudia Diaz, Seda Gürses, Frank Piessens, and Bart Preneel. “FPDetective: dusting the web for fingerprinters." In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, pp. 1129-1140. 2013.

[18] Konstantinos Solomos, Panagiotis Ilia, Sotiris Ioannidis, Nicolas Kourtellis: “TALON: An Automated Framework for Cross-Device Tracking Detection". RAID 2019: 227-241

[19] Iain Thomson. “How TV ads silently ping commands to phones: Sneaky SilverPush code reverse-engineered". The Register. Nov. 2015. https://www.theregister.com/2015/11/20/silverpush_soundwave_ad_tracker/

[20] Edith Ramirez, Maureen K. Ohlhausen, Terrell McSweeny. “Cross-device tracking: An FTC staff report". Tech. rep., 2017

[21] Comments collected on FTC Workshop on Cross-Device Tracking, 16th of November, 2015. https://www.ftc.gov/policy/public-comments/2015/03/initiative-603

[22] Amy Pittman. “The Internet Thinks I’m Still Pregnant", N.Y. TIMES (Sept. 2, 2016). http://www.nytimes.com/2016/09/04/fashion/modern-love-pregnancy-miscarriage-app-technology.html

[23] Marc Groman. “No One Should Be Outed By an Ad", IAPP (Feb. 24, 2015). https://iapp.org/news/a/nai-takes-lgbtstand/

[24] Nicholas Confessore. “Cambridge Analytica and Facebook: The Scandal and the Fallout So Far", N.Y. TIMES (April 4, 2018). https://www.nytimes.com/2018/04/04/us/politics/cambridge-analytica-scandal-fallout.html

[25] Christopher Mims. “The Hacked Data Broker? Be Very Afraid", WALL ST. J. (Sept. 8, 2015). http://www.wsj.com/articles/the-hacked-data-broker-be-very-afraid-1441684860.

[26] Brian Krebs. “Was the Ashley Madison Database Leaked?", KREBS ON SEC. (Aug. 18, 2015). http://krebsonsecurity.com/2015/08/was-the-ashley-madison-databaseleaked/

[27] Roi Perez. “Dropbox Hack Confirmed Real", SC MAGAZINE (Aug. 31, 2016). http://www.scmagazineuk.com/dropboxhack-confirmed-real-68-million-accounts-affected/article/519545/

[28] Brian Krebs. “Data Breach at Health Insurer Anthem Could Impact Millions", KREBS ON SEC. (Feb. 4, 2015). http://krebsonsecurity.com/2015/02/data-breach-at-health-insureranthem-could-impact-millions/

[29] Yehuda Lindell and Benny Pinkas. “Privacy preserving data mining." In Annual International Cryptology Conference, pp. 36-54. Springer, Berlin, Heidelberg, 2000.

[30] Rakesh Agrawal and Ramakrishnan Srikant. “Privacy-preserving data mining." In Proceedings of the 2000 ACM SIGMOD international conference on Management of data, pp. 439-450. 2000.

[31] Valeria Nikolaenko, Udi Weinsberg, Stratis Ioannidis, Marc Joye, Dan Boneh, and Nina Taft. “Privacy-preserving ridge regression on hundreds of millions of records." In 2013 IEEE Symposium on Security and Privacy, pp. 334-348. IEEE, 2013.

[32] Payman Mohassel and Yupeng Zhang. “Secureml: A system for scalable privacy-preserving machine learning." In 2017 IEEE symposium on security and privacy (SP), pp. 19-38. IEEE, 2017.

[33] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. “Federated machine learning: Concept and applications." ACM Transactions on Intelligent Systems and Technology (TIST) 10, no. 2 (2019): 1-19.

[34] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. “(Leveled) fully homomorphic encryption without bootstrapping." ACM Transactions on Computation Theory (TOCT) 6, no. 3 (2014): 1-36.

[35] Nigel P. Smart, F. Vercauteren. “Fully homomorphic simd operations". IACR Cryptology ePrint Archive, 2011.

[36] Craig Gentry, Shai Halevi, Nigel P. Smart. “Fully homomorphic encryption with polylog overhead". EUROCRYPT 2012. LNCS, vol. 7237, pp. 465–482. Springer, Heidelberg (2012).

[37] Mihir Bellare, Alexandra Boldyreva, and Adam O’Neill. “Deterministic and efficiently searchable encryption." In Annual International Cryptology Conference, pp. 535-552. Springer, Berlin, Heidelberg, 2007. (Full version at www.cc.gatech.edu/~aboldyre/papers/bbo.pdf)

[38] Emiliano De Cristofaro and Gene Tsudik. “On the performance of certain private set intersection protocols." IACR (2012): 54.

[39] Yunlu Cai, Chunming Tang, and Qiuxia Xu.“Two-Party Privacy-Preserving Set Intersection with FHE." Entropy 22, no. 12 (2020): 1339.

[40] Christoph Bösch, Pieter Hartel, Willem Jonker, and Andreas Peter. “A survey of provably secure searchable encryption." ACM Computing Surveys (CSUR) 47, no. 2 (2014): 1-51.

[41] Rohit Handa, C. Rama Krishna, and Naveen Aggarwal. “Searchable encryption: A survey on privacy-preserving search schemes on encrypted outsourced data." Concurrency and Computation: Practice and Experience 31, no. 17 (2019): e5201.

[42] Changyu Dong, Giovanni Russello, and Naranker Dulay. “Shared and searchable encrypted data for untrusted servers." Journal of Computer Security 19, no. 3 (2011): 367-397.

[43] Feng Bao, Robert H. Deng, Xuhua Ding, and Yanjiang Yang. “Private query on encrypted data in multi-user settings." In International Conference on Information Security Practice and Experience, pp. 71-85. Springer, Berlin, Heidelberg, 2008.

[44] Peishun Wang, Huaxiong Wang, and Josef Pieprzyk. “An efficient scheme of common secure indices for conjunctive keyword-based retrieval on encrypted data." In International workshop on information security applications, pp. 145-159. Springer, Berlin, Heidelberg, 2008.

[45] National Institute for Science, Technology (NIST): Advanced Encryption Standard (FIPS PUB 197) (November 2001), https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf

[46] Junhyuk Song, Radha Poovendran, Jicheol Lee, and Tetsu Iwata. “The AES-CMAC algorithm." RFC 4493, 2006.

[47] 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, no. 1 (2015): 188-199.

[48] Zongsheng Hou, Neng Zhang, and Leibo Liu. “An Efficient FHE Radix-2 Addition Algorithm in BGV Scheme." In Journal of Physics: Conference Series, vol. 1993, no. 1, p. 012030. IOP Publishing, 2021.

[49] Yao Chen and Guang Gong. “Integer arithmetic over ciphertext and homomorphic data aggregation." In 2015 IEEE Conference on Communications and Network Security (CNS), pp. 628-632. IEEE, 2015.

[50] Craig Gentry, Shai Halevi, and Nigel P. Smart. “Homomorphic evaluation of the AES circuit (updated implementation)." Jan 3 2015.

[51] Silvia Mella and Ruggero Susella. “On the homomorphic computation of symmetric cryptographic primitives." In IMA International Conference on Cryptography and Coding, pp. 28-44. Springer, Berlin, Heidelberg, 2013.

[52] Shai Halevi and Victor Shoup. “Design and implementation of a homomorphic-encryption library." IBM Research (Manuscript) 6, no. 12-15 (2013): 8-36.

[53] Shai Halevi and Victor Shoup. “Algorithms in helib." In Annual Cryptology Conference, pp. 554-571. Springer, Berlin, Heidelberg, 2014.

[54] Shai Halevi and Victor Shoup. “Bootstrapping for helib." In Annual International conference on the theory and applications of cryptographic techniques, pp. 641-670. Springer, Berlin, Heidelberg, 2015.

[55] “Regulation (EU) 2016/679 (General Data Protection Regulation)." April 2016. https://eugdprcompliant.com/personal-data/

[56] Congress.gov. “S.2521 - 113th Congress (2013-2014): Federal Information Security Modernization Act of 2014." December 18, 2014. https://www.congress.gov/bill/113th-congress/senate-bill/2521

[57] Jerome H. Friedman. “Greedy function approximation: a gradient boosting machine." Annals of statistics (2001): 1189-1232.

[58] Jason Brownlee. “How to Configure the Gradient Boosting Algorithm." Machine Learning Mastery. (September 12, 2016) https://machinelearningmastery.com/configure-gradient-boosting-algorithm/

[59] J. R. Quinlan. “Induction of decision trees." Machine learning 1, no. 1 (1986): 81-106.

[60] Loh, Wei-Yin. “Classification and regression trees." Wiley interdisciplinary reviews: data mining and knowledge discovery 1, no. 1 (2011): 14-23.

[61] Wu, Xindong, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan et al. “Top 10 algorithms in data mining." Knowledge and information systems 14, no. 1 (2008): 1-37.

[62] Hyafil Laurent and Ronald L. Rivest. “Constructing optimal binary decision trees is NP-complete." Information processing letters 5, no. 1 (1976): 15-17.

[63] Kenneth P. Burnham and Davfd R. Anderson. “A practical information-theoretic approach." *Model selection and multimodel inference, 2nd edition* (2002).

[64] Jianxun Lian and Xing Xie. “Cross-device user matching based on massive browse logs: The runner-up solution for the 2016 CIKM Cup." arXiv preprint arXiv:1610.03928 (2016).

[65] Minh C. Phan, Yi Tay, and Tuan-Anh Nguyen Pham. “Cross device matching for online advertising with neural feature ensembles: First place solution at CIKM Cup 2016." arXiv preprint arXiv:1610.07119 (2016).

[66] Girma Kejela and Chunming Rong. “Cross-device consumer identification." In 2015 IEEE International Conference on Data Mining Workshop (ICDMW), pp. 1687-1689. IEEE, 2015.

[67] Roberto Díaz-Morales. “Cross-device tracking: Matching devices and cookies." In 2015 IEEE International Conference on Data Mining Workshop (ICDMW), pp. 1699-1704. IEEE, 2015.

[68] Jeremy Walthers. “Learning to rank for cross-device identification." In 2015 IEEE International Conference on Data Mining Workshop (ICDMW), pp. 1710-1712. IEEE, 2015.

[69] Can Karakaya, Hakan Toğuç, Rıdvan Salih Kuzu, and Ali Hakan Büyüklü. “Survey of cross device matching approaches with a case study on a novel database." In 2018 3rd International Conference on Computer Science and Engineering (UBMK), pp. 139-144. IEEE, 2018.

[70] Elena Volkova. “Cross-device tracking with machine learning." Master’s thesis, 2017.

[71] Xiaodong Xiao, Ting Wu, Yuanfang Chen, and Xingyue Fan. “Privacy-Preserved Approximate Classification Based on Homomorphic Encryption." Mathematical and Computational Applications 24, no. 4 (2019): 92.

[72] Adi Akavia, Max Leibovich, Yehezkel S. Resheff, Roey Ron, Moni Shahar, and Margarita Vald. “Privacy-Preserving Decision Tree Training and Prediction against Malicious Server."

[73] Anselme Tueno, Yordan Boev, and Florian Kerschbaum. “Non-interactive private decision tree evaluation." In IFIP Annual Conference on Data and Applications Security and Privacy, pp. 174-194. Springer, Cham, 2020.

[74] PCDT implementation: https://github.com/babeanu-dorian/PCDT

A

MUSE: Multi-user Searchable Encryption

This section offers a paper on a new multi-user searchable encryption scheme that uses the homomorphic evaluation of AES-CMAC to achieve constant-time storage and search complexities.

A

Multi-user searchable encryption using homomorphically-computed hashes

Alexandru Babeanu✉ and Zekeriya Erkin[0000−0001−8932−4703]

Cyber Security Group,
Delft University of Technology, Delft, Netherlands
babeanu.dorian@gmail.com, Z.Erkin@tudelft.nl

Abstract. Conventional encryption is sufficient to preserve the secrecy of data stored on untrusted servers, but comes at the cost of reduced usability, as encrypted data cannot be easily queried. Searchable encryption allows the storage server to perform search queries on encrypted data on behalf of a client, without learning the plaintext data. Existing multi-user searchable encryption schemes compromise on speed to meet security requirements, and as a result offer search complexity at least linear in the number of stored entries. Deterministic encryption of search keywords can be used to achieve constant search complexity, but this approach makes a scheme vulnerable to lookup table attacks. In this paper, we present MUSE, an interactive multi-user searchable encryption protocol, which uses homomorphically computed keyed hashes to offer both constant search complexity and resilience against lookup table attacks. As is the case with most searchable encryption schemes, our protocol encrypts search keywords separately from the associated data. The search keywords are hashed under the secret keys of two distinct parties. Fully homomorphic encryption is employed to evaluate the hash function without revealing the plaintext value to either party. This way, searches become simple table-lookup operations, while neither party is able to individually compute arbitrary hashes and construct lookup tables. The actual data entries are encrypted using a proxy re-encryption scheme, which allows clients to share data access with each other.

Keywords: searchable encryption · homomorphic encryption · remote data storage · secure data storage · privacy

1 Introduction

Remote storage services offer data storage, as well as data availability, accessibility, security, and redundancy. To reduce operational costs, or sometimes due to regulatory reasons, many organisations make use of such services to store their data. However, these services cannot always be trusted, as server administrators would have access to the stored data, as would threat agents in case of a security breach. Conventional encryption allows clients to securely store sensitive information on an untrusted server, but hinders their ability to query the data.

Searchable encryption (SE) allows a server to search in encrypted data on behalf of a client without learning information about the plaintext data. Some schemes encrypt the data in a manner that allows searches to be performed directly on the ciphertext, while others require the client to generate one or more encrypted indices associated with the data. Most SE schemes require for each document stored on the server to be associated with a set of keywords, which can be used in search queries to retrieve matching documents. Based on the number of parties that are allowed to store encrypted data on a server, SE schemes are divided into single-writer and multi-writer schemes. Additionally, depending on the amount of parties that can query and retrieve the stored data, SE schemes are divided into single-reader and multi-reader schemes. A multi-writer/multi-reader SE scheme, also called a multi-user SE scheme, matches the functionality of a generic database, since it allows multiple parties to have store and search capabilities.

In conventional databases, search and storage operations on indexed data have logarithmic or constant time complexity [1], depending on the data structures used in the implementation. In this paper we consider a table lookup operation to have constant time complexity. Most multi-user SE schemes have at best linear complexity for search operations in order to meet security requirements. Bellare et al. [16] describe how deterministic encryption can be used to achieve constant search and storage complexity, and evaluate the security of such schemes. The main issue with deterministic encryption is that it does not offer semantic security, and is therefore susceptible to lookup table attacks. Bellare et al. give an alternative definition of security which deems a scheme that uses deterministic encryption secure on the assumption that the plaintext space has high min-entropy and is therefore resilient to lookup table attacks. This assumption limits the type of data that can be used as search keywords. For example, in the case of storing personally identifiable information, data such as phone numbers, birth dates, names, or email addresses would be desirable search keywords, but also highly susceptible to lookup table attacks due to low plaintext space.

In this paper we present MUSE, an interactive multi-user SE protocol, which makes use of deterministic encryption to achieve constant search time while being resilient to lookup table attacks without requiring a high min-entropy plaintext space. Similarly to most SE schemes, our protocol encrypts the search keywords separately from their associated documents, to allow the server to query the data. These keywords are hashed under the keys of two distinct parties using an interactive algorithm built on the homomorphic evaluation of a keyed hashing algorithm. Similar to other SE schemes [12, 18, 19], MUSE employs the use of a third-party to help with privacy-preserving computations. More specifically, this third-party manages the keys of a fully homomorphic encryption scheme, namely Brakerski et al.'s BGV [21] scheme. The interactive hashing algorithm starts with the client encrypting a keyword value under the BGV encryption scheme and sending the cyphertext to the data storage server. The server evaluates the AES-CMAC [25] hash of the keyword by operating on the homomorphic ciphertext, keying the hash with its own secret key. The result is forwarded to the

third-party, which decrypts the intermediate value and computes the final hash using its own secret key. Thus, a keyed hash is computed without either server learning the original value or the full hash key. Because the hashing algorithm is deterministic, searching for a specific keyword requires a simple table-lookup operation of constant complexity. Additionally, because neither server knows the other one's hash key, they cannot individually compute arbitrary hashes and construct a lookup table. The documents associated with these keywords are encrypted using Shao and Cao's unidirectional proxy re-encryption scheme [28]. This way, clients can share access to their data with each other by enabling the server to re-encrypt documents such that specific clients are able to decrypt them.

2 Related Work

IND1-CKA security, introduced by Goh in [5], describes index security as semantic security against adaptive chosen keywords attacks. The notion of index security requires that the contents of an encrypted document cannot be deduced from its index, i.e. the encrypted form of the keywords associated with it. However, index security does not require the trapdoors of the search queries to be secure. Curtmola et al. [6] introduced two new adversarial models for SE, an adaptive model, IND-CKA1, and a non-adaptive one, IND-CKA2. The IND-CKA models require that nothing is leaked from the remotely stored files or their indices, beyond the outcome and the search pattern of the queries. Additionally, the authors introduce the concept of trapdoor security, which require that query trapdoors do not leak information about the keywords besides what can be inferred from the search and access patterns. IND-CKA1 and IND-CKA2 are considered the standard definitions of security for SE. Bellare et al. [16] introduce the notion of PRIV security for SE schemes that use deterministic encryption. A scheme that offers PRIV security is considered *sufficiently secure* if its plaintext space has high min-entropy. Shen et al. [10] formulate the definition of *full security*, which involves index, trapdoor, and search-pattern privacy.

In the single-writer/single-reader category, the schemes presented in [6–9], achieve optimal search efficiency, i.e. linear in the number of documents that contain the query keyword and constant in everything else. They achieve this by using deterministic keyword hashes for indexing documents and lookup tables for searching. Also in the single-writer/single-reader category, the schemes [10] and [11] use the inner-product between the trapdoor and the searchable content to find the documents that match the search query. This way, they achieve search-pattern privacy at the expense of a search complexity linear in the total number of keywords. Bösch et al. [11] also mention an extension to their scheme which offers access-pattern privacy.

In the single-writer/multi-reader category, Raykova et al. [12] propose a scheme which uses reroutable encryption, and Yang et al. [13] propose a scheme which uses bilinear maps. The scheme in [12] uses a third-party responsible for user authentication and query re-encryption. The scheme achieves sub-linear

search complexity by using deterministic encryption, at the expense of semantic security. The scheme in [13] offers index and query privacy, as well as a search complexity constant in the number of keywords and documents. The main drawback of [13] is the complexity of the storage operation, linear in the number of keywords and documents. Curtmola et al. [6] propose a construction which uses broadcast encryption on top of a single-writer/single-reader SE scheme.

Most multi-writer/single-reader schemes are variations of PEKS [14], the first SE scheme to use asymmetric encryption. Proposed by Boneh et al. in 2004, the scheme uses identity-based encryption, where the keyword acts as the identity. Due to the use of asymmetric encryption, multiple users can generate searchable content encrypted under the public key of the single reader, and only the holder of the private key can query or decrypt the data. Both the store and search operations are constant in the total amount of documents and keywords. In terms of security, the scheme is vulnerable to offline keyword-guessing attacks [20], in which an attacker matches a search-query trapdoor against a table of keyword ciphertexts computed using the reader's public key, to obtain the keywords in the search query. Abdalla et al. [15] offer an improved PEKS scheme, which uses hierarchical identity-based encryption to generate trapdoors that are only valid in a specific time interval. This time interval prevents the server from matching a trapdoor with past or future ciphertexts outside the chosen interval, which makes the scheme more resilient against offline keyword-guessing attacks.

Most multi-writer/multi-reader schemes offer search complexity linear in the total number of stored documents, except for [16]. Most of them use a third-party for user authentication, and sometimes for interactive encryption protocols. Bellare et al. [16] propose a multi-writer/multi-reader SE scheme, which makes the encrypted keywords searchable by appending a hash of the keyword to its ciphertext. This makes both the storage and the search operations very efficient, i.e. constant in the total amount of keywords and documents, but lacks semantic security, and the server can derive the keywords from their hashes through a dictionary attack. Dong et al. [17] propose a protocol which uses an El-Gamal proxy re-encryption scheme in combination with a collision-resistant hash function, in a three-party setting. The protocol employs a fully trusted key-management server, which is separate from the data-storage server, and generates the key pairs used by users to encrypt the search keywords. The key-management server also acts as an access manager that grants and revokes the users' permission to query the data-storage server. The keyword ciphertexts are initially encrypted under each user's secret key, and are then re-encrypted by the server so that they can be matched with queries produced by any user. The protocol offers semantic security under the assumption that the data-storage server is not an authenticated user and cannot generate queries on its own. In terms of efficiency, the storage operation is linear in the amount of keywords associated with a document, while the search operation is linear in the total amount of keyword-document pairs. Bao et al. [18] propose a scheme which uses bilinear maps to allows users with different secret keys to generate the same search index for a given keyword. This scheme introduces a trusted third-party to manage user credentials, which al-

lows for authenticated search queries and the revocation of a user's permission to query the database. The scheme offers query privacy as well as query unforgeability, i.e. only registered users can query the database and users cannot impersonate each other. In terms of efficiency, the storage operation is linear in the amount of keywords associated with a document, while the search operation is linear in the total amount of documents in the database. Wang et al. [19] propose a protocol which allows users to perform conjunctive search queries on the encrypted data, i.e. a document is retrieved only if it matches all the keywords in the search query. The protocol uses a dynamic accumulator for user authentication and a combinatorial accumulator to build a search index from a padded list of hashed keywords. It also makes use of a semi-honest third-party which assists the users in decrypting the data retrieved via search queries. The protocol offers semantic security against chosen-keyword attacks. With regards to efficiency, the storage operation is linear in the maximum amount of keywords associated with a document, and the search operation is linear in the total amount of documents.

3 Cryptographic Preliminaries

We provide a summary of the cryptographic preliminaries employed by our protocol, namely BGV homomorphic encryption, AES-CMAC, and proxy re-encryption.

3.1 BGV Homomorphic Encryption

BGV [21] is an asymmetric non-deterministic fully-homomorphic encryption scheme based on the ring learning with errors problem (RLWE). Both ciphertexts and secret keys are vectors over a polynomial ring $R = \mathbb{Z}[X]/\Phi_m(X)$, which represents the ring of integers over the m-th cyclotomic number field. The plaintext space is the space of polynomials $R_p = R/pR = \mathbb{Z}[X]/(\Phi_m(X), p)$, for some fixed $p \geq 2$, which represents the set of integer polynomial of degree up to $\phi(m) - 1$, reduced modulo p. Since this paper is only concerned with the homomorphic evaluation of binary circuits, the plaintext space is restricted to the space of binary polynomials, R_2.

BGV is fully homomorphic, allowing the computation of both addition and multiplication on encrypted data. Given two ciphertexts $[x]$ and $[y]$, encoding ring elements x and y, a party can compute both $[x+y]$ and $[xy]$ without knowing x or y. At any point in the homomorphic evaluation, there is a current secret key s under which the ciphertext is valid, and a current modulus q, both of which change as the homomorphic evaluation progresses. The polynomial $[< c, s > \mathtt{mod}\ \Phi_m(X)]_q$, obtained from computing the inner product over R_q between the ciphertext vector c and the current secret key s, where q is the current modulus, represents the noise of the ciphertext c. A ciphertext c is *valid* with respect to key s and modulus q if the magnitude of its noise is sufficiently small relative to q, so that it does not wrap around q when performing homomorphic operations.

Homomorphic addition has no effect on the current modulus or key, and the noise magnitude is at most the sum of noises of the arguments. Homomorphic multiplication does not change the current modulus, but it does change the key. If the two input ciphertexts are valid under an n-dimension key s, then the output ciphertext is valid under the n^2-dimension key equal to the tensor product of s with itself. The scheme offers a *key switching* operation, which is used to reduce the size of the key after a multiplication. With respect to noise, the multiplication operation will produce a ciphertext with a noise magnitude equal to the product of the noises of the arguments. Since this may cause the noise to grow too large, the *modulus switching* operation is used before each multiplication. The scheme is instantiated with a list of L moduli, $q_0 < q_1 < ... < q_{L-1}$, where L is the depth of the homomorphic circuit to be evaluated. After a ciphertext reaches the last modulus in the list, it can no longer be used for homomorphic computations, except by using bootstrapping.

Smart and Vercauteren show in [22] that the structure of the BGV plaintext space allows for the evaluation of single instructions, multiple data (SIMD) operations. By setting m odd, the plaintext space R_2 is isomorphic to the direct sum of l copies of $\mathrm{GF}(2^d)$, where $l = \phi(m)/d$. This way, plaintexts can be treated as l-vectors of elements of $\mathrm{GF}(2^d)$, where arithmetic operations over plaintexts corresponds to element-wise operations over l-vectors. The elements of these l-vectors are called slots. In [23], Gentry et al. show how to permute the contents of these slots through the use of automorphisms over R_2, as well as how to raise slot contents to powers of 2 through the use of Frobenius automorphisms.

3.2 AES-CMAC

The Cipher-based Message Authentication Code (CMAC) algorithm is a keyed hash function based on a symmetric-key block cipher. The AES-CMAC [25] algorithm is a specification of CMAC that uses the Advanced Encryption Standard (AES) [26] algorithm as its underlying block cipher.

AES-CMAC takes as input a 128-bit key K and an arbitrary-length message m. Using K, two derived keys K_1 and K_2 are computed. The message m is divided into n blocks of 128 bits, denoted as m_i, where $i \in \{1, ..., n\}$. If necessary, 10^*-padding is used so that the bit-length of m is a multiple of 128. Let $\texttt{AES}(x, K)$ be the output of AES-128 on block x and key K, and let $K_x = K_2$ if the message m was padded, and $K_x = K_1$ otherwise, then the 128-bit output of AES-CMAC is computed as shown in Pseudocode 1.

Pseudocode 1 AES-CMAC algorithm

```
1: procedure AES-CMAC(m_1, m_2, ..., m_n, K, K_x)
2:     h ← AES(m_1, K)
3:     for i ← 2, ..., (n − 1) do
4:         h ← AES(h ⊕ m_i, K)
5:     end for
6:     return AES(h ⊕ m_n ⊕ K_x, K)
7: end procedure
```

In [27], Gentry et al. describe how to modify the operations of AES-128 in order to evaluate the cipher over BGV-encrypted data. Their approach uses packed ciphertexts, which encode one or more 128-bit blocks of data and are operated on using automorphisms and SIMD operations. The polynomial Φ_m is chosen so that it factors modulo 2 into at least 16 irreducible polynomials of degree d, with d divisible by 8. This way, each slot holds one byte of the data, and the number of slots is large enough to store an entire block.

Building on the homomorphic evaluation of AES-128, we can implement the homomorphic evaluation of AES-CMAC for messages with bit-length divisible by 128. Pseudocode 1 can be converted to represent the homomorphic evaluation of AES-CMAC with the following changes in notation:

1. m_i represents the BGV encryption of the i-th 128-bit message block
2. K_x is the no-pad key derived from K
3. $\texttt{AES}(x, K)$ represents the homomorphic evaluation of AES-128
4. $\oplus$ represents homomorphic addition over R_2

3.3 Proxy re-encryption (PRE)

A proxy re-encryption scheme allows a party A to delegate access to data encrypted under its public key pk_A to a different party B, without revealing its secret key sk_A to B. This process is facilitated by a proxy party, which uses a re-encryption mechanism to transform a ciphertext of a message m under a public key pk_A into a ciphertext of the same message m, encrypted with a different public key pk_B. In [28], J. Shao and Z. Cao present a unidirectional single-hop proxy re-encryption scheme that offers IND-CCA security. Their construction uses signatures of knowledge and the Fijisaki-Okamoto transformation, and relies on the Decisional Diffie-Hellman security assumption. The operations that comprise the scheme are as follows:

Setup: Takes as input the bit-length n of messages to be encrypted, and two security parameters k_1 and k_2, and outputs three cryptographic hash functions $H_1 : \{0,\ 1\}^* \to \{0,\ 1\}^{k_1}$, $H_2 : \{0,\ 1\}^* \to \{0,\ 1\}^n$, and $H_3 : \{0,\ 1\}^* \to \{0,\ 1\}^{k_2}$.
Key Generation: Chooses two safe-prime numbers p and q, used to compute the cyclic group order $N^2 = (pq)^2$, and outputs three keys pk, sk, and wsk, where pk is the public encryption key, sk is the long-term secret key, and wsk is the *weak* secret key.
Re-encryption Key Generation: On input a public key pk_Y, a *weak* secret key wsk_X, and the long-term secret key sk_X, outputs the unidirectional re-encrypion key $rk_{X \to Y}$.
Encryption: On input a public key pk and a message $m \in \{0,\ 1\}^n$, outputs the ciphertext $[m]_{pk}$.
Re-encryption: On input a re-encryption key $rk_{X \to Y}$ and a ciphertext $[m]_{pk_X}$, outputs the re-encrypted cyphertext $[m]_{rk_Y}$.
Decryption: On input a ciphertext $[m]_{pk_X}$ or $[m]_{rk_X}$, and a *weak* secret key wsk_X or a long-term secret key sk_X, outputs the plaintext message m.

4 Multi-User Searchable Encryption (MUSE)

Our Multi-User Searchable Encryption (*MUSE*) protocol allows users to store data on a remote server without compromising its secrecy and perform search queries on it with constant time complexity. Additionally, users can query and retrieve data stored by other users, provided they have permission from the user who uploaded it.

4.1 Setting

The setting in which our protocol operates contains two parties of interest, the data storage server (**DS**) and the privacy service (**PS**), as well as any amount of client parties ($\mathtt{C}_i$), where i is a unique client identifier.

$\mathtt{C}_i$ represents the data owner that wants to store and retrieve documents to and from the cloud in a privacy-preserving manner. To store a document, it provides DS with the encrypted document d and an associated set of keywords $\{w_1, ..., w_n\}$, also in encrypted form. To retrieve one or more documents, it provides DS with the encryption of a keyword w, and receives the set of encrypted documents $\{d_1, ..., d_m\}$ associated with w. These documents are encrypted using a proxy re-encryption scheme, which allows clients to share access to the encrypted data with each other.

DS represents the server on which data is stored. It allows authorised clients to store encrypted documents associated with encrypted keywords, as well as retrieve encrypted documents based on encrypted keywords, while deriving no knowledge about the plaintext documents or keywords.

PS represents a semi-honest third-party server which assists DS in hashing the search keywords, without deriving any knowledge about the plaintext documents or keywords.

These parties are assumed to be *honest-but-curious*, meaning that they follow the protocol but derive as much knowledge as possible from the data presented to them. All communication is assumed to be performed over secure channels.

4.2 Protocol Algorithms

Now we describes the algorithms that compose the presented SE protocol: setup, compute hash, grant access, store, retrieve, and revoke access. Table 1 explains the notations used throughout the section.

Setup. PS generates the key pair (pk_{BGV}, sk_{BGV}) used for the BGV cryptosystem and shares the public key pk with DS. PS also generates a 128-bit secret key K_{PS} used to key the AES-CMAC algorithm. DS initializes all the relevant parameters for the proxy re-encryption scheme, and shares the public values with the clients. DS also generates the secret keys K_{DS1} and K_{DS2}, used for the homomorphic computation of AES-CMAC. K_{DS1} is used for the homomorphic computation of AES-128, and K_{DS2} is derived from K_{DS1} using the AES-CMAC key derivation algorithm. DS maintains a mapping $\mu : \{0,1\}^{128} \rightarrow P(I \times D)$,

Table 1: Symbols and Notations

Notation	Description
$[x]_{pk}$	The encryption of x under the public key pk, using the scheme corresponding to pk.
L_w	The fixed bit-length of a padded keyword string, must be a multiple of 128.
$\|x\|$	The bit-length of string x.
0^x	The bit-string cnsisting of x repetitions of 0.
$\|\|$	The concatenation operator.
$pad10^*(x, L)$	$x\|\|10^{L-\|x\|-1}$
$HeAES\ ([x]_{pk},\ K)$	The homomorphic computation of AES-128 with key K, where $[x]_{pk}$ is the BGV-encryption of a 128-bit block x.
$AES\text{-}CMAC\ (x,\ K)$	The computation of AES-CMAC on input string x under AES key K.
$ReKeyGen\ (sk_i,\ pk_j)$	The procedure which generates the re-encryption key $rk_{i\to j}$.
$Dec\ ([x]_{pk}, sk)$	The decryption of $[x]_{pk}$ using the secret key sk, equivalent to x.
$ReEnc\ ([x]_{pk_i},\ rk_{i\to j})$	The re-encryption of $[x]_{pk_i}$ into $[x]_{pk_j}$, using the re-encryption key $rk_{i\to j}$.

where D is the set of encrypted documents, I is the set of client identifiers, and $P(I \times D)$ is the power set of identifier-document pairs. This mapping associates keyword hashes to the corresponding sets of encrypted documents. Initially $\mu(x) = \emptyset$ for all $x \in \{0,1\}^{128}$. DS also maintains a mapping $\rho : I \times I \to R$, where I is the set of client identifiers and R is the set of re-encryption keys that can be generated by the proxy re-encryption scheme. This mapping associates a re-encryption key to each ordered pair of clients. Initially, $\rho(i, j) = \perp$ for all $i, j \in I$. Each client $\mathtt{C}_i$ generates a pair of keys $(pk_i,\ sk_i)$ according to the key generation procedure of the proxy re-encryption scheme, where pk_i denotes the public key used to encrypt the documents that are stored on DS, and sk_i denotes the concatenation on the *weak* and long-term secret keys used for decryption.

Compute hash. Pseudocode 2 describes the auxiliary hashing algorithm performed to maintain the secrecy of the search keywords. DS receives an encrypted string $[x]_{pk_{BGV}}$, composed of a list of $L_w/128$ ciphertexts, where each ciphertext $[x_i]_{pk_{BGV}}$ represents the BGV encryption of the i-th 128-bit block of the input string x. It then homomorphically computes the AES-CMAC hash of x using the algorithm described in section 3.2. This encrypted hash is sent to PS, which decrypts it and uses it to compute another hash, using its own key and the regular AES-CMAC algorithm. It is worth mentioning that under the assumption that DS is honest-but-curios, the *compute hash* algorithm is only run at the request of a client, and that DS cannot use it to compute hashes of arbitrary values and construct a lookup table. In practice, this constraint can be enforced by having PS verify that a client initiated the operation.

Grant access. Pseudocode 3 describes the grant access algorithm which allows a client $\mathtt{C}_i$ to give any other client $\mathtt{C}_j$ access to the documents $\mathtt{C}_i$ has stored on DS. C_i generates a re-encryption key $rk_{i\to j}$ using the procedure described in

section 3.3. C_i then sends $rk_{i\to j}$ to DS, allowing DS to re-encrypt documents stored by C_i so that C_j can decrypt them.

Pseudocode 2 Compute hash algorithm

```
1: procedure DS:HASH([x]_{pk_BGV})
2:     [h]_{pk_BGV} ← [0^128]_{pk_BGV}
3:     for i ← 1...(L_w/128) do
4:         if i = L_w/128 then
5:             [h]_{pk_BGV} ←
                   [h + K_DS2]_{pk_BGV}
6:         end if
7:         [h]_{pk_BGV} ← HeAES(
               [h + x_i]_{pk_BGV}, K_DS1)
8:     end for
9:     return PS:HASH([h]_{pk_BGV})
10: end procedure

11: procedure PS:HASH([h]_{pk_BGV})
12:     h ← Dec([h]_{pk_BGV}, sk_BGV)
13:     return AES-CMAC(h, K_PS)
14: end procedure
```

Pseudocode 3 Grant access algorithm

```
1: procedure C_i:GRANT-ACCESS(j)
2:     rk_{i→j} ← ReKeyGen(sk_i, pk_j)
3:     DS:GRANT-ACCESS(i, j, rk_{i→j})
4: end procedure

5: procedure DS:GRANT-ACCESS
                          (i, j, rk_{i→j})
6:     ρ(i, j) ← rk_{i→j}
7: end procedure
```

Store. Pseudocode 4 describes the store algorithm used to upload documents to the data server along with a set of associated keywords. $\mathtt{C}_i$ first pads the keywords so that they all have the same length L_w. Afterwards, it encrypts them under the BGV scheme, using the encryption key pk_{BGV} received from PS. The document d is encrypted under the PRE scheme using the public key pk_i. The encrypted document and keywords are then sent to DS. DS computes the hash of each keyword using the *compute hash* algorithm, and updates μ so that the hash of each keyword will now be mapped to a set that includes the encryption of d. Each document is paired with the identifier of the client that uploaded it so that it can be re-encrypted using the correct keys when retrieved by other clients.

Search. Pseudocode 5 returns the list of encrypted documents associated with the provided keyword w. $\mathtt{C}_i$ first pads w so that it has length L_w, then sends the BGV-encryption of the padded keyword to DS. DS computes the hash of the keyword using the *compute hash* algorithm, and retrieves the set of documents that corresponds to said hash in the mapping μ. Based on the client identifier paired with each encrypted document, DS decides which ciphertexts will be returned to $\mathtt{C}_i$. If a document d is encrypted under the public key of a client different from i, and $\mathtt{C}_i$ is authorised to access it, i.e. a re-encryption key is available in ρ, then DS will re-encrypt d so that $\mathtt{C}_i$ can decrypt it using private key sk_i.

Revoke access. Pseudocode 6 allows a client $\mathtt{C}_i$ to revoke the access it granted to a client $\mathtt{C}_j$. This algorithm only requires that DS removes the corresponding re-encryption key from ρ. This way, the documents uploaded by $\mathtt{C}_i$ will be ignored in queries performed by $\mathtt{C}_j$, since $\mathtt{C}_j$ would no longer be able to decrypt them.

Pseudocode 4 Store algorithm

```
1: procedure C_i:STORE
                          ({w_1, w_2, ..., w_n}, d)
2:     W ← {[pad10*(w_j, L_w)]_{pk_BGV} |
           j ∈ {1, ..., n}}
3:     DS:STORE(i, W, [d]_{pk_i})
4: end procedure
5: procedure DS:STORE(i, W, [d]_{pk_i})
6:     for all [w]_{pk_BGV} ∈ W do
7:         h ← DS:HASH([w]_{pk_BGV})
8:         μ(h) ← μ(h) ∪ {(i, [d]_{pk_i})}
9:     end for
10: end procedure
```

Pseudocode 5 Search algorithm

```
1: procedure C_i:SEARCH(w)
2:     D_w ← ∅
3:     R ← DS:SEARCH(i,
           [pad10*(w, L_w)]_{pk_BGV})
4:     for all [d]_{pk_i} ∈ R do
5:         D_w ← D_w ∪ {Dec([d]_{pk_i}, sk_i)}
6:     end for
7:     return D_w
8: end procedure
9: procedure DS:SEARCH
                          (i, [w]_{pk_BGV})
10:    h ← DS:HASH([w]_{pk_BGV})
11:    R ← ∅
12:    for all (j, [d]_{pk_j}) ∈ μ(h) do
13:        if i = j then
14:            R ← R ∪ {[d]_{pk_j}}
15:        else if ρ(i, j) ≠ ⊥ then
16:            [d]_{pk_i} ←
                   ReEnc([d]_{pk_j}, ρ(i, j))
17:            R ← R ∪ {[d]_{pk_i}}
18:        end if
19:    end for
20:    return R
21: end procedure
```

Pseudocode 6 Revoke access algorithm

```
1: procedure C_i:REVOKE-ACCESS(j)
2:     DS:REVOKE-ACCESS(i, j)
3: end procedure
4: procedure DS:REVOKE-ACCESS(i, j)
5:     ρ(i, j) ← ⊥
6: end procedure
```

5 Security Evaluation

In this section we define the notions of security relevant for SE schemes and evaluate the security of our protocol with regards to these definitions under the assumption that each party is an *honest-but-curious* actor.

Definition 1 (Read-access). *Let d be a document stored on DS by a party C_i. A party C_j has read-access to d if and only if $i = j$ or there exists a valid re-encryption key $rk_{i \to j}$ stored by DS as a result of the grant access algorithm.*

Definition 2 (Document privacy). *A multi-user searchable encryption scheme offers document privacy if and only if no information can be derived about a document plaintext from its ciphertext without the private key of a party that has read-access to said document.*

Theorem 1. *If the underlying proxy re-encryption scheme is IND-CCA secure, then the MUSE protocol offers document privacy.*

Proof. Let C_t be a target user, pk_t be the public key of C_t for the PRE scheme, ρ_t be the set of re-encryption keys of type $rk_{t\to i}$ with $i \neq t$, and d_t be a target documents that C_t has stored on DS. All the information DS has in relation to d_t consists of the ciphertext $[d_t]_{pk_t}$ computed under public key pk_t, and a set of re-encrypted ciphertexts $R_t = \{[d_t]_{pk_i} \mid rk_{t\to i} \in \rho_t\}$.

In order for DS to be able to obtain document plaintexts, there must exist a polynomial-time algorithm A' which takes as input a document ciphertext $[d_t]_{pk_t}$, a set of re-encryption keys ρ_t, and a set R_t of re-encrypted ciphertexts of d_t under different public keys, and outputs the document plaintext d_t. Let n be the required size of ρ_t and R_t. Note that if A' exists, then the cardinality of both ρ_t and R_t is bound by polynomial complexity.

Definition 9, given in the appendix, describes the IND-CCA security game for proxy re-encryption schemes presented in [29]. Let $A^* = (A_1^*, A_2^*)$ be a polynomial-time adversary for the IND-CCA PRE security game, where A_2^* is as follows:

1. Take as input plaintexts m_0 and m_1, ciphertext c^* encrypted under public key pk^*, and state s.
2. Call O_h n times and retrieve a set π consisting of n public keys with indices in L_H.
3. Use O_{rk} to compute $\rho^* = \{O_{rk}(pk^*, pk_i) \mid pk_i \in \pi\}$.
4. Use O_{reenc} to compute $R^* = \{O_{reenc}(pk^*, pk_i, c^*) \mid pk_i \in \pi\}$.
5. Return the output of running A' on input c^*, ρ^*, R^*.

Since adversary A^* wins the IND-CCA security game for PRE with probability 1, if the underlying PRE scheme is IND-CCA secure, then A' does not exist, and therefore DS cannot obtain document plaintexts. Since DS is the only party that has access to document ciphertexts for which it does not know the decryption key, we can conclude that the *MUSE* protocol offers document privacy if the underlying PRE scheme is IND-CCA secure.

Definition 3 (Query privacy). *A multi-user searchable encryption scheme offers query privacy if and only if the ciphertext of a search query does not reveal information about the plaintext keywords that compose the query.*

Since our protocol relies on deterministic encryption to achieve sub-linear search complexity, it obviously does not offer CPA security for the search queries. Bellare et al. explore this trade-off in [16] and offer a weaker definition of security called PRIV security. According to this definition, a deterministic encryption scheme can be considered secure, provided it has high min-entropy, i.e. it has a sufficiently large plaintext space to be resilient against dictionary attacks.

Theorem 2. *Let Π be a searchable encryption scheme which encrypts a keyword by appending its hash to the ciphertext produced by an underlying asymmetric encryption scheme $\boldsymbol{AE}$. Π is PRIV secure in the random oracle model, if $\boldsymbol{AE}$ is IND-CPA secure and the min-entropy of the data is high enough to preclude a dictionary attack.*

Theorem 2 is proven in [16].

Theorem 3. *If the BGV encryption scheme is IND-CPA secure, then the MUSE protocol offers query privacy, in the random oracle model, under the PRIV definition of security.*

Proof. In *MUSE*, the data made available to DS by a search query is comprised of the keyword ciphertexts under the BGV encryption scheme and the keyword hashes, both computed under the secret keys of PS. Since DS does not know the key used by PS to produce the keyword hashes, the min-entropy of the data is supplemented by the key space of the PS hashing key, making the scheme resilient to dictionary attacks. Similarly, PS cannot execute a dictionary attack on the keyword hashes received from DS, because it lacks the hashing key used by DS. Since the underlying hashing method is resilient to dictionary attacks, theorem 2 implies that if the BGV encryption scheme is IND-CPA secure, then the *MUSE* protocol offers query privacy, in the random oracle model, under the PRIV definition of security.

Definition 4 (Index privacy). *A multi-user searchable encryption scheme offers index privacy if and only if the search indices associated with a document do not reveal information about the document plaintext or the plaintext search keywords associated with the document.*

Theorem 4. *If the MUSE protocol offers query privacy, then it also offers index privacy.*

Proof. The *MUSE* protocol uses the list of double-keyed hashes of the keywords associated with a document as the index of that document. Since these hashes are computed in the same manner as when performing a search operation, it is clear that index computation provides no additional information to any party compared to the evaluation of a search query. Therefore, if a search query does not reveal information about the underlying keywords, the indices do not reveal this information either. Since the search indices are computed independently from the associated documents, they do not contain any information about the document plaintexts besides the associated keywords.

Definition 5 (Search-pattern privacy). *A multi-user searchable encryption scheme offers search-pattern privacy if and only if no party besides the query issuer can verify if two query ciphertexts encrypt identical search queries or if the queries have keywords in common.*

Theorem 5. *The MUSE protocol does not offer search-pattern privacy.*

Proof. Since query keywords are encrypted individually using a deterministic algorithm, both DS and PS can identify the exact number of shared keywords between any two queries.

Definition 6 (Access-pattern privacy). *A multi-user searchable encryption scheme offers access-pattern privacy if and only if no party besides the user who generated the search query is aware of which document ciphertexts were returned in response to said query.*

Theorem 6. *The MUSE protocol does not offer access-pattern security.*

Proof. DS knows the result of matching a search query against a document index for each individual document. Therefore, DS knows exactly which document ciphertexts were returned for each query.

It is also worth noting that, while the protocol is resilient to dictionary attacks, its interactive nature makes it vulnerable to collusion attacks. If DS and PS were to collude, they would be able to retrieve all the search keyword plaintexts, since DS has their BGV ciphertexts for which PS holds the decryption key. If either DS or PS collude with a client, they would gain the ability to compute hashes for chosen values and build a lookup table, which would eventually reveal all search keyword plaintexts. Regarding document privacy, in order for DS to retrieve a document plaintext it would need to collude with a client with read-access to that document.

6 Efficiency Analysis

To test the efficiency of our protocol we build an implementation of it in C++, available at [35]. Our implementation uses the HElib [33] open-source library for its implementation of the BGV homomorphic encryption scheme. We also use the Crypto++ [34] open-source library for its implementation of AES-CMAC, large number arithmetic, and large prime number generation.

As the homomorphic evaluation of AES-128 requires a modulus chain of length 40 [27], the homomorphic evaluation of AES-CMAC requires a modulus chain of length $40 \cdot L_w/128$, where L_w is the bit-length of the input message. Note that the intermediary homomorphic addition has no impact on the count of moduli. For our experiments, we use an input length of 256 bits, which requires a modulus chain of length 80 without using bootstrapping. We also choose the parameters of the BGV scheme as to obtain a security level of 80 or higher, which would be at least equivalent with the security level of AES-128 [24]. To fulfil these requirements, we initialize the BGV cryptosystem with a cyclotomic polynomial of order $m = 65281$, generators $g_1 = 43073$ with order $n_1 = 96$, and $g_2 = 22214$ with order $n_2 = -14$, and a modulus length of $\log q = 1600$.

The underlying proxy-reencryption scheme requires three security parameters k_1, k_2, and k_p. k_1 and k_2 are used to dictate the output length of the cryptographic hash functions used by the scheme, while k_p is used to decide the bit-length of prime numbers p and q, which are used to compute the cyclic group modulo $N^2 = (pq)^2$. The PRE scheme is similar to the RSA cryptosystem, as its computational security relies on the complexity of factoring N. Since the NIST standard recommends a modulus size of at least 2048 bits for RSA [31],

we choose k_p to be 1024, such that N will have 2048 bits. For consistency, we use the same value for k_1 and k_2, even if cryptographic hash functions are considered secure at lower output sizes [32].

We evaluate the efficiency of our protocol with regard to five variables: keyword size, document size, total number of stored documents, number of keywords associated with one document, and number of documents associated with one keyword. For each variable, we run a series of experiments where we set the other variables to the values shown in column *Default value* of Table 2, while the selected variable iterates over the values shown in column *Experiment values*. For each experiment, we measure the run-time of a store operation, a search operation where the retrieved documents were published by the client performing the query, and a search operation where a client retrieves documents stored by a different client, which involves re-encryption. We perform these experiments on the TU Delft HPC cluster, which uses the CentOS Linux operating system, allocating one CPU core, model Intel® Xeon™ E5-2620 v4 2.10GHz, and 16 GB of memory to each experiment. We run each experiment 10 times and average the results. The results are shown in Figures 1 to 5. The search run-times shown in these figures represent both search operations, with and without re-encryption, because there was no difference between the two measurements.

Table 2: Experiment variables and values

Variable	**Default value**	**Experiment values**
keyword size	8 bytes	8, 16, 24, 32 (bytes)
document size	10 MB	1, 1.5, 2, 2.5, 3, 3.5, 4 GB
total number of documents	1	8, 16, 24, 32, 40
keywords / document	1	8, 16, 24, 32, 40
documents / keyword	1	8, 16, 24, 32, 40

The AES-CMAC algorithm is linear in the number of 128-bit blocks in its input, and so is its homomorphic evaluation. As a result, the *compute hash* algorithm is linear in the amount of blocks in the padded keyword. Since both store and search operations encrypt the keywords using the *compute hash* algorithm, we expect their run-times to scale linearly with the amount of 128-bit blocks in the padded keywords. Our expectations are confirmed by the experiment results shown in Figure 1. Note that for this analysis we treat the depth of the homomorphic circuit as a constant, which, in this specific set of experiments, allows for the hashing of keywords of at most two blocks. The encryption, re-encryption, and decryption operations of the underlying PRE scheme are linear in the size of the plaintext size. As these are the only operations MUSE performs in relation to document plaintexts, we expect both store and search operations to be linear with regards to document sizes. This linear relation between run-time and document size is shown in Figure 2. As documents are stored and retrieved independently from each other we expect both operations to be constant in the number of total documents stored on the server. The experiment results shown in Figure 3 corroborate our expectations. Since the keywords of one document

Fig. 1: Run-time analysis of MUSE operations based on keyword size.

Fig. 2: Run-time analysis of MUSE operations based on document size.

Fig. 3: Run-time analysis of MUSE operations based on the total number of stored documents.

Fig. 4: Run-time analysis of MUSE operations based on the number of documents associated with a keyword.

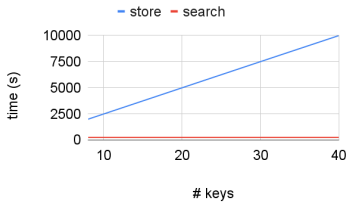

Fig. 5: Run-time analysis of MUSE operations based on the number of keywords associated with a document.

are encrypted separately from the keywords of other documents, we expect the number of documents associated with a keyword to have no influence on the run-time of store operations. On the other hand, a search operation will return all documents that match the query keyword. We therefore expect the search run-time to scale linearly with the number of documents associated with the search keyword. Figure 4 shows that indeed, the number of documents associated with a keyword has no effect on the storage run-time, while causing a linear increase in the search run-time. The store operation encrypts all the keywords associated with the stored document. We therefore expect the store run-time to

A

scale linearly with the number of keywords associated with a document. On the other hand, since keyword hashes are independently mapped to sets of associated documents, we expect the number of keywords associated with a document to have no impact on the search run-time. The graph in Figure 5 shows the expected linear increase in storage run-time, as well as the lack of impact on the search run-time.

7 Conclusions

In this paper we present MUSE, an interactive multi-user searchable encryption protocol that uses deterministic encryption to achieve constant search complexity, without being vulnerable to lookup table attacks. As part of our protocol, we design an interactive keyed hash algorithm which allows two parties to compute a keyed hash value without learning the input value or the full hash key, and thus preventing them from performing dictionary attacks on the hashed values. The design incorporates an algorithm for evaluating AES-CMAC using fully homomorphic encryption, based on Gentry et al.'s algorithm for the homomorphic evaluation of AES-128. We show that MUSE offers document, query, and index privacy for arbitrary search keywords, in the semi-trusted model and under the PRIV definition of security. In terms of short-comings, our protocol does not offer search-pattern or access-pattern privacy, and is vulnerable to collusion attacks due to its interactive nature. We demonstrate that our protocol achieves constant time-complexity with respect to the total number of stored documents, for both search and store operations. Compared to multi-user SE schemes that do not use deterministic encryption, our protocol is more efficient but less secure. Compared to schemes that make use of deterministic encryption, our protocol is comparably efficient, while offering resilience against lookup table attacks.

References

1. Martin Kleppmann. "Designing data-intensive applications: The big ideas behind reliable, scalable, and maintainable systems." O'Reilly Media Inc., 2017.
2. Christoph Bösch, Pieter Hartel, Willem Jonker, and Andreas Peter. "A survey of provably secure searchable encryption." ACM Computing Surveys (CSUR) 47, no. 2 (2014): 1-51.
3. Rohit Handa, C. Rama Krishna, and Naveen Aggarwal. "Searchable encryption: A survey on privacy-preserving search schemes on encrypted outsourced data." Concurrency and Computation: Practice and Experience 31, no. 17 (2019): e5201.
4. Dawn Xiaoding Song, David Wagner, and Adrian Perrig. "Practical techniques for searches on encrypted data." In Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000, pp. 44-55. IEEE, 2000.
5. Eu-Jin Goh. "Secure indexes for efficient searching on encrypted compressed data." Technical Report 2003/216, Cryptology ePrint Archive, 2003. http://eprint.iacr.org/2003/216
6. Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. "Searchable symmetric encryption: improved definitions and efficient constructions." Journal of Computer Security 19, no. 5 (2011): 895-934.

7. Georgios Amanatidis, Alexandra Boldyreva, and Adam O'Neill. "Provably-secure schemes for basic query support in outsourced databases." In IFIP Annual Conference on Data and Applications Security and Privacy, pp. 14-30. Springer, Berlin, Heidelberg, 2007.
8. Peter Van Liesdonk, Saeed Sedghi, Jeroen Doumen, Pieter Hartel, and Willem Jonker. "Computationally efficient searchable symmetric encryption." In Workshop on Secure Data Management, pp. 87-100. Springer, Berlin, Heidelberg, 2010.
9. Melissa Chase and Seny Kamara. "Structured encryption and controlled disclosure." In International conference on the theory and application of cryptology and information security, pp. 577-594. Springer, Berlin, Heidelberg, 2010.
10. Emily Shen, Elaine Shi, and Brent Waters. "Predicate privacy in encryption systems." In Theory of Cryptography Conference, pp. 457-473. Springer, Berlin, Heidelberg, 2009.
11. Christoph Bösch, Qiang Tang, Pieter Hartel, and Willem Jonker. "Selective document retrieval from encrypted database." In International Conference on Information Security, pp. 224-241. Springer, Berlin, Heidelberg, 2012.
12. Mariana Raykova, Binh Vo, Steven M. Bellovin, and Tal Malkin. "Secure anonymous database search." In Proceedings of the 2009 ACM workshop on Cloud computing security, pp. 115-126. 2009.
13. Yanjiang Yang, Haibing Lu, and Jian Weng. "Multi-user private keyword search for cloud computing." In 2011 IEEE Third International Conference on Cloud Computing Technology and Science, pp. 264-271. IEEE, 2011.
14. Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. "Public key encryption with keyword search." In International conference on the theory and applications of cryptographic techniques, pp. 506-522. Springer, Berlin, Heidelberg, 2004.
15. Abdalla, Michel, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. "Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions." In Annual international cryptology conference, pp. 205-222. Springer, Berlin, Heidelberg, 2005.
16. Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. "Deterministic and efficiently searchable encryption." In Annual International Cryptology Conference, pp. 535-552. Springer, Berlin, Heidelberg, 2007. (Full version at `www.cc.gatech.edu/~aboldyre/papers/bbo.pdf`)
17. Changyu Dong, Giovanni Russello, and Naranker Dulay. "Shared and searchable encrypted data for untrusted servers." Journal of Computer Security 19, no. 3 (2011): 367-397.
18. Feng Bao, Robert H. Deng, Xuhua Ding, and Yanjiang Yang. "Private query on encrypted data in multi-user settings." In International Conference on Information Security Practice and Experience, pp. 71-85. Springer, Berlin, Heidelberg, 2008.
19. Peishun Wang, Huaxiong Wang, and Josef Pieprzyk. "An efficient scheme of common secure indices for conjunctive keyword-based retrieval on encrypted data." In International workshop on information security applications, pp. 145-159. Springer, Berlin, Heidelberg, 2008.
20. Jin Wook Byun, Hyun Suk Rhee, Hyun-A. Park, and Dong Hoon Lee. "Off-line keyword guessing attacks on recent keyword search schemes over encrypted data." In Workshop on secure data management, pp. 75-83. Springer, Berlin, Heidelberg, 2006.

21. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. "(Leveled) fully homomorphic encryption without bootstrapping." ACM Transactions on Computation Theory (TOCT) 6, no. 3 (2014): 1-36.
22. Nigel P. Smart, F. Vercauteren. "Fully homomorphic simd operations". IACR Cryptology ePrint Archive, 2011.
23. Craig Gentry, Shai Halevi, Nigel P. Smart. "Fully homomorphic encryption with polylog overhead". EUROCRYPT 2012. LNCS, vol. 7237, pp. 465–482. Springer, Heidelberg (2012).
24. Silvia Mella and Ruggero Susella. "On the homomorphic computation of symmetric cryptographic primitives." In IMA International Conference on Cryptography and Coding, pp. 28-44. Springer, Berlin, Heidelberg, 2013.
25. Junhyuk Song, Radha Poovendran, Jicheol Lee, Tetsu Iwata. "The aes-cmac algorithm." RFC 4493, 2006.
26. National Institute for Science, Technology (NIST): Advanced Encryption Standard (FIPS PUB 197) (November 2001), https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf
27. Craig Gentry, Shai Halevi, Nigel P. Smart. "Homomorphic evaluation of the AES circuit." IACR Cryptology ePrint Archive, 2015.
28. Jun Shao and Zhenfu Cao. "CCA-secure proxy re-encryption without pairings." In International Workshop on Public Key Cryptography, pp. 357-376. Springer, Berlin, Heidelberg, 2009.
29. Zhiguang Qin, Hu Xiong, Shikun Wu, and Jennifer Batamuliza. "A survey of proxy re-encryption for secure data sharing in cloud computing." IEEE Transactions on Services Computing (2016).
30. Pascal Paillier. "Public-key cryptosystems based on composite degree residuosity classes." In International conference on the theory and applications of cryptographic techniques, pp. 223-238. Springer, Berlin, Heidelberg, 1999.
31. Elaine Barker and Q. Dang. "Draft NIST special publication 800-57 part 3 revision 1." Recommendation for Key Management (2014).
32. Q.H. Dang. "Sp 800-107. recommendation for applications using approved hash algorithms." (2009)
33. HElib: https://github.com/homenc/HElib
34. Crypto++: https://www.cryptopp.com/
35. MUSE implementation: https://github.com/babeanu-dorian/MUSE/

A Security Definitions for Proxy Re-encryption

Definition 7 (Derivative of a PRE ciphertext). *Let Σ = (KeyGen, ReKey, Encrypt, ReEncryt, Decrypt) be a proxy re-encryption scheme. Given the challenge ciphertext (pk^*, c^*), then:*

- *The challenge ciphertext (pk^*, c^*) is a derivative of itself.*
- *A pair (pk_j, c_j) is a derivative of the challenge (pk^*, c^*) if (pk_j, c_j) is a derivative of the challenge (pk_i, c_i) and (pk_i, c_i) is also a derivative of (pk^*, c^*).*
- *If a triple (pk_i, pk_j, c_i) has been queried to a re-encryption oracle by the adversary, who in turn obtains a re-encrypted ciphertext c_j as response, then (pk_j, c_j) is a derivative of (pk_i, c_i).*

– *If* (pk_i, pk_j) *has been queried to the re-encryption key generation oracle by the adversary, and Decrypt*$(pk_j, c_j) \in \{m_0, m_1\}$*, then* (pk_j, c_j) *is a derivative of all pairs* (pk_i, c)*.*

Definition 8 (PRE Oracles). *Let* Σ *= (KeyGen, ReKey, Encrypt, ReEncryt, Decrypt) be a proxy re-encryption scheme. Let* L_H *and* L_C *be the index lists of honest and corrupt users respectively. The target user is considered honest and has index* $i^* \in L_H$*. Let* pk_{i^*} *and* sk_{i^*} *be the public and private keys of the target user. The following oracles can be made available to the adversary:*

– O_h*: Takes as input a key pair* (pk_i, sk_i) *outputted by KeyGen, inserts* i *into* L_H*, and outputs the public key* pk_i*.*
– O_c*: Takes as input a key pair* (pk_i, sk_i) *outputted by KeyGen, inserts* i *into* L_C*, and outputs the key pair* (pk_i, sk_i)*.*
– O_{rk}*: Takes as input two public keys* pk_i *and* pk_j*, and outputs a re-encryption key* $rk_{i \to j}$ *as produced by ReKey. The adversary can only make queries in which* $i \neq j$ *and either* $i, j \in L_H$ *or* $i, j \in L_C$*.*
– O_{reenc}*: Takes as input a triple* (pk_i, pk_j, c) *in which* $i \neq j$ *and* $i, j \in L_H \cup L_C$ *and outputs the re-encrypted ciphertext* $c' \leftarrow$ *ReEncrypt*$(rk_{i \to j}, c)$*. The adversary is not allowed to make queries in which* $j \in L_C$ *and* (pk_i, c) *is a derivative of* (pk^*, c^*)*.*
– O_{dec}*: Takes as input a ciphertext* (pk_i, c) *in which* $i \in L_H \cup L_C$*, and outputs* $m \leftarrow$ *Decrypt*(sk_i, c)*. The adversary is not allowed to make queries such that* (pk_i, c) *is a derivative of* (pk^*, c^*)*.*

Definition 9 (CCA security for proxy re-encryption). *Let* Σ *= (KeyGen, ReKey, Encrypt, ReEncryt, Decrypt) be a proxy re-encryption scheme,* $A = (A_1, A_2)$ *be a polynomial-time adversary, and* O_1 *and* O_2 *be the sets of available oracles for* A_1 *and* A_2 *respectively. Both algorithms have access to the key oracles* O_h*,* O_c*, and* O_{rk}*, but access to the* O_{reenc} *and* O_{dec} *oracles is dependent on the attack model. Let* $CCA_{i,j}$ *be an attack model for PRE, and* $t \in \{1, 2\}$*, then* $O_{dec} \in O_t$ *if* $i \geq t$*, and* $O_{reenc} \in O_t$ *if* $j \geq t$*.*

For $i, j \in \{0, 1, 2\}$*,* $b \in \{0, 1\}$ *and the security parameter* $k \in mathbbK$*, the indistinguishability game is defined by the following experiment:*

$\boldsymbol{Exp}^{\texttt{IND-CCA}_{i,j}}_{\Sigma,A,b}(k)$:
$(pk^*, sk^*) \leftarrow$ *KeyGen*(k)*;*
$(m_0, m_1, s) \leftarrow A_1(pk^*)$*;*
$c^* \leftarrow$ *Encrypt*(pk^*, m_b)*;*
$d \leftarrow A_2(m_0, m_1, s, c^*)$*;*
return d*.*

For $i, j \in \{0, 1, 2\}$ *and* $k \in \mathbb{K}$*, the advantage of A to win* $\boldsymbol{Exp}^{\texttt{IND-CCA}_{i,j}}_{\Sigma,A,b}(k)$ *is* $\boldsymbol{Adv}^{\texttt{IND-CCA}_{i,j}}_{\Sigma,A}(k) = |\boldsymbol{Pr}[\boldsymbol{Exp}^{\texttt{IND-CCA}_{i,j}}_{\Sigma,A,1}(t) = 1] - \boldsymbol{Pr}[\boldsymbol{Exp}^{\texttt{IND-CCA}_{i,j}}_{\Sigma,A,0}(t) = 1]|$*.*

The proxy re-encryption scheme Σ *is said to be* $\texttt{IND-CCA}_{i,j}$ *secure if the advantage* $\boldsymbol{Adv}^{\texttt{IND-CCA}_{i,j}}_{\Sigma,A}(k)$ *is negligible.*

A proxy re-encryption scheme Σ *is said to be IND-CCA secure if it is* $\texttt{IND-CCA}_{2,2}$ *secure, in which both algorithms* A_1 *and* A_2 *have access to both* O_{reenc} *and* O_{dec} *oracles.*