Privacy-Preserving Data Processing for Real Use Cases

Inaugural-Dissertation
zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften

der Universität Mannheim

vorgelegt von

M.Sc. Louis Tajan
aus Paris, Frankreich

Mannheim, 2020

Dekan: Dr. Bernd Lübcke, Universität Mannheim
Referent: Prof. Dr. Frederik Armknecht, Universität Mannheim
Korreferent: Prof. Dr. habil. Dirk Westhoff, Hochschule Offenburg

Tag der mündlichen Prüfung: 19. March 2021

Abstract

In the work at hand, we state that privacy and malleability of data are two aspects highly desired but not easy to associate. On the one hand, we are trying to shape data to make them usable and editable in an intelligible way, namely without losing their initial information. On the other hand, we are looking for effective privacy on data such that no external or non-authorized party could learn about their content. In such a way, we get overlapping requirements by pursuing different goals; it is trivial to be malleable without being secure, and vice versa.

We propose four "real-world" use cases identified as scenarios where these two contradictory features are required and taking place in distinct environments. These considered backgrounds consist of firstly, *cloud security auditing*, then *privacy of mobile network users* and *industry 4.0* and finally, *privacy of COVID-19 tracing app users*.

After presenting useful background material, we propose to employ multiple approaches to design solutions to solve the use cases. We combine homomorphic encryption with searchable encryption and private information retrieval protocol to build an effective construction for the could auditing use case. As a second step, we develop an algorithm to generate the appropriate parameters to use the somewhat homomorphic encryption scheme by considering correctness, performance and security of the respective application.

Finally, we propose an alternative use of Bloom filter data structure by adding an HMAC function to allow an outsourced third party to perform set relations in a private manner. By analyzing the *overlapping bits* occurring on Bloom filters while testing the *inclusiveness* or *disjointness* of the sets, we show how these functions maintain privacy and allow operations directly computed on the data structure. Then, we show how these constructions could be applied to the four selected use cases.

Our obtained solutions have been implemented and we provide promising results that validate their efficiency and thus relevancy.

Zusammenfassung

In der vorliegenden Arbeit legen wir dar, dass der Schutz der Privatsphäre und die Formbarkeit von Daten zwei Aspekte sind, die hoch erwünscht, aber nicht leicht miteinander in Einklang zu bringen sind. Einerseits versuchen wir, Daten so zu gestalten, dass sie auf verständliche Art und Weise nutzbar und bearbeitbar sind, d.h. ohne, dass sie ihre ursprüngliche Information verlieren. Andererseits streben wir einen effektiven Datenschutz an, damit externe oder nicht autorisierte Parteien nichts über den Inhalt der Daten erfahren können. Diese Anforderungen überlappen sich jedoch, da sie unterschiedliche Ziele verfolgen; Formbarkeit ist ohne Sicherheit leicht zu erreichen, genauso wie Sicherheit ohne Formbarkeit.

Wir schlagen vier "reale" Anwendungsfälle vor, die als Szenarien identifiziert wurden, in denen diese beiden widersprüchlichen Merkmale erforderlich sind und die in unterschiedlichen Umgebungen stattfinden. Die betrachteten Hintergründe bestehen zunächst aus der Sicherheitsüberprüfung in der Cloud, dann aus dem Datenschutz von Benutzer mobiler Netzwerke und der Industrie 4.0 und schließlich aus dem Datenschutz von Benutzer von COVID-19-Tracing-Anwendungen. Nach der Vorstellung nützlichen Hintergrundmaterials, schlagen wir vor, zur Lösung der Anwendungsfälle mehrere Lösungsansätze zu entwerfen. Wir kombinieren homomorphe Verschlüsselung mit durchsuchbarer Verschlüsselung und einem Protokoll zum Abrufen privater Informationen, um eine effektive Konstruktion für den möglichen Anwendungsfall des Auditing zu erstellen. In einem zweiten Schritt entwickeln wir einen Algorithmus zur Generierung der geeigneten Parameter zur Verwendung des somewhat homomorphen Verschlüsselungsschemas unter Berücksichtigung von Korrektheit, Leistung und Sicherheit der jeweiligen Anwendung.

Schließlich schlagen wir eine alternative Verwendung der Datenstruktur des Bloom-Filters vor, indem wir eine HMAC-Funktion hinzufügen, die es einer ausgelagerten dritten Partei ermöglicht, festgelegte Beziehungen auf private Weise durchzuführen. Durch die Analyse der überlappenden Bits, die bei Bloom-Filtern auftreten, während die Inklusivität oder Disjunktheit der Sätze getestet wird, zeigen wir, wie diese Funktionen die Privatsphäre wahren und Operationen ermöglichen, die direkt auf der Datenstruktur berechnet werden. In der Folge zeigen wir, wie diese Konstruktionen auf die vier ausgewählten Anwendungsfälle angewendet werden könnten.

Die von uns erhaltenen Lösungen wurden implementiert und liefern vielversprechende Ergebnisse, die ihre Effizienz und damit ihre Relevanz validieren.

Acknowledgment

First of all, I would like to express my gratitude to Dirk Westhoff for his trust, his support and our valuable intellectual interactions. I would also like to thank Hochschule Offenburg and in particular Johann Betz for helping me to settle in seamlessly in this new work environment, it has been an indispensable support.

As one may know, a doctoral experience can be as daunting as rewarding. It certainly is a professional and life experience I would recommend even if the last moments of these doctoral years have not been easy to overcome. I am truly glad of this experience and the achievements that have opened me up to new horizons.

For that, I thank Frederik Armknecht from University of Mannheim for welcoming me to my first research project and following and challenging the progress of my investigations over the past four years. Also from University of Mannheim, I thank Christian Gorke, for the valuable discussions and the joint research work during the PAL SAaaS project.

I would also like to thank Avi for the nice conversations in HSO, Alex for showing me the way, Sioul for being the friend he is and Karim for always making the best of things.

For the next adventures to come, I thank Charles D'Aumale whose unfailing interest in the field or in my research was always valuable and inspiring.

Last but definitely not least, I would like to warmly thank my mother, Marianne, and sister, Estrella, for their careful rereading and unconditional support, my grand father, Pierre Rosenstiehl, for the mathematics career inspiration and Nadia for the precious advises and meticulous proofreading. Finally, I thank my beloved and wife-to-be Adèle for the daily support and cheering.

To my father and Maïté.

Contents

Notation

	Searchable Encryption Environment
$\mathtt{SE}$	PEKS scheme
$pk_{\mathtt{SE}}, sk_{\mathtt{SE}}$	public and private keys of the $\mathtt{SE}$ scheme
λ	security parameter
m	message
c	ciphertext
w	keyword
S_w	searchable encryption ciphertext of the keyword w
T_w	trapdoor of the keyword w
	Homomorphic Encryption Environment
$\mathcal{M}$	message space
$\mathcal{C}$	ciphertext space
$\mathcal{E}nc$	encryption function
$\mathcal{D}ec$	decryption function
m_1, m_2	messages
$\odot_{\mathcal{M}}$	operator in $\mathcal{M}$
$\odot_{\mathcal{C}}$	operator in $\mathcal{C}$
N	RSA module
p, q	prime numbers
$\phi(N)$	Euler's function of N
d	integer
$\lambda(n)$	Carmichael function of n
$L(u)$	L function of u
$(\frac{a}{p})$	Legendre symbol of a and p
z	non quadratic residue
$pub.key, sec.key$	public and private keys
c_i	ciphertext
r	Benaloh's ciphertext length in bits
β	part of the Naccache-Stern's public key
R, R_q	rings
χ, χ'	error distributions over R
e_i	element drown from χ
a_i, s	uniformly random elements from R_q
n	SHE's degree of polynomials
t	SHE's value space of the plaintext coefficients
b	SHE's encoding base
q	SHE's value space of the ciphertext coefficients

σ	standard deviation of the error
$\xleftarrow{\$}$	"chosen from the uniform distribution"
pk, sk	SHE's public and private keys
ct, ct'	ciphertexts
γ, δ	sizes of ct and ct'
	Private Information Retrieval Environment
i	database's index
i^*	index requested by the user
n	size of the database
$(x_1, x_2, ..., x_n)$	elements of the database
$\mathcal{U}$	user
$\mathcal{S}$	server
$Q = (q_1, q_2, ..., q_n)$	query from $\mathcal{U}$
R	response from $\mathcal{S}$
$\mathcal{E}nc$	encryption function
$\mathcal{D}ec$	decryption function
m_0, m_1	messages
N	RSA modulo
e, d	RSA public and private exponents
$pub.key, sec.key$	public and private RSA keys
u_0, u_1, u_b	random values
b	bit of the requested message
k	random value
v, m_b	values computed by Bob
k_0, k_1, m'_0, m'_1	values computed by Alice
(i, j)	two-dimensional database's indexes (resp. row and column)
(i^*, j^*)	indexes requested by $\mathcal{U}$
$Q = (\alpha_1, ..., \alpha_{\sqrt{n}}, \beta_1, ..., \beta_{\sqrt{n}})$	query from the user in the case of a two-dimensional database
$R = (R_1, ..., R_{\sqrt{n}})$	response from $\mathcal{S}$
σ_i, u_i, v_i	values computed by $\mathcal{S}$
M	matrix representation of the database
s, t	dimensions of the database
(a, b)	coordinates of the requested element from $\mathcal{U}$
$y_1, ..., y_t$	values $\in \mathbb{Z}_N$ relatively prime to N
$(\frac{y_i}{N})$	Jacobi symbol of y_i and N
$w_{r,j}, z_r$	values computed by $\mathcal{S}$
op	basic operation
$add, mult$	addition and multiplication on the ciphertexts
$testRQ$	quadratic residuosity testing
G, G'	value spaces of the cleartexts and ciphertexts
	Bloom Filter Environment
$\mathcal{AD}, \mathcal{C}, \mathcal{CSP}$	auditor, client and cloud service provider
$\mathcal{W}$	client's whitelist
$\mathcal{L}_1, \mathcal{L}_2$	two sets of the logfile
$\mathcal{A}$, $\mathcal{B}$	generic sets
$INC()$, $DIS()$	*inclusiveness* and *disjointness* operators
$BF_{\mathcal{A}}, BF_{\mathcal{A}\cap\mathcal{B}}, BF_{\mathcal{A}\subseteq\mathcal{B}=\emptyset}$	respective Bloom filter representation of $\mathcal{A}$, *inclusiveness* and *disjointness* relations between $\mathcal{A}$ and $\mathcal{B}$
$bf_{\mathcal{A}}[i]$	i^{th} index of $BF_{\mathcal{A}}$
$n_{\mathcal{A}}, N_{\mathcal{A}}$	cardinality of set $\mathcal{A}$ and maximization of this size
h	HMAC function
K	set of the keys used with function h
n_{key}	amount of keys in K
n^L_{key}, n^U_{key}	lower and upper bounds of n_{key}

m	size of the Bloom filter
$\mathcal{P}_{FP}$	probability of having a false positive to the *inclusiveness* operator
$\mathcal{P}_{FN}$	probability of having a false negative to the *disjointness* operator
$X_{BF_{\mathcal{A}}}$	number of bits set to 1 in $BF_{\mathcal{A}}$
$X_{\cap=\emptyset}$	estimation of the number of bits set to 1 in $BF_{\mathcal{A}\subseteq\mathcal{B}=\emptyset}$ if $\mathcal{A}$ and $\mathcal{B}$ were disjoint
$Y_{BF_{\mathcal{A}}}$	amount of *overlapping bits* in $BF_{\mathcal{A}}$
$Y^*_{BF_{\mathcal{A}}}$	estimation of $Y_{BF_{\mathcal{A}}}$
$Z_{\mathcal{A},\mathcal{B}}$	amount of elements from $\mathcal{A}$ which are not in $\mathcal{B}$
$Z'_{\mathcal{A},\mathcal{B}}$	amount of elements in both $\mathcal{A}$ and $\mathcal{B}$
ρ, μ	standard deviation and mean of the *overlapping bits* distribution
$L_{n_{key}}, L^{\mathcal{A}}_{n_{key}}$	candidates lists of n_{key} used in general, and used in $BF_{\mathcal{A}}$
$L_{n_{\mathcal{A}}}$	candidates list of $n_{\mathcal{A}}$
$\lambda_{n_{key}}, \lambda^{\mathcal{A}}_{n_{key}}, \lambda_{n_{\mathcal{A}}}$	cardinalities of the candidates lists $L_{n_{key}}, L^{\mathcal{A}}_{n_{key}}$ and $L_{n_{\mathcal{A}}}$
$[ob^L_{\mathcal{A}}; ob^U_{\mathcal{A}}]$	range of the *overlapping bits* distribution for $BF_{\mathcal{A}}$
id_i	user i's credentials
t^1_i, t^2_i	starting and ending times of user i's connection to the access point
$\langle A, B, C\rangle$	proximity chain composed by users A, B and C
$[D, E, F]$	infection chain composed by users D, E and F
Tr	threshold
$Pr(X_i)$	probability of being infected of user X_i
BS_N	list of base stations that current node N has been connected to
T	proximity tree
S, S'	sets storing the proximity chains from a proximity tree
R	reproduction number
	Concealed Data Aggregation Environment
$\mathcal{BS}$	base station
$\mathcal{IN}$	intermediate node
$\mathcal{SN}$	sensor node
μ	machine identifier
$\{\alpha^\mu_1, \ldots, \alpha^\mu_n\}$	set of keys of machine μ
n	amount of keys
h	HMAC function
m	size of the Bloom filter
i	sensor node identifier
st_i	status of the machine collected by sensor node i
t_i	temperature of the machine collected by sensor node i
s_i	spin speed of the machine collected by sensor node i
z	amount of sensor nodes from a machine
maj	*majority* rule
$comp$	*completeness* rule
agg	aggregation step
$BF^{(x)}_i$	x-th Bloom filter generated by the sensor node i
$BF^{(x)}_{maj}$	x-th Bloom filter obtained after applying rule *majority*
$BF^{(x)}_{comp}$	x-th Bloom filter obtained after applying rule *completeness*
$BF^{(x)}_{agg}$	x-th Bloom filter obtained after applying the aggregation step
$BF^{(x)}_f$	x-th final Bloom filter obtained by the base station
t_x	time variable
F^j_x	location file stored in user j's smartphone at time t_x
l	proximity limit
E_n	elliptic curve
$\mathbb{F}$	field
$\mathbb{G}$	group

Chapter 1

Introduction

1.1 Privacy vs Malleability

The objective of this doctoral thesis consists of identifying what we could call "real-world" use cases, namely scenarios with requirements and limitations driven by the way societies (as western or European) are established, in particular by their economy and political functioning and by the need for every person to have a minimal control over their personal privacy. We attempt here to weave cryptographically sound sensitive data into process flow. No matter which environment is considered, let it be cloud infrastructure, mobile network, industry 4.0 or others, the existing systems should be forensic ready and usable data collection should be supported. Therefore, gathering building blocks and developing new promising designs to combine privacy and usability of targeted data turns out to be a thrilling and challenging issue.

As an example of obvious benefit of setting a data collection in cloud infrastructure, we could state that it allows to determine responsibilities between a cloud service provider and its clients in case an incident has happened. It also allows to identify how a cyber-crime was perpetrated and to estimate the damage suffered. But such a straightforward approach could suffer from certain limitations, such as not being able to identify relevant data related to attacks exploiting unknown vulnerabilities to the respective parties.

Privacy

After being neglected for quite a time, privacy issue is now fully established as one of the most studied and funded topics of research. Data privacy could be considered as one of the main keystones of modern cryptography along with *data integrity* or *authentication*. We show in Figure 1.1 forecasts established by *Gartner*, an IT research and consulting company, that illustrate the continual increase and development of all IT-security aspects and in particular the ones related to data privacy.

In the work at hand, data privacy is clearly the most important requirement and even if the two concepts of *privacy* and *malleability* may overlap, the first one will always be put forward.

Market Segment	**2017**	**2018**	**2019**
Application Security	2,434	2,742	3,003
Cloud Security	185	304	459
Data Security	2,563	3,063	3,524
Identity Access Management	8,823	9,768	10,578
Infrastructure Protection	12,583	14,106	15,337
Integrated Risk Management	3,949	4,347	4,712
Network Security Equipment	10,911	12,427	13,321
Other Information Security Software	1,832	2,079	2,285
Security Services	52,315	58,920	64,237
Consumer Security Software	5,948	6,395	6,661
Total	**101,544**	**114,152**	**124,116**

Source: Gartner (August 2018)

Figure 1.1. Worldwide security spending by segment, 2017-2019 (Millions of U.S. Dollars).

Malleability

Two aspects of the data, namely substance and form, should be considered to establish the data usability. Indeed, first of all, data on their own should be relevant and valuable to perform processing and testing. They should be correct and reflect all that happened. They should also be understandable, since not readable, to the party that will manipulate them. By understandable we mean usable by the party, that should know to what information the data correspond even if they are obfuscated. This brings us to the second consideration on the data's form. Indeed, in case of required obfuscation on the data, their format should still allow to perform some processing on them. This specific aspect implies very careful attention on the protocol's initialization, such that settings allow processing on the private data and therefore correctness on final results.

1.2 Manipulating Data

According to scenarios and environments considered, multiple questions could be asked on the data's manipulation. For example, what type of data should be considered, how and from which infrastructure could one collect them? More

generally, what parties are involved in the data's manipulation? How and where to store the collected data? Do some of these data need to be protected before being stored and what kind of computations should be performed on them? We attempt, in the remainder of this section, to address these issues.

At first, we give a definition of sensitive data material as considered in the rest of the thesis.

Sensitive material: Any form of data that could be helpful to verify the good conduct of any party. In addition, such data should be protected to preserve privacy to the respective parties.

In his PhD thesis [Rüb16], Rubsamen makes use of the *digital evidence* term to characterize such type of data in cloud environment and in particular cloud auditing. He also uses a definition from SWGDE (the Scientific Working Group on Digital Evidence) which defines digital evidence as "information of probative value that is stored or transmitted in binary form".

In our current work, we prefer the expression *sensitive material* to illustrate the privacy/malleability dichotomy. *Sensitive* displays the obvious need for privacy while *material* indicates the necessity to perform computation and transformation on data.

In cloud auditing environment, several published papers address the digital evidence approach. In [RCN+18] authors state in their work that an attack on cloud infrastructures will leave evidence in the system's memory and the attacker will not be able to erase this evidence within the scan interval. They propose a solution to scan the system and detect such evidence. The type of evidence that is left by the attack could consist of anomalous data in well known kernel or application data structure (e.g. to monitor the kernel's list of active tasks for blacklisted processes). It also could be detected by scanning outgoing network packets for suspicious content. Authors claim that some attacks could leave no trace and therefore their solution will actively detect such attacks. They indicate that by proactively installing "tripwires" inside the VM's memory, the external scan can more easily discover evidence of the attack. This technique implies that the cloud could be flexible and should allow to run external libraries and upload updated code directly on its stack or heap.

As aforementioned, authors from [PHMN16] propose an adaptive evidence collection from cloud infrastructures using attack scenarios. They state that performing an exhaustive evidence collection is not always a viable solution since it could be too cumbersome and energy consuming to effectively analyze all of them. They also think that cloud-related changes as modifications of physical or virtual machine configurations, should be monitored, and the corresponding data collection activities should be adapted accordingly. According to their analysis of *attack scenarios* based on the evidence collection, they represent a sequence of actions an adversary can perform to achieve her criminal goals as compromising a machine, or copying a VM image. By referring to these attacks, they claim to reduce the amount of data to be preserved in the cloud infrastructure and focus their data collection activities on the security breaches that are likely to happen. Authors distinguish different categories of evidence:

Application logs, that are generated by any software installed in the cloud environment, as Content Management Software, VM Managers or Cloud development platforms.

Web server logs, that include information on the VMs' accesses like by example user IP addresses as we use in the first presented use case.

Database logs, that contain information generated by database management systems about queries performed and modifications of databases content.

Network logs, that are generated by both the web server and the operating system running on a physical machine. They record information about how network connections are established, the data traffic on the network and who requested the connection.

System logs, that are generated by the operating systems running on the physical machines belonging to the cloud deployment.

In [GSG12] authors highlight the challenges that an investigator may face while analyzing an extremely large amount of data placed in the cloud by a customer. In [BW11], authors assess the usability of various sources of evidence for investigative purposes in all three major cloud service models (SaaS, PaaS, IaaS). Finally, we mention that in [RPR15], authors list material data as logs, cryptographical proofs or documentation.

The overall framework of collecting and aggregating evidence from the cloud service provider and storing it in evidence stores is presented by Ruebsamen and Reich [RR13], and by Schatz and Clark [SC06] from the Common Digital Evidence Storage Format working group (CDESF). The latter are proposing an evidence framework with an architecture focused on Digital Evidence Bags (DEB).

1.2.1 Collecting

The sensitive material should first be identified and collected. The collection step could be performed for instance, by an auditor, by a network provider company or even directly by the involved party. We could list few trivial requirements on data collection. The sensitive material should be **complete** to avoid any cherry-picked strategy from the collector or manipulation from the data provider. Then, as an optional requirement, only the **necessary** data should be collected. In that sense, we minimize the data' collection to reduce the effort. Finally, we could mention that each type of material needs an **adapted mechanism** as for instance, a log parser for logs or a file retriever for documentation.

In [THGL11] authors stress that evidence stored in the cloud tend to be ephemeral, such as registry entries from Microsoft Windows platforms or temporary internet files that can be misled by customer.

Wolthussen states in [Wol09] that one of the most challenging issues consists of collecting evidence across multiple virtual or physical machines, data centers, specially with different geographical and legal jurisdictions.

In [DS11], authors suggest that evidence should be collected at the virtual machine level, where a web system interfaces with the provider's underlying filesystem and hypervisor. Such an approach allows to have evidence collected "on-demand" by several parties, including customers, providers and lawyers. They developed in [DS13] a tool that provides such functionalities to support forensic acquisition of virtual disks associated with VMs, logs of all APIs requests and firewall logs for the customers' virtual machines.

In [SFM11] authors created a proof-of-concept continuous evidence collection system which could be used to record the deletion and creation of service provisions in the cloud. But monitoring all existing evidence is not a viable solution, as it could be difficult to analyze.

Existing work has mainly focused on how evidence can be collected in the cloud without providing guidance on what data should be preserved. Although sorting techniques have also been proposed as a means of reducing the amount of data to be analyzed in conventional investigations ([PW10]), they have not been applied to target evidence collection activities towards the preservation of the data necessary to investigate the security breaches that are more likely.

1.2.2 Retrieving

This intermediate step introduces the issue of privacy and especially the question of which data should be protected. It is possible that some of the material data are sensitive and require privacy. We dissociate different levels of privacy corresponding to the point where the data should be protected.

No privacy. The data are public and do not need any protection. It could be directly stored after collection.

Semi privacy. The data could be known to the collector party which collects them by as plaintexts. The data then need to be protected from other parties when stored in the evidence store.

Full privacy. No party other than the data provider should be able to read them. Therefore, data are protected upstream and should remain as they are until the end of the whole protocol.

Once stored, data should not be tampered with. Cloud environments have also been suggested as a basis for conducting digital forensic investigations in [RWB10]. In particular, cloud virtual instances and storage can be used, respectively, to gather and store sensible material relative to potential or detected incidents/crimes.

In [LZLY13], authors identify potential problems when storing massive amounts of material data. They specifically address possible information leaks. To solve these issues, they propose an efficient encrypted database model that

is supposed to minimize potential data leaks as well as data redundancy. However, they focus solely on the storage backend and do not provide a workflow that addresses secure data collection as a whole. In [RD14] Redfield and Date propose a system called *Gringotts* that enables secure evidence collection, where evidence data are signed by the system that produces them, before being sent to a central server for archival, using the Evidence Record Syntax (ERS). Authors focus on archiving and preservation of evidence integrity.

1.2.3 Processing

Lastly, we could wonder what kind of computations could be performed on the sensitive material. The type of computations performed on these data should match the one agreed upon during their collection. In other words, the collected data should only be used for the aims it was collected for and not for any alternative purposes such as marketing ones.

Depending on the considered scenario, the processing stage may consist of performing tests on the data like verifying if a VM configuration has changed or if a user's IP address is part of a list of authorized IP addresses. The other type of data processing could be performing an aggregation phase on the sensitive data. To optimize the data-retrieving process or even for the purpose of the use case, a third party could be required to take care of the computation load and thus, perform algebraic operations directly on the data. As one of the mainly used cryptographic primitives that allow such computations on the private representation of the data, we could mention homomorphic encryption.

Another way of performing computations on private data is *multiparty computation*. It consists of having multiple parties jointly computing a function over their inputs while keeping those inputs private to each other. But in such cases, every party involved in the protocol provides sensitive material and plays an active part. As we present later, we differentiate the roles to be played such as to have one party that produces data and another that processes them. Consequently, we set aside this cryptography subfield, which in other circumstances, would have deserved as much consideration.

1.3 Research Contributions and Outline

In this work, we achieved the following goals:

Identifying real-world use cases. Driving by research projects, topics of interest or news events, we identified scenarios where privacy and computation of data are essential.

Evaluating constraints. For each of these use cases, we highlighted data requirements on privacy and malleability aspects.

Construction of valuable solutions. After evaluating existing primitives, we combined some to elaborate new valuable constructions.

Implementing solutions and gathering promising results. We implemented all of our three obtained solutions, thus helps validating their efficiency.

We show in Figure 1.2 how the identified use cases presented in Chapter 2 coincide with proposed solutions developed in Chapter 4.

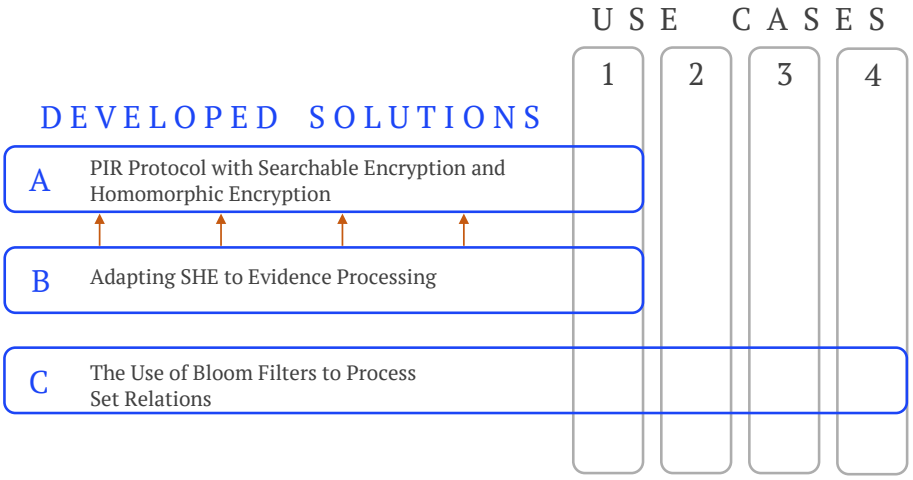

Figure 1.2. How the developed solutions match the proposed use cases. PIR stands for Private Information Retrieval and SHE for Somewhat Homomorphic Encryption.

PAL SAaaS Project

Research project from Nov. 2015 to Apr. 2019 carried out by University of Mannheim with Prof. Frederik Armknecht and Christian Gorke and Hochschule Offenburg with Prof. Dirk Westhoff and Louis Tajan. Funded by the *Baden-Württemberg Stiftung*, the goal of PAL SAaaS is to provide cryptographic mechanisms to support PAL (Privacy, Availability, Liability) in the context of SAaaS (Security Audit as a Service). Several use cases were identified and solutions were proposed. The main part of this thesis has been established as part of this research project.

Industry 4.0 Predictive Maintenance Project

Research project involving multiple professors within Hochschule Offenburg, led by Prof. Matthias Haun and in cooperation with two companies, *Junker* and *Schrempp*. The project started in 2019 and consists of the development of an Industry 4.0-compatible technology for functional and process design predictive and intelligent maintenance systems. The aim of our work package is to establish an end-to-end (E2E) security between the field devices and the instances of the

higher layers of the network in order to increase user acceptance and provide data in a privacy-preserving manner. Outcomes from this project are presented in Sections 2.3 and 4.3.7.

1.3.1 Thesis Organization

This thesis is structured as follows:

Chapter 2 - *Selected Use Cases*, presents the four selected use cases that we state to be relevant to illustrate privacy and malleability's correlation. For each of them we describe the scenario with its parties involved and its threat model.

Chapter 3- *Preliminaries and their Respective Literature* introduces notions and primitives used to develop solutions, along with their respective literature. The different topics correspond to *searchable encryption*, *homomorphic encryption*, *private information retrieval protocols*, *set operations*, *Bloom filter* and *Concealed Data Aggregation protocols*.

Chapter 4 - *Developed Solutions*, presents our three solutions A, B and C to solve the aforementioned use cases. We named our solutions as follows: *Solution A - Combining PIR Protocol with Searchable Encryption and Homomorphic Encryption*, *Solution B - Adapting SHE to Evidence Processing* and *Solution C - Using Bloom Filters to Process Set Relations*.

Chapter 5 concludes our work and summarizes its achievements and limitations.

1.3.2 Publications

- Louis Tajan, Dirk Westhoff, Christian A. Reuter, and Frederik Armknecht. Private information retrieval and searchable encryption for privacy-preserving multi-client cloud auditing. In *11th International Conference for Internet Technology and Secured Transactions, ICITST 2016, Barcelona, Spain, December 5-7, 2016*, pages 162–169. IEEE, 2016. URL: http://dx.doi.org/10.1109/ICITST.2016.7856690

- Louis Tajan, Moritz Kaumanns, and Dirk Westhoff. Pre-computing appropriate parameters: How to accelerate somewhat homomorphic encryption for cloud auditing. In *9th IFIP International Conference on New Technologies, Mobility and Security, NTMS 2018, Paris, France, February 26-28, 2018*, pages 1–6. IEEE, 2018. URL: https://doi.org/10.1109/NTMS.2018.8328713 ***Best Paper Award***.

- Louis Tajan, Dirk Westhoff, and Frederik Armknecht. Private set relations with bloom filters for outsourced SLA validation. *IACR Cryptology ePrint Archive*, 2019:993, 2019. URL: https://eprint.iacr.org/2019/993

- Louis Tajan and Dirk Westhoff. Retrospective tracking of suspects in gdpr conform mobile access networks datasets. In *Proceedings of the Central European Cybersecurity Conference 2019, CECC 2019, Munich, Germany, November 14-15, 2019*, pages 5:1–5:6. ACM, 2019. URL: https://doi.org/10.1145/3360664.3360680

- Louis Tajan, Dirk Westhoff, and Frederik Armknecht. Solving set relations with secure bloom filters keeping cardinality private. In Mohammad S. Obaidat and Pierangela Samarati, editors, *Proceedings of the 17th International Joint Conference on e-Business and Telecommunications, ICETE 2020 - Volume 2: SECRYPT, Paris, France, July 8-10, 2020*, pages 187–197. SciTePress, 2020. URL: https://doi.org/10.5220/0007932301870197

- Louis Tajan and Dirk Westhoff. Approach for GDPR compliant detection of COVID-19 infection chains. *CoRR*, abs/2007.08248, 2020. URL: https://arxiv.org/abs/2007.08248, arXiv:2007.08248

Chapter 2

Selected Use Cases

We present four different use cases where privacy is mainly of interest and processing could be valuable. The first use case is part of our work on the *PAL SAaaS* project while the second one results from an HS Offenburg inner project. The third one results from the *Predictive Maintenance for Industry 4.0* project while the fourth one arises from the COVID-19 crisis of the past months. We gather all of these use cases here, while we propose solutions to solve them in Chapter 4.

2.1 Use Case 1 - Cloud Security Auditing

The constantly growing of cloud business over the past decade results in a rapid increase of different cloud services. While the number of private cloud clients is also rising, commercial interest in usage of cloud providers seems to grow a little less. Reason is that trust in the cloud providers is a concern, or more precisely, provable security of these providers. Having the possibility to prove that a cloud provider behaves correctly according to the agreed on Service Level Agreements (SLA) is a substantial goal. When investigating the area of distributed systems, and in particular service provisioning cloud systems based on SLAs between user and service provider we observe a strong tendency to involve a third party [HX11, ZYS+14, LTS16, RBI17]. The third party is in role of auditing and judging in a semi-automated manner whether the offered services have indeed been successfully provided for a given epoch in the past. Therefore, we are interested in dependable and privacy-preserving building blocks for a third party entity, let it be an auditor or other dependable third parties. By using a kind of whitelist approach an auditor would have to store on the one hand, a whitelist, and on the other hand, some log-lists for a given duration of the past to validate access to an offered service from different users, respectively their identities e.g. ip-addresses. More concretely, we are interested in building blocks which can be used to verify, with a high level of confidence and in a fully privacy preserving manner, whether indeed users from a given group accessed the service, or not.

We want all the parties that own data to upload it to an outsourced third party that will perform solely by itself the required computations. Such scenario has gained relevance with the increased usage of cloud infrastructures. Contrary to multiparty based solutions, we believe that preferably such a protocol class should be non-interactive. For instance, for an indeed practical cloud auditing setting it is required that the user does not have to be available for any validation step after using a given cloud service. We argue that in particular this non-interactive user is valuable, making our approach beneficial compared to competing approaches.

In addition to the trendy increase of auditing in cloud environment, we highlight this use case's relevancy in terms of managing the data at several sides, namely clients, cloud provider and auditor along with its privacy requirements.

2.1.1 Scenario

Cloud service providers often offer many different services, as SaaS, PaaS, IaaS, and a lot more. Clients like companies have many employees which work with different kind of services from the service provider. As an example, a company may allow one department to handle the computation services, while another department is organizing the backups using a cloud storage service. Thus, we are exactly focusing on such clients which consist of different user groups using different cloud services. The client generates an authorization list that defines which users can access to which services. However, there is currently no proof available that the cloud service provider sticks to this authorization list.

Next, a client is usually not able to create these proofs by itself. Therefore we employ a third party, let it be an auditor, who acts on behalf of the clients to perform the checks on the cloud service provider. To be authorized, one user should be an employee from the specific company that required service and in addition, should be connecting to the service from an authorized IP address. An auditor will have to verify that the cloud service provider correctly performed the access control on the company's respective service by testing if IP addresses of the successful users are part of the authorized IP addresses' set. On the contrary, auditor will also have to verify that no authorized user has been denied to connect. With respect to the given privacy requirements, IP addresses from the successful and non-successful sequences as well as from the whitelist have to remain hidden from the auditor.

Additionally, we introduce a fourth party, called the evidence store. It periodically gathers or logs data from the cloud service provider. These data could be encrypted in a way that the store administrator and the auditor do not learn anything about the data, and, on the other hand, the cloud service provider will not know which data the auditor requested. The according framework is depicted in Figure 2.1. We notice that there may be multiple clients.

We identify two Privacy Requirements (PRs) which have to be fulfilled in this scenario:

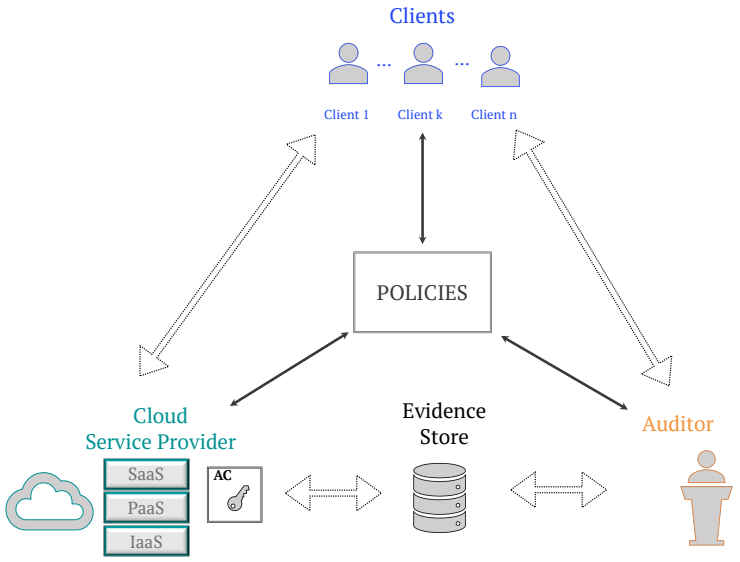

Figure 2.1. Cloud security auditing framework including an evidence store. The parties agree on policies (SLA) before performing the protocols and audits (single line arrow). The communication flow is represented by the double line arrows.

PR1: Client Identity Privacy. The evidence store and the cloud service provider do not learn which audit belongs to which client.

PR2: Client Data Privacy. The evidence store and the auditor do not learn anything about the client data stored at the cloud service provider or the client's associated data sent from the service provider to the store.

Parties Involved

In [RPR15] authors are considering three parties involved in the evidence collection: *Evidence Source*, *Evidence Store* and *Evidence Processor*. Evidences from the *Evidence Source* are encrypted with the *Evidence Processor*'s public key. Therefore the only party blind to the evidence is the *Evidence Store* $\mathcal{ES}$.

For privacy reasons, it is of interest to consider a scenario with no Evidence Store. Indeed, adding another party could make the overall approach more complicate and bring additional threats to the framework. For that reason we also propose a scenario without any storage party.

As presented in Figure 2.2, we consider three parties; a cloud service provider $\mathcal{CSP}$, a client (company) $\mathcal{C}$ and an auditor $\mathcal{AD}$. We define three sets $\mathcal{W}$, $\mathcal{L}_1$ and $\mathcal{L}_2$:

$\mathcal{W} = \{w_1, \ldots, w_{n_\mathcal{W}}\}$ which corresponds to the *whitelist*, set of the authorized IP addresses.

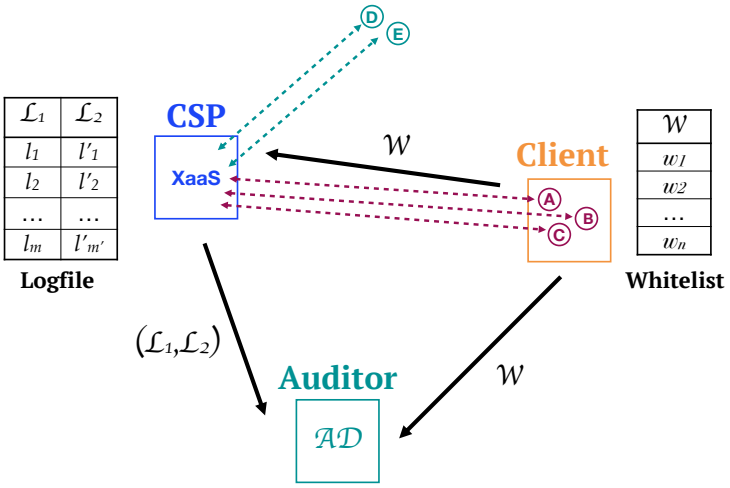

Figure 2.2. Generic framework of the cloud security auditing without $\mathcal{ES}$. Users A, B and C should be authorized to connect by the $\mathcal{CSP}$ while users D and E should not.

$\mathcal{L}_1 = \{l_1, \ldots, l_{n_{\mathcal{L}_1}}\}$ which corresponds to the logfile of IP addresses that successfully connected.

$\mathcal{L}_2 = \{l'_1, \ldots, l'_{n_{\mathcal{L}_2}}\}$ which corresponds to the logfile of IP addresses that failed to connect.

$\mathcal{W}$ is generated and protected by the client himself. Its content is originally sent from $\mathcal{C}$ to $\mathcal{CSP}$ in a non-protected version such that $\mathcal{CSP}$ could perform access control. Nevertheless, it should remain hidden from $\mathcal{AD}$ during the complete auditing process and thus being protected. $\mathcal{L}_1$ and $\mathcal{L}_2$ represent the sets of IP addresses for any connection attempts during the epoch $[t_a; t_b]$, for which auditing is required. They respectively represent the IP addresses' set for all successful and non-successful connections. Both contain sensitive information therefore, their contents should also remain hidden from $\mathcal{AD}$ during whole process.

Set Relations

To verify that $\mathcal{CSP}$ correctly performed access control, $\mathcal{AD}$ has to perform two types of set relations. First of all, to verify that only users with authorized IP addresses succeeded to connect, $\mathcal{AD}$ could perform an *inclusiveness* relation between sets $\mathcal{L}_1$ and $\mathcal{W}$, i.e. to test if all elements from $\mathcal{L}_1$ are included in $\mathcal{W}$. Secondly, to verify that $\mathcal{CSP}$ has rejected only users connecting from non-authorized IP addresses, $\mathcal{AD}$ could perform a *disjointness* relation between $\mathcal{W}$ and $\mathcal{L}_2$. To perform the audit according to the current use case, auditor has to compare both lists $\mathcal{L}_1$ and $\mathcal{L}_2$ with the set $\mathcal{W}$. Such verification consists of the two following steps:

1. Verify that all the IP addresses that have succeeded to connect are authorized ones. In other words verify if all elements from $\mathcal{L}_1$ are also included in $\mathcal{W}$.

$$\mathcal{L}_1 \subseteq \mathcal{W} \quad \equiv \quad \mathcal{L}_1 \cap \mathcal{W} = \mathcal{L}_1 \quad \equiv \quad |\mathcal{L}_1 \cap \mathcal{W}| = |\mathcal{L}_1|$$

2. Verify that none of the IP addresses that did not succeed to connect are authorized ones. In other words, if no elements from $\mathcal{L}_2$ are also included in $\mathcal{W}$.

$$\mathcal{L}_2 \cap \mathcal{W} = \emptyset \quad \equiv \quad |\mathcal{L}_2 \cap \mathcal{W}| = 0$$

Remark: The intersection operator $\cap$ and the inclusion relationship $\subseteq$ are originally used for sets. We could also consider a case where $\mathcal{L}_1$ and $\mathcal{L}_2$ are lists instead of sets. In other words, $\mathcal{L}_1$ and $\mathcal{L}_2$ do not contain solely distinct elements and IP addresses in $\mathcal{L}_1$ and $\mathcal{L}_2$ can even be available multiple times. Therefore, in such concrete case we could redefine such notations as:

$$A \cap B = \{a \in A | a \in B\}$$
$$A \subseteq B \Leftrightarrow \forall a \in A,\ a \in B$$

With such definitions, we remark that in contrary to its regular usage, the $\cap$ operator is not symmetric any more.

2.1.2 Threat Model

We recall that all parties are communicating over a secured and authenticated channel, for instance TLS. On the scenario including an evidence store, out of the four parties, any of these may be malicious. Collusions can even be possible. If either $\mathcal{C}$, $\mathcal{AD}$, or $\mathcal{ES}$ are corrupt, PR1 and PR2 are not violated. In contrast, $\mathcal{CSP}$ could omit the encryption of the data sent to $\mathcal{ES}$ and therefore $\mathcal{ES}$ would learn about client's actions which contradicts PR2. Furthermore, $\mathcal{CSP}$ could forge the data sent to $\mathcal{ES}$ without anyone noticing it. Thus, we state that $\mathcal{CSP}$ does always behave according to the protocol and is at most honest-but-curious. Secure protocols that are developed against malicious adversaries require utilization of different techniques [KK08]. Extending this aspect could be planned for future work. On the other hand, non-availability of $\mathcal{CSP}$ will be detected by $\mathcal{ES}$ immediately, since no data will be sent.

Regarding collusions, we will exclude the client from collusions since he has no further secrets to share with the other parties and will only act in the objective to protect himself. If $\mathcal{AD}$ colludes with $\mathcal{ES}$, both know all the keys used in the PIR protocol and are then able to decrypt $\mathcal{C}$'s information. If $\mathcal{AD}$ colludes with $\mathcal{CSP}$, PR1 is violated, but not PR2, since $\mathcal{ES}$ is honest. If $\mathcal{ES}$ colludes with $\mathcal{CSP}$, PR2 is violated. However, if the client wants to blame other parties, they need to be able to prove that they behaved correctly.

The set relations could be considered as hard-to-solve if we want to achieve a considerable privacy level at the same time. Taking privacy into account means

guaranteeing privacy on the elements of the sets regarding any external party and, in particular, the party which is performing the protocols of *inclusiveness* and *disjointness*. In the no $\mathcal{ES}$-scenario, we consider that on the one hand $\mathcal{C}$ and $\mathcal{CSP}$ behave as *honest-but-curious* and on the other hand that $\mathcal{AD}$ may act as a *malicious* party. Indeed, $\mathcal{CSP}$ already knows the whitelist content while $\mathcal{C}$ will not have access to the logfiles during the auditing protocol since we are not in a multi-party computation model. As the auditor will have access to an obfuscated version of the sets, and we consider that $\mathcal{AD}$ could try to retrieve information about sets' content or cardinality.

We highlight three security requirements (**SR1, SR2** and **SR3**) that should be considered:

SR1. $\mathcal{AD}$ should not be able to modify any part of sets $\mathcal{W}, \mathcal{L}_1, \mathcal{L}_2$ in an undetectable manner and neither should be able to generate its own set and use it to perform the set relations protocols. I.e., performing the protocol with a set from a client along with one generated by the third party should result in a dummy outcome.

SR2. $\mathcal{AD}$ should not be able to learn from $\mathcal{W}$, $\mathcal{L}_1$ or $\mathcal{L}_2$ if:
1) $w_i = l_j$ for some i, j
2) $l_i = l_j$ for some $i \neq j$
In other words, $\mathcal{AD}$ should not be able to learn in how many occurrences an element is present in a set and if a specific element from $\mathcal{W}$ is also in $\mathcal{L}_1$ or $\mathcal{L}_2$. We require a certain level of obfuscation over the computation on the sets. Solely the results of the set relations should be revealed to $\mathcal{AD}$.

SR3. $\mathcal{AD}$ should not learn the sets cardinalities $n_{\mathcal{W}}$, $n_{\mathcal{L}_1}$ and $n_{\mathcal{L}_2}$ from the sets' representations or any results from the protocols.

SR1 and **SR2** are considered as mandatory while **SR3** could be considered as rather optional. Indeed, one may argue that even if the auditor knows the sets cardinalities, he could still not guess which concrete IP addresses are in the sets. But one could easily imagine that, firstly in some use cases, knowing the set cardinality is by itself a leak of privacy, and secondly knowing that a set contains few elements could lead the auditor to infer which ones.

In our three parties model, we recap motivation and behavior that could be adopted by each of them.

Client.

The most convenient model would be to consider the client as honest. The only sensitive data involved are information regarding the company and its employees. In addition, since they are all gathered under the same framework party, w.l.o.g. we could consider the client as **honest-but-curious**. In some

cases, we could consider the client as **malicious** if we argue that client could try to fake the result of the audit and cause $\mathcal{CSP}$ (or $\mathcal{AD}$) to be wrongly blamed. To achieve such a deception, the client could for instance, provide an altered whitelist $\mathcal{W}'$ to the auditor. It will then end up on some incorrect results.

Cloud Service Provider.

Its first role is to perform the access control. In that sense, it is authorized to see all the sensitive data. For the applicability of such a use case, we need to consider that the $\mathcal{CSP}$ is sending the correct evidence to the auditor. Therefore we consider this party as **honest-but-curious**.

Auditor.

We consider that the auditor should not be able to determine the content of the lists $\mathcal{L}_1$ and $\mathcal{L}_2$ and the whitelist $\mathcal{W}$. We also state that such a party could use some trick and manipulate the data in order to determine partial information like knowing how many times an encrypted IP address is present in $\mathcal{L}_1$ or $\mathcal{L}_2$ (see SR2). Therefore we are considering the auditor as **malicious**.

2.2 Use Case 2 - Mobile Users' Data Collection

We present several scenarios related to mobile communication tracking where we could try to enhance privacy. In all of these scenarios, a 3$^{\text{rd}}$ party should verify if some mobile users' identifiers from a private list are included in another list. For instance, such a party could be endorsed by an auditing company. Ideally, all the lists should remain private to each of the involved parties including the evaluating 3$^{\text{rd}}$ party.

Need of GDPR Compliant IT-Solutions.

The General Data Protection Regulation (GDPR) became enforceable beginning 25 May 2018. Its two main objectives are, firstly to enhance the personal data protection by processing them and, secondly to empower the companies in charge of this processing procedure. The European regulation integrates notion of *Privacy by design* which requires data protection to be designed into the development of business processes for products and services. Telecommunication service providers or operating system companies should not only care about securing data "at rest" or being transported but also during computations. Even if GDPR does not regulate national security [HvdSB18], developing private protocols for access logfiles would limit government agencies to solely access private data of specific suspects instead of a massive data analysis of all mobile users. In fact, this could bring the telco providers into the dilemma of fundamental ethical issues. However, in case a telecommunication provider and a government agency choose to collude, no solution could bring privacy enhancement anymore. Nevertheless, we argue that a telco provider respectively Google or Apple do

not have any interest in aggressively colluding with the agencies. Instead, the telco providers may have an interest to act GDPR compliant regarding all stored consumer's location data. The same holds for Apple and Google at least when purely considering the users' geolocation data. Thus, we assume telco providers and other digital players always in the dilemma to act either GDPR compliant or to serve the agencies directives as for instance due to the *Patriot Act* in the US.

2.2.1 Scenarios

Without loss of generality, we identified three different scenarios where users of mobile devices could be aimed by massive data collection and therefore, privacy issued are of value.

Privacy Friendly Tracking for Telecommunication Providers.

Access points, respectively base stations are deployed and as a side effect, collect connection data like identifiers. They also collect connection times of respective mobile devices that connect and communicate via these access points. At one point, a government agency could request the telco provider to compare the access points' logfiles to a list of suspects. For obvious reasons and according to the GDPR, privacy should be preserved in both ways and a 3rd party may be deployed for this. More concretely, the government agency should not be able to read all access logfiles revealing the movement patterns of the mobile users and the mobile telco provider should not be able to infer the suspects. Moreover, neither the involved 3rd party should be able to get insights to access the telco provider's lists and suspected persons. We observe in Figure 2.3 the framework of the scenario.

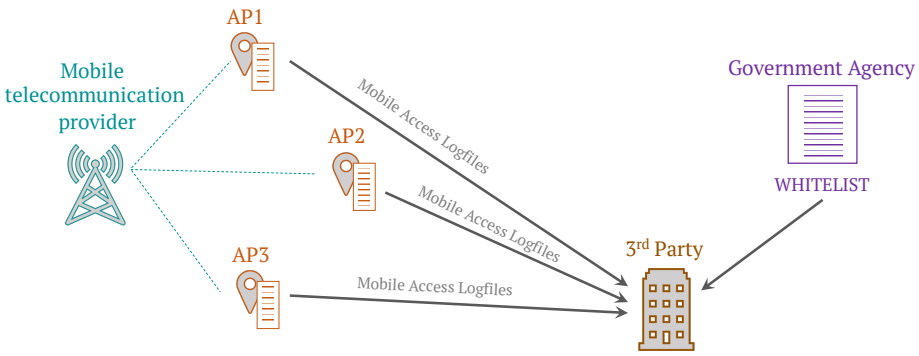

Figure 2.3. UC2 - First scenario - Privacy friendly tracking for a telco provider controlling multiple access points.

Parties:

- **Mobile Telecommunication Provider:** deploys multiple access points (APs) to provide 3G, 4G, LTE or onward access to mobile devices.
- **Government Agency:** would like to verify if some specific mobile users were connected within transmission range of any access points.
- **3rd Party:** will perform the verification and guarantee privacy for all unsuspected mobile user data.

Sets:

- **Access Logfiles:** there exists a mobile access logfile per each access point containing the list of the devices that connected Ids along with other information like time or duration of connection. Identifiers are available in format of Temporary Mobile Subscriber Identity (TMSI) or International Mobile Subscriber Identity (IMSI).
- **Whitelist:** represents a list of suspect users or devices generated by some government agency. Obviously this list is highly sensitive and should mandatorily remain confidential.

Network of WAPs.

In our 2nd sub-use case we consider mobile users that move through a Wireless Local Area Network (WLAN) connecting to different Wireless Access Points (WAP) with Wi-Fi. Each of the WAPs also collects the connection information into an access logfile. Consider this network to be under control of a single administrative party, let it be an IT-administrator within an airport, a mall, a university campus or similar places, or even FreiFunk activities at public places. The users' identifiers could be most probably represented as MAC-addresses. Since almost all WLAN capable mobile devices continuously try to find WLAN access points by using active mode (instead of passive mode) they will reveal their MAC address with the broadcast probe request beacons even if WLAN is only activated and no connection has been established yet. At any time, a government agency would like to verify if any specific mobile device user accessed the WLAN or sent probe request beacons. Also for this scenario we recommend that the 3rd party privately perform the verification. We show in Figure 2.4 the framework of the scenario.

Parties:

- **Hosting Party of Wireless Access Points:** provides 1st-hop internet access via Wi-Fi to the user.
- **User:** connects or sends probe request beacons to the WLAN via different wireless access points.
- **Government Agency:** would like to verify if any specific user have been connected to specific Wireless Access Points.

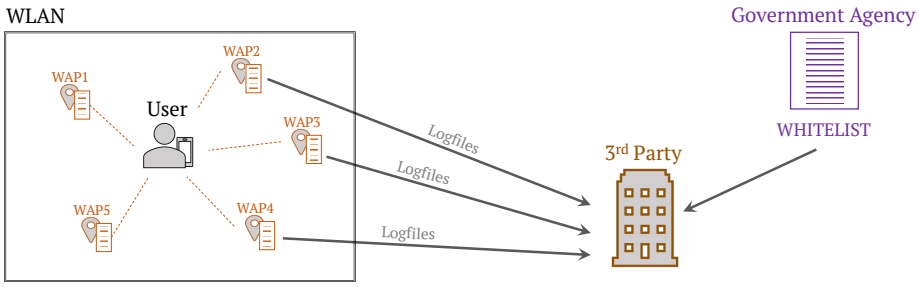

Figure 2.4. UC2 - Second scenario - Network of WAPs.

- **3rd Party:** will perform the verification and guarantee privacy on the government agency and the WLAN along with its respective mobile users.

Sets:

- **Access Logfiles:** there is a logfile per each WAP containing the list of the Ids of the devices that either connected or sent probe requests. The Ids are represented as MAC-addresses.
- **Whitelist:** represents a list of suspected users or devices generated by some government agency. As in the previous use case this list is highly sensitive and should remain private under any circumstances.

Mobile OS Providers.

In the 3rd scenario, we only consider modern smartphones equipped with a mobile operating system (OS) either from Google or from Apple (Android or iOS). We mention that besides smartphones and smartwatches, other embedded devices could be considered, e.g. IP-camera with Android or cars with their on-board units (OBU). Under this pre-requisite, smartphones and other devices running Android or iOS, are collecting location-based data, for instance by continuously making use of the approximation of RSSI (Received Signal Strength Indication) values from nearby telco base stations as well as WLAN access points. Such local logfiles are e.g. *cache_encryptedA.db*, *lockCache_encryptedA.db*, *cache_encryptedB.db* and *cache_encryptedC.db* under iOS for WLAN access points, movement pattern and others. These data constitute a personal logfile in each user's smartphone and all the users' logfiles are retrieved by the operating system companies (Apple, Google) that aggregate them into a massively large master logfile. Modern smartphones equipped with iOS or Andoid harvest these location data, store them locally on the device and from time to time send them in an encrypted manner to Google or Apple. Dhein and Grimm have investigated this swarm mapping based location harvesting approach very accurately and in detail [DG17]. Finally, as for the two

previous scenarios, some government agency could require or even force Apple and Google to access some specific user's itinerary. We show in Figure 2.5 the framework of the scenario.

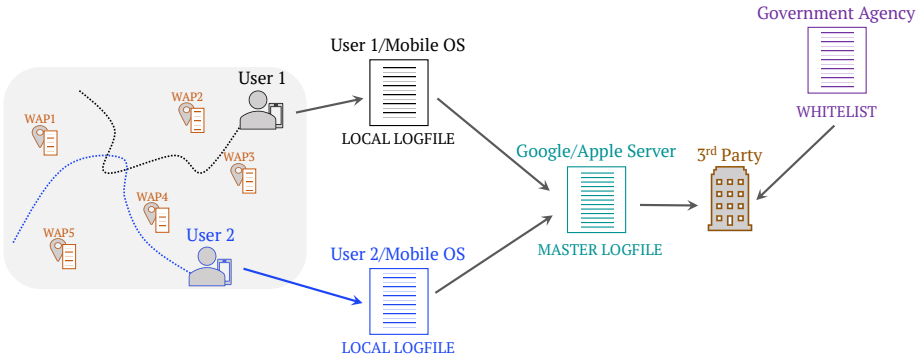

Figure 2.5. UC2 - Third scenario - Mobile OS providers continuously harvesting geo-location data.

Parties:

- **User:** owns a modern smartphone which continuously gets RSSI values from the access points (both WAP and base station).
- **Wireless Access Points (3G, 4G (and onward) mobile base stations as well as WLAN access points):** reveal their signal strength being accumulated into RSSI values at the smartphone side. Also provide various forms of connectivity to the smartphone users.
- **Mobile Operating System Company:** collects location-based information on each user's smartphone and aggregates all the user's logfiles into a master logfile.
- **Government Agency:** would like to verify if users with e.g. smartphones, smart watches, tablets or similar, have been in proximity resp. in the transmission range of either some WAPs or base stations.
- **3rd Party:** will perform the verification and guarantee privacy on the government agency and the operating system company along with its users.

Sets:

- **Local Logfiles:** composed by iOS and Android location-based data received from the RSSIs and stored on each users' smartphone.
- **Master Logfile:** both mobile operating system companies aggregate the personal logfiles of its respective clients into a massively huge master logfile.

- **Whitelist:** represents a list of users or devices generated by some government agency. Highly sensitive and should mandatorily remain confidential.

2.2.2 Threat Model

As we have seen previously, the mobile company has no interest in colluding with the government agency. Indeed, unless any case of force majeure as the *Patriot Act*, the mobile company's reputation relies on how they protect their customers' privacy. Also, we could state that neither the mobile company nor the government agency have any interest in colluding with the 3rd party. Indeed, it is too sensitive for the governement to share any of their data with an external party and the mobile agency could, apart from this protocol, try to sell the data to external parties.

Mobile Company.

By definition, the mobile company owns all clients' sensitive data. Providing privacy to customers against their mobile company is out of scope here, therefore we consider the company as **honest-but-curious** with no active motivation to alter the protocol.

Government Agency.

The agency could legally access customers' data. The whole point of this use case is to propose an alternative protocol where the agency only access the data from suspicious customers. We consider the agency accepting such a protocol. Thereby, it makes no sens to consider any bad behavior from the agency. It is then viewed as an **honest-but-curious** party.

3rd Party.

On the contrary, this party will be invited to participate in the protocol, manipulating sensitive data, which does not occur in the classic way of tracking customers' data. Therefore, the 3rd party could seek to take advantage from its position. Collecting the sensitive data in order to make profit from them should be considered, so we describe the 3rd party as a **malicious** party.

2.3 Use Case 3 - Wireless Sensor Network's Data Aggregation

A wireless sensor network could be considered as a large number of low-cost sensor nodes deployed in a monitoring area. Their objective is to sense, collect, and process cooperatively the information in the network distributed area and then forward the results to a base station. The network could consist of a multihop infrastructure-less network system formed by wireless communication

method. We emphasize that the nodes are space and computationally limited devices which could store only few data, and an highly usage could severely reduce their lifetime. In Figure 2.6 we present a framework of a basic version of WSN.

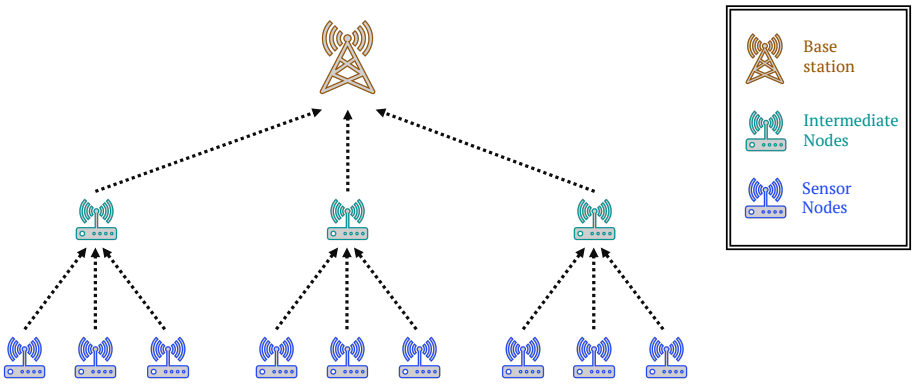

Figure 2.6. Wireless Sensor Network.

There exists various versions of WSNs as widely studied. For instance, we could mention one of the specific types of WSN called Cognitive Wireless Sensor Networks (CWSNs) as described in [Sen13]. It consist of a type of WSNs in which the sensor nodes have the capabilities of changing their transmission and reception parameters according to the radio environment under which they operate in order to achieve reliable and efficient communication and optimum utilization of network resources.

As stated in [YZFC15], WSN's fields of application are manifolds. They are widely used in military defense, industry, agriculture, construction and urban management, biomedical and environmental monitoring, disaster relief, public safety and anti-terrorism, hazardous and harmful regional remote control, and so on which are much accounted by many governments. WSNs have a very important scientific and practical value. To cite a concrete example from [LZDT09], a patient's blood pressure, sugar level and other vital signs are usually of critical privacy concern when monitored by a medical WSN which transmits the data to a remote hospital or doctor's office.

2.3.1 Scenario

Collecting data from the sensor nodes to the base station through intermediate nodes without considering any aggregation functions or protocol will increase the cost in terms of energy and therefore reduce the WSN's lifetime. Indeed, as we see in Figure 2.7, communication consumes much more energy when we do not consider data aggregation from different sensor nodes.

We consider two data processing requirements namely the latency and the accuracy:

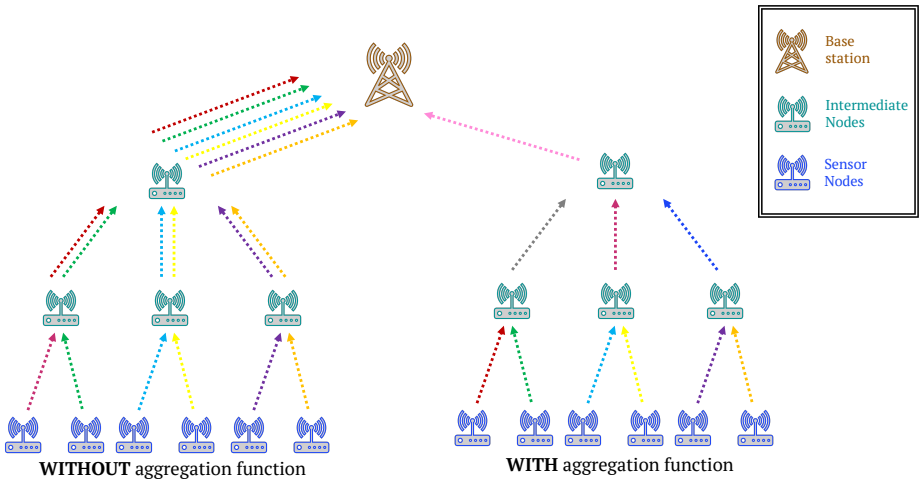

Figure 2.7. Wireless Sensor Network without and with aggregation functions.

Latency. We consider as *latency* the time delay between the generation of the sensor readings at the leaf nodes and the reception of sensor readings at the base station. The latency could be increased for instance by the decryption and re-encryption at each intermediate nodes in a hop-by-hop construction.

Accuracy. The accuracy or correctness corresponds to the difference between the result collected by the sensor nodes and the one received at the base station. A difference could be made because of an adversary or an error that occurred during the aggregation function. Depending on the requirements from the base station, some errors could be tolerated or corrected later.

Parties Involved.

We consider the WSN composed of a base station and sensor or intermediate nodes.

Sensor Nodes ($\mathcal{SN}s$): consist of low-cost sensor nodes which are resource-constrained in terms of computation, communication, storage and power supply. They are especially sensitive to energy consumption thus, a data aggregation protocol could be relevant.

Intermediate Nodes ($\mathcal{IN}s$): placed in between the sensor nodes and the base station. Have to collect data from the sensor nodes and retrieve them to the base station. In case of an optimization using an aggregation protocol, the intermediate nodes will have to perform processing on data.

Base Station ($\mathcal{BS}$): central, powerful party which is responsible for retrieving data from the nodes. Could also be qualified as *server* or *sink*.

For the current scenario we consider the environment of *Industry 4.0*. In such surroundings, factories own machines which are augmented with wireless connectivity and sensors. They are connected to a system such as the base station that can visualize the entire production line and make decisions on its own. We consider in each of the machines, some sensors evaluating the need of maintenance along with other information as machine's temperature or its spin speed. Maintenance is instructed by an higher authority, namely the base station. The scenario is illustrated in Figure 2.8 and takes place like this:

Each machine i owns some sensitive information, as a status st_i or some information as a temperature t_i. As an example we could have:

$$\begin{cases} st_i = ok \text{ or } not\ ok \\ t_i = 45 \end{cases} \tag{2.1}$$

We note that all sensor nodes from the same machine are supposed to own the same values.

For each epoch e_j, $\mathcal{BS}$ needs to retrieve all the machine's values to evaluate the need of maintenance. The intermediate nodes first collect values from a unique machine and perform a **rule** on them. A **rule** could be:

Majority: comparing the values and keeping the one mainly represented. (e.g. a temperature).

Exclusiveness: aggregation of all different values. (e.g. an error message).

Then $\mathcal{IN}s$ aggregate values from different machines to save bandwidth and energy. Needless to say, $\mathcal{IN}s$ should not be able to read the machine's status nor information.

We emphasize that it is not sufficient for the base station to know if at least one machine, or even how many of them need maintenance. We would like the base station to know the status of each specific machine by retrieving their respective values, to know exactly which ones need assistance.

2.3.2 Threat Model

To consider threats, we have to analyze the characteristics of possible adversaries. Another aspect to highlight is how the sensor hardware is protected against intrusion.

Insider adversary - Outsider adversary We consider an adversary as *insider* if it has corrupted a node and remains part of the network. It can then access to all the private information from that node and take its control. The *outsider* adversary settles for listening and analyzing communication between nodes within the network.

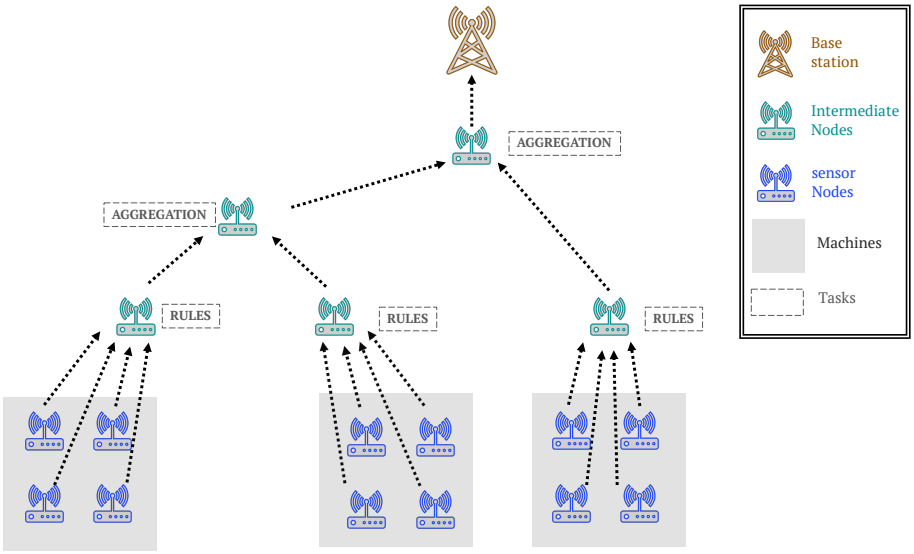

Figure 2.8. Industrial Sensor Network.

Passive adversary - Active adversary A *passive* adversary analyzes the communication traffic to extract sensitive information. An *active* adversary can create or modify a packet in the network. An *active* adversary could thus temper the integrity and freshness of data while a *passive* adversary could violate confidentiality and privacy.

Adversary Model:

Sensor nodes are not equipped with tamper-proof hardware, and the adversary is capable of compromising and fully controlling arbitrary number of sensor nodes. Therefore, we assume all the nodes, namely sensor or intermediate, as **malicious** parties. On the contrary, we assume that $\mathcal{BS}$ is immune to all physical attacks and is trustworthy. We thus consider it as an **honest-but-curious** party. Moreover, the adversary can eavesdrop and alter any messages from honest nodes. In a word, we consider an adversary granted with a full-scale attack capacity against the sensor network except $\mathcal{BS}$.

2.4 Use Case 4 - Detection of COVID-19 Infection Chains

Cases of COVID-19 virus have been reported in more than 190 countries and its spreading has been characterized as pandemic by the *World Health Organization* on 11.03.2020. One of its multiple side effects consists of European democracies

being challenged. Indeed, several countries are collecting location-based data from their own citizens. The *state of emergency* for health reasons has been established in countries as Spain, Portugal, France or Switzerland. Such a specific situation empowers a government to perform actions that would normally not be allowed to undertake. For instance, in Milano, Italy, mobile network operators are providing information on users' traffic to public authorities. In Germany, issues regarding how and for which usage to process the location-based information are ones of the most discussed. Indeed, efforts in Germany are twofold regarding digital support to detect infection chains. First, with an app which is currently under investigation. It consists of using a tracking app with Bluetooth in which a smartphone of an infected user is subsequently informing all devices which have been in proximity (within the beaconing received range at some point in time in the past). Such an approach is very vulnerable due to the requirement of continuously activated Bluetooth. The recently published family of BlueBorne attacks [ARY+19] have shown that mobile devices with activated Bluetooth can easily be remotely executed, e.g. *CVE-2017-0781*, *CVE-2017-0782* or *CVE-2017-14315* and are classified as a severe risk. Moreover, it has been pointed out that the harvesting of contacts via Bluetooth with a tracking app is only properly working in case the app is activated continuously in the foreground, and, moreover, that at least 60% of the smartphone users need to download and continuously use it to indeed have an impact with respect to the identification of infection chains.

Second, telco operators would provide access logfiles of mobile network base stations to RKI (Robert Koch Institute) to support inferring infection chains.

On the contrary, the Netherlands' government decided to not approve a general confinement, for the reason of being incompatible with individual freedom.

For these reasons, we attempt to propose a construction which combine the efficiency to help the public authorities to contain the virus spreading with the possibility to provide privacy with respect to the citizens. Therefore, we concentrate on providing a privacy-preserving solution for the 2nd effort currently done within Germany.

We recall that GDPR's two main objectives are to firstly enhance the personal data protection by processing them and to secondly empower the companies in charge of this processing procedure. Even if this regulation does not apply on fields as public health or national security [HSB19], weaving the proposed Bloom filter based private protocols into infection chains investigation would limit government agencies to solely identify users with high probability of being infected instead of a massive data analysis of all mobile users.

2.4.1 Scenario

A government agency, which role is to reduce the spreading of the COVID-19 virus in its country, knows different pairs of infected persons (A, B). Its objective here, is to identify all the possible paths which relies user A to user B and considers the case where infection of user B is a consequence of user A's infection. By retrieving all possible paths (surely it could also turn out that no

path exists and the infection of users A and B was unrelated), the agency could identify all the users within this path that may be also infected by the virus and try to contact them. Indeed, different mobile device's users close to the same mobile base station at the same time could potentially spread the virus in case of one being infected. To do so, the agency is analyzing connection data provided by a telco company. The connection logs are collected on the base stations which are providing network access to the users' mobile devices.

Parties Involved.

Four parties are involved in the scenario:

Users: could be infected by the COVID-19 virus. They are connecting to the base stations to access the mobile network.

Telco company: provides network to the users via several base stations. It also provides log data from the network connections to government agencies.

Base stations: are distributed over several countries, provide network to the users' mobile devices and collect connection data.

Government agency: aims to identify "infection chains" in order to contact the possible infected users and counteract the virus pandemic.

Collecting Connection Data

Any time a user is connecting to the mobile network using base station j, the following connection information is collected and aggregated by the telco company:

$$(id_i, t_i^1, t_i^2)$$

with id_i the user's credentials and t_i^1 and t_i^2 respectively the starting and ending times of its connection to the access point. Such connection data should be considered as sensitive regarding the location privacy of the users. Indeed, on the one hand the base stations are using usernames to characterize the users and on the other hand only the telco company could generate and access the connection information from the base stations.

Chapter 3

Preliminaries and their Respective Literature

We introduce notions and primitives used to develop solutions in Chapter 4, along with their respective literature. The different topics correspond to searchable encryption, homomorphic encryption, private information retrieval protocols, set operations, Bloom filter and Concealed Data Aggregation protocols. We stress that all of these notions allow directly or indirectly to perform computations on private data.

3.1 Searchable Encryption

Cryptographic primitive searchable encryption offers secure search functions over encrypted data, that is the search executing party does not learn any information about the plaintext data. In general, keyword indexes are built to securely and efficiently perform search queries. The two main techniques used in searchable encryption are Searchable Symmetric Encryption (SSE) and Public Key Encryption with keyword Search (PEKS). The latter, proposed by Boneh *et al.* [BCOP04] will be employed in our proposed solution in Section 4.1 and is defined as follows.

The PEKS scheme $\mathtt{SE}$ consists of the five algorithms:

$\mathtt{SE}.\mathrm{KeyGen}(\lambda) \to (pk_{\mathtt{SE}}, sk_{\mathtt{SE}})$: Given a security parameter λ, the public/private key pair $(pk_{\mathtt{SE}}, sk_{\mathtt{SE}})$ is generated.

$\mathtt{SE}.\mathrm{Enc}(pk_{\mathtt{SE}}, m) \to c$: Given the public key $pk_{\mathtt{SE}}$ and a message m, it generates a ciphertext c.

$\mathtt{SE}.\mathrm{PEKS}(pk_{\mathtt{SE}}, w) \to S_w$: Given the public key $pk_{\mathtt{SE}}$ and a keyword w, it generates a PEKS ciphertext S_w of w.

$\texttt{SE}.\text{Trapdoor}(sk_{\texttt{SE}}, w) \rightarrow T_w$: Given a keyword w and the private key $sk_{\texttt{SE}}$, it produces a trapdoor T_w.

$\texttt{SE}.\text{Test}(pk_{\texttt{SE}}, S_w, T_{w'}) \rightarrow \{0,1\}$: Given the public key $pk_{\texttt{SE}}$, a searchable encryption ciphertext S_w, and a trapdoor $T_{w'}$, it outputs 1 (`true`) if $w = w'$ or 0 (`false`) otherwise.

3.2 Homomorphic Encryption

3.2.1 Definition and Classification of Homomorphic Cryptosystems

Definition 1. *A cryptosystem is called homomorphic if an algebraic operation processed on ciphertexts is equivalent to one processed on plaintexts.*

More formally, let it be $\mathcal{M}$ the messages space, $\mathcal{C}$ the ciphertexts space and $\mathcal{E}nc$ an encryption function:

$$\forall m_1, m_2 \in \mathcal{M},\ \mathcal{E}nc(m_1 \odot_{\mathcal{M}} m_2) \leftarrow \mathcal{E}nc(m_1) \odot_{\mathcal{C}} \mathcal{E}nc(m_2)$$

with $\odot_{\mathcal{M}}$ and $\odot_{\mathcal{C}}$, operators in respectively $\mathcal{M}$ and $\mathcal{C}$. $\leftarrow$ means "could be directly computed from" without any decryption.

Homomorphic encryption allows valid computations directly on the ciphertext, generating an encrypted result which matches the result according operations performed on the plaintext, when decrypted. An evaluation algorithm is added to encryption and decryption which operates on ciphertexts.

According to the type of computation, we distinguish several cryptoschemes' types: *partially homomorphic*, *somewhat homomorphic* or *fully homomorphic*.

Partially homomorphic. These cryptoschemes solely allow one type of computation on the ciphertexts. For instance, the Paillier's scheme [Pai99] is additively homomorphic, i.e. given the public key and two ciphertexts one can compute the encryption of the addition of both respective plaintexts .

$$\mathcal{D}ec(\mathcal{E}nc(m_1) \times \mathcal{E}nc(m_2)) \equiv m_1 + m_2$$

Another example is the "unpadded RSA" scheme where a multiplication of two ciphertexts is equivalent to a multiplication of two cleartexts.

$$\mathcal{D}ec(\mathcal{E}nc(m_1) \times \mathcal{E}nc(m_2)) = x_1 \times m_2$$

These cryptoschemes allow an unlimited amount of additions or multiplications but not both at once.

Somewhat homomorphic. Allow an unlimited amount of additions and a limited amount of multiplications. For instance, there is the Brakerski and Vaikuntanathan's scheme [BV11a]. We could also cite [DPSZ12] where somewhat homomorphic encryption schemes are used for multiparty computation.

Fully homomorphic. In [Gen09], Craig Gentry firstly introduced a fully homomorphic encryption scheme which allows unlimited amounts of additions and multiplications on the ciphertexts. Nevertheless, this scheme is far from being practical due to its computation complexity, and since currently, none of the known FHE schemes are efficient enough to provide applicability [ABC+15]. We restrict ourselves to Paillier's and SHE instead.

This kind of cryptoschemes, mainly asymmetric, is part of the modern technique of cryptography. It brings additional features that are highly relevant according to its scope of application. Obviously these new features bring another level of complexity which could classify this kind of cryptography as heavyweight compared to classical schemes.

3.2.2 Partially Additive Cryptosystems

As we will see in the PIR protocols' description, we are particularly interested in the additive feature of the homomorphic cryptosystems. Therefore, we look in detail different kinds of additive schemes in order to compare their complexities.

First, we give some mathematical reminders, required to a perfect understanding of the schemes.

Definition 2. *Let it be N a product of two prime integers p and q, we consider the **Euler's function** as $\phi(N)$ where $\phi(N) = (p-1)(q-1)$.*

Property 1. *$N = p * q$ is an **RSA module** with p and q two prime integers and $p < q$. p does not divide $q-1$, in other words, $pgcd(N, \phi(N)) = 1$.*

Definition 3. *The **Carmichael function** of a positive integer d, as $\lambda(d)$, consists of the smallest integer m such as $a^m \equiv 1 \mod d$ for all integer a between 1 and d that is coprime to d.*

Property 2. *For all $w \in \mathbb{Z}^*_{N^2}$*

$$w^{\lambda(N)} = 1 \mod N \text{ and } w^{N\lambda(N)} = 1 \mod N^2.$$

Definition 4. *We consider an integer q as **quadratic residue modulo** p if there exists an interger x such that:*

$$x^2 \equiv q \mod p.$$

Definition 5. *For $u \in \mathbb{Z}^*_{N^2}$, we define the L **function** as follows:*

$$L(u) = \tfrac{u-1}{N} \mod N^2$$

Definition 6. *The **Legendre symbol** $(\frac{a}{p})$ of an integer a and an odd prime p consists of:*

$$(\frac{a}{p}) = \begin{cases} 1 & \textit{if } a \textit{ is a quadratic residue modulo } p \textit{ and } a \not\equiv 0 \mod p \\ -1 & \textit{if } a \textit{ is a non-quadratic residue modulo } p \\ 0 & \textit{if } a \equiv 0 \textit{ mod } p \end{cases}$$

Definition 7. *The **Jacobi symbol** $(\frac{a}{n})$ corresponds to the product of the Legendre symbols of the prime factors of n:*

$$(\frac{a}{n}) = (\frac{a}{p_1})^{\alpha_1} \times (\frac{a}{p_2})^{\alpha_2} \times \cdots \times (\frac{a}{p_k})^{\alpha_k}$$

where $n = p_1^{\alpha_1} \times p_2^{\alpha_2} \times \cdots \times p_k^{\alpha_k}$

Examples of Additive Cryptosystems

We present now several additive homomorphic cryptosystems:

Goldwasser-Micali [GM82]. Developed by Shafi Goldwasser and Silvio Micali in 1982, this cryptosystem is a bitwise encryption scheme. It is defined by the following functions:

- *Keygen*: z is generated as a non quadratic residue such that its Jacobi symbol $(\frac{z}{N}) = 1$, in particular if $pgcd(p, q) \equiv 3 \mod 4$ then $N - 1$ is one of these elements. We have the two following keys: $pub.key = (N, z)$ and $sec.key = (p, q)$.
- *Enc*: to encrypt the i-th bit of a message $m_i \in \{0, 1\}$, we randomly generate $y \in \mathbb{Z}_N^*$ and compute $c_i = y^2 z^{m_i} \mod N$.
- *Dec*: to decrypt c_i we compute the following Jacobi symbol:

$$(\frac{c_i}{N}) = (\frac{c_i}{p})(\frac{c_i}{q}) = \begin{cases} 1 * 1 & \textit{then } m_i = 0 \\ \textit{else} & m_i = 1 \end{cases}$$

- Properties:
  - adding two ciphertexts $\mathcal{E}nc(x_1 \oplus x_2) = \mathcal{E}nc(x_1) * \mathcal{E}nc(x_2)$, with $\oplus$ the addition operator within $\mathbb{Z}_2$.
  - multiplying a ciphertext with a cleartext k: $\mathcal{E}nc(kx \mod 2) = \mathcal{E}nc(x)^k$.
- Security: based on the quadratic residuosity problem, i.e. guessing if a random element x from $\mathbb{Z}_N^*$ is a square.

Benaloh [Ben94]. This cryptosystem is derivated from the Goldwasser-Micali's presented in 1994 by Josh Benaloh. The message space is r bits but with a pretty long decrypting process.

- *Keygen*: on select an RSA module such that r divides $p - 1$ but not $q - 1$ neither $\frac{p-1}{r}$. Therefore, we get that r divides $\phi(N)$ and $pgcd(r, \frac{\phi(N)}{r}) = 1$. We randomly generate $g \in \mathbb{Z}_N^*$ such that g is not a r-th power residue. We get the following keys: $pub.key = (N, r, g)$ and $sec.key = (p, q)$.

- $\mathcal{E}nc$: to encrypt a message $m \in \mathbb{Z}_r$ we randomly generate $z \in \mathbb{Z}_N^*$ and we compute $c = g^m z^r \mod N$.
- $\mathcal{D}ec$: to decrypt c we test, for all $j \in \mathbb{Z}_r$, if $(cg^{-j})^{\frac{\phi(N)}{r}} = 1 \mod N$ (i.e. cg^{-j} is a r-th power residue) then $m = j$.
- Properties:
  - adding two ciphertexts $\mathcal{E}nc(x_1 + x_2) = \mathcal{E}nc(x_1) * \mathcal{E}nc(x_2)$
  - multiplying a ciphertext with a cleartext k: $\mathcal{E}nc(kx) = \mathcal{E}nc(x)^k$
- Security: based on the hardness to decide if a random element from $\mathbb{Z}_N^*$ is a r-th power residue modulo a composite integer. It generalizes the quadratic residuosity problem but unfortunately, the decryption process gets pretty long. There exists an optimization based on the Pollard's rho algorithm [Pol78] with a decrypting complexity of $O(\sqrt{r})$.

Naccache-Stern [NS98]. In 1998, David Naccache and Jacques Stern proposed an alternative version to enhance the message space and reduce decryption time. It uses the Pohlig-Hellman algorithm [PH78].

- $\mathcal{K}eygen$: we select an RSA module N and $\beta = \prod_{i=1}^{k} p_i$ a divisor of $\phi(N)$ such that $pgcd(\beta, \frac{\phi N}{\beta}) = 1$ and with p_i integers relatively prime and $g \in \mathbb{Z}_N^*$ with an order equal to a multiple of β. We get the following keys: $pub.key = (N, \beta, g)$ and $sec.key = (\phi(N), p_1, p_2, \ldots, p_k)$.
- $\mathcal{E}nc$: to encrypt a message $m \in \mathbb{Z}_\beta$ we randomly generate $z \in \mathbb{Z}_N^*$ and compute $c = g^m z^\beta \mod N$.
- $\mathcal{D}ec$: at first, we determine the m_i as follows: we compute $c_i = c^{\frac{\phi(N)}{p_i}} \mod N$ and as $c_i \equiv g^{\frac{m_i \phi(N)}{p_i}} \mod N$ then we compare c_i to $g^{\frac{j\phi(N)}{p_i}}$ with $j \in N$ and $1 < j < p_i - 1$. After retrieving all the m_i, as $m_i \equiv m \mod p_i$, we can retrieve m with the use of the Chinese remainder theorem.
- Properties:
  - adding two ciphertexts $\mathcal{E}nc(x_1 + x_2) = \mathcal{E}nc(x_1) * \mathcal{E}nc(x_2)$
  - multiplying a ciphertext with a cleartext k: $\mathcal{E}nc(kx) = \mathcal{E}nc(x)^k$
- Security: based on the higher residuosity problem, i.e. a generalization of the quadratic residuosity problem when $d = 2$.

Okamoto-Uchiyama [OU98]. Presented in 1998 by Tatsuaki Okamoto and Shigenori Uchiyama, this scheme allows to decrypt without the need of exhaustive research. Indeed, in the previous schemes, we've performed some tests for every power. Now we use an RSA module as $N = p^2 q$ and we notice that $(1 + p)^m = 1 + mp \mod p^2$ using the binomial theorem.

- $\mathcal{K}eygen$: we then select an RSA module as $N = p^2q$ with p of size k. We select $g \in \mathbb{Z}_N^*$ such that $g^{p-1} \mod p^2$ has order p. Then we get: $pub.key = (N, k, g)$ and $sec.key = (p, q)$.
- $\mathcal{E}nc$: for encrypting a message $m \in [0, 2^{k-1}]$ we randomly generate $z \in \mathbb{Z}_N^*$ and we compute $c = g^m z^N \mod N$.
- $\mathcal{D}ec$: we perform the following formula: $m = \frac{(c^{p-1} - 1 \mod p^2)/p}{(g^{p-1} - 1 \mod p^2)/p} \mod p$.
- Properties:
  - adding two ciphertexts $\mathcal{E}nc(x_1 + x_2) = \mathcal{E}nc(x_1) * \mathcal{E}nc(x_2)$
  - multiplying a ciphertext with a cleartext k: $\mathcal{E}nc(kx) = \mathcal{E}nc(x)^k$
- Security: Based on the factorization using the same technique as before. If an attacker knows the encryption function, then by encrypting a value higher than p and asking for a decryption, he could get p and therfore break the RSA module.

Paillier [Pai99]. Presented by Paillier in 1999, this scheme extends Okamoto and Uchiyama's idea with a module as $N^2 = p^2q^2$. It is based on the Carmichael function properties.

- $\mathcal{K}eygen$: we generate $g \in \mathbb{Z}_{N^2}^*$ such that g has order N (we could select $g = 1 + kN$ with $k \in \mathbb{N}$).
  Then we get: $pub.key = (N, g)$ and $sec.key = (\lambda(N))$
- $\mathcal{E}nc$: to encrypt a message $m \in \mathbb{Z}_N$ we randomly generate $r \in \mathbb{Z}_N^*$ and compute $c = g^m r^N \mod N^2$.
- $\mathcal{D}ec$: to decrypt c we compute $m = \frac{L(c^{\lambda(N)} \mod N^2)}{L(g^{\lambda(N)} \mod N^2)} \mod N$.
- Properties:
  - adding two ciphertextss $Enc(x_1 + x_2) = Enc(x_1) * Enc(x_2)$
  - multiplying a ciphertext with a cleartext k: $Enc(kx) = Enc(x)^k$
- Security: Based on the hardness to distinguish between a residue of order $N \mod N^2$ and a non-residue of order N.

Analyzing and Comparing the Additive Cryptosystems

In Section 3.2.2, a relevant aspect to compare the schemes is the expansion ratio (i.e. length's increase of a message when it is encrypted). We easily notice that the smaller the expansion is, the better the cryptosystem is in terms of communication complexity.

3.2.3 Somewhat Homomorphic Encryption

The Ring Learning with Errors Problem

The Ring Learning with Errors (RLWE) problem, presented by Lyubashevsky *et al.*in [LPR10], corresponds to the larger Learning with errors (LWE) problem

Table 3.1. Comparing the additive cryptosystems (from our own contribution).

	Message space	Cyphertext space	Expansion
Goldwasser-Micali	$\mathbb{Z}_2$	$\mathbb{Z}_N$	$\log(N)$
Benaloh	$\mathbb{Z}_r$	$\mathbb{Z}_N$	$\frac{\log(N)}{\log(r)}$
Naccache-Stern	$\mathbb{Z}_\sigma$	$\mathbb{Z}_N$	≈ 4
Okamoto-Uchiyama	$\mathbb{Z}_p$	$\mathbb{Z}_N$	≈ 3
Paillier	$\mathbb{Z}_N$	$\mathbb{Z}_{N^2}$	≈ 2

specialized to polynomial rings over finite fields. The solution to the RLWE problem may be reducible to the NP-Hard Shortest Vector Problem (SVP) in Lattice. The RLWE assumption is characterized by multiple parameters, rings $R := \mathbb{Z}[x]/\langle f(x)\rangle$ and $R_q := R/qR$ for some degree n, integer polynomial $f(x) \in \mathbb{Z}[x]$, a prime integer $q \in \mathbb{Z}$ and an error distribution χ over R. The ring R_q thus represents the ring of degree n polynomials modulo $f(x)$ with coefficients in $\mathbb{Z}_q$. With these configuration, addition in this ring is done componentwise in their coefficients and multiplication is simply polynomial multiplication modulo $f(x)$ and q. Let $s \xleftarrow{\$} R_q$ be a uniformly random generated ring element. The assumption says that given any polynomial number of samples of the form $(a_i, b_i = a_i \times s + e_i) \in (R_q)^2$, where a_i is uniformly random in R_q and e_i is drawn from the error distribution χ, the b_i's are computationally indistinguishable from uniform in R_q. This specific case of the RLWE assumption where ring elements are represented as polynomials has been highlighted by Brakerski and Vaikuntanathan in [BV11b].

The SHE Scheme

The studied encryption scheme here is the somewhat homomorphic encryption scheme presented by Brakerski and Vaikuntanathan in 2011 [BV11b]. This scheme requires to firstly encode the message into a polynomial representation subsequently to encrypt it with the respective encryption function resulting in a couple of polynomials. Figures 3.1 and 3.2 illustrate the steps to encrypt a message.

Message Space. The plaintext polynomial space corresponds to $R_t = \mathbb{Z}_t[x]/(x^n + 1)$ with two parameters t and n. The polynomial degree n limits the amount of coefficients while parameter t bounds the coefficients' size.

In [Bie14], Bieberstein considers one additional parameter comparing to the presentation of this scheme by Brakerski *et al.*: the base b. This parameter

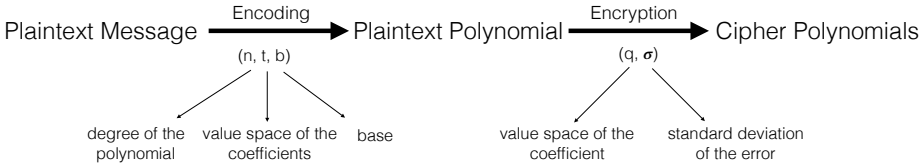

Figure 3.1. Encoding and encryption path for a message.

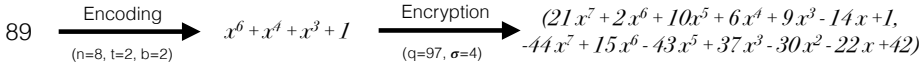

Figure 3.2. A concrete example with a message $m = 89$.

represents the base of all polynomials. In other words, when encoding and decoding the messages, we have $x = b$. In a classical manner we consider this parameter equal to 2 (as it is presented in [NLV11]). In this classical configuration, we are encoding the message with coefficient $c_i \in \{0, \ldots, b-1\}$. Bieberstein proposes that b could be different than 2 or t. However, he did not suggest anything regarding t while in [NLV11], t should be prime and less than q.

Distributions.

Uniform Distribution When we write $d \xleftarrow{\$} S$ that means d chosen from the uniform distribution over some finite set S. In the latter sections we specify it to the ring R_q.

Gaussian Distribution We let the distribution $\chi = D_{\mathbb{Z}^n,\sigma}$ to indicate the n-dimensional discrete Gaussian distribution. To sample a vector $x \in \mathbb{Z}^n$ from this distribution, sample $y_i \in \mathbb{R}$ from the Gaussian standard deviation σ and set $x_i := \lfloor y_i \rceil$, where $\lfloor \cdot \rceil$ represents rounding to the nearest integer. Using the isomorphism mentioned above, we treat χ as the error distribution over integer degree n polynomials defined by the probability density function in [NLV11]:

$$\forall \mathbf{e} \in \mathbb{Z}^n : \Pr[\mathbf{e} \leftarrow D_{\mathbb{Z}^n,\sigma}] = \frac{e^{-\pi \|\mathbf{e}\|^2/\sigma^2}}{\sum_{e \in \mathbb{Z}^n} e^{-\pi \|\mathbf{e}\|^2/\sigma^2}} \tag{3.1}$$

We let χ' be a noise distribution like χ, only with larger standard deviation σ' as it is presented in [BV11b].

$$\sigma' \geq 2^{\omega(log\ n)} \times \sigma \tag{3.2}$$

Key Generation. We are considering here the public-key version of the scheme such that a key pair is generated. The secret key is set as $sk = s \in R_q$

where we sample a ring element $s \xleftarrow{\$} \chi$. The public key corresponds to a RLWE instance: $pk = (a_0, b_0 = a_0 s + te_0)$. $a_0 \xleftarrow{\$} R_q$ is uniformly randomized and $e_0 \xleftarrow{\$} \chi$ a small error.

Encryption. Given the public-key $pk = (a_0, b_0)$ and a message $m \in R_q$, elements $v, e' \xleftarrow{\$} \chi$ and $e'' \xleftarrow{\$} \chi'$ are set and the ciphertext is defined as $ct = (c_0, c_1) = (b_0 v + te'' + m, -(a_0 v + te'))$.

Decryption. Given the secret key $sk = s$ and a ciphertext $ct = (c_0, c_1)$, we first compute $\widetilde{m} = c_0 + c_1 s \in R_q$. Secondly, we output the decrypted message $m \equiv \widetilde{m} \mod t$. In case of a ciphertext with more than two elements, the generic decryption formula is:

$$\widetilde{m} = \sum_{i=0}^{d} c_i \times s^i \tag{3.3}$$

Addition. Given two ciphertexts $ct = (c_0, \ldots, c_d)$ and $ct' = (c'_0, \ldots, c'_d)$, the addition of the two corresponds to the following ciphertext:

$$ct_{add} = ct + ct' = (c_0 + c'_0, \ldots, \ c_d + c'_d) \tag{3.4}$$

Namely, addition is done by coordinate-wise vector addition of the ciphertext vectors. If the two ciphertexts have a different length, we should pad the shorter ciphertext with zeroes on the most significant bits. The output of the addition of two ciphertexts $ct = (c_0, \ldots, c_\delta)$ and $ct' = (c'_0, \ldots, c'_\gamma)$ contains $\max(\delta+1, \gamma+1)$ ring elements. Therefore, addition does not increase the number of elements in the ciphertext vector.

Multiplication. Given $ct = (ct_0, ct_1)$ and $ct' = (ct'_0, ct'_1)$, the multiplication computes $ct_{mult} = (ct_0 ct'_0, ct_0 ct'_1 + ct'_0 ct_1, ct_1 ct'_1)$. Homomorphic multiplication increases the size of the ciphertext. The output of the multiplication of the two ciphertexts $ct = (c_0, \ldots, c_\delta)$ and $ct' = (c'_0, \ldots, c'_\gamma)$ contains $\delta + \gamma + 1$ ring elements and will be represented as:

$$\left(\sum_{i=0}^{\delta} c_i v^i\right) \times \left(\sum_{i=0}^{\gamma} c'_i v^i\right) = \sum_{i=0}^{\delta+\gamma} \tilde{c}_i v^i \tag{3.5}$$

3.3 Private Information Retrieval Protocol

3.3.1 Definition

Definition 8. *In a Private Information Retrieval (PIR) protocol, a server possesses a database and a user would like to retrieve the i*-th element of it. At the end of the protocol, user will get the element and server will know nothing about i.*

This protocol was first introduced by Chor *et al.* in [CGKS95]. We consider the protocol presented by Ostrovsky and Skeitk in [OI07] as the most practical one. We denote this protocol as the **Classical Protocol** and we summarize it as follows. Let i^* be the index requested by the user $\mathcal{U}$ for a n-tuple $(x_1, x_2, \ldots, x_n)$ representing a one-dimensional database owned by a server $\mathcal{S}$.

1. $\mathcal{U}$ generates a query $Q = (q_1, q_2, \ldots, q_n)$, where for all indexes $i \in \{1, \ldots, n\}$:
$$q_i = \begin{cases} \mathcal{E}nc(0) \; if \; i \neq i^* \\ \mathcal{E}nc(1) \; if \; i = i^* \end{cases} \tag{3.6}$$
and sends it to the server.
2. $\mathcal{S}$ computes and sends back response $R = \sum_{i=1}^{n} q_i x_i$.
3. $\mathcal{U}$ decrypts response and gets requested data $\mathcal{D}ec(R) = x_{i^*}$. Indeed,
$$\begin{aligned} \mathcal{D}ec(R) &= \mathcal{D}ec(q_0 x_0 + \cdots + q_{i^*} x_{i^*} + \cdots + q_n x_n) \\ &= \mathcal{D}ec(\mathcal{E}nc(0)x_0 + \cdots + \mathcal{E}nc(1)x_{i^*} + \cdots + \mathcal{E}nc(0)x_n) \\ &= 0x_0 + \cdots + 1x_{i^*} + \cdots + 0x_n = x_{i^*} \end{aligned} \tag{3.7}$$

Here, $\mathcal{E}nc$ and $\mathcal{D}ec$ respectively correspond to encryption and decryption function of an additive homomorphic scheme (e.g. Paillier's [Pai99]).

A more straightforward and naive solution to this problem could be the server transmitting the whole database to the user. Regarding the database's size, such an approach could easily become unrealistic. We distinguish two types of PIR protocol:

Computational Private Information Retrieval (CPIR). This class of protocols consists of a unique version of database and a server with a limited computational power. Therefore, privacy is guarantee against any malicious server that could not compute any information from the user's request and its response.

Information Theoretic Private Information Retrieval (ITPIR). In this protocols' class, multiple servers possess a database copy but do not communicate with each other.

In the work at hand, we limit our research to CPIR protocols. A particular case of CPIR protocols which provides privacy on both parties is the Symmetric Private Information Retrieval.

Symmetric Private Information Retrieval (SPIR). In addition to a classical protocol of PIR, a symmetric PIR brings additional privacy on the database owner. Indeed, the user does not get any additional information than the requested data. This protocol is similar to a 1-out-of-2 Oblivious Transfer problem.

We present the Oblivious Transfer problem and its 1-out-of-2 case in detail.

Oblivious Transfer (OT). This problem was firstly introduced by Michael O. Rabin [Rab05] in 1981. Similarly to the PIR protocol, it consists of a server that provides data without knowing which data in particular. Also, receiver learns nothing about other messages. Three types of OT protocols have been developed; the 1-out-of-2 OT, the 1-out-of-n OT and the k-out-of-n OT.

We give details of the first category **1-out-of-2 OT**:

1. Alice owns 2 messages, m_0 and m_1, and has to transmit one of the two to Bob without learning which one will be transmitted.
2. Alice generates RSA keys including a RSA modulo N along with public and private exponents e and d: $pub.key = (N, e)$ and $sec.key = (d)$.
3. Alice randomly generates u_0 and u_1 and sends them to Bob along with the public key.
4. Bob selects b as 0 or 1 regarding the requested message and selects the respective u_b.
5. Bob generates a random value k and obfuscates u_b computing $v = (u_b + k^e) \mod N$ and sends it to Alice.
6. Alice computes the two possible values of k: $k_0 = (v - u_0)^d \mod N$ and $k_1 = (v - u_1)^d \mod N$.
7. Alice processes the two messages with the respective keys: $m'_0 = m_0 + k_0$ and $m'_1 = m_1 + k_1$ and sends them to Bob.
8. Bob selects the one he can read with k, and retrieves $m_b = m'_b - k$.

The 1-out-of-n OT protocol is a generic case where Alice owns n messages. In the k-out-of-n version, Bob would like to retrieve k messages among the n messages from Alice.

3.3.2 Examples of PIR Protocols

We present several PIR protocols to understand their mechanisms and to the objective of comparing them and selecting the more efficient. We consider two parties in the following protocols: a user $\mathcal{U}$ and a server $\mathcal{S}$ which owns a database. Both of the parties are limited to probabilistic and polynomial computations. We use an homomorphic encryption scheme with their respective encryption and decryption functions $\mathcal{E}nc$ and $\mathcal{D}ec$. The homomorphic scheme could be for example the Paillier's presented in Section 3.2.2. In addition to the **Classical Protocol** [OI07] we present three alternative PIR protocols:

Two-Dimensional Protocol [OI07]. This version is similar to the previous one except the database of n elements which is a two-dimensional tabular. Each element from the database is represented by a couple of indexes (i, j)

with $i, j \in \{1, \ldots, \sqrt{n}\}$, where i corresponds to its row and j its column. The requested element from $\mathcal{U}$ is represented by (i^*, j^*).

1. $\mathcal{U}$ generates the request $Q = (q_1, q_2, \ldots, q_{\sqrt{n}})$ as follows: for all indexes $i \in \{1, \ldots, \sqrt{n}\}$,

$$q_i = \begin{cases} \mathcal{E}nc(0 \; if \; i \neq i^* \\ \mathcal{E}nc(1) \; if \; i = i^* \end{cases} \tag{3.8}$$

2. $\mathcal{S}$ computes for each row: $R_j = \sum_{i=1}^{\sqrt{n}} q_i x_{ij}$. Its sends back the following response: $R = (R_1, \ldots, R_{\sqrt{n}})$.
3. $\mathcal{U}$ retrieves the vector and decrypts the required element(i.e. the j^*-th element): $\mathcal{D}ec(R_{j^*}) = x_{i^*j^*}$.

We notice that despite a complexity improvement, $\mathcal{U}$ is able to decrypt the whole i^*-th row from the database, namely $\sqrt{n}$ elements. This protocol is clearly not a symmetric PIR since user could retrieve additional elements than the one requested.

Chang Protocol [Cha04]. This version has been proposed in 2004 by Yan-Cheng Chang, it brings back symmetrical feature on privacy even with a two-dimensional database as in the *Two-Dimensional Protocol*.

1. $\mathcal{U}$ generates a request $Q = (\alpha_1, \ldots, \alpha_{\sqrt{n}}, \beta_1, \ldots, \beta_{\sqrt{n}})$ as follows: for all indexes $i, j \in \{1, \ldots, \sqrt{n}\}$,

$$\alpha_i = \begin{cases} \mathcal{E}nc(0) \; if \; i \neq i^* \\ \mathcal{E}nc(1) \; if \; i = i^* \end{cases} \quad and \quad \beta_j = \begin{cases} \mathcal{E}nc(0) \; si \; j \neq j^* \\ \mathcal{E}nc(1) \; si \; j = j^* \end{cases} \tag{3.9}$$

2. $\mathcal{S}$ computes for each row: $\sigma_i = \sum_{j=1}^{\sqrt{n}} \beta_j x_{i,j} \mod N^2$ with N the RSA module, and decomposes it as: $\sigma_i = u_i N + v_i$.
   $\mathcal{S}$ computes $u = \sum_{j=1}^{\sqrt{n}} \alpha_j u_j \mod N^2$ and $v = \sum_{j=1}^{\sqrt{n}} \alpha_j v_j \mod N^2$ and sends these values to $\mathcal{U}$.
3. $\mathcal{U}$ receives the couple (u, v) and decrypts as:

$$\mathcal{D}ec(\mathcal{D}ec(u)N + \mathcal{D}ec(v)) = x_{i^*j^*} \tag{3.10}$$

RQ Protocol [KO97]. This version is based on the quadratic residuosity problem and could be used if the requested data is a bit. Indeed, the database from $\mathcal{S}$ consists of a matrix M composed by $n = s * t$ bits. $\mathcal{U}$ would like to retrieve the x_{i^*}-th bit, with $i^* = (a, b)$, a its row and b its column.

1. $\mathcal{U}$ randomly selects t values $y_1, \ldots, y_t \in \mathbb{Z}_N$, relatively prime to N and with a Jacobi symbol of 1 (i.e. $\forall i \ (\frac{y_i}{N}) = 1$) such that y_i is a quadratic residuosity for $i \neq b$ and y_b is not. $\mathcal{U}$ sends these values to $\mathcal{S}$.
2. For each element from the matrix M, $\mathcal{S}$ computes:

$$w_{r,j} = \begin{cases} y_j^2 \ if \ M_{r,j} = 0 \\ y_j \ if \ M_{r,j} = 1 \end{cases} \tag{3.11}$$

   Then, for each column, $\mathcal{S}$ computes:

$$z_r = \prod_{j=1}^{t} w_{r,j} \tag{3.12}$$

   The request's response is $R = (z_1, \ldots, z_t)$.
3. $\mathcal{U}$ solely considers the requested element, namely z_a and tests if it is a quadratic residuosity. If it is, then $b_i = 0$, if not $b_i = 1$.

3.3.3 Protocols Comparison

After a detailed description of the four types of PIR protocol, we look at their respective complexity. First, we express the following complexities. The ones from $\mathcal{U}$ to generate the request (**UPL**) and to process its response (**DWL**), and the ones from $\mathcal{S}$ to perform computations on database and to generate the respective response (**COMPUTES**). Second, we determine the communication's complexity in terms of sizes of the messages for the ones sent from $\mathcal{U}$ to $\mathcal{S}$ ($\longrightarrow$) and their responses ($\longleftarrow$). We recall the relevant parameters: n the database's size, k the size of the requested element by $\mathcal{U}$ (in bits). And we set *op* as a basic operation like multiplication or modular exponentiation.

For instance, for Paillier's cryptosystem [Pai99], encryption corresponds to two exponentiations and one multiplication, which makes three *op*. Decryption corresponds to one exponentiation, one subtraction, one division and one multiplication, which makes four *op*. On server's side, *add* corresponds to a multiplication and *mult* to a modular exponentiation. Then, with a classical PIR, request generation and server's computations are in $O(n)$ while the user's decryption is in $O(1)$.

For RQ protocol, we do not express the encryption's complexity which consists in generating n values relatively prime to N quadratic residues. We express the quadratic residuosity testing by *testRQ*. Remaining computations are related to data' size. Table 3.2 expresses computation and communication complexities.

To interpret these results, we propose Table 3.3.

TABLE 3.2. COMPLEXITIES' COMPARISON OF DIFFERENT PIR PROTOCOLS (FROM OUR OWN CONTRIBUTION).

Protocols		$\mathcal{U}$		$\mathcal{S}$	Communication	
		UPL	**DWL**	**COMPUTES**	$\longrightarrow$	$\longleftarrow$
Paillier	*Classic* [OI07]	$n \times \mathcal{E}nc \equiv 3n \times op \mapsto \boldsymbol{O(n)}$	$1\mathcal{D}ec \equiv 4op \mapsto \boldsymbol{O(1)}$	$n \times (add + mult) \equiv 2n \times op \mapsto \boldsymbol{O(n)}$	$n \times \log \lvert G' \rvert$	$log \lvert G' \rvert$
	DB 2-dim [OI07]	$\sqrt{n} \times \mathcal{E}nc \equiv 3\sqrt{n} \times op \mapsto \boldsymbol{O(\sqrt{n})}$	$1\ \mathcal{D}ec \equiv 4op \mapsto \boldsymbol{O(1)}$	$\sqrt{n} \times \sqrt{n} \times (add + mult) \equiv 2n \times op \mapsto \boldsymbol{O(n)}$	$\sqrt{n} \times \log \lvert G' \rvert$	$\sqrt{n} \times \log \lvert G' \rvert$
	Chang [Cha04]	$2\sqrt{n} \times \mathcal{E}nc \equiv 6\sqrt{n}.op \mapsto \boldsymbol{O(\sqrt{n})}$	$3\mathcal{D}ec + add + mult \equiv 14\ op \mapsto \boldsymbol{O(1)}$	$\sqrt{n} \times \sqrt{n} \times (add + mult) + \sqrt{n} \times (div + mod) + 2\sqrt{n} \times (add + mult) \equiv (2n + 6\sqrt{n})op \mapsto \boldsymbol{O(n)}$	$2\sqrt{n} \times \log \lvert G' \rvert$	$2 \times \log \lvert G' \rvert$
RQ [KO97]		$n \times op \mapsto \boldsymbol{O(n)}$	$k \times testRQ \equiv 2k \times op \mapsto \boldsymbol{O(k)}$	$n \times k(3 \times mult) \equiv 3kn \times op \mapsto \boldsymbol{O(kn)}$	$n \times \log \lvert G \rvert$	$k \times \log \lvert G \rvert$

TABLE 3.3. PROS AND CONS OF DIFFERENT PIR PROTOCOLS (FROM OUR OWN CONTRIBUTION).

Protocols		PROS/CONS
Paillier	*Classic* [OI07]	⊕ user only retrieves requested element: Symmetric PIR ⊖ a significant complexity on server's side
	DB 2-dim [OI07]	⊕ better complexity ⊖ user learns every elements on row i* from database
	Chang [Cha04]	⊕ Symmetric PIR ⊖ more computations on both user's and server's side
RQ [KO97]		⊖ bit-wise processing, therefore not realistic if $\boldsymbol{k}$ is large needs to generate a large amount of prime numbers and quadratic residues.

3.3.4 Related Work

Outsourcing of a PIR protocol is explored when protocols allow multiple database readers [HG13, JJK+13]. The aspect of retrieving encrypted data, stored in an outsourced database described in [FD13], is applied to a cloud scenario, but without PIR. Other work employs Reed-Solomon and LDPC codes to explore the possibility of retrieving multiple records of a database [GKL10, FR12]. Another way of achieving this property is k-out-of-n Oblivious Transfer protocol first introduced in 2002 by Mu *et al.* [MZV02] and then widely developed [CT08, Wan15, LH14].

3.4 Set Operations

Multiple types of operations could be performed on sets. In the three aforementioned use cases, we aim to test set relations and some of them could be reduced to compute some operations cardinality. For instance, being able to compute cardinality of the intersection of two sets indicates whether they have elements in common or if one set is included in the other one. For privacy concerns it could be of interest to solely reveal its cardinality instead of the intersection itself. Therefore, we propose later on a solution to solve two kinds of set relations namely the *inclusiveness* and the *disjointness* defined as follows:

Definition 1 (Inclusiveness)**.** *Let $\mathcal{A}$ and $\mathcal{B}$ be finite sets. $\mathcal{A}$ is included in $\mathcal{B}$,*

i.e. $\mathcal{A} \subset \mathcal{B}$*, iff all elements from* $\mathcal{A}$ *are included in* $\mathcal{B}$ *:* $\forall a \in \mathcal{A} : a \in \mathcal{B}$.

Definition 2 (Disjointness)**.** *Let* $\mathcal{A}$ *and* $\mathcal{B}$ *be finite sets.* $\mathcal{A}$ *and* $\mathcal{B}$ *are disjoint iff none of the elements from* $\mathcal{A}$ *are included in* $\mathcal{B}$*. In other words,* $\mathcal{A} \cap \mathcal{B} = \emptyset$ *:* $\forall a \in \mathcal{A} : a \notin \mathcal{B}$.

Computing disjointness test as a two-party secure computation problem has been proposed in several papers [FNP04, KM05, HW06, YWPZ08] based on homomorphic encryption, Pedersen commitments for [FNP04, KM05], on "testable and homomorphic commitments" on polynomial representations for [HW06], and Sylvester matrix and Lagrange interpolation for [YWPZ08]. But none of these are adapted to a third party scenario or provide privacy on the sets' sizes.

3.5 Bloom Filters

3.5.1 Definition

A Bloom filter is a data structure introduced by Burton Howard Bloom in 1970 [Blo70]. It is used to represent a set of elements. With a Bloom filter representing a certain set, one can verify whether an element is a member of this set. Such a data structure consists of a tabular of m bits which is associated to n_{key} public hash functions. At first, all the m bits are initialized to 0. Moreover two functions namely *add()* and *test()* are available. To add an element to the Bloom filter, one has to compute the hashes of this element with each of respective n_{key} hash functions. Then, set the bit to 1 for each position corresponding to a hash value. To test whether one element is included in the Bloom filter with the *test()* function, one has, similarly, to compute the respective hash values of this element and verify if respective bits are set to 1. If at least one of these bits is set to 0, then we know for sure that the tested element is not a member of the set represented by the Bloom filter (i.e. no false negative for *test()* could happen). On the contrary, with some probability, the *test()* function could retrieve a false positive. Indeed, even if all the bits that have been verified are set to 1, the tested element may not be part of the set represented by the Bloom filter. There exist multiple applications for this approach. It is used by web browser or web site to optimize cache or site recommendations. With database query operations it could also, for example, reduce the disk lookups for non-existent rows or columns. Finally we could also mention its relevancy with respect to cryptocurrencies by for instance accelerating the wallet synchronization of Bitcoin or fasten the finding of logs on Ethereum blockchain.

To express the probability of having a false positive when performing function *test()* we introduce the *overlapping bit* notion.

Definition 3 (Overlapping bit)**.** *When adding an element to a Bloom filter, a certain bit has to be set to 1 but this bit is already set to 1.*

The probability of having an overlapping bit is null when the Bloom filter is still blank, and it grows along with the number of inserted elements. We express

this probability as following with $X_{BF_{\mathcal{A}}}$ the amount of bits already set to 1 in $BF_{\mathcal{A}}$ at a specific point in time:

$$\mathcal{P}_{ob} = \frac{X_{BF_{\mathcal{A}}}}{m} \tag{3.13}$$

We could then express the average amount of different bits added to the Bloom filter when adding one new element to it:

$$X_{add} = n_{key} + \sum_{i=1}^{n_{key}} \left((-1)^i \times \frac{\sum_{j=i}^{n_{key}-1} \binom{j}{i}}{m^i}\right) \tag{3.14}$$

And we could generalize it to the average amount of bits added to the Bloom filter when adding N new elements to it:

$$X_{add}(N) = n_{key} \times N + \sum_{i=1}^{n_{key}\times N} \left((-1)^i \times \frac{\sum_{j=i}^{(n_{key}\times N)-1} \binom{j}{i}}{m^i}\right) \tag{3.15}$$

By observing the current state of a Bloom filter representing finite set $\mathcal{A}$ of $n_{\mathcal{A}}$ elements, one can express the exact amount of overlapping bits as the value $Y_{BF_{\mathcal{A}}}$:

$$Y_{BF_{\mathcal{A}}} = (n_{\mathcal{A}} \times n_{key}) - X_{BF_{\mathcal{A}}} \tag{3.16}$$

3.5.2 Related Work

Adding privacy to Bloom filters has been yet investigated in different works. The first way of doing that consists of directly encrypting the Bloom filter with homomorphic encryption, as Kerschbaum did in [Ker12] and [Ker11] and also as developed in [BC04] and [ZPH+17]. In [Ker12] we notice that Kerschbaum proposes an "outsourced" version of his protocol but requires homomorphic multiplications between Bloom filters of encrypted elements. Also, the protocol disables the server to learn about the intersection size. In [NK09] authors add privacy by using two approaches: blind signature schemes and oblivious pseudorandom functions. The second type of approaches, as we suggest in the current work, consists of replacing the traditional hash functions of the Bloom filter by HMAC functions. We explain how these existing solutions [Goh03, SBR09, KNV09, SGM02, LG, QLW07] do not fit our requirements and therefore our solution brings novelty. First of all, we highlight the fact that none of the following solutions provide privacy on the sets cardinality. In [Goh03] Goh associates Bloom filters with a keyed pseudo-random function to allow a private member testing in the Bloom filter. This is in particular one aspect we do not want the auditor be able to do. In [KNV09] authors propose a solution that approximately computes the dot product protocol using Bloom filters. Contrary to our outsourced requirement, this solution is a two-parties interactive protocol. Also, authors use keyed versions of SHA-1 as the Bloom filter hash functions but do neither motivate this choice nor explain its benefit. They implemented their solution and show results with relatively small

parameters (less than 5 keys and no Bloom filters larger than 5000 bits). They conclude that their solution roughly supports the use of more than four hash functions. In [LG] authors expose a construction of Bloom filters along with HMAC protocol in a wireless sensor aggregation scenario. Their approach is somehow similar but the base station (equivalent to the auditor here) shares HMAC keys directly with each of the nodes. Therefore, the merging of Bloom filters from different nodes does not allow any operation since different keys are used. Also, the base station performs testing on the Bloom filter and holds all the secret keys. In [QLW07], still by combining the Bloom filter approach with HMAC (or keyed hash functions), authors propose a solution to compute membership of elements in a set. Therefore, they manipulate Bloom filters of unique elements that leads to data leakage regarding the amount of elements. Also having one Bloom filter per element could be very costly, especially when considering thousands of them. In [SGM02], authors combine some version of filters with HMAC in the objective to compute the sets intersections. But in such protocol, parties (here the data sources), which perform the intersection, need the HMAC key to return the result. Such a construction does not fit our requirements where in particular the active party (the auditor) should not hold the key for privacy reasons. Indeed, authors consider that all parties hold the key and thus are all trustable. In [SBR09] authors compute Dice-coefficient, which represent similarities between two strings in a private manner using Bloom filters representation. They present some experimental results produced from relatively small Bloom filters parameters that shows limitations on having a large amount of bits in Bloom filters to keep set sizes protected. They motivate the use of HMAC by arguing that it adds an additional security layer to their protocol. We notice that works that provide results by testing their solution are space limited. On the contrary, in the work at hand, we expose results with real-world parameters as 10^4 elements inserted in the Bloom filters of 10^{10} bits. Other works [AM14, EFG+15, Bf12, LYC+06], that deserve attention, compute set relations and operations using the Bloom filters approach but in a multi-party model, that does not fit our requirements. Solving the set intersection cardinality implies solving the disjointness problem as in [EFG+15]. But this work relies on knowing the sizes of the sets and does not fit our outsourced configuration.

In Table 3.4 we recap what aforementioned approaches provide, compared to our proposed solution. For each approach we precise if it can be used in an outsourced model where an untrustable third party performs the computations, if functions *add()*, *test()* and the set relations *inclusiveness* (INC) and *disjointness* (DIS) are computable by the third party.

In [MS17] authors applied the Bloom filter to key exchange mechanisms in wireless sensor network (WSN) environment while in [TNS+17], authors use Bloom filters in WSN environment to minimize overhead and duplication of packets and also to reduce energy usage. They provide control over broadcast overhead. Instead of having each node broadcasting to all the nodes like anarchy, they provide a protocol based on a neighboring pattern where nodes generate and broadcast Bloom filters to avoid unwanted duplication of packets. Here

TABLE 3.4. COMPARISON OF THE APPROACHES (FROM OUR OWN CONTRIBUTION).

Approach	Outs.	*add()*	*test()*	INC/DIS	Card. privacy
Encrypted BF [Ker11] [Ker12, BC04, ZPH+17]	yes	yes	yes	yes (obfuscated results)	no
Blind Signature [NK09]	no	-	-	no	no
BF with HMAC [Goh03]	yes	no	yes	no	no
-[KNV09]	no	-	-	no	no
-[LG]	yes	yes	yes	no	no
-[QLW07]	yes	no	yes	yes	no
- [SGM02]	yes	yes	yes	no	no
- [SBR09]	yes	no	no	no	no
PSI [AM14, EFG+15] [Bf12, LYC+06]	no	-	-	no	yes
Our approach	yes	no	no	yes	yes

Bloom filters are used to optimize communication process and not in a privacy preserving manner. The only retrieved information are nodes' ID of "urgent case nodes". They also investigated the neighbor separation that should be considered to reach all the nodes and minimize duplication.

3.6 Concealed Data Aggregation

3.6.1 Definition

Concealed data aggregation (CDA) is a type of protocols firstly introduced in 2005 by Westhoff *et al.*[GWS05]. It consists of an end-to-end privacy preservation and "en route" aggregation of nodes readouts in wireless sensor network (WSN). Other security aspects could be consider as data integrity preservation or replay protection, while processing encrypted data at intermediate nodes. Analyzing the WSN environment and its respective threat model will directly set the security requirements our CDA protocol will have to fulfill. That been said, the main motivation to develop CDA, instead of having a hop-by-hop aggregation construction or even no aggregation at all is the resource-constrained devices. Indeed, the sensor nodes (leaves) and the intermediate nodes are limited in terms of energy, memory, bandwidth or lifetime. Applying data aggregation helps reducing these costs. On the contrary data aggregation increases security vulnerabilities.

3.6.2 Privacy Constructions and Related Work

To guarantee protection against malicious insider adversaries in case of a compromised intermediate node, we consider protocols providing end-to-end security. Contrary to hop-by-hop aggregation construction, end-to-end encryption consists of data encrypted at the sensor nodes. Then these data are aggregated and finally decrypted at the base station and exclusively there. We give

some overview of different types of existing CDA along with their cryptographic features using literature review [PJ16].

Encrypted Data Classification

In [WMLD04], authors propose a model for categorizing encrypted messages in wireless sensor networks. A classifier, namely an intermediate sensor node, is embedded with a set of searching keywords in encrypted format, and matches an encrypted message with a keyword. The classifier then takes a decision between forwarding the message without aggregation, forwarding the message after aggregation or dropping the duplicate message. We highlight that here the classifier only access encrypted sensor readouts and encrypted keywords.

Symmetric Homomorphic Encryption

Using symmetric-key based cryptosystems provides two main advantages which are translated into bandwidth and energy saving. They have a small message expansion and their computation cost is significantly less than asymmetric-key based operations. On the contrary, since same key is used for encryption across all nodes, it increases security vulnerabilities. In such a scenario, all sensor nodes along with the base station own the secret key. We list the different symmetric cryptosystems used to build CDA in literature and we present their respective characteristics:

- *Domingo-Ferrer's cryptosystem* [Dom02]:
  - **Characteristics**: Supports addition, subtraction, multiplication and division over encrypted data. The data should be encrypted using the same secret key. Secure against known-plaintext attacks. Is also used in [GWS05, WGA06, RKP07].
  - **Pros**: Supports homomorphic multiplication of ciphertexts.
  - **Cons**: Has a large message expansion and a considerably high power consumption. Is unsecured for some parameters settings and all data should be encrypted with the same key.
- *CMT cryptosystem* [CCMT09]:
  - **Characteristics**: Is an additive homomorphic cryptosystem which uses a pseudo-random function to generate the cryptographic keys. Is secured in a computational-complexity theoretic setting. Is also used in [PKP11, RM13, PPL07].
  - **Pros**: Is a viable alternative for real-world applications thanks to the use of pseudo-random function.
  - **Cons**: Requires more bandwidth and needs sensor nodes' identity-related information to perform decryption.
- *Armknecht et al. enhancement* [AWGH08]:

  - **Characteristics**: Uses a bi-homomorphic encryption function to encrypt data as well as cryptographic keys.
  - **Pros**: Do not need sensor nodes' identity-related information anymore.

- *Onen and Molva* [ÖM07]:
  - **Characteristics**: Uses a layer-wise security mechanism and a key-attribution algorithm.
  - **Pros**: Reduces impact of threats as node compromise attacks. Uses multiple pair-wise keys between nodes to suppress encryption layers.
  - **Cons**: Needs a pre-distribution mechanism, a re-keying requirement and a fixed network topology.

- *Di Pietro et al.* [PMM09]:
  - **Characteristics**: Adapts a concept of delayed aggregation and peer monitoring for integrity preservation.
  - **Pros**: Mitigates effects of node compromise attacks and nodes failures.
  - **Cons**: Allows a compromised node to reveal not only its secret key but also to recover the secret keys of neighboring nodes.

Asymmetric Homomorphic Encryption

Mykletun et al. evaluated viability of asymmetric-key based homomorphic cryptosystems in WSN environment [MGW06, PWC10]. Authors highlight characteristics that asymmetric cryptosystems should have as, the *ciphertext indistinguishability* or the *message expansion minimization*.

We list the different asymmetric cryptosystems used to build CDA in literature and we give their respective characteristics:

- *Goldwasser-Micali's cryptosystem* [GM84]:
  - **Characteristics**: Is provably secured against Chosen-Plaintext Attacks (IND-CPA). Supports privacy homomorphism for X-OR operations. Is also used in [SJM13, SYY99, AGW05, AKSX04].
  - **Pros**: Could be used to compute MIN and MAX functions at aggregator nodes [SJM13]. Supports multiplicative homomorphic operations over encrypted data [SYY99]. And performs the secure comparaisons of encrypted data [AGW05] or both using order-preserving encryption [AKSX04].
  - **Cons**: Encrypts data bit-by-bit and therefore has a substantial message expansion.

- *Okamoto-Uchiyama's cryptosystem* [OU98]:

  - **Characteristics**: Is an additive homomorphic cryptosystem semantically secured under the p-subgroup assumption. Its security depends on intractability of factoring $n = p^2q$. Is also used in [MGW06].
  - **Pros**: Allows less expensive cost for decryption than EC-ElGamal.
  - **Cons**: Requires larger ciphertext size than EC-ElGamal. Cryptosystem's parameters should be chosen such that n is sufficiently large.

- *Elliptic Curve Paillier* [Pai00]:
  - **Characteristics**: Is an additive homomorphic, probabilistic and semantically secured. Is based on trapdooring discrete logarithm on elliptic curve over a ring $E_{n=p^2q^2}$.
  - **Cons**: Has a significant message expansion and vulnerability against adaptive Chosen-Ciphertext Attacks (CCA2). Allows the secret key to be recovered from publicly available data as shown in [Gal02].

- *Elliptic Curve ElGamal* [Kob87]:
  - **Characteristics**: Is defined on an elliptic curve over a finite field $\mathbb{F}$, and supports additive homomorphism. Is based on an intractability of solving the Elliptic Curve Discrete Logarithm Problem (ECDLP). Is also used in [GWMA07, AM09, UHW, LHSC10].
  - **Pros**: Requires only 160-bit key-size which improves bandwidth efficiency, energy utilization, and storage capabilities in WSNs.
  - **Cons**: Has at least a 4-to-1 message expansion ratio, and its reverse mapping function requires to solve the ECDLP.

- *Elliptic Curve Boneh* [BGN05]:
  - **Characteristics**: Is an additive homomorphic cryptosystem over an elliptic curve group $\mathbb{G}$, which supports arbitrary addition operations and a single multiplication operation over encrypted data. Is based on the intractability of the subgroup decision problem. Is also used in [BGM10b, BGM10a, OX11, ZYH14].
  - **Cons**: Requires the use of bilinear pairing which increases computation cost compared to EC ElGamal.

Chapter 4

Developed Solutions

In Chapter 2 we presented four real-life use cases before describing all necessary preliminaries in Chapter 3. In the current chapter, we propose solutions to solve these use cases as synthesized in Figure 4.1. Solution A is a first proposition to address the first use case; *Cloud Security Auditing*. In Solution B we propose an accurate analysis along with an algorithm to optimize the using of Solution A. Finally, Solution C represents a whole new approach which in addition to solve the first use case, also address the other three use cases.

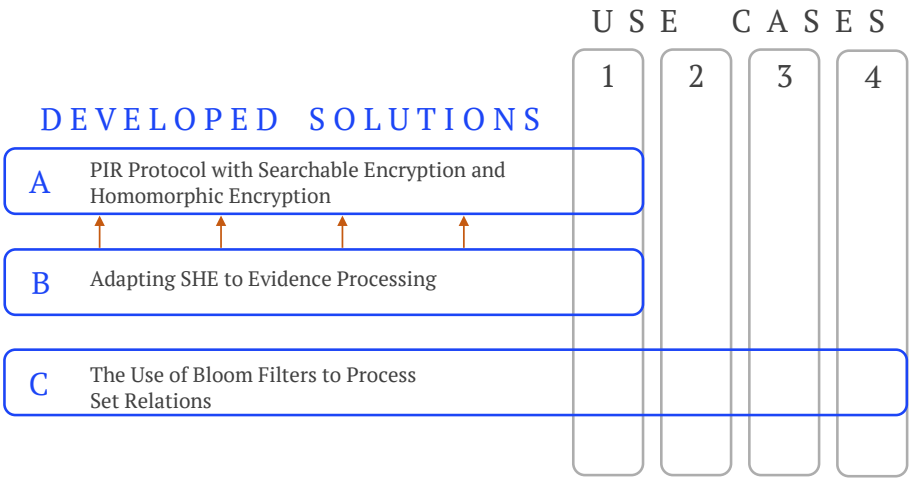

Figure 4.1. How the developed solutions match the proposed use cases.

4.1 Solution A - Combining PIR Protocol with Searchable Encryption and Homomorphic Encryption

We propose here a construction of different building blocks. In order to achieve requirement **PR1**, a Private Information Retrieval (PIR) protocol will be used with respect to envisioned properties *secure data storage* and *query flexibility* which we will explain next. Due to PIR protocol's features, $\mathcal{AD}$ will be able to hide on behalf of which client the audit is performed. This solution is based on the initial work [LRW14] from Lopez *et al.* but both properties are new to the realm of PIR.

Secure data storage, our first enhancement, means that information from evidence store's database that $\mathcal{AD}$ requests, will no longer be stored in plaintext but in a certain encrypted manner. In that way, the evidence store's administrator, in addition to not be able to see the requested element, will not have access to the plaintext content of the stored evidence itself. This additional feature is interesting for requirement **PR2**. Indeed, sensitive data will be hidden to $\mathcal{ES}$ as well as all information retrieved by $\mathcal{AD}$ from $\mathcal{CSP}$.

Query flexibility describes our second enhancement. Our PIR protocol is able to deal with multiple elements. More precisely, with the classical version of PIR [OI07], $\mathcal{AD}$ would retrieve exactly one element of information per query. Using the scheme presented in this work, in addition to retrieve a unique element, one query also allows $\mathcal{AD}$ to receive the output of a function. In its simplest form this can be either the sum or the product of different elements from the evidence store $\mathcal{ES}$. However, also more enhanced functions using both operations, a moderate number of times at the same time are possible.

4.1.1 Cloud Auditing Construction

In this section we provide a description of the auditing framework and their parties developed to solve *use case 1*. The reader could refer to Section 2.1 where the use case is described in detail. We recall Figure 4.2 to a better understanding of the scenario.

Auditing Framework

Here we present parties involved in the cloud audit framework. We assume that all parties communicate using an authenticated and secure channel, as for example TLS.

Cloud Service Provider ($\mathcal{CSP}$). $\mathcal{CSP}$ represents the cloud service provider who offers different services to clients, for example Software as a Service (SaaS).

Audit Controller ($\mathcal{AC}$). As a part of $\mathcal{CSP}$, $\mathcal{AC}$ collects and aggregates useful digital evidence from the $\mathcal{CSP}$'s services. Besides collecting, $\mathcal{AC}$ also

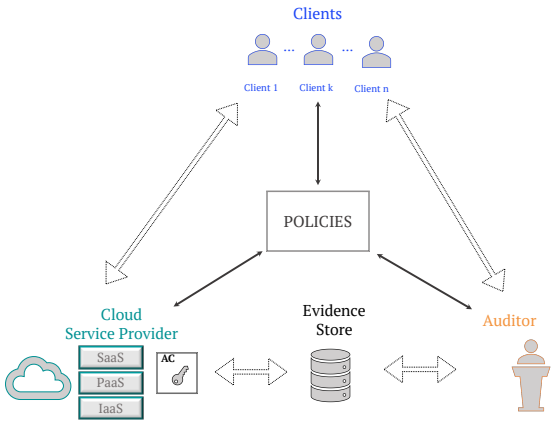

Figure 4.2. The security cloud auditing framework with the different parties. They agree on policies as SLA (single line arrow) before performing the audit's protocol. The communication flow is represented by double line arrows.

encrypts the data and sends them to $\mathcal{ES}$.

Evidence Store ($\mathcal{ES}$). $\mathcal{ES}$ gathers data from (multiple) $\mathcal{AC}$ and stores data in a database. It computes responses to the Auditor.

Auditor ($\mathcal{AD}$). On behalf of clients, $\mathcal{AD}$ performs verifications on $\mathcal{CSP}$'s services by performing checking on data stored in $\mathcal{ES}$.

Client ($\mathcal{C}$). Client corresponds to a set of entities, for example companies, who further consist of a set of users each. Client engages the $\mathcal{CSP}$'s services and wants to have a provable evidence that the $\mathcal{CSP}$ is behaving correctly. He instructs the auditor to check the $\mathcal{CSP}$ regularly.

All parties agree on certain policies before performing the PIR protocol, as for example service level agreements.

Multi-Client Auditing Scenario

We generalize the use case scenario originally presented in [LRW14] with multiple clients. We initially suppose some companies outsourcing some of their internal IT services to $\mathcal{CSP}$. That service provider is specialized in providing Software as a Service (SaaS). A companies' security policy states that while any employee may use the service provided by $\mathcal{CSP}$, this may only be done when connecting from their own company's network. This ensures additional security systems, e.g. transparent security proxies deployed by companies, are not circumvented. Technically speaking this means:

- Each company specifies a list of IP addresses which are classified as *authorized*, or *unauthorized*.

- Employees of each company can access the service provided by $\mathcal{CSP}$ only if they connect using an authorized IP address.

An audit of the $\mathcal{CSP}$'s access control logs consists of verifying that such policies are being adhered to, and that no access from an unauthorized location has occurred.

Combination of Two Homomorphic Encryption Schemes

In our solution, we propose a PIR protocol on encrypted data. Thus, two encryption schemes `Scheme1` and `Scheme2` have to be combined and their compatibility has to be guaranteed.

`Scheme1`: Used to encrypt data stored in $\mathcal{ES}$. Should be at least additively homomorphic, such that the computation of a sum of entries over a row or column is possible. However, multiplicative operations are also preferable.

`Scheme2`: Used to encrypt the query and perform the PIR protocol. Due to PIR protocol's requirements, this scheme should be at least additively homomorphic.

At next we evaluate possible combinations of encryption candidates with respect to the requirements derived from use case 1 and presented in Section 2.1.

We first consider Paillier's scheme for both schemes. For a single retrieval, this candidate will fit perfectly. However, to retrieve the sum of elements,PIR protocol should result in the ciphertexts' product. Product of elements could not be retrieved if `Scheme2` is Paillier's one due to the fact that it is only additively homomorphic.

Then, we consider `Scheme1` as the SHE scheme from [BV11a] and `Scheme2` as Paillier's scheme. With SHE scheme, elements from the database have to be encoded into a polynomial representation and subsequently encrypted as a tuple containing two polynomial elements denoted as $(c_{i,0}, c_{i,1})$. As presented in Section 3.3, for a PIR query $Q = (q_1, q_2, \ldots, q_n)$ and η the encoding polynomial degree, response will be:

$$R = \left(\sum_{i=1}^{n} (c_{i,0} q_i), \sum_{i=1}^{n} (c_{i,1} q_i) \right)$$

where

$$c_{i,j} q_i = \sum_{k=0}^{\eta} c_{i,0,j} q_i x^k, \ j \in \{0, 1\}$$

Paillier's decryption yields the sum of encrypted data and a modulo reduction is required before decryption phase. Since `Scheme1` is asymmetric, and the party which encrypts the data does not own the decryption key, it is not capable of decrypting them. With `Scheme2`, Paillier's scheme is also asymmetric and only the public key is shared. The polynomial representation significantly

increases computation effort. Parts of PIR computations will have to be performed on each coefficient of the cipher which may increase computation cost. This also holds for communication performance and size of the messages.

Finally we consider `Scheme1` and `Scheme2` as SHE scheme. The PIR is now using the same scheme as the one used to encrypt database entries. This means that the party which encrypts data could also generate PIR requests and vice versa. Therefore we will, either use the same keys for different PIR requests, or we will have to make a new encryption over the complete database per new PIR request. Obviously, for a practical setting both approaches lack feasibility.

To conclude the investigation on our first developed solution, since the PIR protocol retrieves a sum and a product of encrypted data the most suitable solution is SHE for `Scheme1` and Paillier's scheme for `Scheme2`.

The PIR Protocol

One objective of our protocol is to hide which client is requesting the audit, meaning which data of the evidence store is affected by the computation. With an additive homomorphic scheme such as Paillier's, this kind of privacy requirement could be fulfilled. We recall that in classical PIR protocol [OI07] client sends a different query for each information he is interested in and subsequently would be able to compute sum or product at the client-side. Besides poor bandwidth performance, in that way the number of queries obviously reveals how many entries the querier is interested in. In addition to the Paillier's scheme, we will need a searchable encryption scheme to perform some test directly on encrypted data of the evidence store in a confidentiality preserving manner. With the use of these two schemes, we will be able to retrieve encrypted sum and product of encrypted data in a way that the evidence store will not learn which data is affected.

4.1.2 Protocol

In this section, we first show how digital evidence is collected and stored by the evidence store. Then we introduce the PIR protocol instantiation and apply it to use case 1 - Cloud Security Auditing.

Digital Evidence

First of all, $\mathcal{CSP}$ generates data from its access control. This sensitive information are then provided to the evidence store $\mathcal{ES}$, which makes them available to the auditor in order to validate $\mathcal{CSP}$' contractual duties. Some of these data have to be encrypted according to confidentiality agreements between $\mathcal{C}$ and $\mathcal{CSP}$, since this may be part of security policies. $\mathcal{C}$ authorizes $\mathcal{CSP}$ to log access control as evidence of compliance of the policy presented above. They apply restrictions that consist of having sensitive information protected against data leakage, insiders, outsiders as well as third-party auditors and the evidence store. For instance, device and geolocation of each companies' users (e.g. IP address)

are considered as personal sensitive data. Since $\mathcal{AD}$ should be able to perform different kinds of audits, $\mathcal{CSP}$ needs to provide several information regarding its provided services to $\mathcal{ES}$. The following list of data fields are collected and sent to $\mathcal{ES}$.

Index. Unique identifier for a data row.

Timestamp. Date and time when the evidence was collected.

AC. Identifier of the audit controller which collects the evidence.

Client. Identifier of the user's company.

User. Identifier of the user who accesses the controlled resource.

IP$\star$. IP address from which the connection was attempted.

Availability$\star$. $\mathcal{CSP}$'s fulfilled service availability Av $\in \{0,1\}$ during service usage (availability as defined in the SLA).

Connection$\star$. Describes if a connection was successfully established by Out $\in \{0,1\}$.

Authentication. The authentication denotes if the connecting IP address was authorized or not by Auth $\in \{0,1\}$.

A star $\star$ marks these entries as encrypted. As we will explain in the upcoming audit goals description, $\mathcal{AD}$ will have to test if each IP address has been previously authorized by $\mathcal{C}$. This is why we are using encryption paradigm Public Key Encryption with keyword Search SE such that $\mathcal{AD}$ and $\mathcal{ES}$ do not learn IP addresses. Each IP address collected will be encrypted and aggregated by $\mathcal{AC}$ as $\mathtt{SE}.\,\mathrm{PEKS}(pk_{\mathtt{SE}}, IP)$.

Regarding the *availability* and *connection* data, we are using the SHE scheme, to allow evidence store to perform computations without learning about these sensitive data. In the SHE scheme presented in Section 3.2.3, the encryption function generates a polynomial representation of the ciphertext. We get $((\alpha_0,\ldots,\alpha_\eta),(\beta_0,\ldots,\beta_\eta)) \leftarrow \mathtt{SHE}.\,\mathrm{Enc}(syk_{\mathtt{SHE}}, m)$ where α_i and β_i are coefficients and η is the degree of the polynomial representing message m's encryption.

In Table 4.1 we give a database overview of the service logfile.

Table 4.1. Evidence store database overview of an authentication-connection log.

Index	Timestamp	AC	Client	User	IP	Av	Out	Auth
1	09.02.16	AC58	C2	U4	$\mathtt{SE}.\,\mathrm{PEKS}(IP_1)$	$\mathtt{SHE}.\,\mathrm{Enc}(1)$	$\mathtt{SHE}.\,\mathrm{Enc}(1)$	1
2	09.02.16	AC34	C8	U2	$\mathtt{SE}.\,\mathrm{PEKS}(IP_2)$	$\mathtt{SHE}.\,\mathrm{Enc}(1)$	$\mathtt{SHE}.\,\mathrm{Enc}(0)$	1
...			...		...		...	
n	10.02.16	AC15	C1	U4	$\mathtt{SE}.\,\mathrm{PEKS}(IP_n)$	$\mathtt{SHE}.\,\mathrm{Enc}(0)$	$\mathtt{SHE}.\,\mathrm{Enc}(0)$	0

Regarding the keys' distribution, $\mathcal{C}$ knows solely $sk_{\texttt{SHE}}, sk_{\texttt{SE}}$ while $\mathcal{AD}$ knows all of the secret keys, i.e. $sk_{\texttt{SHE}}, sk_{\texttt{SE}}, sk_{\texttt{PA}}$. $\mathcal{CSP}$ and $\mathcal{ES}$ do not know any of the mentioned keys.

Initialization of the PIR Protocol

Before performing the audit and having the store checking, if IP addresses from its database are authorized, an initialization step is required.

At first, all clients send their lists of authorized IP addresses to both $\mathcal{AD}$ and $\mathcal{CSP}$. Then, $\mathcal{AD}$ generates trapdoors using the searchable encryption algorithm described in Section 3.1, and sends all of them to $\mathcal{ES}$. Each time a new piece of data is received from $\mathcal{AC}$, $\mathcal{ES}$ will test IP addresses using trapdoors and will fill in the database rows *Auth* with 1, if the IP address is authorized or 0 otherwise.

After testing authorization of all different IP addresses, we get eight different combinations of values $\{0,1\}^3$ which are interpreted as presented in Table 4.2.

Table 4.2. Interpretation of different results regarding service availability and access control correctness. When Authentication is not successful (0), we only consider access control.

Availability	**Connection**	**Auth**	**Interpretation**	
			service	*access control*
0	0	0	–	correct
0	0	1	not available	not correct
0	1	0	–	not correct
0	1	1	available	correct
1	0	0	–	correct
1	0	1	available	not correct
1	1	0	–	not correct
1	1	1	available	correct

Access Control Checking & Availability Validation of the Service

We state that here, the main goal of the auditor is to verify that access control has been correctly performed by $\mathcal{CSP}$. More concretely, we define an acceptable case when all successful connections were only made from authorized IP addresses and no connection attempt made from authorized IP addresses failed. In Table 4.2 we observe that four of the cases express that the access control succeeded and the remaining four that it failed. As aforementioned, another goal for the auditor in this scenario could be to check to what extend the service itself has been available. The availability field from the database expresses if the service was fully available during whole connection duration. In an audit scenario, it is relevant to consider the service availability only in the case

of connections attempted from authorized users. To enable $\mathcal{AD}$ to respond to such type of verification we assume a relaxed but still meaningful definition of service availability.

Definition 4 (Availability)**.** *We suppose a service to be X % available iff X % of* authorized *users connection attempts were successful.*

This helps $\mathcal{AD}$ to verify at the end of an auditing period, the percentage of offered service availability based on provided digital evidence. It just has to perform some simple arithmetic operations on it to judge if SLA has indeed been fulfilled. A concrete query $\mathcal{C}$ requests to $\mathcal{AD}$ could be: *Tell me if the access control was done correctly and if the service was available. If not, give me the percentage of errors.* In [LRW14], queries as "Separately, the total of successful and non-successful connections from unauthorized IP addresses" or "Probability of success when trying to connect from valid and invalid IP address" were solved in a privacy preserving manner. We also encrypt the data fields *Availability* and *Connection* to add confidentiality on the evidence store. Again, we emphasize that with the PIR protocol, $\mathcal{ES}$ will not be able to learn which company's data $\mathcal{AD}$ is asking for.

For the protocol, $\mathcal{AD}$ has to hide client's identity. To do so, it has to generate a PIR request. First, $\mathcal{ES}$ sends the client ($\{C_j\}_{j=1}^{k}$) and index ($\{I_j\}_{j=1}^{k}$) data fields to $\mathcal{AD}$ so it could know the log database's architecture. It also has to generate a pair of keys $(pk_{\mathtt{PA}}, sk_{\mathtt{PA}})$ according $\mathtt{PA}$. KeyGen function. Algorithm 1 corresponds to the PIR request generation performed by $\mathcal{AD}$.

Algorithm 1 Generating PIR request $Q = \{q_j\}_{j=1}^{k}$ for client $C_{i'}$ and k data.

procedure *Request*$(\{C_j\}_{j=1}^{k}, \{I_j\}_{j=1}^{k}, pk_{\mathcal{P}a})$

  for $i \leftarrow 1, k$ **do**
    if $C_{I_i} = C_{i'}$ **then**
      $q_i \leftarrow \mathtt{PA}.\mathrm{Enc}(pk_{\mathtt{PA}}, 1)$
    else
      $q_i \leftarrow \mathtt{PA}.\mathrm{Enc}(pk_{\mathtt{PA}}, 0)$
    end if
  end for
  return $(Q = \{q_j\}_{j=1}^{k}, I_1, I_k, pk_{\mathtt{PA}})$
end procedure

We consider six values provided by our protocol and which allow $\mathcal{AD}$ to respond to the following query: "Tell me if the access control was done correctly and if the service was available. If not, provide the percentage of errors."

- V_1: total amount of connection attempts
- V_2: amount of connections made from authorized IP addresses

- V_3: amount of connections made from authorized IP addresses with a successful connection
- V_4: amount of connections made from authorized IP addresses with a $\mathcal{CSP}$'s service being available
- V_5: quantity of connections made from authorized IP addresses with a successful connection while $\mathcal{CSP}$'s service is available
- V_6: the amount of connections made from unauthorized IP addresses with a successful connection

To compute these values, $\mathcal{ES}$ accesses *availability*, *connection* and *authentication* columns from the database. Algorithm 2 describes these computations.

Algorithm 2 Computing V_1, V_2, V_3 ,V_4, V_5, and V_6 for k data.

procedure *Computation*($\{q_j\}_{j=1}^{k}$, $pk_{\mathcal{P}a}$, I_1, I_k, $\{Av_j\}_{j=1}^{k}$, $\{Out_j\}_{j=1}^{k}$, $\{Auth_j\}_{j=1}^{k}$)

  $V_1 \leftarrow \texttt{PA}.\text{Enc}(pk_{\texttt{PA}}, 0)$
  $V_2 \leftarrow \texttt{PA}.\text{Enc}(pk_{\texttt{PA}}, 0)$
  $V_3 \leftarrow \texttt{SHE}.\text{Enc}(pk_{\texttt{SHE}}, 0)$
  $V_4 \leftarrow \texttt{SHE}.\text{Enc}(pk_{\texttt{SHE}}, 0)$
  $V_5 \leftarrow \texttt{SHE}.\text{Enc}(pk_{\texttt{SHE}}, 0)$
  $V_6 \leftarrow \texttt{SHE}.\text{Enc}(pk_{\texttt{SHE}}, 0)$

  for $i \leftarrow I_1, I_k$ **do**
    $V_1 \leftarrow \texttt{PA}.\text{Add}(pk_{\texttt{PA}}, V_1, q_i)$
    if $Auth_i = 1$ **then**
      $V_2 \leftarrow \texttt{PA}.\text{Add}(pk_{\texttt{PA}}, V_2, q_i)$
      $Prod_i \leftarrow \texttt{SHE}.\text{Mult}(Av_i, Out_i)$
      for $j \leftarrow 1, (2\eta + 2)$ **do**
        $V_{3j} \leftarrow \texttt{PA}.\text{Add}(pk_{\texttt{PA}}, V_{3j}, \texttt{PA}.\text{Mult}(pk_{\texttt{PA}}, q_i, Out_i))$
        $V_{4j} \leftarrow \texttt{PA}.\text{Add}(pk_{\texttt{PA}}, V_{4j}, \texttt{PA}.\text{Mult}(pk_{\texttt{PA}}, q_i, Av_i))$
        $V_{5j} \leftarrow \texttt{PA}.\text{Add}(pk_{\texttt{PA}}, V_{5j}, \texttt{PA}.\text{Mult}(pk_{\texttt{PA}}, q_i, Prod_i))$
      end for
    else
      for $j \leftarrow 1, (2\eta + 2)$ **do**
        $V_{6j} \leftarrow \texttt{PA}.\text{Add}(pk_{\texttt{PA}}, V_{6j}, \texttt{PA}.\text{Mult}(pk_{\texttt{PA}}, q_i, Out_i))$
      end for
    end if
  end for
  return (V_1, V_2, V_3, V_4, V_5, V_6)
end procedure

With these six values, $\mathcal{AD}$ decrypts V_1, V_2 and all coefficients of V_3, V_4, V_5, V_6 with Paillier decryption function $\texttt{PA}.\text{Dec}$ which yields values V_1^*, V_2^*, V_3^*, V_4^*,

V_5^*, and V_6^*. Then, it processes the homomorphic decryption function `SHE.Dec` to V_3^*, V_4^*, V_5^*, and V_6^* yielding plaintext values V_3', V_4', V_5', and V_6' with $V_1' = V_1^*$ and $V_2' = V_2^*$. $\mathcal{AD}$ then concludes audit by performing Algorithm 3.

Algorithm 3 Audit resulting.

```
1: procedure Result(V_1', V_2', V_3', V_4', V_5', V_6')
2:
3:     CE ← V_2' − V_3' + V_6'
4:     AE ← V_2' − V_5'
5:     TE ← V_2' − V_5' + V_6'
6:
7:     if TE = 0 then
8:         return The access control was perfectly performed and the service
   fully available.
9:     else if AE = 0 then
10:        return The service was fully available but the access control failed in
   (CE/V_1')% of the situations.
11:    else if CE = 0 then
12:        return The service was fully available but the access control failed in
   (AE/V_1')% of the situations.
13:    else
14:        return The access control failed or the service was unavailable in
   (TE/V_1')% of the situations.
15:    end if
16: end procedure
```

In Figure 4.3 we have an overall recap of this use case's audit protocol.

4.1.3 Evaluation and Results

Security Analysis

In our framework, we combine SHE scheme from Brakerski and Vaikuntanathan with Paillier's scheme. This implies to verify their compatibility. Indeed, the arithmetic form of the values which are computed and transmitted during audit corresponds to Paillier's ciphertext. Concretely, it implies that all these values are computed modulo a parameter N^2. Since our enhancement of PIR protocol processes on encrypted data, a restriction consists of having the encrypted data stored in the database not larger than value N^2. As the selected cryptosystem for these pieces of data is the SHE scheme, they will be encoded into polynomial. We, then argue that each polynomial's coefficient should be strictly smaller than N^2. According to this limitation, we could analyze the security level of these two cryptosystems. Paillier's cryptosystem is a probabilistic asymmetric algorithm based on intractability hypothesis of the decisional composite residuosity assumption given at next:

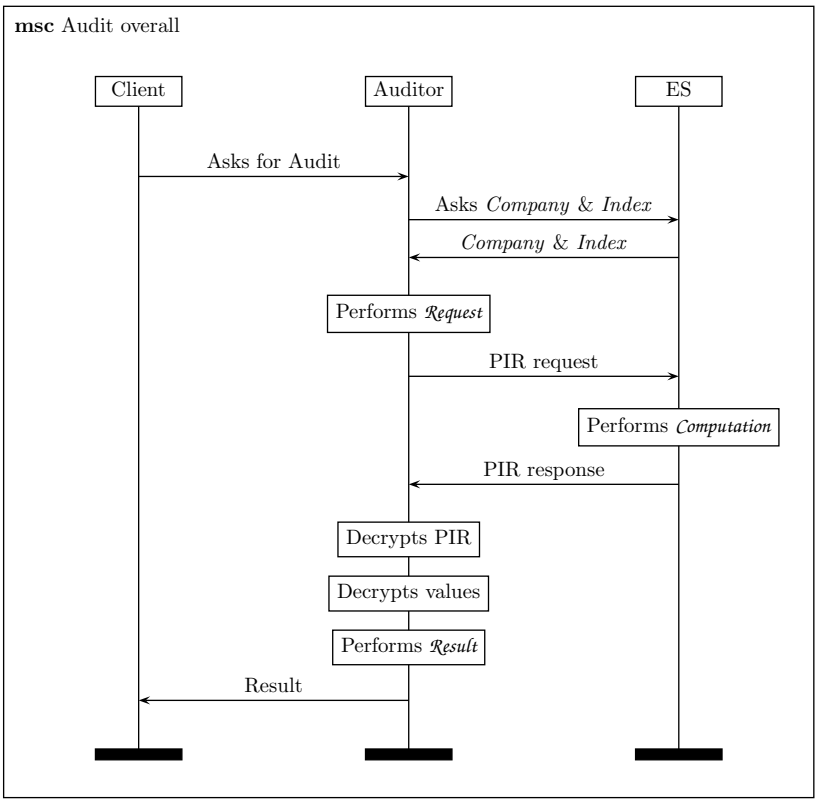

Figure 4.3. The overall protocol for retrieving the extend of correctness of the access control performed by $\mathcal{CSP}$.

Decisional Composite Residuosity Assumption (DCRA). It is a mathematical assumption which states that, given a composite n and an integer z, it is hard to decide whether z is a n-residue modulo n^2 or not.

Security of the SHE cryptosystem [BV14] relies on the RLWE problem which is defined in Section 3.2.3.

PEKS scheme of Boneh is used independently. It is provable secure against chosen keyword attacks and its security relies on difficulty to solve the Bilinear Diffie-Hellman problem (BDH). We precise the following definition:

Bilinear Map. Let G_1 and G_2 be two groups of prime order p. A bilinear map from $G_1 \times G_2$ to G_2 is a function e: $G_1 \times G_2 \to G_2$ such that for all $u,\ v \in G_1,\ a, b \in \mathbb{Z}, e(u^a, v^b) = (u, v)^{ab}$.

Bilinear Diffie-Hellman problem. Let G be a cyclic group of order q and let g be a generator of G. Given $\{g, A = g^a, B = g^b, C = g^c\}$ where $a, b, c \in \mathbb{Z}_q$, compute $e(g, g)^{abc}$.

The success probability of any probabilistic, polynomial-time algorithm $\mathcal{A}$ in solving BDH in G is defined as:

$$Succ_{\mathcal{A},G}^{BDH} = Prob[\mathcal{A}(g, g^a, g^b, g^c) = e(g,g)^{abc}] \tag{4.1}$$

BDH assumption. For any probabilistic, polynomial–time algorithm $\mathcal{A}$, $Succ_{\mathcal{A},G}^{BDH}$ is negligible.

Computation Cost and Performance

For each party involved in our protocol, we express its computation effort in terms of execution times. To do so, we distinguish three different types of protocol steps. First, the *Initialization* process, where PEKS and `SHE`'s keys are generated by each client. During this step, the list of authorized IP addresses is also transmitted to $\mathcal{AD}$ by each client and equivalent trapdoors are computed by $\mathcal{AD}$. The second type consists of all tasks processed on a daily basis, meaning the data processing. Concretely, $\mathcal{AC}$ encrypts sensitive parts of data using the somewhat homomorphic and searchable encryption schemes and then transfers data to $\mathcal{ES}$. We also consider the test of each IP address with `SE`. Test function by $\mathcal{ES}$ with each client's trapdoor. Finally, the last type of steps consists of the audit use case. This protocol could be divided into five steps: *PIR Request*, *Computation*, *PIR Decryption*, *Values Decryption* and *Result*. Let n be the amount of evidence collected and stored in $\mathcal{ES}$ at a given time when an audit is requested. Value e is the number of clients making use of $\mathcal{AD}$ and p is the maximum number of authorized IP addresses by each client.

We specify parameters n, e, and p to estimate the concrete execution times of auditing process. Let us suppose that data from 10 different companies ($e = 10$) are located in $\mathcal{ES}$ and all of these companies are using the same auditor $\mathcal{AD}$. Each of these companies has 100 employees and each of these employees connects to the service offered by $\mathcal{CSP}$ on an average of 3 times a day. Considering 230 working days per year, we finally get the approximative number of 69×10^4 elements ($n = 10 \times 100 \times 3 \times 230$) after one year of evidence collection. Also, for each company we consider the maximum amount of allowed IP addresses of 256 ($p = 256$). For each cryptographic function, we express in Table 4.3 its approximative execution time. For the three cryptoschemes, we give the used configuration measurements from Table 4.3:

- **Paillier scheme**. Measurements have been made with a CPU configuration of Intel Core i5 M520 2.40GHz × 4, with a C++ implementation and a secret keys of 1024 bits.

- **PEKS scheme**. For this cryptosystem, we consider as in [LRW14] that the operation which requires most of computational effort is the bilinear mapping (also known as pairing). We use Lynn's [Lyn07] benchmarking where the evaluation time of this bilinear map is 11 ms. In the considered benchmarking, CPU configuration corresponds to Intel Pentium III at 1

GHz with an implementation using PBC library [Lyn07] and the Supersingular elliptic curve $y^2 = x^3 + x$ of 512 bits. The pairing function is only used with SE. PEKS and SE. Test functions.

- **SHE scheme**. We consider here results presented in [NLV11]. Configuration is a CPU of Intel Core 2 at 2.1 GHz, an implementation in algebra system MAGMA, a degree of ring polynomials $\eta = 512$ and a maximal ciphertext amount of polynomials of $D = 2$.

All of these execution times from Table 4.3 are obtained with a unique CPU. In a professional setting, using multiple CPUs and parallelization, one could drastically reduce these execution times. In Table 4.3, for each party, three execution times are given: $T1$ which corresponds to the time for initialization phase, $T2$ which is the time spent to process every new piece of data and finally $T3$ which represents the complete time of execution of an audit when the database size is $n = 69 \times 10^4$. We consider here that $\mathcal{CSP}$, $\mathcal{AD}$ and $\mathcal{ES}$ possess 6 CPUs each and, when the algorithm allows it, we integrate parallelization in the final execution times. We get 2.6×10^{-1} s for the client ($T1$), 1.79 h for $\mathcal{AD}$ ($T3$), 3.53×10^{-1} s for $\mathcal{CSP}$ ($T2$). For $\mathcal{ES}$ we obtain 4.73×10^{-1} s for $T2$ and 0.8 h for $T3$, respectively.

TABLE 4.3. PERFORMANCE OF CRYPTOGRAPHIC FUNCTIONS IN OUR PIR PROTOCOL.

Party	Step	Algorithm	Execution time (s)	# of calls
Client	Initialization	SE. KeyGen(λ)	–	1
		SHE. KeyGen(λ)	2.6×10^{-1}	1
Auditor	Initialization	SE. Trapdoor	–	cp
	PIR Request	PA. KeyGen(λ)	1.498×10^{-1}	1
		PA. Enc	2.503×10^{-2}	n
	PIR Decrypt	PA. Dec	2.314×10^{-2}	$4D\eta + 2$
	Values Decrypt	SHE. Dec	1.8×10^{-2}	4
	Result	Arithmetic functions	–	6
ES	Data	SE. Test	1.1×10^{-2}	p
	Computation	PA. Enc	2.503×10^{-2}	2
		SHE. Enc	3.53×10^{-1}	4
		PA. Add	1.50×10^{-5}	$(6\eta + 8)n$
		PA. Mult	1.282×10^{-4}	$(6\eta + 6)n$
		SHE. Mult	5.6×10^{-2}	n
CSP	Data	SHE. Enc	3.53×10^{-1}	4
		SE. PEKS	1.1×10^{-2}	1

With this particular choice of database size, we get an approximative execution time of the complete audit protocol of 2.6 hours. In Figure 4.4, the first

graphic illustrates this execution time according to database's size. To also get a precise idea of the computation cost to process data, the second graphic shows execution times according the amount of new data daily collected.

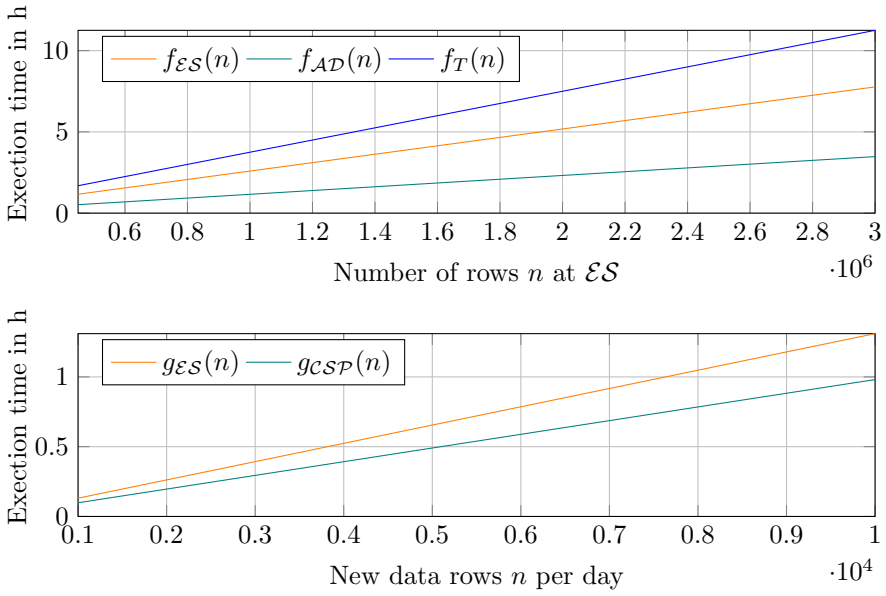

Figure 4.4. $f_{\mathcal{ES}}$, $f_{\mathcal{AD}}$ and f_T denote execution time in hours according to the size of the database at $\mathcal{ES}$, $\mathcal{AD}$ and in total. $g_{\mathcal{ES}}$ and $g_{\mathcal{CSP}}$ denote the execution time in hours to process the data according to the amount of new data collected per day for $\mathcal{ES}$ and $\mathcal{CSP}$.

Communication Cost

Finally, we consider the communication cost of the audit protocol between $\mathcal{ES}$ and $\mathcal{AD}$. By referring to Figure 4.3, we are looking here at three messages. First the *Company* and *Index* database's fields transmitted by $\mathcal{ES}$ to $\mathcal{AD}$, then the PIR request from $\mathcal{AD}$ to $\mathcal{ES}$ and finally the PIR response in the opposite direction. We could easily establish that the size of these three pieces of data depends on the number of elements in the database (n) and on the choice of Paillier's keys size i.e. the security parameter. Indeed, some Paillier's ciphertexts are included in the PIR's request and response. The size of these ciphertexts is exactly equal to twice the Paillier's modulo N size. In the chosen configuration of $n = 69 \times 10^4$ and q twice the size of N of 1024 bits, we have a size of data transmitted of 25.53 Mbits, a PIR request of 0.7 Gbits and its response of 6.14 Mbits. In general, both sizes of data transmitted and PIR response depend linearly on the database size n, e.g. for $n = 10^5$ the data transmitted is 3.7 Mbits and 600 Kbits for the PIR response. Lastly, the PIR request's size depends linearly on n and q, e.g. for $n = 10^5, q = 1024$, the PIR request corresponds to 102.4 Mbits.

4.2 Solution B - Adapting SHE to Evidence Processing

As seen in the first presented solution, we need to use a somewhat homomorphic encryption scheme that fits the scenario. Concretely we are aiming for an homomorphic encryption scheme, which supports in a cost-efficient manner many addition operations but only few multiplicative operations. The original SHE scheme is still a promising candidate for our setting.

It is in our pre-dominant interest to accelerate arithmetic operations on encrypted data, such that they can be performed also on a relatively large number of data. It turns out that this is a cross-layer effort. Parameters from the use case that shall be supported, need to be mapped to the performed number of arithmetic operations and size of the required cleartext value space. They also have to fit the encoding layer (plaintext polynomials), and finally the encryption layer (ciphertext polynomials). Since many of these parameters turn out to be mutually dependent, a proper configuration algorithm is required. Such configuration orchestrates them to apply SHE in a speed-optimized and secure way for a given setting. We will independently analyze parameters regarding correctness, performance and security aspects of a specific SHE scheme and we will merge them in order to draw this proper configuration algorithm. The selected somewhat homomorphic scheme is the one introduced by Brakerski and Vaikuntanathan in 2011 [BV11b] which we carefully adjust to our concrete privacy enhancing cloud auditing use case.

4.2.1 Discussion of the SHE Parameters

Parameters should be chosen according to guarantee i) the **correctness**, ii) an acceptable **performance** and iii) an appropriate **security** level.

Correctness Considerations

To guarantee correctness of the resulting value after having computed a certain amount of additions and multiplications on the ciphertexts, we have to ensure three aspects of the ciphertext evolution. The growing of information's degree, the growing of information's coefficient and finally the growing of error. Indeed, even if the ciphertext is not growing by itself because of the q and x^n+1 modulo reductions, these aspects are growing inside of it. In Figure 4.5 we can see the two ways information is spreading. The choice of parameters n, t and q will determine the maximal value that the degree and the coefficients could be. The parameters have been previously presented in Section 3.2.3. By performing computations on the ciphertexts, the plaintext information weaved into them will be able to grow up to these limitations.

We have seen in Section 3.2 that addition and multiplication could be performed directly on ciphertexts. In addition to consider the amount of executions of these two types of operations, we should also be aware of their order of execution. Indeed, this will impact the plaintext's maximal possible value and thus

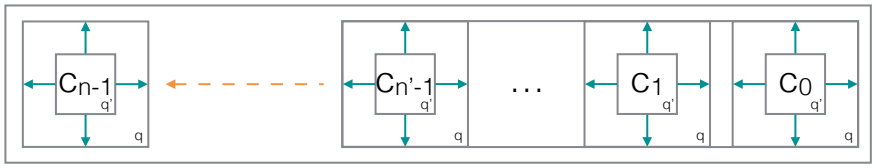

Figure 4.5. The two-dimensional growing of the information in the ciphertext polynomials.

the correctness of the final result. That is the reason why, as we will see later, we are considering an arithmetic tree to integrate the operations' order. Let's define:

$$\|ct\|_\infty := \max(|\ c_0\ |, |\ c_1\ |, \ldots, |\ c_\delta\ |) \tag{4.2}$$

the infinite norm of a ciphertext that corresponds to the maximal size of its coefficients in plaintext dimension.

For example, we first encode a plaintext value $m \in \mathbb{Z}_\mathcal{N}$ into a *message polynomial* p and we choose an encoding method where coefficient values are minimized. We get:

$$p(X) = c_0 + c_1 \times X + \ldots + c_{n-1} \times X^{n-1} \tag{4.3}$$

with $\forall i \in \{0, \ldots, n-1\},\ 0 \leqslant c_i < b$, and so $\|p\|_\infty = b - 1$. This polynomial is then encrypted by SHE scheme into a ciphertext couple $ct = \{ct_0, ct_1\}$ and we obtain $\|ct\|_\infty = q - 1$.

Let us now suppose that we are performing an encrypted addition between ciphertexts ct and ct' where ciphertext ct' has been similarly produced ($\|ct'\|_\infty = q - 1$). Since the encrypted addition is an arithmetic addition performed coefficient-wise as presented in Section 3.2.3, the resulting ciphertext ct_{add} will have the same norm due to modulo, i.e. $\|ct_{add}\|_\infty = q - 1$. However, in the plaintext space, the resulting polynomial will have its norm doubled i.e. $\|p_{add}\|_\infty = 2b + 2$. We could then generalize to:

$$\|Add(p, p')\|_\infty = \|p\|_\infty + \|p'\|_\infty \tag{4.4}$$

To be able to optimize the choice of parameter t (i.e. choosing it as small as possible while still ensuring security and correctness), the best solution would be to predict the maximal retrieved coefficient after performing the arithmetic tree according to the given use case on plaintexts. Parameter t will have to be greater than the infinite norm of the resulting corresponding polynomial.

We also have to consider limitations from the initial work [BV11b] in the Key Dependent Messages (KDM) scenario:

- $t > \sigma\sqrt{n}$ (i.e. a sample from χ resides in R_t with all but negligible probability).
- $t < 2^{-\omega(logn)} \times \sigma^{-d} \times q$ with $ct = (ct_0, \ldots, ct_d)$.

Let us now consider the encrypted multiplication. We define the degree function:

$$deg(p) := i \tag{4.5}$$

such that i is the degree of polynomial p. If we perform an addition between two ciphertexts, since addition is computed on respective coefficients, information will not spread to higher coefficients. On the contrary, when we perform a multiplication between ciphertexts, the information will affect higher degree coefficients. When we multiply two polynomials p and p', we can denote

$$deg(p_{mult}) = deg(p) + deg(p') \tag{4.6}$$

if $deg(p), deg(p') > 1$. In the ciphertext space, the degree of the encrypted polynomial will not increase because of the $x^n + 1$ modulo and at any time $deg(ct) \leqslant n - 1$. The use of multiplication on the ciphertext will result in a limitation of parameter n. Indeed, if after too many multiplications, $deg(p_{mult}) > n - 1$ the modulo wrapping will remove information and the final decryption will not be correct. For that reason, with the degree of the "fresh" encoded polynomials and the arithmetic tree, we will have to predict the degree of the resulting polynomial such that parameter n remains less than this degree.

Performance Considerations

In [NLV11], Naehrig *et al.* state for a reduction of the parameters' size to improve performance. In the following section, we aim to analyze how these parameters are manageable and concretely what their limitations are. We should determine which form could be the best with respect to the objective to reduce the computation cost of the cryptographic functions. We face two opposite strategies: firstly, having a high polynomial degree with small coefficients and secondly having a low maximal degree with significant coefficients.

We recall that parameter b is the base that is used to encode the message into a polynomial. If we choose a small base, by definition, we could get a small message space reduction t but that will imply the use of a large polynomial degree n. Vice versa, a choice of a larger base b will increase the lower bound of t since the "fresh" coefficients will be larger and will grow faster with computations. Choosing a larger encoding base could also reduce the lower bound of n and thus help reducing the homomorphic operation complexity.

Choosing the base b and selecting how to encode the *message polynomial* have to be considered. Regarding b, we could consider two opposite strategies to encode the messages into polynomials. First we could encode any message with coefficients $c_i \in \{0, \ldots, b-1\}$. With this strategy we will get a maximal degree for the encoding polynomial. The second strategy will be to have on the contrary a minimal degree for the *message polynomial*. Then we could get the first coefficient such that $c_0 = m$ (for m the complete plaintext message) and $c_i = 0$ for all $0 < i < n$. By this manner we get an encoding polynomial of degree 0 and therefore parameter n will not have to be that large.

To be able to represent a specific message space, we have to adjust the three parameters n, t and q, respectively the polynomial degree, the value spaces of

the coefficients of the cleartext and ciphertext. As message space, we should not consider only the initial message space but also the final one. This corresponds to the potential values of the messages after being computed in the ciphertext space.

In [NLV11], authors consider that a wise strategy regarding performance would be to reduce parameters t, q or n. They suggest to encode the messages as a list of bits. In other terms, as a polynomial with coefficients equal to $\{0, 1\}$. Such a strategy will directly require the largest parameter n for a given value space. Indeed, the minimal value of n should be equal to the logarithm of the maximal encoded message. Also, this technique implies a use of the base (parameter b) equal to 2, otherwise, not all the values could have been encoded (see Section 3.2.3). As we will see later, this seems to not be the optimal strategy to set the parameters.

We are now investigating how reducing the size of the polynomials will impact the performance of the encryption, decryption and computations of the ciphertexts. In a first step, we will see how a variation of parameter n impacts the performance. Subsequently we do this for parameter q. We are expressing the cryptographic functions in terms of elementary operations to see how they are connected to parameter n. We consider here "fresh" ciphertexts, (i.e. ciphertext with two polynomials) and polynomials multiplication performed with the FFT [1] algorithm:

$$\begin{aligned}\mathbf{1\ Enc_{data}} &\rightarrow 3\ Sample_{poly} + 3\ Add_{poly} + 2\ Mult_{poly} + 3n\ Mult_{poly} \\ &\rightarrow 3n\ Sample_{coef} + (2n\log n + 7n - 8)\ Add_{coef} \\ &+ (2n\log n + 7n - 8)\ Mult_{coef} + (4n\log n + 13n - 20)\ Mod_{coef}\end{aligned}$$

$$\mathbf{1\ Add_{data}} \rightarrow 2\ Add_{poly} \rightarrow 2n\ Add_{coef} + 2n\ Mod_{coef}$$

$$\begin{aligned}\mathbf{1\ Mult_{data}} &\rightarrow 4\ Mult_{poly} + 4\ Red_{poly} + 1\ Add_{poly} \\ &\rightarrow (4n\log n + 9n - 16)\ Add_{coef} + (4n\log n + 8n - 16)\ Mult_{coef} \\ &\quad + (21n - 40)\ Mod_{coef}\end{aligned}$$

$$\begin{aligned}\mathbf{1\ Dec_{data}} &\rightarrow 3\ Mult_{poly} + 1\ Add_{poly} \\ &\rightarrow (3n\log n + 7n - 12)\ Add_{coef} + (3n\log n + 6n - 12)\ Mult_{coef} \\ &\quad + (6n\log n + 16n - 30)\ Mod_{coef}\end{aligned}$$

The results show that the complexity of the four cryptographic functions is linearithmic in term of n.

[1] Fast Fournier Transform (FFT) algorithm which optimizes the multiplication's performance.

We now analyze the complexity at a deeper level. We consider the coefficients encoded with the *two's complement* encoding technique [vN93].

$$\begin{aligned}\mathbf{1\ Enc_{data}} \rightarrow\ & 3n\ Sample_{coef} + [(2n\log n + 7n - 8)(10(\log q)^2 - 14\log q + 8) \\ & + (4n\log n + 13n - 20)(5\log q - 2)]\ Add_{bit} \\ & + [(2n\log n + 7n - 8)(3(\log q)^2 - 2\log q + 1) \\ & + (4n\log n + 13n - 20)\log q]\ Mult_{bit}\end{aligned}$$

$$\mathbf{1\ Add_{data}} \rightarrow 4n\ (5\log q - 2)\ Add_{bit} + 4n\log q\ Mult_{bit}$$

$$\begin{aligned}\mathbf{1\ Mult_{data}} \rightarrow\ & [(4n\log n + 9n - 16)(10(\log q)^2 - 14\log q + 8) \\ & + (21n - 40)(5\log q - 2)]\ Add_{bit} \\ & + [(4n\log n + 9n - 16)(3(\log q)^2 - 2\log q + 1) \\ & + (21n - 40)\log q]\ Mult_{bit}\end{aligned}$$

$$\begin{aligned}\mathbf{1\ Mult_{data}} \rightarrow\ & [(3n\log n + 7n - 12)(10(\log q)^2 - 14\log q + 8) \\ & + (6n\log n + 16n - 30)(5\log q - 2)]\ Add_{bit} \\ & + [(3n\log n + 7n - 12)(3(\log q)^2 - 2\log q + 1) \\ & + (6n\log n + 16n - 30)\log q]\ Mult_{bit}\end{aligned}$$

We see here that the complexity is quadratic in term of $\log q$, or in term of parameter q, it is polylogarithmic. We could then affirm that a reduction of n is the predominant parameter to save performance even if that should imply an increase in q. This must be the guiding principle that leads our aimed algorithm.

Security Considerations

We are now taking security aspects into account by integrating a security measurement against a specific attack, currently considered as the most promising one. As in [NLV11], we are considering the distinguishing attack presented in [LP11]. In this work, authors analyze the security of a lattice-based encryption scheme based on the LWE problem. This scheme is an instance of an abstract system described by Micciancio [Mic10] that generalizes all the schemes of [Reg09, PVW08, GPV08]. The cryptosystem could be assimilated to the one we are currently using, since it is based on the same mathematical problem and both of them are characterized by the same parameters.

We are considering this attack and expressing the robustness of the scheme according to the selected parameters. As it is also done in [NLV11], we express the security of the scheme through the logarithm of the running time Θ (in seconds) of the attack itself under each specific parameters' configuration. This

amount of security depends on the three parameters n, q and l. Parameter l corresponds to the size of the messages that have to be encrypted and is expressed in bits. Equation (4.7) from [LP11] shows how we quantify the security:

$$log(\Theta) = \frac{1.8\ (2n+l)^2}{n\ log(q)} - 110 \tag{4.7}$$

As it is presented in the appendix section of [DPSZ12], the security of this cryptosystem could also be reduced to the hardness of the RLWE assumption (defined in Section 3.2.3). Parameter σ, the standard deviation of the used Gaussian distribution, relies on the security of such a problem. This parameter should be chosen so that it could avoid combinatorial style attack. An attack of this kind is presented in [AG11] which breaks LWE in time $2^{O(\|e\|_\infty^2)}$ with high probability, where e is the LWE error vector. Since e is chosen by the discrete Gaussian distribution with standard deviation σ, if we pick σ large enough then this attack should be prevented. Thus, according to [DPSZ12] choosing $\sigma \geqslant 8$ will ensure avoiding combinatorial attacks. Also, we define the second standard deviation $\sigma' = n \times \sigma$ to obtain slightly stronger overall security by making the public key better protected than any ciphertext.

4.2.2 Protocol

We integrate the three previously presented considerations on correctness, performance and security in order to obtain an algorithm that generates the perfect parameters to a suitable usage. This algorithm allows a user, respectively in our case a database administrator, to select appropriate and performance saving parameters for his own use case. We emphasize that our approach is generic, with the only pre-condition being that one can derive an arithmetic tree from its given use case. The algorithm's input arguments T, M and λ correspond respectively to the arithmetic tree of computations, the set of the value spaces of the computed data and the security parameter. In other words, M_i corresponds to the value spaces of the i^{th} leaf of T.

The overall protocol is presented in Algorithm 4 - `Generating_SHE_Parameters` and we describe each of its steps in detail:

> **Set** n**.** As we have seen in Section 4.2.1, parameter n should be chosen as a power of 2, as small as possible. Thus we initialize $n = 2$ in the algorithm.
>
> **Set** b**.** To initialize the encoding base, we consider parameters n and M_i and we apply the restriction formula $b \geqslant \lceil \sqrt[n]{Max(M_i)} \rceil$.
>
> **Compute** p_{result}**.** In the objective to select parameter t, we firstly have to compute a prediction of the resulting polynomial after the decryption phase (see Algorithm 5). In that way, we will select parameter t such that the *t-reduction* during the decryption will not remove any information as described in Section 4.2.1. For this step, the concrete use case's arithmetic

tree T is necessary to consider the order of the homomorphic operations and to which data they are processing. Then, we verify that $deg(p_{result}) \geqslant n$. If not, parameter b should be set greater in order to reduce $deg(p_{result})$.

Set t. We select parameter t prime such that $t > \|p_{result}\|_\infty$. In other words, t should be greater than the maximal possible value of coefficients from the resulting polynomial.

Set σ and σ'. As we have seen in Sections 3.2.3 and 4.2.1, to avoid combinatorial style attacks we set the standard deviations $\sigma = 8$ and $\sigma' = n \times \sigma$.

Set q. We set modulo q regarding others parameters. q should be a sub-exponential prime such that $q \equiv 1 \pmod{2n}$. To perform correctly the decryption phase, q should also fulfill the inequation $q > 2 \times \|p_{result}\|_\infty \times (t\sigma n^{1,5})^{D+1}$. Algorithm 6 enables to compute D.

Compute $log(\Theta)$. We compute the security level of the generated parameters and verify if it fulfills the security parameter λ. If not, we start over Algorithm 4 with a greater parameter n.

4.2.3 Evaluation and Results

Use Cases

To illustrate the choice of the correct parameters we have implemented our algorithms. We first execute the algorithms on the *Use Case 1 - Cloud Security Auditing* by using an implementation of the SHE scheme. Then we also apply the algorithms to another use case - *Billing Service* to support and verify that the generated parameters are indeed the best choice in terms of performance while at the same time providing appropriate security. These two specific use cases are derived from cloud security auditing [Clo], to illustrate our purpose. Nevertheless, we would like to emphasize that our algorithms could be applied to any scenario as long as the SHE scheme is needed. The user will have to provide his arithmetic tree as well as the message spaces of the different values used during the scenario and define the required level of security.

SHE Implementation. For the SHE running we are using the initial work of Bieberstein. For his Master Thesis [Bie14], he implemented the SHE scheme in *C++* with the use of two arithmetic libraries *Flint* and *GMP*. His work was purely based on the symmetric version and we modified it in order to use it as a public-key scheme. The need of the scheme with private/public keys was motivated by the use cases that could require the cloud infrastructure as investigated within the PAL-SAaaS project. The measurements presented in Tables 4.5 and 4.7 have been made with a CPU configuration of Intel Core i5 M520 2.40GHz x 4.

Algorithm 4 `Generating_SHE_Parameters`

Input: M, T, λ
Output: n, b, t, q, σ, σ'

Set $n \leftarrow \sqrt{2}$
Set $log(\Theta) \leftarrow 0$
while $log(\Theta) < \lambda$ **do**
    Set $n \leftarrow n^2$
    Set $b \leftarrow \lceil \sqrt[n]{Max(M)} \rceil$
    if $b < 2$ **then**
        $b \leftarrow 2$
    end if
    Compute $p_{result} \leftarrow$ `Estimate_ResultingPoly_fromArithmeticTree` $(n, b, T.root)$
    while $deg(p_{result}) \geqslant n$ **do**
        $b \leftarrow b + 1$
        Compute $p_{result} \leftarrow$ `Estimate_ResultingPoly_fromArithmeticTree` $(n, b, T.root)$
    end while
    Set t prime s.t. $t > \|p_{result}\|_\infty$
    Set $\sigma \leftarrow 8$
    Set $\sigma' \leftarrow n \times \sigma$
    Compute $D \leftarrow$ `Compute_MultDepth_fromArithmeticTree`$(T.root)$
    Set q prime s.t. $q \equiv 1 \pmod{2n}$ and $q > 2 \times \|p_{result}\|_\infty \times (t\sigma n^{1,5})^{D+1}$
    Compute $log(\Theta) \leftarrow (1.80 \times (2n + \lceil log(Max(M)) \rceil)^2)/(n \times log(q)) - 110$
end while
return σ, σ', b, n, t and q

Application to Use Case 1 - Cloud Security Auditing. The first considered use case corresponds to the one presented in Section 2.1. The cloud auditor is checking on behalf of a client that the $\mathcal{CSP}$'s access control has been performed correctly. Some computations will be performed directly on the evidence in the Evidence Store ($\mathcal{ES}$) with the aim to retrieve some amounts of specific data. The considered evidence is partially encrypted with the SHE scheme. To compute a set, we add all of the encryption regarding one field in the $\mathcal{ES}$ database and we get the total amount of connection attempts with this characteristic. To get an intersection of sets we first multiply two characteristics for each connection attempt and subsequently add all the products. We then get the total amount of connection attempts with the two characteristics. In Figure 4.6 we show a preview of the database stored in the $\mathcal{ES}$ simplified to two characteristics, along with the operation circuit of the computations that will have to be performed on the data.

We run our algorithms for this use case and obtain the following set of parameters $\{t = 20011,\ b = 2,\ n = 4096,\ log(q) = 84,\ \sigma = 8,\ \sigma' = 32768\}$ for a standard level of security of 128 bits ($\lambda = 128$). The use case's algorithm will

Algorithm 5 `Estimate_ResultingPoly_fromArithmeticTree`

Input: n, b, ν
Output: p_{result}

1: **if** $\nu = null$ **then**
2: return
3: **end if**
4: `Estimate_ResultingPoly_fromArithmeticTree`$(n, b, \nu.left\_son)$
5: `Estimate_ResultingPoly_fromArithmeticTree`$(n, b, \nu.right\_son)$
6: **if** $\nu.left\_son = null$ AND $S2_i = null$ **then**
7: $deg(\nu) \leftarrow \lfloor log_b(\nu.M) \rfloor$
8: **if** $\nu.M < b$ **then**
9: $\|\nu\|_\infty \leftarrow \nu.M$
10: **else**
11: $\|\nu\|_\infty \leftarrow b - 1$
12: **end if**
13: **else**
14: **if** $\nu.type = (+)$ **then**
15: $deg(\nu) \leftarrow max(deg(\nu.left\_son), deg(\nu.right\_son))$
16: $\|\nu\|_\infty \leftarrow \|\nu.left\_son\|_\infty + \|\nu.right\_son\|_\infty$
17: **else if** $\nu.type = (\times)$ **then**
18: $deg(\nu) \leftarrow deg(\nu.left\_son) + deg(\nu.right\_son)$
19: $\|\nu\|_\infty \leftarrow 2 \times [min(deg(\nu.left\_son), deg(\nu.right\_son)) + 1] \times \|\nu.left\_son\|_\infty \times \|\nu.right\_son\|_\infty$
20: **end if**
21: **end if**
22: $p_{result} \leftarrow \nu$
23: return p_{result}

perform 10^4 times a multiplication between two polynomials which represent a simple encryption of 0 or 1. Table 4.5 shows the different running times when we run the use case with the parameters obtained with our algorithms for n equals some power of two as well as the appropriate level of security against the distinguishing attack. We reasonably get results showing that increasing the polynomial degree makes the running times relatively bigger.

Table 4.5. UC1 runtimes (in ms) and level of security for different SHE parameters settings

t	b	n	$qBits$	Encryption	Addition	Multiplication	Decryption	Audit (in s)	$log(\Theta)$
20011	**2**	**4096**	**84**	22.773	3.980	40.64	10.186	534.345	241.171
20011	2	8192	87	47.814	7.937	81.409	21.420	1012.756	568.041
20011	2	16384	90	99.290	15.965	183.258	45.624	2220.649	1200.800

Algorithm 6 `Compute_MultDepth_fromArithmeticTree`

Input: ν
Output: *int* D
1: $D \leftarrow 0$
2: **if** $\nu.type = (+)$ **then**
3: $D \leftarrow max($`Compute_MultDepth_fromArithmeticTree`$(\nu.left\_son),$ `Compute_MultDepth_fromArithmeticTree`$(\nu.right\_son))$
4: **else if** $\nu.type = (\times)$ **then**
5: $D \leftarrow 1 +$ `Compute_MultDepth_fromArithmeticTree`$(\nu.left\_son)+$ `Compute_MultDepth_fromArithmeticTree`$(\nu.right\_son)$
6: **else**
7: return 0
8: **end if**
9: return D

Table 4.4. Use Case 1 - Cloud Security Auditing database sample

Authorized $a_i \in [\![0,1]\!]$	Connected $c_i \in [\![0,1]\!]$
$\mathcal{E}nc(a_1)$	$\mathcal{E}nc(c_1)$
$\mathcal{E}nc(a_2)$	$\mathcal{E}nc(c_2)$
...	...
$\mathcal{E}nc(a_{10000})$	$\mathcal{E}nc(c_{10000})$

Application to a Billing Service use case. This second use case represents a billing process for cloud service's usage executed by an auditor on encrypted data from a database. In this scenario, an auditor needs to sum up all the connection times of a client to a cloud service. The database stores every connection attempt (in this case, 10^4) with, for each of them, the duration - *Time*, the price to charge (per units of duration) - *Price*, if the connection was made from an authorized user - *Authorized* and if the user succeeded to connect - *Connected*. All of the database components are encrypted with the specific SHE encryption scheme presented in Section 3.2.3. Regarding the characteristic of being made from an authorized user or not and being successful or not, we express it by having an encryption of 1 to the respective field and an encryption of 0 otherwise. For each client, the cloud auditor multiplies the service usage with the cost and with the two fields *Authorized* and *Connected* which represent tags. Finally, the auditor sums up all values together and sends the final sum to the client. We show in Figure 4.7 a preview of the database stored in the Evidence Store ($\mathcal{ES}$), along with the operation circuit of the computations that will have to be performed on the data. For each line, we do a multiplication between time and cost and another one between the two tag fields. Subsequently we multiply the two resulting products. Finally, the summation of all of these 10^4 products

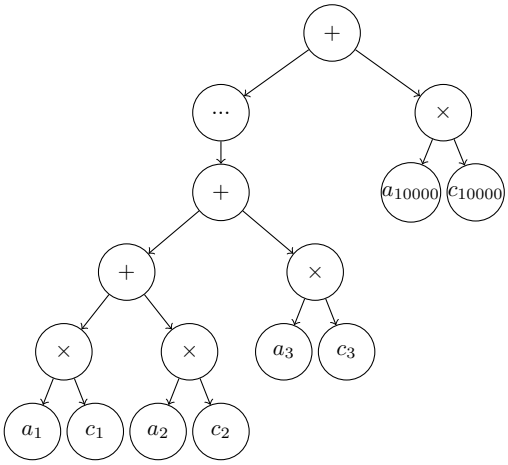

Figure 4.6. UC1 - Cloud Security Auditing arithmetic tree obtained from the SQL query > SELECT SUM (Authorized × Connected) FROM UC1.

is performed. An important aspect to notice is that since we are multiplying products, we get a multiplication depth of three.

Table 4.6. Use Case - Billing Service database sample

Time $t_i \in [\![0, 86400]\!]$	Price $p_i \in [\![0, 10]\!]$	Authorized $a_i \in [\![0, 1]\!]$	Connected $c_i \in [\![0, 1]\!]$
$\mathcal{E}nc(t_1)$	$\mathcal{E}nc(p_1)$	$\mathcal{E}nc(a_1)$	$\mathcal{E}nc(c_1)$
$\mathcal{E}nc(t_2)$	$\mathcal{E}nc(p_2)$	$\mathcal{E}nc(a_2)$	$\mathcal{E}nc(c_2)$
...	...	...	...
$\mathcal{E}nc(t_{10000})$	$\mathcal{E}nc(p_{10000})$	$\mathcal{E}nc(a_{10000})$	$\mathcal{E}nc(c_{10000})$

We run our algorithm `Generating_SHE_Parameters` for the Billing Service UC with a standard level of security of 128 bits ($\lambda = 128$) and obtain the following set of parameters $\{t = 320009,\ b = 2,\ n = 8192,\ log(q) = 178,\ \sigma = 8,\ \sigma' = 65536\}$.

Independently of our resulting set of parameters, we run the use case with several other sets of parameters to illustrate the accuracy of our algorithms in term of performance. In Table 4.7 we show the different running times as well as the level of the security against the distinguishing attack. In this specific scenario, we consider that $\mathcal{CSP}$ has encrypted and aggregated 2.10^4 rows in the ES database. It makes no sense to express the total time of the database generation since each data has been aggregated one by one when a connection attempt occurred. The *Audit* field corresponds to the execution of the computations circuit by the auditor. We notice that we could get a very small parameter n thanks to an increase of the base, but to fulfill an acceptable level of security against the distinguishing attack we must consider a large polynomial degree as

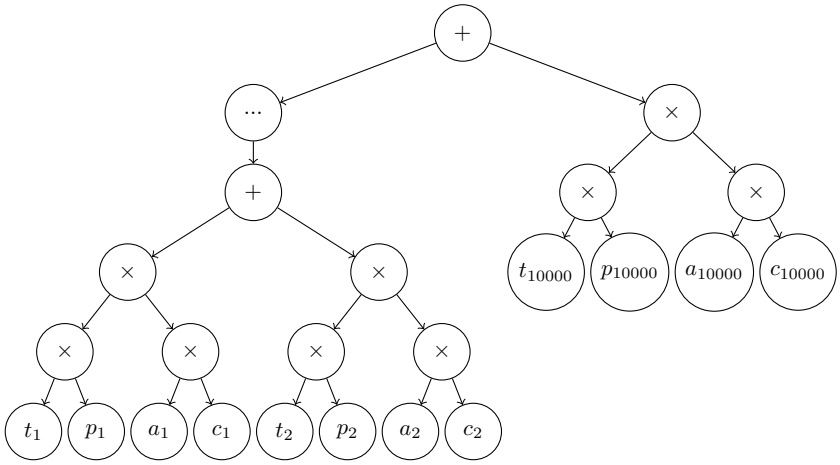

Figure 4.7. UC - Billing Service arithmetic tree obtained from the SQL query > SELECT SUM ((Time × Price)×(Authorized × Connected)) FROM UC BILLING SERVICE.

Table 4.7. UC Billing Service runtimes (in ms) and level of security for different SHE parameters settings

t	b	n	$log(q)$	Encryption	Addition	Multiplication	Decryption	Audit (in s)	$log(\Theta)$
320009	**2**	**8192**	**178**	47.735	8.757	78.101	30.850	4585.882	223.956
320009	2	16384	184	99.956	17.433	175.129	63.754	9905.253	533.620
320009	2	32768	190	215.479	34.710	380.250	137.817	31921.553	1134.161

$n = 8192$ for a security of 223 bits. In Table 4.7, the first line corresponds to the parameters obtained by our algorithm `Generating_SHE_Parameters`. The two following ones correspond to the optimized parameters sets when we set degree n as the very next power of two. By observing the resulting running times, we could see that for every line, the running time of each function doubles. For instance, if we select a parameter n four times too large ($n = 16384$) the running time of the auditing process will be approximately four times longer than with the optimized n selected by our algorithm `Generating_SHE_Parameters`.

Regarding the running times of both use cases, we notice that every running time of the cryptographic functions of the SHE scheme grows linearly in terms of the degree n of the polynomial representation. In our scenario, a perfect selection of the scheme's parameters could then lead to major time savings for the auditor as well as for the $\mathcal{CSP}$.

4.3 Solution C - Using Bloom Filters to Process Set Relations

We propose here a protocol to solve what we present as *private outsourced inclusiveness test* and *private outsourced disjointness test*. We suppose two parties, let it be Alice and Bob each owning a dataset of elements, respectively $\mathcal{A}$ and $\mathcal{B}$. They would like to know, for example, if their datasets are disjoint, i.e. if $\mathcal{A} \cap \mathcal{B} = \emptyset$. To do so, they will ask a third party, $\mathcal{S}$ to perform the verification. Alice (resp. Bob) does not want to reveal any information on her dataset including its size to any other party.

We propose a solution based on Bloom filters [Blo70] representation along with keyed hash functions as HMAC to solve both of the set relations. We recall that Bloom filters are space-efficient data structures used to represent sets that allow to perform set membership checks; a more complete description can be found in Section 3.5. One may argue that a simple pseudonymization could be sufficient for the scenario sketched above, as only apply a keyed hash function on the sets [CC04b]. However, even if the pseudonymization function remains private to any other party than the Bloom filter owners, one may directly gain knowledge on the number of common elements of two Bloom filters. Such a naive approach will also reveal which pseudonym is present in none, one or both sets. Conversely, Bloom filter representation has the particular feature of adding obfuscation to sets.

We argue that contrarily to multiparty based solutions [KS05], it has relevance if such a protocol class is non-interactive. It could for example be applied to a scenario of networking where a social network server should verify if any users have relations in common. In particular we apply this solution to the four selected use cases in Chapter 2 as for instance, it could be applied to cloud auditing use cases (UC1) where a third party auditor should perform verification on logfiles and whitelists or scenarios of mobile users tracking (UC2).

In the following sections, we present how we tuned the Bloom filters usage to enhance privacy on the sets' content and cardinality. After describing our functions in detail, we analyse the correctness and privacy of our protocol, in particular by presenting one possible attack on the sets' cardinality. Finally we explain how we applied this solution to our four use cases and present their respective results.

4.3.1 Privacy Enhancements

Using Bloom filters with privacy-sensitive scenarios is not as common as it is for classical applications. Our approach, which consists of making such a technique fitting privacy-sensitive use cases, is based on the use of a public keyed collision-resistant hash function (e.g. MAC function) with a set of n_{key} private keys instead of n_{key} public hash functions. Without loss of generality we decide to use an HMAC function to solve our bunch of use cases. That being

said, any party that does not hold the keys cannot use the *test()* function to verify directly whether a specific element is included in the Bloom filter or not. The other security benefit when using an HMAC function is that even if the function is publicly released, any party that does not hold the keys cannot add additional elements. More formally, we define a Bloom filter of size m of a set $\mathcal{A} = \{a_1, \ldots, a_{n_\mathcal{A}}\}$, with a set of n_{key} keys $K = \{k_1, \ldots, k_{n_{key}}\}$ and an HMAC function $h_k : \{0,1\}^* \to \{1, \ldots, m\}$ with $k \in K$ as:

$$BF\big(\mathcal{A}, (h_k)_{k\in K}\big) = bf_\mathcal{A}[j]_{1\leqslant j\leqslant m} \tag{4.8}$$

$$where \;\; bf_\mathcal{A}[j] = 1 \; if \; \exists \; (i,\kappa) \; s.t. \; h_{k_\kappa}(a_i) = j$$

$$bf_\mathcal{A}[j] = 0 \; otherwise$$

For a better understanding, we use the simplified notation $BF_\mathcal{A}$ to represent the Bloom filter of set $\mathcal{A}$.

The second privacy enhancement we add aims to keep parameter n_{key} private in order to avoid revealing the sets' cardinalites. Having parameter n_{key} publicly released in addition to privacy requirements on cardinality, requires too many overlapping bits in the Bloom filters. Indeed, the naive technique to retrieve the cardinality of the set by simply looking at its respective Bloom filter would be to divide its amount of bits set to one, by parameter n_{key}. There exists an optimized technique introduced by Swamidass and Baldi [SB07] that computes $n^*_\mathcal{A}$, an approximation of the number of distinct elements inserted in $BF_\mathcal{A}$, with $X_{BF_\mathcal{A}}$ the amount of bits set to 1 in the Bloom filter:

$$n^*_\mathcal{A} = -\frac{m}{n_{key}} ln\Big[1 - \frac{X_{BF_\mathcal{A}}}{m}\Big] \tag{4.9}$$

Such a technique requires even more overlapping bits to mislead any prospective attacker. We see that by making parameter n_{key} private, one could not be able to compute $n^*_\mathcal{A}$ anymore. We give examples in Table 4.8, for several parameter' configurations. We simulate the generation of 10^3 Bloom filters $BF_\mathcal{A}$ and we estimate the sets' cardinality with the naive technique and the one from Swamidass and Baldi (S&B). By referring to the results obtained using S&B technique, we could agree on the necessity of keeping n_{key} private to gain privacy on the sets' cardinality. However, in Section 4.3.4 we show how an attacker could still retrieve information on cardinalities even with such precaution. One may question the complexity of keeping size of K private or the effort to store a large amount of keys. We could then slightly modify the protocol to have a unique key k. Indeed, outcome of $h_k(x)$ could be divided in n_{key} equal size fragments, each indicating Bloom filter's index to increment. As it is suggested in [LG], in case of a fragment too small to contain information of size m, we may perform multiple rounds of HMAC operations with different salts. Of course, in such protocol modification, parameter n_{key} and sets' size are not disclosed.

TABLE 4.8. SET CARDINALITY ESTIMATIONS WITH THE NAIVE AND SWAMIDASS AND BALDI (S&B) TECHNIQUE FOR DIFFERENT PARAMETERS' CONFIGURATIONS.

$n_{\mathcal{A}}$	m	n_{key}	$X_{BF_{\mathcal{A}}}$	$n^*_{\mathcal{A}}$ with *naive*	$n^*_{\mathcal{A}}$ with *S&B*
100	1.44×10^3	10	[682 - 758]	[68.2 - 75.8]	[92.4 - 107.6]
100	6.5×10^7	1700	[169824 - 171718]	[100.0 - 101.1]	[99.9 - 100.0]
1000	2×10^9	950	[949730 - 949822]	[999.7 - 999.8]	[999.9 - 1000.1]

4.3.2 Protocol Functions

Initialization.

$\mathbf{h}, \mathbf{n_{key}}, \mathbf{m}, \mathbf{K} \leftarrow$Setup: One of the two parties Alice or Bob should first choose and generate the Bloom filter's parameters: a dimension m, an HMAC function h, an amount of keys n_{key} and the set of keys $K = \{\kappa_1, \ldots, \kappa_{n_{key}}\}$. Alice generates parameters by performing the following protocol:

- randomly choose $n_{key} \in [n^L_{key}; n^U_{key}]$ with n_{key}, n^L_{key} and n^U_{key} integers.
- set m such that $X_{\cap=\emptyset} < n^L_{key}$.

Values n^L_{key} and n^U_{key} are public and we determine them later considering correctness and privacy in Section 4.3.3. Restriction on parameter m corresponds to a correctness consideration which we explain in greater detail in Section 4.3.3. Alice then selects the public HMAC function h, generates the n_{key} respective keys and privately shares parameters $\{h, n_{key}, m, K\}$ with Bob.

$\mathbf{BF}_{\mathcal{A}}, \mathbf{BF}_{\mathcal{B}} \leftarrow$Create($\mathcal{A}, \mathcal{B}$): Alice (resp. Bob) generates the Bloom filter of its data $\mathcal{A} = \{a_1, \ldots, a_{n_{\mathcal{A}}}\}$ (resp. $\mathcal{B} = \{b_1, \ldots, b_{n_{\mathcal{B}}}\}$).

$$BF_{\mathcal{A}} = BF\big(\mathcal{A}, (h_\kappa)_{\kappa \in K}\big) = bf_{\mathcal{A}}[j]_{1 \leqslant j \leqslant m}$$
$$BF_{\mathcal{B}} = BF\big(\mathcal{B}, (h_\kappa)_{\kappa \in K}\big) = bf_{\mathcal{B}}[j]_{1 \leqslant j \leqslant m}$$

Inclusiveness Protocol.

This operator allows to verify if one set is included in another. It has to be performed directly on Bloom filters of the respective sets. To determine if $\mathcal{A}$ is included in $\mathcal{B}$ we define $\mathbf{BF}_{\mathcal{A} \subseteq \mathcal{B}} \leftarrow \mathbf{INC}(\mathbf{BF}_{\mathcal{A}}, \mathbf{BF}_{\mathcal{B}})$:

$$\begin{aligned} &bf_{\mathcal{A} \subseteq \mathcal{B}}[j]_{1 \leqslant j \leqslant m} \leftarrow INC(BF_{\mathcal{A}}, BF_{\mathcal{B}}) \\ &where \;\; 0 \leftarrow bf_{\mathcal{A} \subseteq \mathcal{B}}[j] \; if \; (bf_{\mathcal{A}}[j] = 1 \;\wedge\; bf_{\mathcal{B}}[j] = 0) \\ &1 \leftarrow bf_{\mathcal{A} \subseteq \mathcal{B}}[j] \; otherwise. \end{aligned} \tag{4.10}$$

We remark that such operator is equivalent to the bitwise binary operator combination:

$$INC(BF_{\mathcal{A}}, BF_{\mathcal{B}}) \equiv \neg(BF_{\mathcal{A}})\ OR\ \ BF_{\mathcal{B}} \tag{4.11}$$

$\mathcal{S}$ firstly computes the inclusion protocol on the two respective Bloom filters of sets $\mathcal{A}$ and $\mathcal{B}$ to test if $\mathcal{A} \subseteq \mathcal{B}$, namely if all the elements from Alice's set are included in Bob's set:

$$INC(BF_{\mathcal{A}}, BF_{\mathcal{B}}) = BF_{\mathcal{A}\subseteq\mathcal{B}} = bf_{\mathcal{A}\subseteq\mathcal{B}}[j]_{1\leqslant j\leqslant m}$$

Then $\mathcal{S}$ expresses value $X_{\mathcal{A}\subseteq\mathcal{B}}$ which corresponds to the number of bits set to 1 in the resulting Bloom filter:

$$X_{\mathcal{A}\subseteq\mathcal{B}} = \sum_{j=1}^{m} bf_{\mathcal{A}\subseteq\mathcal{B}}[j] \tag{4.12}$$

$\mathcal{S}$ tests whether $X_{\mathcal{A}\subseteq\mathcal{B}} = m$ and can conclude that $\mathcal{A} \subseteq \mathcal{B}$ if no false positive occurred. Otherwise we have $\mathcal{A} \nsubseteq \mathcal{B}$ with certainty.

We give an example of the *inclusiveness* relation with tiny sets and parameters. We test if both sets $\mathcal{A}$ and $\mathcal{A}'$ are included in set $\mathcal{B}$.

$\mathcal{A} = \{x_1, x_2\}$, $\mathcal{A}' = \{x_1, x_4\}$, $\mathcal{B} = \{x_1, x_2, x_3\}$
$m = 12$, $n_{key} = 3$, $K = \{k_1, k_2, k_3\}$

$h_{k_1}(x_1) = 10$, $h_{k_1}(x_2) = 1$, $h_{k_1}(x_3) = 5$, $h_{k_1}(x_4) = 9$
$h_{k_2}(x_1) = 11$, $h_{k_2}(x_2) = 10$, $h_{k_2}(x_3) = 12$, $h_{k_2}(x_4) = 2$
$h_{k_3}(x_1) = 2$, $h_{k_3}(x_2) = 5$, $h_{k_3}(x_3) = 8$, $h_{k_3}(x_4) = 1$

BF_A	1	1	0	0	1	0	0	0	0	1	1	0
$BF_{A'}$	1	1	0	0	0	0	0	0	1	1	1	0
BF_B	1	1	0	0	1	0	0	1	0	1	1	1

$BF_{A\subseteq B}$	0	0	0	0	0	0	0	0	0	0	0	0
$BF_{A'\subseteq B}$	0	0	0	0	0	0	0	0	1	0	0	0

By computing the *inclusiveness* relation we see that $\mathcal{A} \subseteq \mathcal{B}$. On the contrary $\mathcal{A}' \nsubseteq \mathcal{B}$ since $bf_{\mathcal{A}'}[9] = 1$ and $bf_{\mathcal{B}}[9] = 0$.

Disjointness Protocol.

This set relation allows us to check that no elements from one set are included in another one. In other words, it verifies the claim that two sets are disjoint. Such a test function is not trivial. Indeed, if we use Bloom filters, it is not sufficient to

highlight cases where a bit has been set to 1 at the same index in the respective Bloom filters. We define this operator as $\mathbf{BF}_{\mathcal{A}\cap\mathcal{B}=\emptyset} \leftarrow \mathbf{DIS}(\mathbf{BF}_{\mathcal{A}}, \mathbf{BF}_{\mathcal{B}})$:

$$\begin{aligned} &bf_{\mathcal{A}\cap\mathcal{B}=\emptyset}[j]_{1\leqslant j\leqslant m} \leftarrow DIS(BF_{\mathcal{A}}, BF_{\mathcal{B}}) \\ &where \;\; 1 \leftarrow bf_{\mathcal{A}\cap\mathcal{B}=\emptyset}[j] \; if \; (bf_{\mathcal{A}}[j] = 1 \;\wedge\; bf_{\mathcal{B}}[j] = 1) \\ &0 \leftarrow bf_{\mathcal{A}\cap\mathcal{B}=\emptyset}[j] \; otherwise. \end{aligned} \tag{4.13}$$

We remark that this operator is equivalent to the bitwise logical-and operator:

$$DIS(BF_{\mathcal{A}}, BF_{\mathcal{B}}) \equiv BF_{\mathcal{A}} \; AND \; BF_{\mathcal{B}}. \tag{4.14}$$

To verify that no elements from Alice's dataset are included Bob's, $\mathcal{S}$ performs the disjointness relation on respective Bloom filters of $\mathcal{A}$ and $\mathcal{B}$:

$$DIS(BF_{\mathcal{A}}, BF_{\mathcal{B}}) = BF_{\mathcal{A}\cap\mathcal{B}=\emptyset} = \; bf_{\mathcal{A}\cap\mathcal{B}=\emptyset}[j]_{1\leqslant j\leqslant m}$$

Then $\mathcal{S}$ expresses $X_{\mathcal{A}\cap\mathcal{B}=\emptyset}$ which corresponds to the number of bits set to 1 in the resulting Bloom filter:

$$X_{\mathcal{A}\cap\mathcal{B}=\emptyset} = \sum_{j=1}^{m} bf_{\mathcal{A}\cap\mathcal{B}=\emptyset}[j] \tag{4.15}$$

$\mathcal{S}$ compares it such that:

if $X_{\mathcal{A}\cap\mathcal{B}=\emptyset} < n_{key}^{L}$ then $\mathcal{A}$ and $\mathcal{B}$ are distinct

if $X_{\mathcal{A}\cap\mathcal{B}=\emptyset} \geqslant n_{key}^{L}$ then $\mathcal{A}$ and $\mathcal{B}$ have at least one element in common

Indeed, for each element included in both sets, we get n_{key} times a bit set to 1 in the resulting Bloom filter. However, we could still get such a bit set to 1 due to a bit set to 1 in $BF_{\mathcal{A}}$ and $BF_{\mathcal{B}}$ stemming from different elements originally added to the Bloom filters. We call such a case a false negative for the disjointness relation since the auditor will state that the sets are not disjoint while they are. We will discuss the probability of this occurrence in the following sections.

We give another toy example of the *disjointness* relation between $BF_{\mathcal{A}}$, $BF'_{\mathcal{A}}$ and $BF_{\mathcal{B}}$.

$\mathcal{A} = \{x_4, x_5\}$, $\mathcal{A}' = \{x_1, x_4\}$, $\mathcal{B} = \{x_1, x_2, x_3\}$
$m = 12$, $n_{key} = 3$, $n_{key}^{L} = 3$, $n_{key}^{U} = 5$, $K = \{k_1, k_2, k_3\}$

$h_{k_1}(x_1) = 10$, $h_{k_1}(x_2) = 1$, $h_{k_1}(x_3) = 5$, $h_{k_1}(x_4) = 9$, $h_{k_1}(x_5) = 6$
$h_{k_2}(x_1) = 11$, $h_{k_2}(x_2) = 10$, $h_{k_2}(x_3) = 12$, $h_{k_2}(x_4) = 2$, $h_{k_2}(x_5) = 1$
$h_{k_3}(x_1) = 2$, $h_{k_3}(x_2) = 5$, $h_{k_3}(x_3) = 8$, $h_{k_3}(x_4) = 1$, $h_{k_3}(x_5) = 7$

BF_A	1	1	0	0	0	1	1	0	1	0	0	0
$BF_{A'}$	1	1	0	0	0	0	0	0	1	1	1	0
BF_B	1	1	0	0	1	0	0	1	0	1	1	1

$BF_{A\cap B=\varnothing}$	1	1	0	0	0	0	0	0	0	0	0	0
$BF_{A'\cap B=\varnothing}$	1	1	0	0	0	0	0	0	0	1	1	0

We get $X_{\mathcal{A}\cap\mathcal{B}=\emptyset} = 2$ and $X_{\mathcal{A}'\cap\mathcal{B}=\emptyset} = 4$. Since we have $X_{\mathcal{A}\cap\mathcal{B}=\emptyset} < n_{key}^L$, we definitively know that $\mathcal{A}$ and $\mathcal{B}$ are disjoint. On the contrary, we get $X_{\mathcal{A}'\cap\mathcal{B}=\emptyset} \geqslant n_{key}^L$, therefore $\mathcal{A}'$ and $\mathcal{B}$ might be disjoint. Indeed, with such toy configuration we have $X_{\mathcal{A}\cap\mathcal{B}=\emptyset} = 2$ and do not get $n_{key}^L << X_{\mathcal{A}\cap\mathcal{B}=\emptyset}$. With $X_{\mathcal{A}'\cap\mathcal{B}=\emptyset} = 4$ we hesitate between either $\mathcal{A}'$ and $\mathcal{B}$ disjoint or having one element in common.

4.3.3 Correctness Analysis

In this section we analyze correctness of our two proposed relations. We recall the Bloom filter approach allows false positives but no false negative on the *test()* function. Nevertheless, with our proposal we are not considering the *test()* function and we thus focus on overlapping bits of the Bloom filters resulting from our set relations.

Correctness of the Inclusiveness Relation.

For the *inclusiveness* relation, we stress that only false positives could happen and not false negatives. Indeed, after performing $INC(BF_{\mathcal{A}}, BF_{\mathcal{B}})$, if there is an index j with $bf_{\mathcal{A}\subseteq\mathcal{B}}[j] = 0$, we have $bf_{\mathcal{A}}[j] = 1$ and $bf_{\mathcal{B}}[j] = 0$, then with certainty, at least one element from $\mathcal{A}$ does not belong to $\mathcal{B}$. In practice, if the outcome of the auditing process states that $\mathcal{A} \nsubseteq \mathcal{B}$ then we have a probability of correctness of 1.

On the other hand, if we get $\mathcal{A} \subseteq \mathcal{B}$ as a result, such outcome is not necessarily correct. Indeed, we get a probability of correctness equal to $1 - \mathcal{P}_{FP}$, with $\mathcal{P}_{FP}$ the probability of having a false positive. $\mathcal{P}_{FP}$ could be expressed in terms of parameters n_{key}, m and $n_{\mathcal{W}}$ denoting the amount of elements inserted in $BF_{\mathcal{B}}$. The probability that our *inclusiveness* relation outcomes a false positive whereas one element a_1 from $\mathcal{A}$ is not in $\mathcal{B}$ is equivalent to the one to have $test(\mathcal{B}, a_1)$ resulting *true* with the same parameters. We detail the value of $\mathcal{P}_{FP}$:

First, we denote the probability that after inserting $n_{\mathcal{A}}$ elements, a certain bit is equal to 1 is:

$$1 - (1 - \frac{1}{m})^{n_{key} \times n_{\mathcal{A}}} \tag{4.16}$$

If we consider that $Z_{\mathcal{A},\mathcal{B}}$ elements from $\mathcal{A}$ are not included in $\mathcal{B}$, the probability

of having a false positive after computing *inclusiveness* relation is:

$$\mathcal{P}_{FP} \geqslant \left(1 - (1 - \frac{1}{m})^{n_{key} \times n_{\mathcal{A}}}\right)^{n_{key} \times Z_{\mathcal{A},\mathcal{B}}} \tag{4.17}$$

$$\mathcal{P}_{FP} \simeq \left(1 - (1 - \frac{1}{m})^{X_{add}(n_{\mathcal{A}})}\right)^{X_{add}(Z_{\mathcal{A},\mathcal{B}})} \tag{4.18}$$

With $X_{add}()$ presented in (3.15).

We give here a toy example of the *inclusiveness* relation between $BF_{\mathcal{A}}$ and $BF_{\mathcal{B}}$ and where we obtain a false positive case.

$\mathcal{A} = \{x_1, x_5\}$, $\mathcal{B} = \{x_1, x_2, x_3\}$
$m = 12$, $n_{key} = 3$, $n^L_{key} = 3$, $n^U_{key} = 5$, $K = \{k_1, k_2, k_3\}$

$h_{k_1}(x_1) = 10$, $h_{k_1}(x_2) = 1$, $h_{k_1}(x_3) = 5$, $h_{k_1}(x_5) = 12$
$h_{k_2}(x_1) = 11$, $h_{k_2}(x_2) = 10$, $h_{k_2}(x_3) = 12$, $h_{k_2}(x_5) = 5$
$h_{k_3}(x_1) = 2$, $h_{k_3}(x_2) = 5$, $h_{k_3}(x_3) = 8$, $h_{k_3}(x_5) = 1$

BF_A	1	1	0	0	1	0	0	0	0	1	1	0
BF_B	1	1	0	0	1	0	0	1	0	1	1	1

$BF_{A \subseteq B}$	0	0	0	0	0	0	0	0	0	0	0	0

By performing the *inclusiveness* relation *INC()* on $BF_{\mathcal{A}}$ and $BF_{\mathcal{B}}$ we get $\mathcal{A} \subseteq \mathcal{B}$ but this is not correct since $x_5 \notin \mathcal{B}$.

We test our implementation of Bloom filters construction for multiple parameters' configurations. We express in Table 4.9 the percentage of false positive obtained while performing 10^4 times the *inclusiveness* relation between sets $\mathcal{A}$ and $\mathcal{B}$, and we verify that it corresponds to the respective $\mathcal{P}_{FP}$ expressed in (4.18).

Correctness of the Disjointness Relation.

For the *disjointness* relation, we have on the contrary no case of false positive, but a case of false negative may happen. Indeed, if we get $X_{\mathcal{A} \cap \mathcal{B} = \emptyset} < n^L_{key}$, it means that $BF_{\mathcal{A}}$ and $BF_{\mathcal{B}}$ have less than n^L_{key} (and thus less than n_{key}) indexes i where $bf_{\mathcal{A}}[i] = 1$ and $bf_{\mathcal{B}}[i] = 1$. It is then not possible that $\mathcal{A}$ and $\mathcal{B}$ have common elements. Regarding the false negative scenario, it could happen if we get too many *resulting overlapping bits* in the resulting Bloom filter $BF_{\mathcal{A} \cap \mathcal{B} = \emptyset}$, which we define as:

Definition 5 (Resulting overlapping bit)**.** *When there exists a specific index i, where $bf_{\mathcal{A}}[i] = bf_{\mathcal{B}}[i] = 1$ and these two bits are coming from different elements.*

TABLE 4.9. COMPARISON OF THE FALSE POSITIVE PERCENTAGE RESULTING FROM THE RUNNING OF THE INCLUSIVENESS RELATION'S IMPLEMENTATION AND THE PROBABILITY FROM (4.18).

$n_{\mathcal{B}}$	n_{key}	m	$\mathcal{P}_{FP}$	$\%_{FP}$
10^2	10^1	1×10^3	1.02×10^{-2}	9.00×10^{-3}
10^2	7×10^2	1×10^4	5.28×10^{-1}	5.38×10^{-1}
10^3	5	5×10^3	1.01×10^{-1}	1.01×10^{-1}
10^3	10^2	2×10^4	5.09×10^{-1}	5.12×10^{-1}
10^4	5×10^2	1×10^6	3.40×10^{-2}	4.00×10^{-2}
10^4	10^3	1.5×10^6	2.80×10^{-1}	2.20×10^{-1}

Therefore, a false negative consists of a case where $\mathcal{A}$ and $\mathcal{B}$ have no elements in common but $\mathcal{S}$ gets $X_{\mathcal{A}\cap\mathcal{B}=\emptyset} \geqslant n_{key}^L$, i.e. more than n_{key}^L *resulting overlapping bits* happened. To avoid such a case, we have to accurately tune parameters such that in a case of distinct sets $\mathcal{A}$ and $\mathcal{B}$, the respective value $X_{\mathcal{A}\cap\mathcal{B}=\emptyset}$ will never (with acceptable probability) be greater than n_{key}^L. To do so, Alice has to carefully select parameters n_{key} and m such that $X_{\cap=\emptyset} < n_{key}^L$. Value $X_{\cap=\emptyset}$ represents the expected value of $X_{\mathcal{A}\cap\mathcal{B}=\emptyset}$ when performing the *disjointness* protocol on two distinct sets $\mathcal{A}$ and $\mathcal{B}$. To express value $X_{\cap=\emptyset}$, we firstly give the probability of having a bit set to 1 for any index j in both Bloom filters $BF_{\mathcal{A}}$ and $BF_{\mathcal{B}}$, knowing that $\mathcal{A}$ and $\mathcal{B}$ are distinct:

$$p(bf_{\mathcal{A}}[j] = 1 \ \wedge \ bf_{\mathcal{B}}[j] = 1) = p(bf_{\mathcal{A}}[j] = 1) \ \times \ p(bf_{\mathcal{B}}[j] = 1) \tag{4.19}$$

$$= (1 - (1 - \frac{1}{m})^{n_{key} \times n_{\mathcal{A}}}) \ \times \ (1 - (1 - \frac{1}{m})^{n_{key} \times n_{\mathcal{B}}})$$

Finally, the expected amount of bits set to 1 in both $BF_{\mathcal{A}}$ and $BF_{\mathcal{B}}$ at the same index resulting from distinct set elements is:

$$X_{\cap=\emptyset} = m \ \times \ (1 - (1 - \frac{1}{m})^{X_{add}(n_{\mathcal{A}})}) \ \times \ (1 - (1 - \frac{1}{m})^{X_{add}(n_{\mathcal{B}})}) \tag{4.20}$$

When we have $Z'_{\mathcal{A},\mathcal{B}}$ common elements inserted in both Bloom filters, we get $X_{\mathcal{A}\cap\mathcal{B}=\emptyset} \simeq Z'_{\mathcal{A},\mathcal{B}} \times n_{key} + X_{\cap=\emptyset}$. Therefore, if Alice takes care that $X_{\cap=\emptyset}$ never gets greater or equal to n_{key}^L, then $\mathcal{S}$ could notice when the two sets have common elements even in the case of $Z'_{\mathcal{A},\mathcal{B}} = 1$. We compare the value of $X_{\cap=\emptyset}$ obtained by (4.20) with the value of $X_{\mathcal{A}\cap\mathcal{B}=\emptyset}$ obtained by running 10^4 times our implementation. We show our results in Table 4.10.

TABLE 4.10. COMPARISON OF THE OVERLAPPING BITS ON THE DISJOINTNESS RELATION BETWEEN TWO DISTINCT SETS AND ITS PREDICTION FROM (4.20).

$n_{\mathcal{A}}$	$n_{\mathcal{B}}$	n_{key}	m	$X_{\cap=\emptyset}$	$X_{\mathcal{A}\cap\mathcal{B}=\emptyset}$
10^2	10^2	10	1×10^3	3.99×10^2	4.00×10^2
10^2	10^2	10	1×10^6	9.99×10^{-1}	1.02×10^0
10^2	10^2	700	1×10^4	9.99×10^3	9.98×10^3
10^2	10^3	700	1×10^7	4.71×10^3	4.72×10^3
10^3	10^3	5	5×10^7	5.00×10^{-1}	5.02×10^{-1}
10^3	10^3	100	2×10^4	1.97×10^4	1.97×10^4
10^3	10^3	1500	1×10^7	1.94×10^5	1.94×10^5
10^3	10^4	1000	1×10^7	6.01×10^5	6.02×10^5
10^4	10^4	1500	1×10^8	1.94×10^6	1.94×10^6

Choosing Parameters Regarding Correctness.

In the classical use of Bloom filters as presented in [Blo70], some usage recommendations are made to generate parameters n_{key} and m:

$$m = -\frac{n_{\mathcal{A}} \times ln\ (\mathcal{P}_{FP})}{(ln\ 2)^2} \tag{4.21}$$

$$n_{key} = \frac{m}{n_{\mathcal{A}}} \times ln\ 2 \tag{4.22}$$

We stress that initially Bloom filters are not supposed to hold such relations testing as *inclusiveness* or *disjointness*. Therefore, the considerations on generation of n_{key} and m are manifold. In Table 4.11 we provide some examples of parameters m and n_{key} such that the false positive probability $\mathcal{P}_{FP}$ of *inclusiveness* relation is acceptable and the expected amount $X_{\cap=\emptyset}$ is significantly smaller than, for instance, $n^L_{key} = 5$.

TABLE 4.11. APPROPRIATE PARAMETERS m AND n_{key} CONSIDERING $\mathcal{P}_{FP}$ AND $X_{\cap=\emptyset}$ WITH FIXED $n_{\mathcal{A}}$ AND $n_{\mathcal{B}}$.

$n_{\mathcal{A}}$	$n_{\mathcal{B}}$	m	n_{key}	$\mathcal{P}_{FP}$	$X_{\cap=\emptyset}$
10^2	5×10^1	10^6	10	9.9×10^{-31}	0.50
10^2	10^3	10^7	10	9.9×10^{-41}	0.99
10^3	10^3	7.62×10^9	1861	~ 0	4.54×10^2
10^4	2×10^2	9.47×10^9	1468	~ 0	4.55×10^2

We notice that in Table 4.11, parameters n_{key} and m are significantly smaller than the ones considered later on, but we recall that we care here solely about correctness of the set relations and not about privacy of the set cardinality.

4.3.4 Privacy Analysis

In this section we show how our solutions fulfill privacy requirements in terms of content and cardinality.

Distribution of the Overlapping Bits.

In this section we analyze the characteristics of *overlapping bits* occurring throughout the basic step of Bloom filters generation. To complete this analysis we run our Bloom filter's implementation for several parameters' configurations 10^3 times each. We start by illustrating in Figure 4.8 the distribution of $Y_{BF_{\mathcal{A}}}$ in two parameters' configurations namely $\{n_{\mathcal{A}} = 50,\ m = 1.3 \times 10^7,\ n_{key} = 200\}$ and $\{n_{\mathcal{A}} = 200,\ m = 1.4 \times 10^7,\ n_{key} = 100\}$. We recall $Y_{BF_{\mathcal{A}}}$ to be the exact amount of *overlapping bits* as presented in (3.16).

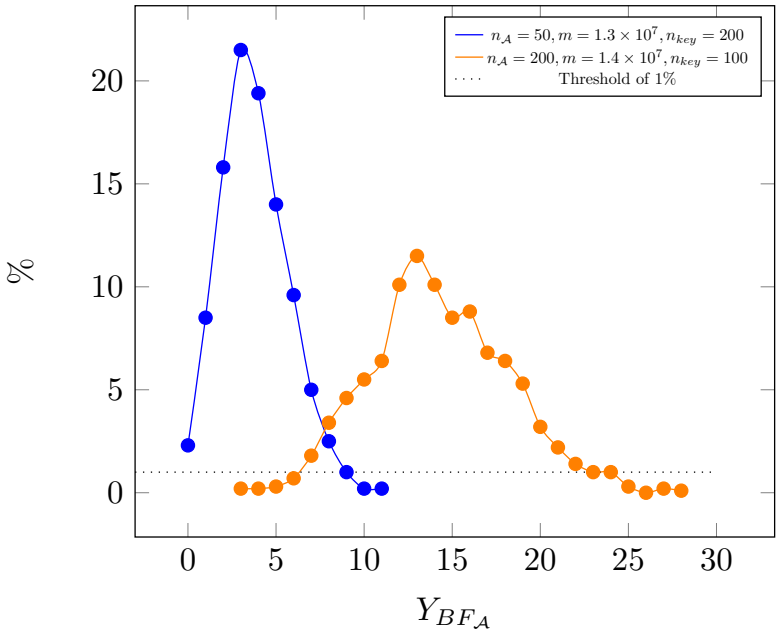

Figure 4.8. Overlapping bits distribution for 10^3 generated Bloom filters of the same set.

In Figure 4.9, we show the distributions for Bloom filters of two sets with similar cardinalities ($n_{\mathcal{A}} \approx n_{\mathcal{B}}$) and the same parameters' configuration. We also show the distribution of the amount of bits in the Bloom filter corresponding to the result of the disjointness operator between the two previous Bloom filters representing two distinct sets. We have $\{n_{\mathcal{A}} = 1000, m = 1.44 \times 10^4, n_{key} = 10\}$ and $\{n_{\mathcal{B}} = 1100, m = 1.44 \times 10^4, n_{key} = 10\}$.

In Figure 4.10 we show a case where the parameters are generated following recommendations from (4.21) and (4.22) for set $\mathcal{B}$ and 100 distinct elements inserted in the Bloom filters. We use the same parameters' configuration with

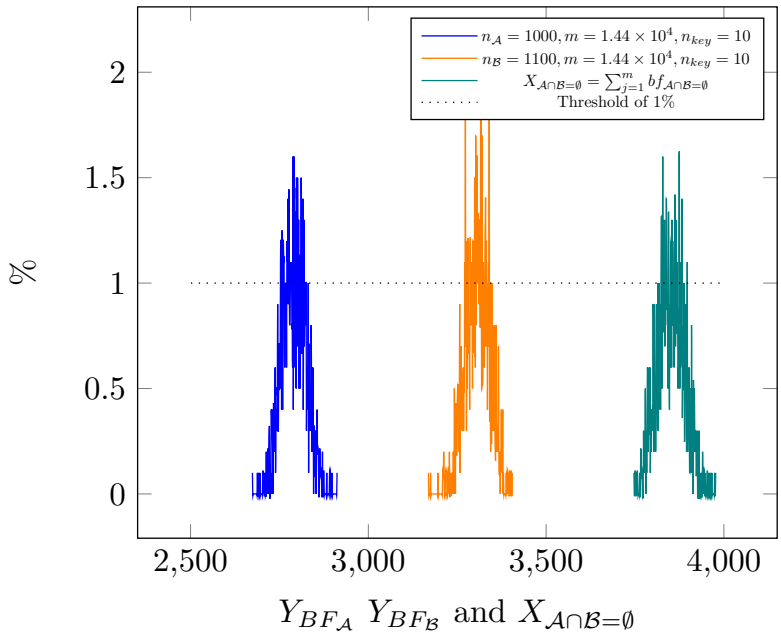

Figure 4.9. Overlapping bits distribution when $n_{\mathcal{A}} \approx n_{\mathcal{B}}$.

set $\mathcal{A}$ containing only 10 distinct elements ($n_{\mathcal{A}} \ll n_{\mathcal{B}}$). We have $\{n_{\mathcal{A}} = 10, m = 1.44 \times 10^3, n_{key} = 10\}$ and $\{n_{\mathcal{B}} = 100, m = 1.44 \times 10^3, n_{key} = 10\}$. As it is illustrated, we obtain such a distribution by running the generation of 10^3 Bloom filters for each parameters' configuration.

From these distributions we noted several characteristics. First, the more elements we add to the Bloom filter, the larger the *overlapping bits* range is. For instance, if we follow recommendations from (4.21) and (4.22), and we insert only 10 elements, we get a range of overlapping bits to approximately 10. When we have 100 inserted elements the range increases to approximately 40. Since our protocols use an HMAC function that generates a uniform random distribution, we could consider that the overlapping bits follow a normal distribution. If it is the case, to consider 99.7% of the possible overlapping bits values for a Bloom filter representing a specific set, one could define them in range $[\mu - 3\sigma; \mu + 3\sigma]$ with μ the mean and σ the standard deviation. Setting parameters in the aim to tune the distribution to get an acceptable overlapping bits range regarding the aimed level of privacy could be performed. In Table 4.12 we see how the standard deviation of the *overlapping bits* distribution varies depending on parameters and the amount of inserted elements. As a second characteristic, we observe that when we have two sets with highly distant cardinalities $n_{\mathcal{A}} \ll n_{\mathcal{B}}$ (or *resp.* $n_{\mathcal{B}} \ll n_{\mathcal{A}}$), the number of overlapping bits in the Bloom filter of the smaller set $Y_{BF_{\mathcal{A}}}$ (*resp.* $Y_{BF_{\mathcal{B}}}$) decreases substantially and that of the larger set increases substantially. Having too few overlapping bits in a Bloom filter could

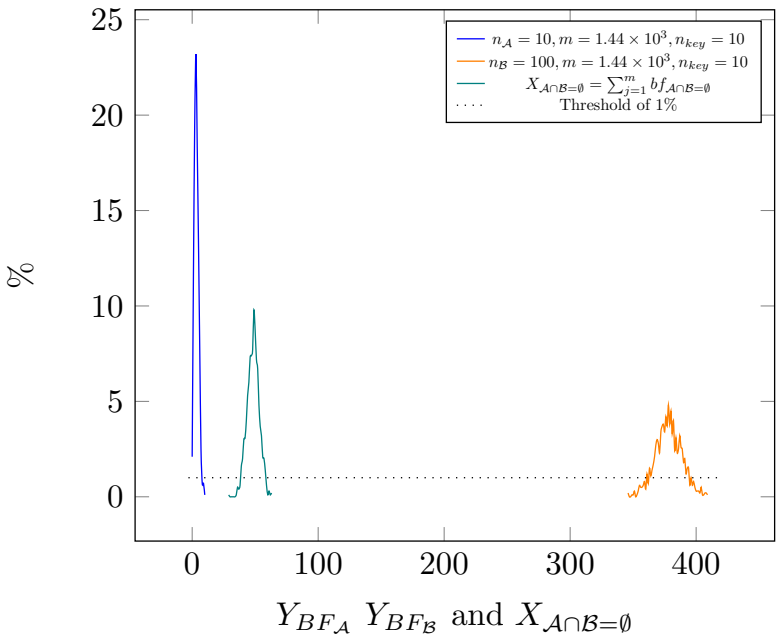

Figure 4.10. Overlapping bits distribution when $n_\mathcal{A} \ll n_\mathcal{B}$.

be problematic, especially if it could be predictable by the attacker. By running tests we notice that no matter which n_{key} is picked or how many elements are inserted in the Bloom filters, if the ratio $\frac{n_\mathcal{A}}{n_\mathcal{B}}$ remains the same, then, firstly the expected amounts of overlapping bits in $BF_\mathcal{A}$ and $BF_\mathcal{B}$ remain approximately the same and secondly $Y_{BF_\mathcal{A}}$ (resp. $Y_{BF_\mathcal{B}}$) is quite small. Moreover, we see that it is even worse if we keep decreasing the ratio $\frac{n_\mathcal{A}}{n_\mathcal{B}}$. One solution to maintain an acceptable range of overlapping bits in the smaller set's Bloom filter, even if we have a significant difference in the cardinalities, could be to use a greater domain $[n_{key}^L; n_{key}^U]$. Indeed, for the same ratio $\frac{n_\mathcal{A}}{n_\mathcal{B}}$, we obtain greater *overlapping bits* ranges.

Content Privacy.

First, we claim that no attacker can determine which concrete elements are included in both of the sets $\mathcal{A}$ and $\mathcal{B}$. Such a characteristic holds by means of the Bloom filter's construction. Indeed, each element of the sets is mapped with the HMAC function. Such a primitive is constructed from a cryptographic hash function and therefore benefits from its one-wayness characteristic. This means, that the only straightforward manner to get any knowledge on Bloom filters' content would be to use the *test()* function which is only computable by Alice and Bob.

Table 4.12. The average and standard deviation of the overlapping bits distribution in Bloom filters $BF_{\mathcal{A}}$ and $BF_{\mathcal{B}}$ for different parameters' configurations.

$n_{\mathcal{A}}$	$n_{\mathcal{B}}$	m	n_{key}	$\mu_{\mathcal{A}}$	$\sigma_{\mathcal{A}}$	$\mu_{\mathcal{B}}$	$\sigma_{\mathcal{B}}$
10	100	1.44×10^3	10	3.3	1.73	278.6	10.3
100	100	2.2×10^7	1700	653.9	24.6	654.9	26.1
10	100	2×10^6	1700	72.3	8.4	7023.4	77.9
100	1000	6×10^7	500	20.8	4.44	2075.8	45.3

Cardinality Privacy.

In this section, we focus on the ability of any attacker to retrieve the cardinality of the sets from one or multiple versions of the Bloom filter's representation of the set. We recall that there exists an optimized manner to get the cardinality of a set from its Bloom filter representation as explained in Section 4.3.1, the *S&B technique*. Without any *overlapping bit*, getting the result is therefore straightforward. On the contrary, having multiple overlapping bits will lead any non-authorized party to misinterpret the cardinality. To ensure this, the ratio of the amount of *overlapping bits* over parameter n_{key} should be large.

Another aspect to consider with such an approach is that anytime an attacker gets a different Bloom filter of the same set $\mathcal{A}$, he can get closer to its cardinality. Indeed, the cardinality of $\mathcal{A}$ will always be greater or equal to $\frac{X_{BF_{\mathcal{A}}}}{n_{key}}$. By accessing $X_{BF_{\mathcal{A}}}$ from several different Bloom filters of the same set $\mathcal{A}$, he therefore can refine his cardinality guessing.

We also notice that having an acceptable probability of false negative and an acceptable level of privacy for set cardinalities are contradictory strategies. Indeed, our approach to solve the *disjointness* set relation is based on reducing the amount of *overlapping bits* to avoid confusion on having common elements.

Sets Cardinalities Attack.

We present here a possible attack performed by any external attacker or even the server $\mathcal{S}$, that aims to retrieve cardinalities $n_{\mathcal{A}}$ and $n_{\mathcal{B}}$. To do so, $\mathcal{S}$ will firstly try to determine parameter n_{key} used by Alice and Bob. $\mathcal{S}$ knows that $n_{key} \in [n_{key}^L; n_{key}^U]$ and that n_{key} is a factor of the amount of bits inserted in both Bloom filters. From these pieces of information and by observing the Bloom filters, $\mathcal{S}$ will start by generating candidate lists. The candidates list for n_{key} is represented as $L_{n_{key}} = \{l_1, \ldots, l_{\lambda_{n_{key}}}\}$ with $\lambda_{n_{key}}$ a security parameter which represents the size of this list. $\mathcal{S}$ also considers two sub-lists $L^{\mathcal{A}}_{n_{key}} = \{l_1, \ldots, l_{\lambda_{\mathcal{A}}}\}$ and $L^{\mathcal{B}}_{n_{key}} = \{l_1, \ldots, l_{\lambda_{\mathcal{B}}}\}$ which correspond to the lists of factors regarding $BF_{\mathcal{A}}$ and $BF_{\mathcal{B}}$ before the cross-checking which leads to $L_{n_{key}}$. Finally we have $Y_{BF_{\mathcal{A}}} \in [ob_{\mathcal{A}_1}; ob_{\mathcal{A}_2}]$ and $Y_{BF_{\mathcal{B}}} \in [ob_{\mathcal{B}_1}; ob_{\mathcal{B}_2}]$ amounts of *overlapping bits* in the Bloom filters.

In each Bloom filter, some *overlapping bits* could have occurred, therefore the attacker knows that regarding $BF_{\mathcal{A}}$, n_{key} is a factor of $X_{BF_{\mathcal{A}}}$ or $(X_{BF_{\mathcal{A}}}+1)$ or $(X_{BF_{\mathcal{A}}}+2)\dots$ Similarly holds for $BF_{\mathcal{B}}$. It means that $L^{\mathcal{A}}_{n_{key}}$ (resp. $L^{\mathcal{B}}_{n_{key}}$) is composed by elements l_j which verify the two following characteristics:

$$l_j \in [n^L_{key}; n^U_{key}] \text{ and } l_j | x_{\mathcal{A}} \text{ with } x_{\mathcal{A}} \in [X_{BF_{\mathcal{A}}} + ob_{\mathcal{A}_1}; X_{BF_{\mathcal{A}}} + ob_{\mathcal{A}_2}]$$
$$(\text{resp. } l_j | x_{\mathcal{B}} \text{ with } x_{\mathcal{B}} \in [X_{BF_{\mathcal{B}}} + ob_{\mathcal{B}_1}; X_{BF_{\mathcal{B}}} + ob_{\mathcal{B}_2}])$$

Finally, we get $L_{n_{\mathcal{A}}} = (l_i)_{i\in[1;\lambda_{n_{\mathcal{A}}}]}$ the list of candidates for $n_{\mathcal{A}}$ with $\lambda_{n_{\mathcal{A}}}$ the amount of elements in $L_{n_{\mathcal{A}}}$. Similarly we have $L_{n_{\mathcal{B}}} = (l_i)_{i\in[1;\lambda_{n_{\mathcal{B}}}]}$ the list of candidates for $n_{\mathcal{B}}$ with $\lambda_{n_{\mathcal{B}}}$ the amount of elements in $L_{n_{\mathcal{B}}}$.

The first step of the attack consists of listing all the common factor of $\{X_{BF_{\mathcal{A}}}, (X_{BF_{\mathcal{A}}}+1), (X_{BF_{\mathcal{A}}}+2), \dots\}$ and $\{X_{BF_{\mathcal{B}}}, (X_{BF_{\mathcal{B}}}+1), (X_{BF_{\mathcal{B}}}+2), \dots\}$ to generate lists $L^{\mathcal{A}}_{n_{key}}$ and $L^{\mathcal{B}}_{n_{key}}$. Then, $\mathcal{S}$ will intersect the two lists to generate the candidates list $L_{n_{key}}$.

The second phase of the attack is to translate $L_{n_{key}}$ into lists $L_{n_{\mathcal{A}}}$ and $L_{n_{\mathcal{B}}}$. $\mathcal{S}$ could use the *S&B technique* [SB07] to approximate size $n_{\mathcal{A}}$. Since parameter m is public and value $X_{BF_{\mathcal{A}}}$ is directly computable, we have the following function:

$$n^*_{\mathcal{A}}(n_{key}) = -\frac{m}{n_{key}} ln\Big[1 - \frac{X_{BF_{\mathcal{A}}}}{m}\Big] \tag{4.23}$$

When we look closely at list $L_{n_{key}}$, we notice that if some elements are following each other, they are translated to the same candidate of $n_{\mathcal{A}}$. In other words, multiple elements from $L_{n_{key}}$ correspond to the same element from $L_{n_{\mathcal{A}}}$. Thus we get $\lambda_{n_{\mathcal{A}}} \leqslant \lambda_{n_{key}}$.

Attack on Three Cases. We have implemented the *sets cardinalities attack* and we run it with different levels of knowledge for the attacker. We differentiate three cases namely the *best case*, the *in-between case* and the *worst case*. They could be seen as three attacker models where level of knowledge is minimal in the *best case* and maximal in the *worst case*.

Best Case. $\mathcal{S}$ knows nothing about the expected *overlapping bits* distribution. It considers that any amounts of *overlapping bits* could occur and thus the attack will not result in any valuable information on the sets' cardinality. Indeed, the attack will return $\lambda_{\mathcal{A}} = \lambda_{\mathcal{B}} = \lambda_{n_{key}} = (n^U_{key} - n^L_{key})$. In such ideal case we get a perfect privacy on the sets cardinality.

In-Between Case. In such a case we consider that $\mathcal{S}$ has more knowledge about the *overlapping bits* that affect the Bloom filters. We assume that $\mathcal{S}$ is not able to predict the exact *overlapping bits* distribution but could approximately estimate its ranges $[ob_{\mathcal{A}_1}; ob_{\mathcal{A}_2}]$ and $[ob_{\mathcal{B}_1}; ob_{\mathcal{B}_2}]$. $\mathcal{S}$ will perform the attack with these maximized default ranges.

Worst Case. In the worst case assumption, $\mathcal{S}$ knows the exact overlapping bits distribution for both Bloom filters. With this knowledge, $\mathcal{S}$ could associate a weight to any elements from $L_{n_{key}}$ and *de facto* elements from $L_{n_\mathcal{A}}$ and $L_{n_\mathcal{B}}$. $\mathcal{S}$ sets weight for each element from $L_{n_{key}}$ based on the *overlapping bits* distribution of each Bloom filter.

Example of an Attack in the Three Cases. We give a toy example of the attack obtained with our implementation of the protocol. The parameters selected by Alice and Bob are the following: $n_{key} = 17$ with $n^L_{key} = 10$ and $n^U_{key} = 30$, $m = 1.5 \times 10^8$, $n_\mathcal{A} = 1024$ and $n_\mathcal{B} = 2305$.

$\mathcal{S}$ gets $BF_\mathcal{A}$ and $BF_\mathcal{B}$ and counts their respective bits set to 1; $X_{BF_\mathcal{A}} = 17407$ and $X_{BF_\mathcal{B}} = 39179$. It is then listing factors as described in Section 4.3.4 by considering *overlapping bits* amounts as $Y_{BF_A}, Y_{BF_B} \in [0; 10]$.

Table 4.13. Results of the set cardinality attack

Y_{BF_A}/Y_{BF_B}	$X_{BF_\mathcal{A}} + Y_{BF_A}$	$L^\mathcal{A}_{n_{key}}$	$X_{BF_\mathcal{B}} + Y_{BF_B}$	$L^\mathcal{B}_{n_{key}}$
0	1707	13	39179	29
1	1708	16, 17	39180	10, 12, 15, 20, 30
2	1709	21	39181	$\emptyset$
3	1710	10	39182	11, 13, 22, 26
4	1711	23	39183	$\emptyset$
5	1712	12	39184	16
6	1713	11	39185	17
7	1714	$\emptyset$	39186	14, 18, 21
8	1715	15, 27	39187	$\emptyset$
9	1716	14, 28	39188	$\emptyset$
10	1717	$\emptyset$	39189	$\emptyset$

$\mathcal{S}$ selects the common factors from $L^\mathcal{A}_{n_{key}}$ and $L^\mathcal{B}_{n_{key}}$ and narrows the candidates list to $L_{n_{key}} = \{10, 11, 12, 13, 14, 15, 16, 17, 21, 23\}$. We note that $\mathcal{S}$ has reduced the candidates list by more than half. Indeed, before the attack we had $\lambda_{n_{key}} = 21$ and now $\lambda_{n_{key}} = 10$.

Then the attacker translates $L_{n_{key}}$ into $L_{n_\mathcal{A}}$ and $L_{n_\mathcal{B}}$ with the *(S&B) technique*:

$$L_{n_\mathcal{A}} = \{757, 829, 1024, 1088, 1161, 1243, 1339, 1451, 1583, 1741\}$$
$$L_{n_\mathcal{B}} = \{1704, 1866, 2305, 2449, 2612, 2799, 3014, 3265, 3562, 3918\}$$

We give more examples of the attack in particular for each of the three cases considered as attacker model.

Best Case. $\mathcal{S}$ could only translate the information that n_{key} is any integer and $n_{key} \in [n^L_{key}; n^U_{key}]$. Therefore, $\mathcal{S}$ reduces the candidates list to $L_{n_\mathcal{A}} = [\lceil n^*_\mathcal{A}(n^U_{key})\rceil; \lfloor n^*_\mathcal{A}(n^L_{key})\rfloor]$ and gets its cardinality as $\lambda_{n_\mathcal{A}} = \lfloor n^*_\mathcal{A}(n^L_{key})\rfloor -$

$\lceil n^*_{\mathcal{A}}(n^U_{key})\rceil + 1$ since in this case the attacker could not yet exclude any value from $L_{n_\mathcal{A}}$. We notice that in some cases, $\lambda_{n_\mathcal{A}}$ could be very small even without performing the attack. It is the case especially when we have a quite small $n_\mathcal{A}$. In Table 4.14 we give some examples of $L_{n_\mathcal{A}}$ and $\lambda_{n_\mathcal{A}}$ without any attack.

TABLE 4.14. RESPECTIVE $L_{n_\mathcal{A}}$ AND $\lambda_{n_\mathcal{A}}$ FOR SEVERAL PARAMETER' CONFIGURATIONS.

$n_\mathcal{A}$	m	n^L_{key}	n^U_{key}	$X_{BF_\mathcal{A}}$	$L_{n_\mathcal{A}}$	$\lambda_{n_\mathcal{A}}$
10	2×10^6	1500	2000	16935	$[9; 11]$	3
100	2×10^6	1500	2000	162968	$[85; 113]$	29
100	6×10^7	500	1000	49981	$[51; 100]$	50
1000	6×10^7	500	1000	497936	$[501; 1000]$	500

In-Between Case. In Table 4.15 we show the results of the attack with such a limited knowledge.

TABLE 4.15. RESULTS OF THE ATTACK.

$n_\mathcal{A}$	$n_\mathcal{B}$	m	n_{key}	$[ob_{\mathcal{A}_1}; ob_{\mathcal{A}_2}]$	$[ob_{\mathcal{B}_1}; ob_{\mathcal{B}_2}]$	$\lambda_\mathcal{A}$	$\lambda_\mathcal{B}$	$\lambda_{n_{key}}$	$\lambda_{n_\mathcal{A}}$	$\lambda_{n_\mathcal{B}}$
100	1000	6×10^7	500	[0 - 50]	[1500 - 2500]	38	315	38	38	38
10	100	2×10^6	1700	[0 - 150]	[6000 - 8000]	46	425	46	3	3
100	100	2.2×10^7	1700	[500- 1000]	[500 - 1000]	142	140	136	29	29

Worst Case. For this attacker model, we give examples with three different parameters' configurations. In each case, we run 10^3 Bloom filter generations of each specific set to obtain a precise *overlapping bits* distribution. Then, we provide all the retrieved candidates for parameter n_{key} and how they are translated into $n_\mathcal{A}$ and $n_\mathcal{B}$ candidates along with their respective weights. In particular we set a weight of 1.00 for the most probable candidate for parameter n_{key} and then, we set the other ones proportionately to this one.

We start by a parameters' configuration of $\{n_\mathcal{A} = 100,\ n_\mathcal{B} = 1000,\ m = 6 \times 10^7,\ n_{key} = 500,\ n^L_{key} = 500,\ n^U_{key} = 1000\}$. Before the attack $\mathcal{S}$ knows that $L_{n_\mathcal{A}} = [50; 100]$ and $L_{n_\mathcal{B}} = [500; 1000]$ and therefore deduces that $\lambda_{n_\mathcal{A}} = 51$ and $\lambda_{n_\mathcal{B}} = 501$. In Table 4.16 we show the results of the attack obtained by running our implementation with *overlapping bits* distribution obtained by generating 10^3 Bloom filters of two sets. We see in Table 4.16 that $\mathcal{S}$ gets greater weights for candidates which are not the correct ones. In particular we have six couples of candidates with an equal or greater probability than the correct couple $(n_\mathcal{A}, n_\mathcal{B})$.

The second selected parameters' configuration is $\{n_\mathcal{A} = 10,\ n_\mathcal{B} = 100,\ m = 2 \times 10^6,\ n_{key} = 1700,\ n^L_{key} = 1500$ and $n^U_{key} = 2000\}$. Before the attack, $\mathcal{S}$ knows that $L_{n_\mathcal{A}} = [8; 11]$and $L_{n_\mathcal{B}} = [85; 113]$ and therefore deduces that $\lambda_{n_\mathcal{A}} = 3$ and $\lambda_{n_\mathcal{B}} = 29$. In Table 4.17 we show the attack's results. With the attack, we get

TABLE 4.16. EXAMPLE OF TRANSLATING $L_{n_{key}}$ INTO $L_{n_\mathcal{A}}$ AND $L_{n_\mathcal{B}}$ FOR THE SETS CARDINALITIES ATTACK.

Element from $L_{n_{key}}$	Weight	Element from $L_{n_\mathcal{A}}$ and $L_{n_\mathcal{B}}$	Weight
500	**0.67**	**100 and 1000**	**0.67**
505	1.00	99 and 990	1.00
532	0.02	94 and 940	0.02
625	0.67	80 and 800	0.67
633	0.02	79 and 790	0.02
641	0.94	78 and 780	0.94
658	0.02	76 and 760	0.02
685	0.14	73 and 730	0.14
862	0.85	58 and 580	0.85
877	0.43	57 and 570	0.43
893	0.02	56 and 560	0.02
909	1.00	55 and 550	1.00
926	0.17	54 and 540	0.17
1000	0.67	50 and 500	0.67

TABLE 4.17. EXAMPLE OF TRANSLATING $L_{n_{key}}$ INTO $L_{n_\mathcal{A}}$ AND $L_{n_\mathcal{B}}$ FOR THE SETS CARDINALITIES ATTACK.

Element from $L_{n_{key}}$	Weight	Element from $L_{n_\mathcal{A}}$ and $L_{n_\mathcal{B}}$	Weight
1545, 1546, 1547	0.10, 0.53, 1.00	110 and 11	1.63
1700, 1701, 1702	**0.16, 0.71, 0.84**	**100 and 10**	**1.71**
1889, 1890	0.33, 0.71	90 and 9	1.04

$L_{n_\mathcal{A}} = \{9, 10, 11\}$ and $L_{n_\mathcal{B}} = \{90, 100, 110\}$ with $\lambda_{n_\mathcal{A}} = \lambda_{n_\mathcal{B}} = 3$. We also get the greater weight for the correct candidates.

We give a last example where $n_\mathcal{A} \approx n_\mathcal{B}$. The selected parameters are $\{n_\mathcal{A} = 100,\ n_\mathcal{B} = 100,\ m = 2.2 \times 10^7,\ n_{key} = 1700,\ n^L_{key} = 1500,\ n^U_{key} = 2000\}$. Before the attack $\mathcal{S}$ knows that $L_{n_\mathcal{A}} = \lambda_{n_\mathcal{A}} = [85; 113]$ and $\lambda = \lambda_{n_\mathcal{B}} = 29$. In Table 4.18 we show the resulting $L_{n_\mathcal{A}}$ and $L_{n_\mathcal{B}}$.

Finally we get $L_{n_\mathcal{A}} = L_{n_\mathcal{B}} = [85; 113]$ and $\lambda_{n_\mathcal{A}} = \lambda_{n_\mathcal{B}} = 29$, namely the same lists than before performing the attack. $\mathcal{S}$ gets information on whether one candidate is more probable than another regarding the attached weight. We notice in this example that the correct value for $n_\mathcal{A}$ and $n_\mathcal{B}$ (100) is not the most probable, and even $\mathcal{S}$ gets eight candidates with an higher or equivalent probability of being the correct size. We see then that in some cases, the attack allows the attacker to reduce its lists $L_{n_\mathcal{A}}$ and $L_{n_\mathcal{B}}$ but in other cases it only provides approximation on the probability of being the actual size.

We have shown in this section, depending on the considered security model, how much information any attackers could gain on the sets' cardinality. We have seen that increasing domains of parameters n_{key} and m could allow to

TABLE 4.18. EXAMPLE OF TRANSLATING $L_{n_{key}}$ INTO $L_{n_{\mathcal{A}}}$ AND $L_{n_{\mathcal{B}}}$ FOR THE SETS CARDINALITIES ATTACK.

Element from $L_{n_{key}}$	Weight	Element from $L_{n_{\mathcal{A}}}$ and $L_{n_{\mathcal{B}}}$	Weight
1504	0.46	113	0.46
1518	0.61	112	0.61
1531	0.21	111	0.21
1545	0.54	110	0.54
1560	0.18	109	0.18
1574	0.93	108	0.93
1589	0.43	107	0.43
1604	0.64	106	0.64
1619	0.75	105	0.75
1634, 1635	0.18	104	0.36
1650	0.54	103	0.54
1666	0.21	102	0.21
1683	0.68	101	0.68
1700	**0.71**	**100**	**0.71**
1717	0.68	99	0.68
1734, 1735	0.21	98	0.42
1752	0.29	97	0.29
1771	0.61	96	0.61
1789	0.46	95	0.46
1808	0.46	94	0.46
1828	0.93	93	0.93
1848	0.61	92	0.61
1868	0.96	91	0.96
1889	0.71	90	0.71
1910	1.00	89	1.00
1932	0.61	88	0.61
1954	0.82	87	0.82
1976	0.18	86	0.18
2000	0.71	85	0.71

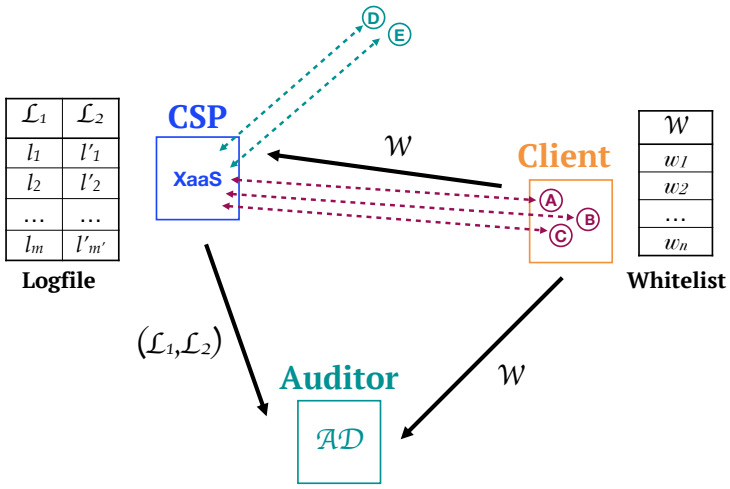

Figure 4.11. Generic framework of the cloud security auditing. Users A, B and C should be authorized to connect by the $\mathcal{CSP}$ while users D and E should not.

reduce such information. As we will see now, by applying this solution to our use cases, we recall that the most predominant privacy consideration is on the content of the sets. Preserving the cardinality's privacy is therefore secondary and providing a certain amount of this particular privacy could be considered as sufficient.

4.3.5 On Cloud Security Auditing

We show here how the *inclusiveness* and *disjointness* protocols could be used to solve the first presented use case; Use Case1 - Cloud Security Auditing.

Protocol

For simplicity of presentation, we present our protocols with only one client involved. We remark that one can easily adapt it to a use case with multiple clients using the same cloud provider. We also emphasize the fact that, to process the two set relations, the considered Bloom filters should be similarly generated, namely with the same size m, keyed hash function and set of keys K. First we recall the two privacy enhancements from tuning the classical use of Bloom filter. Then we present the *inclusiveness* and *disjointness* set relations before explaining how the parameters should be selected to guarantee a certain level of correctness on these two relations.

We recall use case 1 framework in Figure 4.11. Firstly, $\mathcal{C}$ performs function *Setup()* and generates the Bloom filter parameters: the dimension m, the HMAC function h, the amount of keys n_{key} and the set of keys $K = \{\kappa_1, \ldots, \kappa_{n_{key}}\}$.

Then, by the use of *Create()* function, $\mathcal{C}$ (resp. $\mathcal{CSP}$) generates the Bloom filter of its data $\mathcal{W} = \{w_1, \ldots, w_{n_\mathcal{W}}\}$ (resp. $\mathcal{L}_1 = \{l_1, \ldots, l_{n_{\mathcal{L}_1}}\}$ and $\mathcal{L}_2 = \{l'_1, \ldots, l'_{n_{\mathcal{L}_2}}\}$).

$$\begin{aligned} BF_\mathcal{W} &= BF\big(\mathcal{W}, (h_\kappa)_{\kappa \in K}\big) = bf_\mathcal{W}[j]_{1 \leqslant j \leqslant m} \\ BF_{\mathcal{L}_1} &= BF\big(\mathcal{L}_1, (h_\kappa)_{\kappa \in K}\big) = bf_{\mathcal{L}_1}[j]_{1 \leqslant j \leqslant m} \\ BF_{\mathcal{L}_2} &= BF\big(\mathcal{L}_2, (h_\kappa)_{\kappa \in K}\big) = bf_{\mathcal{L}_2}[j]_{1 \leqslant j \leqslant m} \end{aligned}$$

$\mathcal{AD}$ computes the *inclusiveness* protocol on the two respective Bloom filters of sets $\mathcal{L}_1$ and $\mathcal{W}$ to test if $\mathcal{L}_1 \subseteq \mathcal{W}$, namely if all the authorized connections have been made from authorized IP addresses:

$$INC(BF_{\mathcal{L}_1}, BF_\mathcal{W}) = BF_{\mathcal{L}_1 \subseteq \mathcal{W}} = bf_{\mathcal{L}_1 \subseteq \mathcal{W}}[j]_{1 \leqslant j \leqslant m}$$

Then $\mathcal{AD}$ expresses $X_{\mathcal{L}_1 \subseteq \mathcal{W}}$ which corresponds to the number of bits set to 1 in the resulting Bloom filter:

$$X_{\mathcal{L}_1 \subseteq \mathcal{W}} = \sum_{j=1}^{m} bf_{\mathcal{L}_1 \subseteq \mathcal{W}}[j] \tag{4.24}$$

$\mathcal{AD}$ tests if $X_{\mathcal{L}_1 \subseteq \mathcal{W}} = m$ and can conclude that $\mathcal{L}_1 \subseteq \mathcal{W}$ if no false positive occurred. Otherwise we have $\mathcal{L}_1 \nsubseteq \mathcal{W}$ with certainty.

To verify that no authorized user failed to connect to the service offered by $\mathcal{CSP}$, $\mathcal{AD}$ performs the disjointness relation on the respective Bloom filters of $\mathcal{W}$ and $\mathcal{L}_2$:

$$DIS(BF_\mathcal{W}, BF_{\mathcal{L}_2}) = BF_{\mathcal{W} \cap \mathcal{L}_2 = \emptyset} = \ bf_{\mathcal{W} \cap \mathcal{L}_2 = \emptyset}[j]_{1 \leqslant j \leqslant m}$$

Then $\mathcal{AD}$ expresses $X_{\mathcal{W} \cap \mathcal{L}_2 = \emptyset}$ which corresponds to the number of bits set to 1 in the resulting Bloom filter:

$$X_{\mathcal{W} \cap \mathcal{L}_2 = \emptyset} = \sum_{j=1}^{m} bf_{\mathcal{W} \cap \mathcal{L}_2 = \emptyset}[j] \tag{4.25}$$

$\mathcal{AD}$ compares it such that:

if $X_{\mathcal{W} \cap \mathcal{L}_2 = \emptyset} < n^L_{key}$ then $\mathcal{W}$ and $\mathcal{L}_2$ are distinct

if $X_{\mathcal{W} \cap \mathcal{L}_2 = \emptyset} \geqslant n^L_{key}$ then $\mathcal{W}$ and $\mathcal{L}_2$ have at least one element in common

By referring to the security requirements presented in Section 2.1.2 we could outline that *SR1* and *SR2* are fulfilled. Indeed:

SR1. If $\mathcal{AD}$ does not know the HMAC's keys $K = \{k_1, \ldots, k_{n_{key}}\}$, it cannot generate its own Bloom filter or add any element to an existing one and perform the set relations. Indeed, using HMAC function requires that all the considered Bloom filters are generated with the same keys.

SR2. These two sub-requirements are fulfilled thanks to the Bloom filter inherent characteristics. Indeed, the first one holds because of the fact that all elements inserted in a Bloom filter are mixed together and it is not possible, even from the same Bloom filter, to distinguish between them. The second sub-requirement holds since when the whitelist Bloom filter is created, even if an element occurs multiple times in the logfile, it will be added to the respective Bloom filter only once.

We show as well that our solution fulfills the third security requirement *SR3*:

SR3. The overlapping bits property of the Bloom filters allows to hide the exact number of elements in the whitelist or the logfile. However, $\mathcal{AD}$ is able to determine the amount of bits set to 1 in the Bloom filters. It could then deduct the following information: $n_{\mathcal{A}} \geqslant \frac{X_{BF_{\mathcal{A}}}}{n_{key}}$. By keeping parameter n_{key} secret to $\mathcal{AD}$, we consider the cardinalities obfuscated and SR3 could still be considered as fulfilled to a certain level.

Evaluation and Results

We have implemented our protocols of the set relations in Java. The following measurements have been made with CPU configuration of Intel Core i5 M520 2.40GHz × 4.

Results on the Cloud Auditing Use Case. First of all, running the implementation allowed us to express a precise value of the false positive (resp. false negative) rate. We test our solution with parameters suiting the cloud security auditing use case meaning a whitelist of 10^3 or 10^4 elements and logfiles from 10^2 to 10^4 elements. To test the false positive case, we generate Bloom filters of a whitelist $\mathcal{W}$ and a logfile $\mathcal{L}_1$ with different amounts of IP addresses. The tested parameters configurations are displayed in Table 4.19. Every IP addresses inserted in $BF_{\mathcal{L}_1}$ are also inserted in $BF_{\mathcal{W}}$ except for one. For every parameters configuration we test the *inclusiveness* relation $INC(\mathcal{L}_1, \mathcal{W})$ 10^3 times. We performed the same experimental protocol with the *disjointness* relation. In both cases we obtained 0.00% of false positives or false negatives.

Performance. In Table 4.19 we show the performance of our two set relations. We run 10^4 times each set operator for more accuracy and the computation times are expressed in seconds. We see that performance times decrease linearly depending on parameter m, indeed as presented, the set relations are equivalent to bit-wise operations on the Bloom filters. That being said, we also notice that the performance times considering the set cardinality privacy are very acceptable especially in an auditing use case.

TABLE 4.19. RUNNING TIMES OF THE TWO SET RELATIONS IN SECONDS.

$n_{\mathcal{W}}$	$n_{\mathcal{L}_1}$	$n_{\mathcal{L}_2}$	m	n_{key}	n_{key}^L	n_{key}^U	Time for $INC(\mathcal{L}_1, \mathcal{W})$	Time for $DIS(\mathcal{L}_2, \mathcal{W})$
10^3	10^3	10^3	1.18×10^9	733	500	2000	2.57×10^{-1}	2.16×10^{-1}
10^3	10^3	10^3	7.62×10^9	1861	500	2000	2.51×10^{-1}	7.44×10^{-1}
10^4	9×10^3	2×10^2	2.93×10^9	816	500	2000	2.29×10^{-1}	2.80×10^{-1}
10^4	9×10^3	2×10^2	9.47×10^9	1468	500	2000	2.16×10^{-1}	8.67×10^{-1}

4.3.6 On Retrospective Tracking of Suspects in GDPR Conform Mobile Access Networks Datasets

In this section we show in detail how the proposed disjointness protocol could be applied to the aforementioned use case 2. We recall that the three sub-use cases are dealing with mobile tracking of suspects with the objective of a privacy friendly handling of all unsuspected mobile user's personal geo-location information. Finally, we present results from our implementation of the third scenario and evaluate its correctness and privacy aspects.

In the following sections we refer to the Bloom filter's function *Setup()*, *Create()* and *DIS()* that we introduced in Section 4.3.2.

Protocols

Tracking of Suspect Entities on Telco Datasets Ensuring Privacy for Unsuspected Users. First of all, the mobile telecommunication provider initiates the Bloom filter's parameters by processing the *Setup()* function. For each of its access points, the mobile telecommunication provider generates a Bloom filter to which they add the relevant information (TMSI/IMSI) for any connection at the harvesting time. In other words, each of the Bloom filters represent the list of the devices that connected themselves to the respective access points during a certain period of time.

When the government agency requests to intersect the access points' access logfiles with its whitelist of suspected entities, the respective telco provider shares the Bloom filter's parameters (size m and keys K) with the agency. Then, the agency generates the Bloom filter corresponding to its whitelist with the relevant parameters by means of the *Create()* function.

After receiving all the Bloom filters, the 3$^{\text{rd}}$ party (e.g. Interpol or BKA in Germany, responsible for wiretapping, or similar players in other countries) performs the disjointness function *DIS()* between the whitelist's Bloom filter and each access point's Bloom filters. Every time the function returns that sets are not disjoint, the 3$^{\text{rd}}$ party could notify the agency that at least one element from the whitelist has been connected to a certain access point. On the contrary, if the *DIS()* function returns a positive outcome, it means the sets are disjoint and therefore no suspect entity from the agency's whitelist has been connected to any access point during the specific period.

Finally, in case of a match, the government agency due to its legal exception, could request the involved access point logfile directly from the telco.

Tracking of Suspect Entities on WLAN Network Providers Datasets Ensuring Privacy for Unsuspected Users. Similarly to the first use case, the WLAN administrator initiates the Bloom filter's parameters by processing the *Setup()* function. The administrator of the WAPs generates a Bloom filter in which they add the MAC address of each device that either connected or used the active mode to detect the nearest WLAN access point.

When the government agency requests to intersect the WAPs' logfiles with its whitelist, the WLAN administrator shares the Bloom filter's parameters with the agency. Then, the agency generates a Bloom filter of its whitelist with the relevant parameters by means of the *Create()* function.

After receiving all the Bloom filters, the 3$^{\text{rd}}$ party performs the *DIS()* function between the whitelist's Bloom filter and each WAPs' Bloom filters. Any time the function returns that the sets are not disjoint, the 3$^{\text{rd}}$ party could notify the agency that at least one suspect from the whitelist has connected to a certain access point.

Finally, in case of a match, the agency could request the access point logfile directly to the WLAN administrator.

Tracking of Suspect Entities on Mobile OS Providers Datasets Ensuring Privacy for Unsuspected Users. In our last use case, we face two types of logfiles. A local logfile generated by each mobile device and a master logfile which aggregates all the local logfiles coming from the same mobile operating system company. At each time t_j, the user's mobile device (smartphone, tablet or similar embedded devices equipped with Android or iOS) generates a personal logfile composed by tuples $(RSSI_i/WAP_i)$ with $i \in [1; n]$ which represent the mobile device's distances from the n wireless access points let it be $WAPs$ or BSs. The bunch of these data indicates the exact location of the phone's user at any time t_j. After retrieving all the logfiles from the users' smartphones, the operating system company creates the master logfile as follows. For each WAP_i, it creates a Bloom filter BF_i with the *Create()* function and adds a user's identification in form of its MAC address to it if there exists a tuple $(RSSI_i/WAP_i)$ where $RSSI_i < l$ in its respective personal logfile. Variable l represents a certain limit to which we could assume that the user has been in proximity to a certain WAP. In other words, a mobile user is added to a WAP's Bloom filter if he has been nearby this wireless access point at a certain time. We show in Figure 4.12 the generation of the master logfile from two personal logfiles located at different smartphones.

Finally, the 3$^{\text{rd}}$ party receives reveal parts of the master logfile and the whitelist's Bloom filter and performs the *DIS()* function to determine if one of the elements from the whitelist has been in transmission range of one of the wireless access points. Please note that the privacy extension of the 3$^{\text{rd}}$ use case may be particularly helpful when suspects shall be identified on European ground and Apple or Google are willing to support EU-GDPR.

In Algorithm 7 we show how the inherent system of the modern mobile_OS already integrates the geo-location harvesting approach at each user j's device for $j \in [1; n']$.

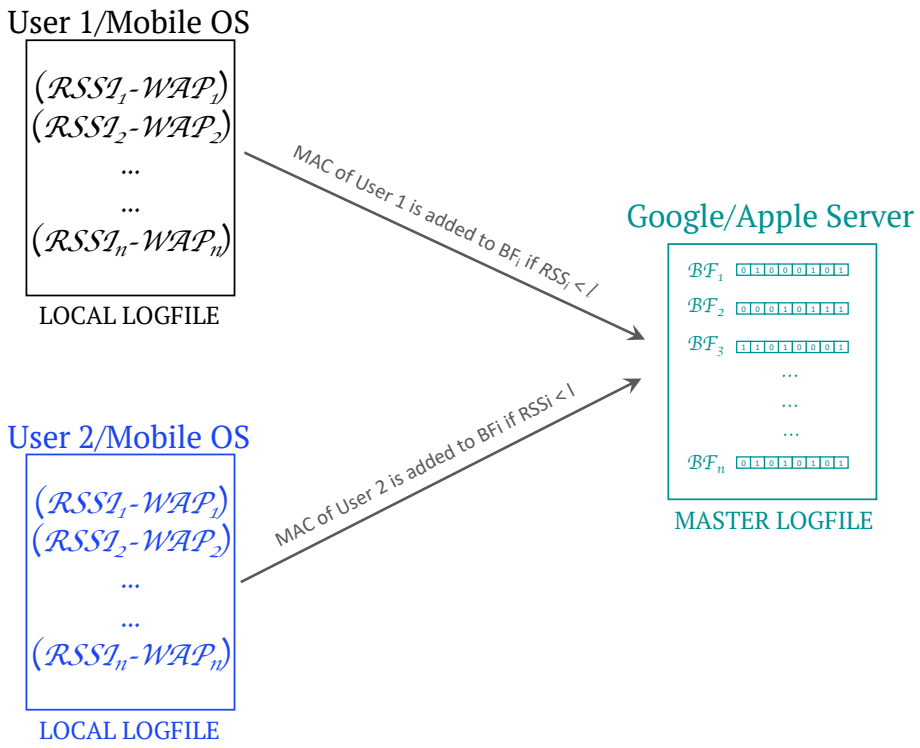

Figure 4.12. Generation of the master logfile from two smartphone users' logfiles.

Algorithm 7 Local logfile generation by the mobile OS geo-location harvesting feature

for all t_x **do**
  $Create\_file(F_x^j)$ with
  for $i \in [1; n]$ **do**
    $F_x^j.add((RSSI_i; WAP_i))$
  end for
  Send F_x^j to its respective Mobile OS company
end for

In Algorithms 8 to 10 we provide the pseudo-code performed by the three active parties of the protocol. We remark that although the parameters' generation could takes place at the government agency's side, we decided to place it at the OS company's side to be more adjustable to the relevant sizes of the larger sets or access logfiles. In such a way, it makes more sense to have the company adjusting the parameters to its logfile and estimating the whitelist's size with a standard value rather than the opposite.

We consider all the communication channels between the parties, and especially when transmitting the parameters, secured e.g. out-of-band signaling, TLS.

Algorithm 8 Parameters initialization and BF generation by the mobile OS company's data center

$\mathbf{h}, \mathbf{n_{key}}, \mathbf{m}, \mathbf{K} \leftarrow$Setup
Send $\mathbf{h}, \mathbf{n_{key}}, \mathbf{m}, \mathbf{K}$ to the Government Agency
for $i \in [1; n]$ **do**
  $\mathbf{BF_i} \leftarrow$Create(h, m, K)
  for $j \in [1; n']$ **do**
    in F_x^j:
    if $RSSI_i < l$ **then**
      add j to BF_i
    end if
  end for
  Send $\mathbf{BF_i}$ to the 3rd party
end for

Algorithm 9 BF generation by the government agency

$\mathbf{BF}_{\mathcal{W}} \leftarrow$Create$(h, m, K, \mathbf{Whitelist})$
Send $\mathbf{BF}_{\mathcal{W}}$ to the 3rd party

Algorithm 10 Verification phase by the 3rd party

for $i \in [1; n]$ **do**
  if $\mathbf{DIS}(\mathbf{BF_i}, \mathbf{BF}_{\mathcal{W}}) = \mathbf{False}$ **then**
    Send i to the Agency
  end if
end for

Evaluation and Results

Discussion on Correctness. We see in this section what kind of parameters should be selected to perform the function on the current use cases which obviously, due to the size of the sets, are varying very much. As it is presented with

the *Setup()* function, the parameters initialization relies on the cardinalities of the sets, i.e. how many different elements have to be added to the Bloom filters.

Regarding the whitelist, namely the list of suspects from a government agency, we could estimate its scale up to at most 10^2 elements. Also, we could estimate the maximal size of Bloom filters from the master logfile by considering e.g. a location as *Gare du Nord* in Paris, one of the busiest places in Europe with approximately 7×10^5 visitors per day. According to the 2018 study on the French numeric usage from the ARCEP [ARC18], we know that 75% of the French population owns a smartphone and therefore we could estimate the maximal size of the logfiles from use cases 1 and 2 to a Bloom filter with 5×10^5 elements. Due to OS market allocation, we could estimate the maximal Bloom filter retrieving all the Android users connecting to the *Gare du Nord's* WAPs for a day to 4×10^5 in use case 3.

We tested our solution with an implementation of the disjointness operator to validate the correctness of the retrieved results and to give a rough estimation of the processing times. In Table 4.20 we give examples of relevant parameters that produced successful computations along with the running time of the disjointness function in seconds. We precise that we have implemented the disjointness protocol in Java and the measurements from Table 4.20 have been made with a CPU configuration of Intel Core i5 M520 2.40GHz $\times$ 4.

TABLE 4.20. EXAMPLES FOR THE DISJOINTNESS RUNNING TIMES OF ALGORITHM 10.

Use Case	Logfiles Size	Whitelist Size	m	n_{key}	Running Time [sec]
1 & 2	10^4	10^2	4.40×10^9	1416	7.41×10^{-1}
1 & 2	5×10^4	10^2	8.1×10^9	861	3.36×10^1
3	10^5	10^2	6.90×10^9	561	1.3×10^1
3	5×10^5	10^2	1.27×10^{10}	561	1.9×10^2

Discussion on Privacy. The main advantage of using the solution presented in Section 4.3 is to provide privacy on the cardinality of the sets. Indeed, with the use of another approach or the classical way of using Bloom filters, the 3rd party would be able to learn how many suspects form the agency's whitelist. As described in Section 4.3.1, keeping n_{key} secret enables a much more complete privacy on the whitelist and logfiles. The other benefit of this approach and its particularity of using an HMAC function instead of a bunch of public hash functions is avoiding the 3rd party to create its own Bloom filter. In case of a malicious 3rd party trying to retrieve identities of the suspects, one may easily imagine that it generates a Bloom filter with a unique element and performs the disjointness function between this Bloom filter and the whitelist's one. In that way the 3rd party could test if a specific element is included in the suspect list.

For that reason, using secret keys to generate a valid Bloom filter enhances the privacy aspect of the protocol. By applying our solution to real world scenarios, we could imagine several examples of role distribution. We imagine several role distributions for the third sub-use case *Tracking of Suspect Entities on Mobile OS Providers Datasets Ensuring Privacy for Unsuspected Users*. In the first case we have a mobile telecommunication provider from a country A and a government agency from a country B. To perform the *disjointness* protocol, the 3^{rd} party could be performed by an international institution like e.g. Interpol or Europol to act as a go-between. In another scenario, the mobile telecommunication provider and the government agency are from the same country A. To guarantee privacy rights to their own citizens, the 3^{rd} party could be endorsed either by a friendly government agency from a foreign country B or by a semi-trusted private company. Such a friendly government agency (resp. subcontractor company) needs to be, reliable enough in the sense of correctly performing the sensitive protocol, but not to the point of blind trust and secret key sharing.

4.3.7 On Concealed Data Aggregation in WSN

Protocol

To solve Use Case 3 we could adapt our private construction of the Bloom filters to form a sort of *concealed data aggregation* protocol. As recalled in Section 3.6, the main objective of CDA protocols is to make the intermediate nodes collecting and aggregating the data in order to save computation and communication power. Such a protocol's class fits perfectly our use case 3 which main privacy requirement consists of disabling the intermediate nodes ($\mathcal{IN}s$) of gaining any information from the retrieved data from the sensor nodes. Using Bloom filters with HMAC functions instead of public hash functions could be particularly suitable.

One of the main characteristic of Bloom filters consists of the fact that, when the amount of data in the filter (number of elements) increases, its size does not necessary increase. Such a feature is highly valuable especially in aggregation scenarios. We could also find such attributes with homomorphic encryption but in this case, to avoid increasing the size of the ciphertext, the element values should be processed together (e.g. added or multiplied). This makes it harder, even impossible, to decompose and retrieve the original elements. Moreover, using homomorphic encryption in aggregation environment, means that all the elements have to be encrypted with the same key. This brings a security weakness in case of corrupted nodes. To avoid such a drawback we propose to use a different set of keys for each machine.

In Figure 4.13 we recall our solution to the third use case based on the privacy-preserving Bloom filters which construction is the following:

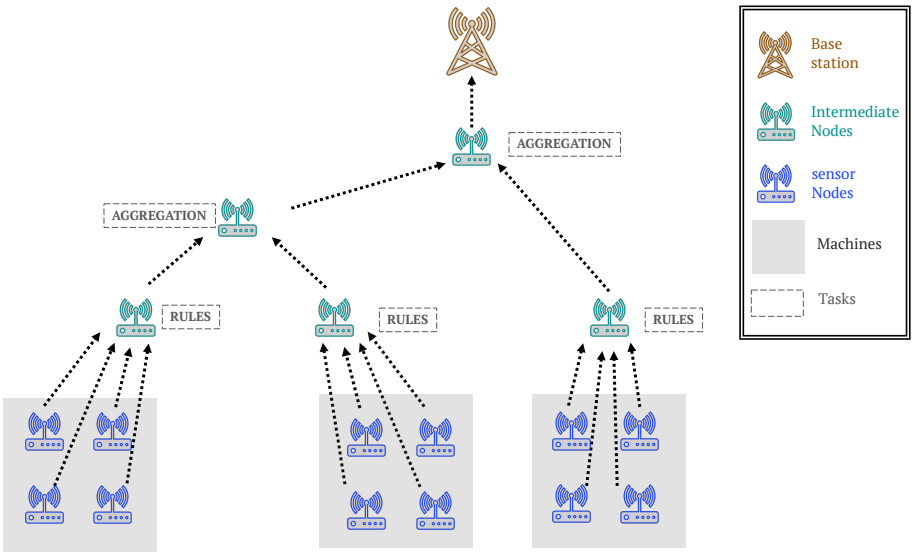

Figure 4.13. Industrial Sensor Network.

Initialization.

$\mathbf{h}, \mathbf{m}, \{\alpha_{\mathbf{1}}^{\mu}, \ldots, \alpha_{\mathbf{n}}^{\mu}\} \leftarrow$**Setup:** The base station $\mathcal{BS}$ should first choose and generate the Bloom filter parameters: the HMAC function h and the dimension m and should generate a distinct set of keys for each machine μ: $\{\alpha_1^{\mu}, \ldots, \alpha_n^{\mu}\}$. $\mathcal{BS}$ distributes the keys' set to all $\mathcal{SN}s$ from the respective machines along with the Bloom filter parameters.

$\mathbf{BF_i^{(1)}}, \mathbf{BF_i^{(2)}}, \mathbf{BF_i^{(3)}}, \cdots \leftarrow$**Create($\mathbf{st_i}, \mathbf{t_i}, \mathbf{s_i}, \ldots$):** Each $\mathcal{SN}$ i creates its own Bloom filter $BF_i^{(x)}$ for each information. For instance the 1^{st} corresponds to the status and the 2^{nd} to the temperature of respective machine μ. Thus, $\mathcal{SN}$ i generates its Bloom filters with st_i, t_i and s_i respectively status, temperature and speed of machine μ:

$$BF_i^{(1)} = \begin{cases} BF\big(\{2 \times \mu\}, (h_\kappa)_{\kappa \in \{\alpha_1^{\mu}, \ldots, \alpha_n^{\mu}\}}\big) = bf_i^{(1)}[j]_{1 \leqslant j \leqslant m} \text{ if } st_i = ok \\ BF\big(\{2 \times \mu + 1\}, (h_\kappa)_{\kappa \in \{\alpha_1^{\mu}, \ldots, \alpha_n^{\mu}\}}\big) = bf_i^{(1)}[j]_{1 \leqslant j \leqslant m} \\ \qquad\qquad\qquad\qquad\qquad\qquad \text{if } st_i = not\ ok \end{cases}$$

$$BF_i^{(2)} = BF\big(\{1000 \times \mu + t_i\}, (h_\kappa)_{\kappa \in \{\alpha_1^{\mu}, \ldots, \alpha_n^{\mu}\}}\big) = bf_i^{(2)}[j]_{1 \leqslant j \leqslant m}$$

$$BF_i^{(3)} = BF\big(\{1000 \times \mu + s_i\}, (h_\kappa)_{\kappa \in \{\alpha_1^{\mu}, \ldots, \alpha_n^{\mu}\}}\big) = bf_i^{(3)}[j]_{1 \leqslant j \leqslant m}$$

Therefore, each $\mathcal{SN}$ i owns three Bloom filters or more if other pieces of information from the machine are relevant to be retrieved to $\mathcal{BS}$.

Rules. $\mathcal{IN}$ performs a rule on all the Bloom filters coming from the same machine μ. Two of these rules could be:

Majority: $\mathcal{IN}$ compares the Bloom filters together and keeps the one which value is the most represented. Such a rule could be applied to $BF_i^{(2)}$ or $BF_i^{(3)}$ that concern respectively the temperature or the speed collected by $\mathcal{SN}$ i. We define this rule as:

$$BF_{maj}^{(2)} \leftarrow maj\{BF_{i_1}^{(2)}, BF_{i_2}^{(2)}, \dots, BF_{i_z}^{(2)}\} \tag{4.26}$$

Completeness: $\mathcal{IN}$ sums all the Bloom filters together to get all the possible values. This rule could be applied to $BF_i^{(1)}$ that concerns the machines' status collected by $\mathcal{SN}$ i. We define this rule as $\mathbf{BF_{comp}^{(1)}} \leftarrow \mathbf{comp}\{\mathbf{BF_{i_1}^{(1)}}, \mathbf{BF_{i_2}^{(1)}}, \dots, \mathbf{BF_{i_z}^{(1)}}\}$:

$$\begin{aligned} &bf_{comp}^{(1)}[j]_{1 \leqslant j \leqslant m} \leftarrow comp\{BF_{i_1}^{(1)}, BF_{i_2}^{(1)}, \dots, BF_{i_z}^{(1)}\} \\ &where \;\; 1 \leftarrow bf_{comp}^{(1)}[j] \; if \; \exists i \; st \; bf_i^{(1)}[j] = 1 \\ &0 \leftarrow bf_{comp}^{(1)}[j] \; otherwise. \end{aligned} \tag{4.27}$$

We remark that this operator is equivalent to the bitwise logical-or operator:

$$comp\{BF_{i_1}^{(1)}, BF_{i_2}^{(1)}\} \equiv BF_{i_1}^{(1)} \; OR \; BF_{i_2}^{(1)}. \tag{4.28}$$

Aggregation. $\mathcal{IN}s$ sums the Bloom filters of the same type $(1, 2, \dots)$ together even if they come from different machines. This operation is similar to the *completeness* rule and the bitwise logical-or operator.

$$BF_{agg}^{(1)} \leftarrow agg\{BF_{comp_{i_1}}^{(1)}, BF_{comp_{i_2}}^{(1)}, \dots, BF_{comp_{i_z}}^{(1)}\} \tag{4.29}$$

Retrieving. $\mathcal{BS}$ gets final $BF_f^{(x)}$ and with each set of keys $\{\alpha_1^\mu, \dots, \alpha_n^\mu\}$ tests if a machine μ is defective and requires maintenance:

$$\begin{aligned} \forall \mu, \; if \; (2 \times \mu) \in BF_f^{(1)} \text{ then } st_\mu = ok \\ if \; (2 \times \mu + 1) \in BF_f^{(1)} \text{ then } st_\mu = not \; ok \end{aligned} \tag{4.30}$$

Also, to retrieve the temperature and the spin speed of every machine, $\mathcal{BS}$ performs:

$$\forall \mu, t \; if \; (1000 \times \mu + t) \in BF_f^{(2)} \text{then } t_\mu = t \tag{4.31}$$

$$\forall \mu, s \; if \; (1000 \times \mu + s) \in BF_f^{(3)} \text{then } s_\mu = s \tag{4.32}$$

We remark that having $\mathcal{BS}$ not retrieving any of the two cases of Equation (4.30) for a particular μ, means the status of machine μ has not been retrieved by the nodes network. Similarly, it holds for Equation (4.31) and Equation (4.32) and the machine's temperature and speed.

Evaluations and Results

Discussion on Performance. As classical CDA protocols, the described solution brings space and energy saving. Instead of storing and forwarding all the sensors' data, $\mathcal{IN}s$ are aggregating the data together. This aspect is very valuable since we recall the space and computation power limitation of the nodes in WSN environment.

By applying the *majority* or *completeness* rules or the aggregation step, $\mathcal{IN}s$ are reducing the amount of data to be stored at each level of the WSN. $\mathcal{BS}$ will have to generate the Bloom filters' size m according to the prediction of the amount of data aggregated. That being said, even if the Bloom filters representations contain more bits, the storage saving is significant. Without any data aggregation, the intermediate nodes close to the base station have to forward many more data packets compared with the sensor nodes. With our approach we observe that all the nodes, intermediate or sensor, will have to forward the same amount of data packets. If we consider $\mathcal{BS}$ requesting to be updated on the machines' status and data for each epoch e_j, all the nodes would have to transfer data packets the same amount a single $\mathcal{SN}$ generates Bloom filters. In the protocol's description, it means three Bloom filters, namely one for the machine's status, one for its temperature and one for its spin speed.

Regarding the nodes' workload within the network, the protocol adds intermediate steps as the rules or the aggregation processing. As we explained, these computations are equivalent to bitwise operations between tabular of bits. Such additional efforts are absorbed by strongly reducing the amount of data to process.

Discussion on Correctness. We propose to apply the rules in order to be as correct as possible regarding data retrieval from the machine. Indeed, by applying the rule *completeness* to the machines' status we guarantee that a status **not ok** will always be retrieved by $\mathcal{BS}$ and a maintenance will be requested. We notice that in case of two different nodes collecting different status messages, both of them will be retrieved. Regarding machine's data as temperature or spin speed, we argue that retrieving the most common value could be sufficient. In case of finding no value more frequent than others, we could apply the *majority* rule the other way by removing the data packages corresponding to data seldom presented. Finally, in addition to enabling $\mathcal{BS}$ to link every data with its respective machine, our approach also allows $\mathcal{BS}$ to notice if no information has been collected for a particular machine. Indeed, as discussed previously, for a specific machine index μ, retrieving no data in Equation (4.30), Equation (4.31) nor Equation (4.32) will be detected by $\mathcal{BS}$. Therefore, this approach brings on additional aspect as completeness notification which is provided by only a few existing CDA protocols.

Discussion on Privacy. By employing a Bloom filter representation of the $\mathcal{SN}s$' data we provide privacy to the network. Indeed, all the $\mathcal{IN}s$ that manipulate the data by collecting, aggregating and forwarding them are likely to

retrieve information from the whole process. By making the $\mathcal{SN}s$ obfuscating the data, the $\mathcal{IN}s$ are getting no information on them. Its security level relies on the proper use of the HMAC function, namely correctly choosing the hash function and generating keys of sufficient length according to the standard recommendation. At this point in time, a good usage of HMAC could be to use the *SHA-256* hash function along with keys of 512 bits which generates output hash of 256 bits length.

Contrary to other CDA's approaches, such as the ones using homomorphic encryption, sensor nodes from different machines do not have to share the same keys set. Indeed, homomorphic schemes like the symmetric Domingo-Ferrer's cryptosystem [Dom02] used in the CDA protocol [GWS05] can be used solely by sharing the same symmetric keys to all the sensor nodes so the encrypted data could be aggregated together along the WSN. Even if we agree that distributing different sets of keys to the sensor nodes adds more complexity to the initialization phase, its benefit on the privacy level is highly more valuable. Indeed, having one sensor node corrupted by an attacker will solely compromise the sensitive data for the sensor nodes from the respective machine.

4.3.8 On GDPR Conform Detection of COVID-19 Infection Chains

Protocol

Collecting Connection Data. We recall that for each base station j, the telco company firstly generates and initializes a fresh Bloom filter BF_j represented by a tabular of bits, all set to 0. Any time a user is connecting to the mobile network using base station j, the following connection information is aggregated and added to BF_j:

$$(id_i, t_i^1, t_i^2)$$

with id_i the user's credentials and t_i^1 and t_i^2 respectively the starting and ending times of its connection to the access point.

Proximity Chain - Infection Chain. *As notation rule, we use $\langle\rangle$ to express proximity chains and [] for infection chains.*
A proximity chain consists of a list of users where two successive ones have been at the same location at the same time. To establish a proximity chain, these times of contact should be ordered. In other words, in the proximity chain $\langle A, D, F, E, B\rangle$, the time at which users A and D have been at the same location should precede the one for users D and F (i.e. $[t_A^1; t_A^2] \cap [t_D^1; t_D^2] < [t_D^1; t_D^2] \cap [t_F^1; t_F^2]$).

In addition to be defined as a proximity chain, the list could also represent an infection chain. In this case, all the users composing the chain should have a probability of being infected $Pr(X_i)$ greater than a certain threshold Tr. More concretely, an infection chain $[A, X_1, \ldots, X_n, B]$ is a proximity chain for which

it holds that: $\forall\ X_i : Pr(X_i) > Tr$, otherwise it is solely a proximity chain. Therefore, an infection chain $[A, X_1, \ldots, X_n, B]$ represents how the COVID-19 virus may have spread from an initially infected user A to a consecutive infected user B.

It may happen that one or several subsets of a proximity chain $\langle A, X_1, \ldots, X_n, B\rangle$ are considered as infection chains, e.g. $[A, X_1, \ldots, X_i]$ and/or $[X_j, \ldots, B]$.

Proposed Solution. From any two given infected users A and B, the government agency first aims to identify all the proximity chains $\langle A, id_{X_1}, \ldots, id_{X_n}, B\rangle$. In our protocol, we recall that the telco company provides all the relevant Bloom filters to the government agency. We propose to dissociate three cases:

- **CASE 1**: the smallest possible proximity chain $\langle A, B\rangle$:
  there is a base station BS_j and a Bloom filter $BF_{A,B}$ for set $\{A, B\}$ and $INC(BF_j, BF_{A,B}) = true$.
  Since both users A and B are indeed infected, the proximity chain $\langle A, B\rangle$ is also an infection chain $[A, B]$.

- **CASE 2**: a proximity chain with one intermediate user X $\langle A, id_X, B\rangle$:
  there is a base station BS_{j_1} and a Bloom filter BF_{A,id_X} for set $\{A, id_X\}$ where $INC(BF_{j_1}, BF_{A,id_X}) = true$ and in addition, there is a base station BS_{j_2} and a Bloom filter $BF_{id_X,B}$ for set $\{id_X, B\}$ and $INC(BF_{j_2}, BF_{id_X,B}) = true$. We remark here that we know users A and B but we do not know user X nor his access credential id_X, so the government agency has to search in all base stations for all X_j for which the above two inclusiveness tests INC hold.
  If $Pr(id_X) > Tr$ we can denote $[A, id_X, B]$.

- **CASE 3**: the general case $\langle A, id_{X_1}, \ldots, id_{X_n}, B\rangle$:
  we have $INC(BF_{j_1}, BF_{A,id_{X_1}}) = true \wedge \ldots \wedge INC(BF_{j_n}, BF_{id_{X_n},B}) = true$.

Our solution consists of having the government agency building a data tree structure representing all the proximity chains starting from user A. From this tree, the agency could easily identify the proximity chains from user A to user B. For the next step of the protocol, the government agency has to evaluate the chain to determine its plausibility to actually be an infection chain. We give the outlines of this step but not its evaluation function that we save for the epidemiologists.

We emphasize that at this point, the proximity or infection chains will only reveal usernames of users $X_1, \ldots, X_n$ and not their real identities. At the very end of the protocol, the government agency will request from the telco company the identities of the intermediate infected users.

Generating the Proximity Tree. To obtain a proximity tree, the government agency starts by creating an empty tree T with user A as root. Then,

it processes the recursive algorithm $prox\_tree(A, A, B, t')$ presented in Algorithm 11 with t' the time from when user A could have started the infection process. The recursive algorithm does as follow: first, it generates the list BS_N of base stations that the current node N has been connected to at a time later than t. To test if a user N has been connected to a base station j (i.e. test if $(id_N, t_j^1, t_j^2) \in BF_j$), the government agency receives from the telco company all the Bloom filters composed of each of the 3-tuples (id_N, t_j^1, t_j^2). Then, the government agency performs the inclusiveness testing between the received Bloom filters and BF_i, the Bloom filter corresponding to the connections logfile from BS_j as: $INC(BF_{N,j}, BF_i)$. The next step of the algorithm consists of identifying all the users that visited the base stations from set BS_N at the same moment than user N. As before, the telco company generates Bloom filters with the 3-tuples (id_l, t_l^1, t_l^2) for all users l and all time ranges $[t_l^1; t_l^2]$ that overlap the connection time of user N. To determine which users should be listed, the government agency performs the inclusiveness operator between these Bloom filters and BF_N the one composed by the elements from BS_N. Finally, for every identified users, they are added to the proximity tree T as a leaf of current node N and Algorithm 11 is then recursively processed on the leaves.

An additional aspect to take into account while recursively processing the algorithm is to consider the upper nodes of the current node in the proximity tree. Indeed, we would like to avoid creating some loops in the tree which are irrelevant when dealing with infection problems; if user A infected user C, it makes no sense to consider user C infecting user A in short period of time. The algorithm should then exclude all the users which are already inserted as upper nodes in the tree. Regarding the tree construction, if we consider that user C has been in proximity of user A and id_C is added as a leaf of root A, user A should not be considered anymore as potential leaf of node id_C and so on.

In Figure 4.14 we give a toy example of our recursive algorithm with seven users A, B, C, D, E, F, G, three base stations $BS_{j_1}, BS_{j_2}, BS_{j_3}$ and times as integers in $[0; 24]$. We show the content of connection logfiles from the three base stations and the proximity tree from user A to user B that has been generated by computing $prox\_tree(A, A, B, 0)$. We observe in Figure 4.14 that two users might be in contact around different base stations. Indeed, the resulting proximity chains are $\langle A, C, G, B\rangle$, $\langle A, G, B\rangle$ (with users A and G in proximity around BS_{j_1}), $\langle A, G, B\rangle$ (with users A and G in proximity around BS_{j_3}) and $\langle A, B\rangle$. In case there are evaluated as infection chains, users C and G might also be infected.

Algorithm optimization. With respect to performance, one could consider computing the algorithm on the opposite way, namely with input B as root. To do so, the algorithm should be modified so that time is considered backwards. It starts at ending time (24 for our toy example) and we build the proximity tree by going back in time. We consider as $reverse\_prox\_tree$ this reverse recursive algorithm.

In Figure 4.15 we show the proximity tree obtained after computing

Algorithm 11 $prox\_tree(N, A, B, t)$

Input: a node N from a tree T, users A and B, a time t
Output: a tree T

```
if N = B then
    break
end if
for all BF_j do

    for all t_j^1, t_j^2 > t do

        if (id_N, t_j^1, t_j^2) ∈ BF_j then
            BS_N.add((BS_j, t_j^1, t_j^2))
        end if
        end for
    end for
    for all (BS_Nk, t_Nk^1, t_Nk^2) ∈ BS_N do

        for all id_l do

            for all (t_l^1, t_l^2) | (t_l^1 ⩽ t_Nk^1 ∧ t_l^2 ⩾ t_Nk^1) ∨ (t_l^2 ⩾ t_Nk^2 ∧ t_l^1 ⩽ t_Nk^2) ∨
(t_l^1 ⩽ t_Nk^2 ∧ t_l^2 ⩾ t_Nk^1) do

                if (id_l, t_l^1, t_l^2) ∈ BF_Nk then
                    createLeaf(id_l)
                    prox_tree(id_l, A, B, max(t_Nk^1, t_l^1))
                end if
                end for
            end for
        end for
        if N.leaf = ∅ then
            break
        end if
```

BF_{j1}={ (id_A, 0, 6), (id_C, 2, 9), (id_G, 3, 5), (id_D, 7, 10)}
BF_{j2}={ (id_A, 8, 17), (id_D, 15, 18)}
BF_{j3}={ (id_F, 2, 11), (id_E, 6, 15), (id_G, 8, 24), (id_A, 18, 24), (id_B, 18, 20)}

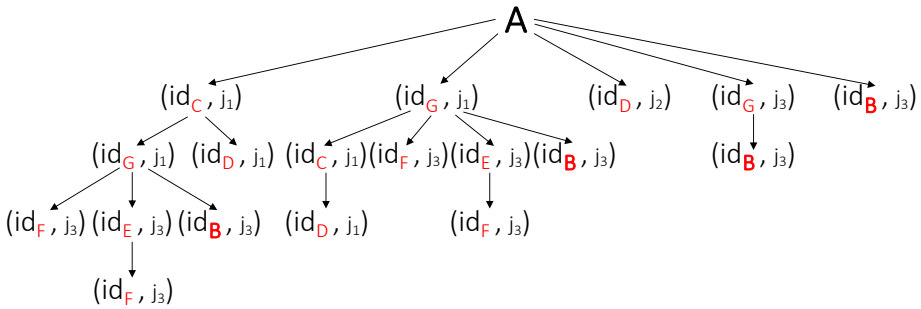

Figure 4.14. Example of connection logfiles from three base stations and the respective proximity tree obtained from $prox\_tree(A, A, B, 0)$. It outcomes three different proximity chains $\langle A, C, G, B\rangle$, $\langle A, G, B\rangle$ and $\langle A, B\rangle$.

$reverse\_prox\_tree(B, A, B, 24)$ from user B considering the time backwards. As expected, the resulting proximity chains are the same than in Figure 4.14 but we remark that the resulting tree is smaller than the one obtained in Figure 4.14. In this specific toy example we notice that obtaining the proximity tree was made faster by reversing our algorithm.

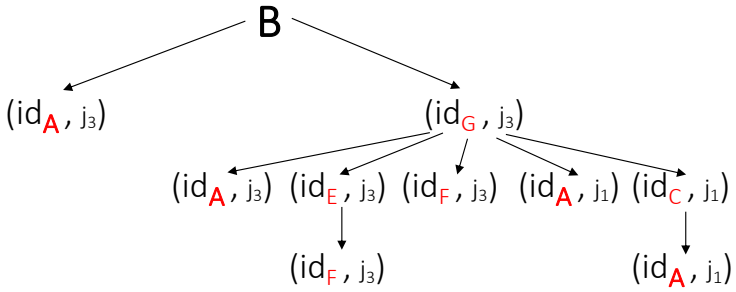

Figure 4.15. Example of a proximity tree obtained from $reverse\_prox\_tree(B, A, B, 24)$ with the same toy example than Figure 4.14. It generates three different proximity chains $\langle A, C, G, B\rangle$, $\langle A, G, B\rangle$ and $\langle A, B\rangle$.

Another aspect we could consider while comparing the two resulting trees, is that the order the tree is being build in and the proximity chain obtained are also reversed. Indeed, in Figure 4.14 we obtain first $\langle A, C, G, B\rangle$ then $\langle A, G, B\rangle$ (via j_1), $\langle A, G, B\rangle$ (via j_3) and finally $\langle A, B\rangle$. In Figure 4.15 we see that we obtain the chains in the exact opposite order with $reverse\_prox\_tree$. Still aiming to optimize the computation time of our algorithm, in particular when

dealing with large numbers of users and base stations, one could simultaneously start the tree generation using the algorithm and its reversed version. For both cases the tree propagates and every time we find a proximity chain in the tree (meaning $N = B$ or $N = A$ for *reverse_prox_tree*) we could store the chain in a set S (or S' for *reverse_prox_tree*). Then for each round (i.e *for* iteration) we test if the two sets have a common element. If not, we continue. In case they have a common proximity chain, we could stop both algorithms and the complete set of proximity chains from users A to B is composed of the addition of sets S and S'.

To illustrate the approach of computing both versions at the same time and, as argued, gain on performance, one could explain:

- if you throw one stone into the water and you want the resulting waves to reach a point in r meters distance, then the circle at the end will encompass many square meters.
- if you throw two stones into the water (one at the original position, the other one at the position you want to reach), the intersection of the resulting waves propagation will be approx. at a distance $r/2$ meters.
- adding the area of these two circles shall be much smaller than the circle's area obtained with one stone.

For example, with $A = \pi \times r^2$ and $r = 10$ $A = 314.159$, and with $r = 5$ the area of the two circles is altogether approximately 160!.

Another level of optimization could be considered in order to identify some of the proximity chains faster as for instance to support the start of a localized quarantine immediately. Instead of storing the chains into S and S', at each propagation round we look at the chains while they are processed so that we stop both algorithms when:

- *prox_tree* has built a path $\langle A, X_1, \ldots, X_i \rangle$
- *reverse_prox_tree* has built a path $\langle B, X_n, \ldots, X_j \rangle$
- and it holds $X_i == X_j$

Then the two parts of the proximity chain could be concatenated to create the proximity chain $\langle A, X_1, \ldots, X_i == X_j, \ldots, X_n, B \rangle$

We could refer to Table 4.21 to see that if we perform both algorithms at the same time in the toy example configuration, we could retrieve the proximity chain $\langle A, C, G, B \rangle$ faster with this second level of optimization.

In Table 4.21 we could observe in detail how we retrieve the proximity chains using the two versions of Algorithm 11 and the optimization with the toy example's configuration. As stated previously, $reverse\_prox\_tree(B, A, B, 24)$ was executed much faster than $prox\_tree(A, A, B, 0)$. Indeed, the original algorithm ended after 18 rounds while the reverse one stopped after the 9^{th} round. Since it is not possible to predict which of the two will finish processing first, computing both in parallel will optimize the retrieving. As for the second level

of optimization, concatenating two parts of proximity chains allows to retrieve $\langle A, C, G, B\rangle$ at round 2 while discovered at round 6 with *prox_tree* and round 9 with *reverse_prox_tree*. It is of value especially when proximity chains are composed by a high number of intermediate users.

TABLE 4.21. CONSTRUCTION OF THE PROXIMITY TREE ROUND BY ROUND WITH *prox_tree* AND *reverse_prox_tree* AND HOW THE OPTIMIZATION COULD BE APPLIED.

Round	*prox_tree*	*reverse_prox_tree*	With optimization
1	C	$A, \langle \mathbf{A}, \mathbf{B}\rangle$	$\langle \mathbf{A}, \mathbf{B}\rangle$ from *reverse_prox_tree*. from *reverse_prox_tree*.
2	G	G	$\langle \mathbf{A}, \mathbf{C}, \mathbf{G}, \mathbf{B}\rangle$ from concatenation of $\langle A, C, G\rangle$ from *prox_tree* and $\langle G, B\rangle$ from *reverse_prox_tree*.
3	F	$A, \langle \mathbf{A}, \mathbf{G}, \mathbf{B}\rangle$	$\langle \mathbf{A}, \mathbf{G}, \mathbf{B}\rangle$ from *reverse_prox_tree*.
4	E	E	
5	F	F	
6	$B, \langle \mathbf{A}, \mathbf{C}, \mathbf{G}, \mathbf{B}\rangle$	F	
7	D	$A, \langle \mathbf{A}, \mathbf{G}, \mathbf{B}\rangle$	$\langle \mathbf{A}, \mathbf{G}, \mathbf{B}\rangle$ from *reverse_prox_tree*.
8	G	C	
9	C	$A, \langle \mathbf{A}, \mathbf{C}, \mathbf{G}, \mathbf{B}\rangle$	
...	...	...	
14	$B, \langle \mathbf{A}, \mathbf{G}, \mathbf{B}\rangle$	-	
...	...	...	
17	$B, \langle \mathbf{A}, \mathbf{G}, \mathbf{B}\rangle$	-	
18	$B, \langle \mathbf{A}, \mathbf{B}\rangle$	-	

The performance gain obtained with our two levels of optimization is downplayed due to the extreme smallness of logfiles in our toy example. But we have seen in Section 4.3.6 that such approach fits perfectly with scenarios with logfiles and therefore Bloom filters up to 10^6 elements.

Algorithm decentralization. The European PEPP-PT consortium is advocating a decentralized approach which has also been investigated in [Vau20]. With our presented optimization, we could integrate such construction by introducing two additional parties besides the ones already presented:

- *Computing party 1* which runs *prox_tree*
- *Computing party 2* which runs *reverse_prox_tree*

This way the agency is only receiving per round the values for X_i (from *computing party 1*) and X_j (from *computing party 2*) and comparing if $X_i == X_j$. Only in the case $X_i == X_j$ do we obtain that *computing party 1* is sending $\langle A, X_1, \ldots, X_i\rangle$ and *computing party 2* is sending $\langle X_j, \ldots, X_n, B\rangle$ to the agency. With such a construction, multiple parties are involved in the computation and the whole effort does not rely on the government agency.

Algorithm complexity. One could easily see by analyzing the obtained results in Figure 4.14 and Figure 4.15 that the size of the resulting tree will depend on the size of the base stations' logfiles. These logfiles will naturally depend on the amount of users and thus connections during the particular time. The more base stations and users there are, the more logfiles will be numerous and fully filled. In our toy example, we have 11 connection entries in all combined base stations as displayed in Figure 4.14. They result in a tree with respectively 19 and 10 nodes by computing $prox\_tree$ and $reverse\_prox\_tree$. We also recall that in case we find the final user of the wanted infection chain (user B in our example) in the tree, the algorithm reaches a break instruction and therefore the respective sub-tree is no longer explored. A high activity of this particular user could then reduce the tree's spreading. As seen previously, one of the two algorithms will be faster to execute without being able to predict which one and applying the presented optimization could reduce the complexity to the faster one.

Proximity Chain Evaluation. From all the proximity chains $\langle A, id_{X_1}, \ldots, id_{X_n}, B\rangle$ obtained by performing the aforementioned protocol, the government agency should determine if users X_i might also be infected. To do so, the agency could estimate the users' probability of being infected and compare it to a threshold (i.e. $Pr(X_i) > Tr$). Such a probability obviously depends, among others, on the respective neighbors within the chain. We consider the probability value computed as a function $infection(previous\_node, contact\_time, contact\_distance, reproduction\_number, saturation)$ where $saturation$ shall denote the percentage of infected persons within the human population of a region, which obviously changes over time.

More precisely, in Germany the reproduction number R, which is defined as the mean number of people infected by a case, was 3 at the beginning of the COVID-19 crisis and by 17.04.2020 could be reduced to 0.7 (and meanwhile $R = 1.1$). Clearly this number is only an average but still indicates that inference from a proximity chain to an infection chain very much depends on the concrete time and location entities met during the pandemic wave. Similar numbers also exist for other countries as for instance $R = 0.8$ for Belgium at 17.04.2020. Another important observation is that since a proximity chain can easily build up over a period of weeks, $Pr(X_i)$ may significantly vary. But only if all probabilities are larger than Tr the agency can at least argue having identified a possible infection chain.

It goes without saying that it is out of scope to determine the $infection$ function. On the one hand, specialists emphasize the high contagiousness of the virus but on the other hand, having two users connecting to the same base station at the same time does not necessarily imply any physical contact between the two.

Without being able to determine the exact probability of a user to be infected by another one, we could propose a model to evaluate the probability of a proximity chain becoming an infection chain. First, we know that users A

and B are infected and we would like to determine if user B has been infected due to user A or via another chain and other infection events. Therefore, applying probability theory to such a problem is relevant and reflects the *chain* characteristic of it.

We define as $Pr(X_i)$ the following conditional probability $P(X_i|X_{i-1})$ of the event "X_{i-1} has infected X_i knowing that X_{i-1} is already infected". It holds that $Pr(X_1 \cap \cdots \cap X_n) = \prod_{i=1}^{n} Pr(X_i)$. Considering a proximity chain $\langle A, X_1, \ldots, X_n, B\rangle$, there is a clear tendency that the overall probability to have user B infected due to user A is inversely proportional to the length of the proximity chain. We propose the following probability model for evaluating a proximity chain:

$$\text{For each } \langle A, X_1, \ldots, X_n, B\rangle, \text{ if } \prod_{i=1}^{n} Pr(X_i) \geq T_r \text{ then } [A, X_1, \ldots, X_n, B].$$

The proximity tree obtained at the previous stage of the protocol contains nodes with users' credentials and only these usernames are revealed. It is only in case a proximity chain turns out to be an infection chain that, the agency will request from the telco company the real identities of the users composing the chain. Therefore, users' identity are solely revealed in case of *infection* function outcomes so. Moreover, we recall that during the overall process no additional location information of other users listed in the mobile operator's logfile are revealed to the agency.

Recursivity of the Infection Detection. One may notice that a trivial optimization would be to switch users A and B in the sense that "infection of user A is coming from user B". In Figure 4.16 we show the proximity tree obtained from our algorithm by computing $prox\_tree(B, B, A, 0)$ with our toy example logfiles. We notice that it results in a very different tree than in Figure 4.14 obtained by $prox\_tree(A, A, B, 0)$. In case the government agency holds some information on the infection time of users A and B, for example that user A has been infected before user B, only one direction should be considered by the agency.

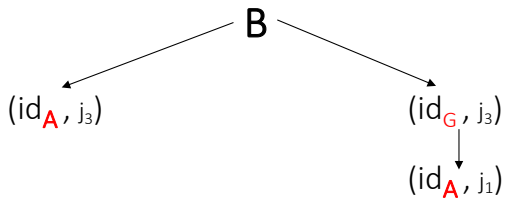

Figure 4.16. Example of a proximity tree from user B to user A obtained from $prox\_tree(B, B, A, 0)$. It results in two different proximity chains $\langle B, A\rangle$ and $\langle B, G, A\rangle$.

To be the most efficient, the government agency should perform a final step in the protocol. All the users identified as infected at the previous stage (i.e. all X_i where $Pr(X_i) > Tr$) should be considered as new users A and respectively B in the proposed solution. Indeed, our protocol is initiated with users tuples (A, B) already identified as infected by the agency. The freshly identified users are thus incrementing the list of known infected persons and the protocol should be applied to them to optimize the search. In such a way, the most infected users could be identified and contacted.

Evaluation and Results

Discussion on Location Privacy. We argue that the proposed solution provides privacy for the users by two different means. Firstly by using only personal credentials as usernames and secondly thanks to the Bloom filter's construction and its obfuscation feature. Indeed, as explained previously, the real identities of users are not provided and stored in the Bloom filters nor the logfiles. The telco company uses usernames to distinguish users and the private mapping will be provided to the government agency solely on-demand, when a user is identified as being part of an infection chain.

The second aspect of location privacy is given by the Bloom filters based approach as presented in Section 4.3.1 which allows to compute relations among logfiles while keeping these data sets private. We recall that such an approach uses an HMAC function instead of a bunch of public hash functions and therefore only the telco company could create the Bloom filters and no other party. To this extent, the government agency could not try to retrieve locations of a specific user by generating a Bloom filter with a unique element and performs the inclusiveness relation between this Bloom filter and the ones from base stations. For that reason, using secret keys to generate a valid Bloom filter enhances the privacy aspect of the protocol. Finally we recall that secret keys are generated and stored only at the telco company side and are not required by the government agency to perform our protocol.

Our detailed protocol supports a government agency to track possible COVID-19 infection chains and therefore identify plausible infected mobile users. Throughout the entire protocol, the agency will only handle usernames which do not allow to retrieve the users' identities and therefore their privacy will be preserved. Solely in the case of possible infection by the life-threatening COVID-19 virus, real identities will be revealed to the agency, that will be able to contact them and provide medical support. In such way, the telco companies act GDPR compliant and could still guarantee a certain level of location privacy to their clients. We could stress that if data stem from the 'in proximity' mobile telco's logfile, it means that two devices have been in the same transmission range of a base station. In the worst case they can still have a $2 \times r$ distance (easily 500 m or more). However, if the same approach can be applied to the RSSI based Swarm-mapping approach for Android or iOS collected data then 'in proximity' has a much better accuracy [DG17]. In particular also the WiFiLocationHarvest file of each mobile device contains timestamp, latitude,

longitude, trip-id, speed, course at an amazing accuracy which comes close to the accuracy required to check if two devices got nearer than 2 m (infection distance). And, moreover, compared to the promoted App based approach with Bluetooth from *Germany Fraunhofer Institutes* and others in the RSSI based approach the mobile's WLAN and Bluetooth can be off, and yet, simply due to the measured RSSI from the access point, the approach provides the location data of the devices equipped with such modern mobile operating systems.

To conclude, applying the Bloom filters approach to *Use Case 4 - Detection of COVID-19 Infection Chains* may be a good starting point for debating a reasonable GDPR compliant detection of COVID-19 infection chains since we argue it does not provide additional privacy-leakage to other parties than those who already have the knowledge of our location data.

Chapter 5

Conclusion

As a lesson learned from the PAL SAaaS project, aiming to broadly solve all the cloud auditing scenarios or all cases where privacy and malleability are requested, is not realistic. For that reason, we concentrated our work on establishing specific and relevant use cases that we claim to be achievable in everyday life. We tried to diversify the scope of application by covering *tasks delegation* issues in cloud security auditing as well as *preventing mass-surveillance* for mobile network users. We also included *government surveillance* issues in the very latest context of COVID-19 pandemic. Finally, we addressed *optimization* issues in industry 4.0 environment along with the highly critical aspect of preventing machines' data breaches.

After identifying relevant "real-world" use cases, we used two appropriate strategies to provide solutions, that we could characterize as *combination* and *distortion*.

The first strategy *combination* consists of combining existing approaches as presented in ***Solution A*** - *Combining PIR Protocol with Searchable Encryption and Homomorphic Encryption*. Since making these cryptographic protocols usable together is not straightforward, we analyzed and identified the most relevant of homomorphic scheme, PIR protocol or searchable encryption scheme and proposed an efficient protocol. We applied this proposal to ***Use Case 1*** - *Cloud Security Auditing* to allow a third party auditor to securely perform verifications of the cloud provider, on behalf of users. To be even more complete, we developed in ***Solution B*** - *Adapting SHE to Evidence Processing* an algorithm that allows any user to optimally initialize the somewhat homomorphic scheme, and therefore making *Solution A* even more efficient. As a matter of fact, our second solution allows any user to select the most fit parameters considering the three highlighted aspects of *correctness*, *performance* and *security*.

The second strategy *distortion* corresponds to distort well-known solutions that, originally had a completely different purpose; thinking outside the box. This is precisely what we did by redirecting the use of Bloom filters in ***Solution C*** - *Using Bloom Filters to Process Set Relations*. We recall that originally, Bloom filter data structure was established to make usable applications in which

systems are resource-limited and could not support too large amounts of data. Without neglecting this optimization feature, we used the obfuscation aspect brought by the structure itself, along with the "one-wayness" feature of hash functions, to develop set relations processed in a private manner. To that end, we shown how to perform the *private outsourced inclusiveness test* and the *private outsourced disjointness test* on data sets. Such defined protocols could therefore be applied to a great amount of scenarios. In particular we shown how it could be used to solve the four presented use cases. In addition to the aforementioned first use case, we applied the disjointness protocol to ***Use Case 2*** - *Mobile Users' Data Collection* to allow a third party to intersect mobile connection logfiles with a government whitelist of suspect users. Then we applied our Bloom filter-based construction to perform a version of *Concealed Data Aggregation* protocol in the environment of industry 4.0 in ***Use Case 3*** - *Wireless Sensor Network's Data Aggregation*. Finally, with use of the inclusiveness function, we developed a protocol to generate proximity tree and infection chain to solve ***Use Case 4*** - *Detection of COVID-19 Infection Chains*, highlighting the up-to-dayness value of our work. Indeed, such construction will allow a government agency to identify suspected chains of infection and to contact the probable infected users in the objective to contain the COVID-19 pandemic.

Throughout the fulfillment of this doctoral research, not resulting in a global and broad-based solution led us to investigate approaches at a very much precise level. In fact, it gave us the satisfaction of assimilating and mastering several different topics as cloud auditing or wireless sensor networks.

Such a well received limitation enables multiple new use cases and scenarios to be addressed in future work.

List of Figures

List of Tables

List of Algorithms

Acronyms

AC Audit Controller

AP Access Point

API Application Programming Interface

ARCEP Autorité de Régulation des Communications Electroniques et des Postes

BDH Bilinear Diffie-Hellman

BF Bloom Filter

BKA Bundeskriminalamt

CCA2 Adaptive Chosen-Ciphertext Attack

CDA Concealed Data Aggregation

CDESF Common Digital Evidence Storage Format

COVID-19 Coronavirus Disease 2019

CPIR Computational Private Information Retrieval

CPU Central Processing Unit

CSP Cloud Service Provider

CVE Common Vulnerabilities and Exposures

CWSN Cognitive Wireless Sensor Network

DB Database

DCRA Decisional Composite Residuosity Assumption

DEB Digital Evidence Bag

E2E End-to-End

ECDLP Elliptic Curve Discrete Logarithm Problem

ERS Evidence Record Syntax

ES Evidence Store

FFT Fast Fournier Transform

FHE Fully Homomorphic Encryption

GDPR General Data Protection Regulation

GMP GNU Multiple Precision arithmetic library

HE Homomorphic Encryption

HMAC Hash-based Message Authentication Code

IaaS Interface as a Service

IMSI International Mobile Subscriber Identity

IN Intermediate Node

IND-CPA Indistinguishability under Chosen-Plaintext Attack

iOS iPhone Operating System

IP Internet Protocol

IT Information Technology

ITPIR Information Theoretic Private Information Retrieval

KDM Key Dependent Message

LDPC Low-Density Parity-Check

LTE Long Term Evolution

LWE Learning With Error

MAC Media Access Control

NP Non-deterministic Polynomial time

OBU On-Board Unit

OS Operating System

OT Oblivious Transfer

PaaS Platform as a Service

PAL Privacy Availability Liability

PBC Pairing-Based Cryptography

PEKS Public Key Encryption with keyword Search

PEPP-PT Pan-European Privacy-Preserving Proximity Tracing

PIR Private Information Retrieval

PR Privacy Requirement

PSI Private Set Intersection

RKI Robert Koch Institute

RLWE Ring Learning With Error

RSA Rivest Shamir and Adleman

RSSI Received Signal Strength Indication

SAaaS Security Audit as a Service

SaaS Software as a Service

SE Searchable Encryption

SHA Secure Hash Algorithm

SHE Somewhat Homomorphic Encryption

SLA Service-Level Agreement

SPIR Symmetric Private Information Retrieval

SQL Structured Query Language

SR Security Requirement

SSE Searchable Symmetric Encryption

SVP Shortest Vector Problem

SWGDE Scientific Working Group on Digital Evidence

TLS Transport Layer Security

TMSI Temporary Mobile Subscriber Identity

UC Use Case

VM Virtual Machine

WAP Wireless Access Point

WLAN Wireless Local Area Network

WSN Wireless Sensor Network

XaaS Anything as a Service

Bibliography

[ABC+15] Frederik Armknecht, Colin Boyd, Christopher Carr, Kristian Gjøsteen, Angela Jäschke, Christian A. Reuter, and Martin Strand. A guide to fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2015:1192, 2015. URL: http://eprint.iacr.org/2015/1192.

[AG11] Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In *Proceedings of the 38th International Colloquim Conference on Automata, Languages and Programming - Volume Part I*, ICALP'11, pages 403–415, Berlin, Heidelberg, 2011. Springer-Verlag. URL: http://dl.acm.org/citation.cfm?id=2027127.2027170.

[AGW05] Mithun Acharya, Joao Girão, and Dirk Westhoff. Secure comparison of encrypted data in wireless sensor networks. In Eitan Altman and Holger Karl, editors, *3rd International Symposium on Modeling and Optimization in Mobile, Ad-Hoc and Wireless Networks (WiOpt 2005), 4-6 April 2005, Trentino, Italy*, pages 47–53. IEEE Computer Society, 2005. URL: https://doi.org/10.1109/WIOPT.2005.44.

[AKSX04] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Order-preserving encryption for numeric data. In Gerhard Weikum, Arnd Christian König, and Stefan Deßloch, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004*, pages 563–574. ACM, 2004. URL: https://doi.org/10.1145/1007568.1007632.

[AM09] Julia Albath and Sanjay Madria. Secure hierarchical data aggregation in wireless sensor networks. In *2009 IEEE Wireless Communications and Networking Conference, WCNC 2009, Proceedings, Budapest, Hungary, 5-8 April 2009*, pages 2420–2425. IEEE, 2009. URL: https://doi.org/10.1109/WCNC.2009.4917960, doi:10.1109/WCNC.2009.4917960.

[AM14] Vikas G. Ashok and Ravi Mukkamala. A scalable and efficient privacy preserving global itemset support approximation using bloom filters. In Vijay Atluri and Günther Pernul, editors, *Data and Applications Security and Privacy XXVIII - 28th Annual IFIP WG 11.3 Working Conference, DBSec 2014, Vienna, Austria, July 14-16, 2014. Proceedings*, volume 8566 of *Lecture Notes in Computer Science*, pages 382–389. Springer, 2014. URL: http://dx.doi.org/10.1007/978-3-662-43936-4_26.

[ARC18] ARCEP. Baromètre du numérique. 2018. URL: https://www.arcep.fr/uploads/tx_gspublication/barometre-du-numerique-2018_031218.pdf.

[ARY+19] Muder Almiani, Abdul Razaque, Liu Yimu, Meer Jaro Khan, Tang Minjie, Mohammed Alweshah, and Saleh Atiewi. Bluetooth application-layer packet-filtering for blueborne attack defending. In *Fourth International Conference on Fog and Mobile Edge Computing, FMEC 2019, Rome, Italy, June 10-13, 2019*, pages 142–148. IEEE, 2019. URL: https://doi.org/10.1109/FMEC.2019.8795354, doi:10.1109/FMEC.2019.8795354.

[AWGH08] Frederik Armknecht, Dirk Westhoff, Joao Girão, and Alban Hessler. A lifetime-optimized end-to-end encryption scheme for sensor networks allowing in-network processing. *Computer Communications*, 31(4):734–749, 2008. URL: https://doi.org/10.1016/j.comcom.2007.10.019.

[BC04] Steven M. Bellovin and William R. Cheswick. Privacy-enhanced searches using encrypted bloom filters. Cryptology ePrint Archive, Report 2004/022, 2004. https://eprint.iacr.org/2004/022.

[BCOP04] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In Cachin and Camenisch [CC04a], pages 506–522. URL: http://dx.doi.org/10.1007/978-3-540-24676-3_30.

[Ben94] Josh Benaloh. Dense probabilistic encryption. In *Proceedings of the workshop on selected areas of cryptography*, pages 120–128, 1994.

[Bf12] Martin Burkhart and Xenofontas Dimitropoulos fontas. Fast private set operations with sepia. 2012. URL: https://pdfs.semanticscholar.org/c79a/c8b0ff2f7377fc12d0d8f23ba74aecd8db3b.pdf?_ga=2.250376646.1532399169.1562684991-2082322444.1554797542.

[BGM10a] J. M. Bahi, C. Guyeux, and A. Makhoul. Efficient and robust secure aggregation of encrypted data in sensor networks. In *2010 Fourth International Conference on Sensor Technologies and Applications*, pages 472–477, July 2010. doi:10.1109/SENSORCOMM.2010.76.

[BGM10b] Jacques M. Bahi, Christophe Guyeux, and Abdallah Makhoul. Secure data aggregation in wireless sensor networks: Homomorphism versus watermarking approach. In Jun Zheng, David Simplot-Ryl, and Victor C. M. Leung, editors, *Ad Hoc Networks - Second International Conference, ADHOCNETS 2010, Victoria, BC, Canada, August 18-20, 2010, Revised Selected Papers*, volume 49 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 344–358. Springer, 2010. URL: https://doi.org/10.1007/978-3-642-17994-5_23, doi:10.1007/978-3-642-17994-5_23.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In Joe Kilian, editor, *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2005. URL: https://doi.org/10.1007/978-3-540-30576-7_18.

[Bie14] Arndt Bieberstein. An implementation of somewhat homomorphic encryption scheme from the ring learning with errors. Master's thesis, Hochschule Furtwangen University, Furtwangen, Germany, 2014.

[Blo70] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, July 1970. URL: http://doi.acm.org/10.1145/362686.362692.

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 97–106. IEEE Computer Society, 2011. URL: https://doi.org/10.1109/FOCS.2011.12.

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 505–524. Springer, 2011. URL: http://dx.doi.org/10.1007/978-3-642-22792-9_29.

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. *SIAM J. Comput.*, 43(2):831–871, 2014. URL: http://dx.doi.org/10.1137/120868669.

[BW11] Dominik Birk and Christoph Wegener. Technical issues of forensic investigations in cloud computing environments. In Robert F.

Erbacher, Roy H. Campbell, and Yong Guan, editors, *2011 IEEE Sixth International Workshop on Systematic Approaches to Digital Forensic Engineering, SADFE 2011, Oakland, CA, USA, May 26, 2011*, pages 1–10. IEEE Computer Society, 2011. URL: https://doi.org/10.1109/SADFE.2011.17.

[CC04a] Christian Cachin and Jan Camenisch, editors. *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*. Springer, 2004.

[CC04b] Tim Churches and Peter Christen. Some methods for blindfolded record linkage. *BMC Med. Inf. & Decision Making*, 2004. URL: https://doi.org/10.1186/1472-6947-4-9.

[CCMT09] Claude Castelluccia, Aldar C.-F. Chan, Einar Mykletun, and Gene Tsudik. Efficient and provably secure aggregation of encrypted data in wireless sensor networks. *TOSN*, 5(3):20:1–20:36, 2009. URL: https://doi.org/10.1145/1525856.1525858.

[CGKS95] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, 23-25 October 1995*, pages 41–50. IEEE Computer Society, 1995. URL: http://dx.doi.org/10.1109/SFCS.1995.492461.

[Cha04] Yan-Cheng Chang. Single database private information retrieval with logarithmic communication. In Huaxiong Wang, Josef Pieprzyk, and Vijay Varadharajan, editors, *Information Security and Privacy: 9th Australasian Conference, ACISP 2004, Sydney, Australia, July 13-15, 2004. Proceedings*, volume 3108 of *Lecture Notes in Computer Science*, pages 50–61. Springer, 2004. URL: https://doi.org/10.1007/978-3-540-27800-9_5.

[Clo] Cloud Auditing Data Federation (CADF). https://www.dmtf.org/standards/cadf.

[CT08] Cheng-Kang Chu and Wen-Guey Tzeng. Efficient k-out-of-n oblivious transfer schemes. *J. UCS*, 14(3):397–415, 2008. URL: http://dx.doi.org/10.3217/jucs-014-03-0397.

[DG17] Andreas Dhein and Rüdiger Grimm. Standortlokalisierung in modernen smartphones - grundlagen und aktuelle entwicklungen. *Informatik Spektrum*, 40(3):245–254, 2017. URL: https://doi.org/10.1007/s00287-016-0964-7.

[Dom02] Josep Domingo-Ferrer. A provably secure additive and multiplicative privacy homomorphism. In Agnes Hui Chan and Virgil D.

Gligor, editors, *Information Security, 5th International Conference, ISC 2002 Sao Paulo, Brazil, September 30 - October 2, 2002, Proceedings*, volume 2433 of *Lecture Notes in Computer Science*, pages 471–483. Springer, 2002. URL: https://doi.org/10.1007/3-540-45811-5_37.

[DPSZ12] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*, pages 643–662. Springer, 2012. URL: https://doi.org/10.1007/978-3-642-32009-5_38.

[DS11] Josiah Dykstra and A.T. Sherman. Understanding issues in cloud forensics: Two hypothetical case studies. *Journal of Network Forensics*, 3:19–31, 01 2011. URL: https://www.researchgate.net/publication/286192780_Understanding_Issues_in_cloud_forensics_Two_hypothetical_case_studies/link/569670be08ae1c427903c76b/download.

[DS13] Josiah Dykstra and Alan Sherman. Design and implementation of frost: Digital forensic tools for the openstack cloud computing platform. *Digital Investigation*, 10:S87–S95, 08 2013. doi:10.1016/j.diin.2013.06.010.

[EFG+15] Rolf Egert, Marc Fischlin, David Gens, Sven Jacob, Matthias Senker, and Jörn Tillmanns. Privately computing set-union and set-intersection cardinality via bloom filters. In Ernest Foo and Douglas Stebila, editors, *Information Security and Privacy - 20th Australasian Conference, ACISP 2015, Brisbane, QLD, Australia, June 29 - July 1, 2015, Proceedings*, volume 9144 of *Lecture Notes in Computer Science*, pages 413–430. Springer, 2015. URL: http://dx.doi.org/10.1007/978-3-319-19962-7_24.

[FD13] Bernardo Ferreira and Henrique Domingos. Searching private data in a cloud encrypted domain. In João Ferreira, João Magalhães, and Pável Calado, editors, *Open research Areas in Information Retrieval, OAIR '13, Lisbon, Portugal, May 15-17, 2013*, pages 165–172. ACM, 2013. URL: http://dl.acm.org/citation.cfm?id=2491783.

[FNP04] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In Cachin and Camenisch [CC04a], pages 1–19. URL: http://dx.doi.org/10.1007/978-3-540-24676-3_1.

[FR12] Matthieu Finiasz and Kannan Ramchandran. Private stream search at the same communication cost as a regular search: Role of LDPC codes. In *Proceedings of the 2012 IEEE International Symposium on Information Theory, ISIT 2012, Cambridge, MA, USA, July 1-6, 2012*, pages 2556–2560. IEEE, 2012. URL: http://dx.doi.org/10.1109/ISIT.2012.6283979.

[Gal02] Steven D. Galbraith. Elliptic curve paillier schemes. *J. Cryptology*, 15(2):129–138, 2002. URL: https://doi.org/10.1007/s00145-001-0015-6, doi:10.1007/s00145-001-0015-6.

[Gen09] Craig Gentry. *A Fully Homomorphic Encryption Scheme*. PhD thesis, Stanford, CA, USA, 2009. AAI3382729. URL: https://crypto.stanford.edu/craig/craig-thesis.pdf.

[GKL10] Jens Groth, Aggelos Kiayias, and Helger Lipmaa. Multi-query computationally-private information retrieval with constant communication rate. In Phong Q. Nguyen and David Pointcheval, editors, *Public Key Cryptography - PKC 2010, 13th International Conference on Practice and Theory in Public Key Cryptography, Paris, France, May 26-28, 2010. Proceedings*, volume 6056 of *Lecture Notes in Computer Science*, pages 107–123. Springer, 2010. URL: http://dx.doi.org/10.1007/978-3-642-13013-7_7.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In Harry R. Lewis, Barbara B. Simons, Walter A. Burkhard, and Lawrence H. Landweber, editors, *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA*, pages 365–377. ACM, 1982. URL: https://doi.org/10.1145/800070.802212.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984. URL: https://doi.org/10.1016/0022-0000(84)90070-9.

[Goh03] Eu-Jin Goh. Secure indexes. *IACR Cryptology ePrint Archive*, 2003:216, 2003. URL: http://eprint.iacr.org/2003/216.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 197–206. ACM, 2008. URL: https://doi.org/10.1145/1374376.1374407.

[GSG12] George Grispos, Tim Storer, and William Bradley Glisson. Calm before the storm: The challenges of cloud computing in digital

forensics. *IJDCF*, 4(2):28–48, 2012. URL: https://doi.org/10.4018/jdcf.2012040103.

[GWMA07] Joao Girão, Dirk Westhoff, Einar Mykletun, and Toshinori Araki. Tinypeds: Tiny persistent encrypted data storage in asynchronous wireless sensor networks. *Ad Hoc Networks*, 5(7):1073–1089, 2007. URL: https://doi.org/10.1016/j.adhoc.2006.05.004, doi:10.1016/j.adhoc.2006.05.004.

[GWS05] Joao Girão, Dirk Westhoff, and Markus Schneider. CDA: concealed data aggregation for reverse multicast traffic in wireless sensor networks. In *Proceedings of IEEE International Conference on Communications, ICC 2005, Seoul, Korea, 16-20 May 2005*, pages 3044–3049. IEEE, 2005. URL: https://doi.org/10.1109/ICC.2005.1494953.

[HG13] Yizhou Huang and Ian Goldberg. Outsourced private information retrieval. In Ahmad-Reza Sadeghi and Sara Foresti, editors, *Proceedings of the 12th annual ACM Workshop on Privacy in the Electronic Society, WPES 2013, Berlin, Germany, November 4, 2013*, pages 119–130. ACM, 2013. URL: http://doi.acm.org/10.1145/2517840.2517854.

[HSB19] Chris Hoofnagle, Bart Sloot, and Frederik Borgesius. The european union general data protection regulation: What it is and what it means. *Information & Communications Technology Law*, 28:1–34, 02 2019. doi:10.1080/13600834.2019.1573501.

[HvdSB18] Chris Jay Hoofnagle, Bart van der Sloot, and Frederik Zuiderveen Borgesius. The european union general data protection regulation: What it is and what it means. In *UC Berkeley Public Law Research Paper*, 2018. URL: http://dx.doi.org/10.2139/ssrn.3254511.

[HW06] Susan Hohenberger and Stephen A. Weis. Honest-verifier private disjointness testing without random oracles. In George Danezis and Philippe Golle, editors, *Privacy Enhancing Technologies, 6th International Workshop, PET 2006, Cambridge, UK, June 28-30, 2006, Revised Selected Papers*, volume 4258 of *Lecture Notes in Computer Science*, pages 277–294. Springer, 2006. URL: https://doi.org/10.1007/11957454_16.

[HX11] Shuai Han and Jianchuan Xing. Ensuring data storage security through a novel third party auditor scheme in cloud computing. In *2011 IEEE International Conference on Cloud Computing and Intelligence Systems, CCIS 2011, Beijing, China, September 15-17, 2011*, pages 264–268. IEEE, 2011. URL: https://doi.org/10.1109/CCIS.2011.6045072.

[JJK+13] Stanislaw Jarecki, Charanjit S. Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner. Outsourced symmetric private information retrieval. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 875–888. ACM, 2013. URL: http://doi.acm.org/10.1145/2508859.2516730.

[Ker11] Florian Kerschbaum. Public-key encrypted bloom filters with applications to supply chain integrity. In Yingjiu Li, editor, *Data and Applications Security and Privacy XXV - 25th Annual IFIP WG 11.3 Conference, DBSec 2011, Richmond, VA, USA, July 11-13, 2011. Proceedings*, volume 6818 of *Lecture Notes in Computer Science*, pages 60–75. Springer, 2011. URL: https://doi.org/10.1007/978-3-642-22348-8_7.

[Ker12] Florian Kerschbaum. Outsourced private set intersection using homomorphic encryption. In Heung Youl Youm and Yoojae Won, editors, *7th ACM Symposium on Information, Compuer and Communications Security, ASIACCS '12, Seoul, Korea, May 2-4, 2012*, pages 85–86. ACM, 2012. URL: http://doi.acm.org/10.1145/2414456.2414506.

[KK08] Murat Kantarcioglu and Onur Kardes. Privacy-preserving data mining in the malicious model. *IJICS*, 2(4):353–375, 2008. URL: http://dx.doi.org/10.1504/IJICS.2008.022488.

[KM05] Aggelos Kiayias and Antonina Mitrofanova. Testing disjointness of private datasets. In Andrew S. Patrick and Moti Yung, editors, *Financial Cryptography and Data Security, 9th International Conference, FC 2005, Roseau, The Commonwealth of Dominica, February 28 - March 3, 2005, Revised Papers*, volume 3570 of *Lecture Notes in Computer Science*, pages 109–124. Springer, 2005. URL: https://doi.org/10.1007/11507840_13.

[KNV09] Murat Kantarcioglu, Robert Nix, and Jaideep Vaidya. An efficient approximate protocol for privacy-preserving association rule mining. In Thanaruk Theeramunkong, Boonserm Kijsirikul, Nick Cercone, and Tu Bao Ho, editors, *Advances in Knowledge Discovery and Data Mining, 13th Pacific-Asia Conference, PAKDD 2009, Bangkok, Thailand, April 27-30, 2009, Proceedings*, volume 5476 of *Lecture Notes in Computer Science*, pages 515–524. Springer, 2009. URL: https://doi.org/10.1007/978-3-642-01307-2_48.

[KO97] Eyal Kushilevitz and Rafail Ostrovsky. Replication is NOT needed: SINGLE database, computationally-private information retrieval. In *38th Annual Symposium on Foundations of Computer Science, FOCS '97, Miami Beach, Florida, USA, October 19-22, 1997*,

pages 364–373. IEEE Computer Society, 1997. URL: https://doi.org/10.1109/SFCS.1997.646125.

[Kob87] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209, 1987.

[KS05] Lea Kissner and Dawn Xiaodong Song. Privacy-preserving set operations. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*, pages 241–257. Springer, 2005. URL: http://dx.doi.org/10.1007/11535218_15.

[LG] Zhijun Li and Guang Gong. Efficient data aggregation with secure bloom filter in wireless sensor networks. URL: http://cacr.uwaterloo.ca/techreports/2012/cacr2012-32.pdf.

[LH14] Der-Chyuan Lou and Hui-Feng Huang. An efficient t-out-of-n oblivious transfer for information security and privacy protection. *Int. J. Communication Systems*, 27(12):3759–3767, 2014. URL: http://dx.doi.org/10.1002/dac.2573.

[LHSC10] Y. Lin, B. He, H. Sun, and Y. Chen. Cds: Concealed data sorting scheme in wireless sensor networks. In *2010 International Computer Symposium (ICS2010)*, pages 370–375, Dec 2010. doi:10.1109/COMPSYM.2010.5685484.

[LP11] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for lwe-based encryption. In Aggelos Kiayias, editor, *Topics in Cryptology - CT-RSA 2011 - The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, 2011*, Lecture Notes in Computer Science. Springer, 2011.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2010. URL: http://dx.doi.org/10.1007/978-3-642-13190-5_1.

[LRW14] Jose M. Lopez, Thomas Rübsamen, and Dirk Westhoff. Privacy-friendly cloud audits with somewhat homomorphic and searchable encryption. In *I4CS*, pages 95–103, 2014. URL: http://dx.doi.org/10.1109/I4CS.2014.6860559.

[LTS16] Sebastian Lins, Heiner Teigeler, and Ali Sunyaev. Towards a bright future: Enhancing diffusion of continuous cloud service auditing by third parties. In *24th European Conference on Information Systems, ECIS 2016, Istanbul, Turkey, June 12-15, 2016*, page Research Paper 130, 2016. URL: http://aisel.aisnet.org/ecis2016_rp/130.

[LYC+06] Pierre K. Y. Lai, Siu-Ming Yiu, K. P. Chow, C. F. Chong, and Lucas Chi Kwong Hui. An efficient bloom filter based solution for multiparty private matching. In Hamid R. Arabnia and Selim Aissi, editors, *Proceedings of the 2006 International Conference on Security & Management, SAM 2006, Las Vegas, Nevada, USA, June 26-29, 2006*, pages 286–292. CSREA Press, 2006. URL: https://dblp.org/rec/bib/conf/csreaSAM/LaiYCCH06.

[Lyn07] B. Lynn. *On the implementation of pairing-based cryptosystems*. Stanford University, 2007. URL: https://books.google.de/books?id=sy0hAQAAIAAJ.

[LZDT09] Na Li, Nan Zhang, Sajal K. Das, and Bhavani M. Thuraisingham. Privacy preservation in wireless sensor networks: A state-of-the-art survey. *Ad Hoc Networks*, 7(8):1501–1514, 2009. URL: https://doi.org/10.1016/j.adhoc.2009.04.009.

[LZLY13] Zehui Li, Ruoqing Zhang, Zichen Li, and Yatao Yang. An efficient massive evidence storage and retrieval scheme in encrypted database. pages 25–25, 01 2013. doi:10.1049/cp.2013.2469.

[MGW06] Einar Mykletun, Joao Girão, and Dirk Westhoff. Public key based cryptoschemes for data concealment in wireless sensor networks. In *Proceedings of IEEE International Conference on Communications, ICC 2006, Istanbul, Turkey, 11-15 June 2006*, pages 2288–2295. IEEE, 2006. URL: https://doi.org/10.1109/ICC.2006.255111.

[Mic10] Daniele Micciancio. Duality in lattice cryptography. In *Public key cryptography*, page 2, 2010. URL: https://cseweb.ucsd.edu/~daniele/papers/DualitySlides.pdf.

[MS17] Anup Kumar Maurya and V. N. Sastry. Secure and efficient authenticated key exchange mechanism for wireless sensor networks and internet of things using bloom filter. In *3rd IEEE International Conference on Collaboration and Internet Computing, CIC 2017, San Jose, CA, USA, October 15-17, 2017*, pages 173–180. IEEE Computer Society, 2017. URL: https://doi.org/10.1109/CIC.2017.00032.

[MZV02] Yi Mu, Junqi Zhang, and Vijay Varadharajan. m out of n oblivious transfer. In Lynn Margaret Batten and Jennifer Seberry, editors, *Information Security and Privacy, 7th Australian Conference, ACISP 2002, Melbourne, Australia, July 3-5, 2002, Proceedings*, volume 2384 of *Lecture Notes in Computer Science*, pages 395–405. Springer, 2002. URL: http://dx.doi.org/10.1007/3-540-45450-0_30.

[NK09] Ryo Nojima and Youki Kadobayashi. Cryptographically secure bloom-filters. *Trans. Data Privacy*, 2(2):131–139, 2009. URL: http://www.tdp.cat/issues/abs.a015a09.php.

[NLV11] Michael Naehrig, Kristin E. Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In Christian Cachin and Thomas Ristenpart, editors, *Proceedings of the 3rd ACM Cloud Computing Security Workshop, CCSW 2011, Chicago, IL, USA, October 21, 2011*, pages 113–124. ACM, 2011. URL: https://dl.acm.org/citation.cfm?id=2046682.

[NS98] David Naccache and Jacques Stern. A new public key cryptosystem based on higher residues. In Li Gong and Michael K. Reiter, editors, *CCS '98, Proceedings of the 5th ACM Conference on Computer and Communications Security, San Francisco, CA, USA, November 3-5, 1998.*, pages 59–66. ACM, 1998. URL: https://doi.org/10.1145/288090.288106.

[OI07] Rafail Ostrovsky and William E. Skeith III. A survey of single-database private information retrieval: Techniques and applications. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *Public Key Cryptography - PKC 2007, 10th International Conference on Practice and Theory in Public-Key Cryptography, Beijing, China, April 16-20, 2007, Proceedings*, volume 4450 of *Lecture Notes in Computer Science*, pages 393–411. Springer, 2007. URL: http://dx.doi.org/10.1007/978-3-540-71677-8_26.

[ÖM07] Melek Önen and Refik Molva. Secure data aggregation with multiple encryption. In Koen Langendoen and Thiemo Voigt, editors, *Wireless Sensor Networks, 4th European Conference, EWSN 2007, Delft, The Netherlands, January 29-31, 2007, Proceedings*, volume 4373 of *Lecture Notes in Computer Science*, pages 117–132. Springer, 2007. URL: https://doi.org/10.1007/978-3-540-69830-2_8.

[OU98] Tatsuaki Okamoto and Shigenori Uchiyama. A new public-key cryptosystem as secure as factoring. In Kaisa Nyberg, editor, *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*, volume 1403 of

Lecture Notes in Computer Science, pages 308–318. Springer, 1998. URL: https://doi.org/10.1007/BFb0054135.

[OX11] Suat Ozdemir and Yang Xiao. Integrity protecting hierarchical concealed data aggregation for wireless sensor networks. *Computer Networks*, 55(8):1735–1746, 2011. URL: https://doi.org/10.1016/j.comnet.2011.01.006, doi:10.1016/j.comnet.2011.01.006.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999. URL: http://dx.doi.org/10.1007/3-540-48910-X_16.

[Pai00] Pascal Paillier. Trapdooring discrete logarithms on elliptic curves over rings. In Tatsuaki Okamoto, editor, *Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings*, volume 1976 of *Lecture Notes in Computer Science*, pages 573–584. Springer, 2000. URL: https://doi.org/10.1007/3-540-44448-3_44.

[PH78] Stephen C. Pohlig and Martin E. Hellman. An improved algorithm for computing logarithms over gf(p) and its cryptographic significance (corresp.). *IEEE Trans. Information Theory*, 24(1):106–110, 1978. URL: https://doi.org/10.1109/TIT.1978.1055817.

[PHMN16] Liliana Pasquale, Sorren Hanvey, Mark Mcgloin, and Bashar Nuseibeh. Adaptive evidence collection in the cloud using attack scenarios. *Computers & Security*, 59:236–254, 2016. URL: https://doi.org/10.1016/j.cose.2016.03.001.

[PJ16] Keyur Parmar and Devesh C. Jinwala. Concealed data aggregation in wireless sensor networks: A comprehensive survey. *Computer Networks*, 103:207–227, 2016. URL: https://doi.org/10.1016/j.comnet.2016.04.013, doi:10.1016/j.comnet.2016.04.013.

[PKP11] Stavros Papadopoulos, Aggelos Kiayias, and Dimitris Papadias. Secure and efficient in-network processing of exact SUM queries. In Serge Abiteboul, Klemens Böhm, Christoph Koch, and Kian-Lee Tan, editors, *Proceedings of the 27th International Conference on Data Engineering, ICDE 2011, April 11-16, 2011, Hannover, Germany*, pages 517–528. IEEE Computer Society, 2011. URL: https://doi.org/10.1109/ICDE.2011.5767886.

[PMM09] Roberto Di Pietro, Pietro Michiardi, and Refik Molva. Confidentiality and integrity for data aggregation in WSN using peer monitoring. *Security and Communication Networks*, 2(2):181–194, 2009. URL: https://doi.org/10.1002/sec.93.

[Pol78] John M Pollard. Monte carlo methods for index computation (mod p). *Mathematics of computation*, 32(143):918–924, 1978.

[PPL07] S. Peter, K. Piotrowski, and P. Langendoerfer. On concealed data aggregation for wsns. In *2007 4th IEEE Consumer Communications and Networking Conference*, pages 192–196, Jan 2007. doi:10.1109/CCNC.2007.45.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. *A Framework for Efficient and Composable Oblivious Transfer*, pages 554–571. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. doi:10.1007/978-3-540-85174-5_31.

[PW10] Stephen Pearson and Richard Watson. *Digital triage forensics: processing the digital crime scene.* Syngress, 2010. URL: https://www.sciencedirect.com/book/9781597495967/digital-triage-forensics.

[PWC10] Steffen Peter, Dirk Westhoff, and Claude Castelluccia. A survey on the encryption of convergecast traffic with in-network processing. *IEEE Trans. Dependable Sec. Comput.*, 7(1):20–34, 2010. URL: https://doi.org/10.1109/TDSC.2008.23.

[QLW07] Ling Qiu, Yingjiu Li, and Xintao Wu. Preserving privacy in association rule mining with bloom filters. *J. Intell. Inf. Syst.*, 29(3):253–278, 2007. URL: https://doi.org/10.1007/s10844-006-0018-8.

[Rab05] Michael O. Rabin. How to exchange secrets with oblivious transfer. *IACR Cryptology ePrint Archive*, 2005:187, 2005. URL: http://eprint.iacr.org/2005/187.

[RBI17] Syed S. Rizvi, Trent A. Bolish, and Joseph R. Pfeffer III. Security evaluation of cloud service providers using third party auditors. In Hani Hamdan, Djallel Eddine Boubiche, Homero Toral-Cruz, Sedat Akleylek, and Hamid Mcheick, editors, *Proceedings of the Second International Conference on Internet of things and Cloud Computing, ICC 2017, Cambridge, United Kingdom, March 22-23, 2017*, pages 106:1–106:6. ACM, 2017. URL: https://doi.org/10.1145/3018896.3025154.

[RCN+18] Sundaresan Rajasekaran, Harpreet Singh Chawla, Zhen Ni, Neel Shah, Emery Berger, and Timothy Wood. CRIMES: using evidence to secure the cloud. In Paulo Ferreira and Liuba Shrira, editors,

Proceedings of the 19th International Middleware Conference, Middleware 2018, Rennes, France, December 10-14, 2018, pages 40–52. ACM, 2018. URL: https://doi.org/10.1145/3274808.3274812.

[RD14] Catherine M. S. Redfield and Hiroyuki Date. Gringotts: Securing data for digital evidence. In *35. IEEE Security and Privacy Workshops, SPW 2014, San Jose, CA, USA, May 17-18, 2014*, pages 10–17. IEEE Computer Society, 2014. URL: https://doi.org/10.1109/SPW.2014.11.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):34:1–34:40, September 2009. URL: http://doi.acm.org/10.1145/1568318.1568324.

[RKP07] Shu Qin Ren, Dong Seong Kim, and Jong Sou Park. A secure data aggregation scheme for wireless sensor networks. In Parimala Thulasiraman, Xubin He, Tony Li Xu, Mieso K. Denko, Ruppa K. Thulasiram, and Laurence Tianruo Yang, editors, *Frontiers of High Performance Computing and Networking ISPA 2007 Workshops, ISPA 2007 International Workshops SSDSN, UPWN, WISH, SGC, ParDMCom, HiPCoMB, and IST-AWSN Niagara Falls, Canada, August 28 - September 1, 2007, Proceedings*, volume 4743 of *Lecture Notes in Computer Science*, pages 32–40. Springer, 2007. URL: https://doi.org/10.1007/978-3-540-74767-3_4.

[RM13] M. B. O. Rafik and F. Mohammed. Sa-spkc: Secure and efficient aggregation scheme for wireless sensor networks using stateful public key cryptography. In *2013 11th International Symposium on Programming and Systems (ISPS)*, pages 96–102, April 2013. doi:10.1109/ISPS.2013.6581500.

[RPR15] Thomas Rübsamen, Tobias Pulls, and Christoph Reich. Security and privacy preservation of evidence in cloud accountability audits. In Markus Helfert, Víctor Méndez Muñoz, and Donald Ferguson, editors, *Cloud Computing and Services Science - 5th International Conference, CLOSER 2015, Lisbon, Portugal, May 20-22, 2015, Revised Selected Papers*, volume 581 of *Communications in Computer and Information Science*, pages 95–114. Springer, 2015. URL: https://doi.org/10.1007/978-3-319-29582-4_6.

[RR13] Thomas Rübsamen and Christoph Reich. Supporting cloud accountability by collecting evidence using audit agents. In *IEEE 5th International Conference on Cloud Computing Technology and Science, CloudCom 2013, Bristol, United Kingdom, December 2-5, 2013, Volume 1*, pages 185–190. IEEE Computer Society, 2013. URL: http://dx.doi.org/10.1109/CloudCom.2013.32.

[Rüb16] Thomas Rübsamen. *Evidence-based accountability audits for cloud computing*. PhD thesis, University of Plymouth, UK,

2016. URL: http://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.698105.

[RWB10] Denis J. Reilly, Chris Wren, and Tom Berry. Cloud computing: Forensic challenges for law enforcement. In *5th International Conference for Internet Technology and Secured Transactions, ICITST 2010, London, United Kingdom, November 8-10, 2010*, pages 1–7. IEEE, 2010. URL: http://ieeexplore.ieee.org/document/5678033/.

[SB07] S. Joshua Swamidass and Pierre Baldi. Mathematical correction for fingerprint similarity measures to improve chemical retrieval. *Journal of Chemical Information and Modeling*, 47(3):952–964, 2007. PMID: 17444629. URL: https://doi.org/10.1021/ci600526a.

[SBR09] Rainer Schnell, Tobias Bachteler, and Jörg Reiher. Privacy-preserving record linkage using bloom filters. *BMC Med. Inf. & Decision Making*, 9:41, 2009. URL: https://doi.org/10.1186/1472-6947-9-41.

[SC06] Bradley Schatz and Andrew J. Clark. An open architecture for digital evidence integration. In Andrew J. Clark, Mark McPherson, and George M. Mohay, editors, *AusCERT Asia Pacific Information Technology Security Conference : Refereed R&D Stream*, pages 15–29, Gold Coast, Queensland, May 2006. University of Queensland. URL: http://eprints.qut.edu.au/21119/.

[Sen13] Jaydip Sen. A survey on security and privacy protocols for cognitive wireless sensor networks. *CoRR*, abs/1308.0682, 2013. URL: http://arxiv.org/abs/1308.0682, arXiv:1308.0682.

[SFM11] Clay Shields, Ophir Frieder, and Mark Maloof. A system for the proactive, continuous, and efficient collection of digital forensic evidence. *Digital Investigation*, 8:S3 – S13, 2011. The Proceedings of the Eleventh Annual DFRWS Conference. URL: http://www.sciencedirect.com/science/article/pii/S1742287611000260.

[SGM02] Gunther Schadow, Shaun J. Grannis, and Clement J. McDonald. Discussion paper: Privacy-preserving distributed queries for a clinical case research network. In *Proceedings of the IEEE International Conference on Privacy, Security and Data Mining - Volume 14*, CRPIT '14, pages 55–65, Darlinghurst, Australia, Australia, 2002. Australian Computer Society, Inc. URL: http://dl.acm.org/citation.cfm?id=850782.850790.

[SJM13] Bharath K. Samanthula, Wei Jiang, and Sanjay Madria. A probabilistic encryption based MIN/MAX computation in wireless sensor networks. In *2013 IEEE 14th International Conference on*

Mobile Data Management, Milan, Italy, June 3-6, 2013 - Volume 1, pages 77–86. IEEE Computer Society, 2013. URL: https://doi.org/10.1109/MDM.2013.18.

[SYY99] Tomas Sander, Adam Young, and Moti Yung. Non-interactive cryptocomputing for nc1. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, FOCS '99, pages 554–, Washington, DC, USA, 1999. IEEE Computer Society. URL: http://dl.acm.org/citation.cfm?id=795665.796534.

[THGL11] Mark John Taylor, John Haggerty, David Gresty, and David J. Lamb. Forensic investigation of cloud computing systems. *Network Security*, 2011(3):4–10, 2011. URL: https://doi.org/10.1016/S1353-4858(11)70024-1.

[TKW18] Louis Tajan, Moritz Kaumanns, and Dirk Westhoff. Pre-computing appropriate parameters: How to accelerate somewhat homomorphic encryption for cloud auditing. In *9th IFIP International Conference on New Technologies, Mobility and Security, NTMS 2018, Paris, France, February 26-28, 2018*, pages 1–6. IEEE, 2018. URL: https://doi.org/10.1109/NTMS.2018.8328713.

[TNS+17] Anum Talpur, Thomas Newe, Faisal Karim Shaikh, Adil Amjad Sheikh, Emad A. Felemban, and Abdelmajid Khelil. Bloom filter based data collection algorithm for wireless sensor networks. In *2017 International Conference on Information Networking, ICOIN 2017, Da Nang, Vietnam, January 11-13, 2017*, pages 354–359. IEEE, 2017. URL: https://doi.org/10.1109/ICOIN.2017.7899458.

[TW19] Louis Tajan and Dirk Westhoff. Retrospective tracking of suspects in gdpr conform mobile access networks datasets. In *Proceedings of the Central European Cybersecurity Conference 2019, CECC 2019, Munich, Germany, November 14-15, 2019*, pages 5:1–5:6. ACM, 2019. URL: https://doi.org/10.1145/3360664.3360680.

[TW20] Louis Tajan and Dirk Westhoff. Approach for GDPR compliant detection of COVID-19 infection chains. *CoRR*, abs/2007.08248, 2020. URL: https://arxiv.org/abs/2007.08248, arXiv:2007.08248.

[TWA19] Louis Tajan, Dirk Westhoff, and Frederik Armknecht. Private set relations with bloom filters for outsourced SLA validation. *IACR Cryptology ePrint Archive*, 2019:993, 2019. URL: https://eprint.iacr.org/2019/993.

[TWA20] Louis Tajan, Dirk Westhoff, and Frederik Armknecht. Solving set relations with secure bloom filters keeping cardinality private. In

Mohammad S. Obaidat and Pierangela Samarati, editors, *Proceedings of the 17th International Joint Conference on e-Business and Telecommunications, ICETE 2020 - Volume 2: SECRYPT, Paris, France, July 8-10, 2020*, pages 187–197. SciTePress, 2020. URL: https://doi.org/10.5220/0007932301870197.

[TWRA16] Louis Tajan, Dirk Westhoff, Christian A. Reuter, and Frederik Armknecht. Private information retrieval and searchable encryption for privacy-preserving multi-client cloud auditing. In *11th International Conference for Internet Technology and Secured Transactions, ICITST 2016, Barcelona, Spain, December 5-7, 2016*, pages 162–169. IEEE, 2016. URL: http://dx.doi.org/10.1109/ICITST.2016.7856690.

[UHW] Osman Ugus, Alban Hessler, and Dirk Westhoff. Performance of additive homomorphic ec-elgamal encryption for tinypeds. *6. Fachgespräch Sensornetzwerke*, page 55.

[Vau20] Serge Vaudenay. Centralized or decentralized? the contact tracing dilemma. Cryptology ePrint Archive, Report 2020/531, 2020. https://eprint.iacr.org/2020/531.

[vN93] John von Neumann. First draft of a report on the EDVAC. *IEEE Annals of the History of Computing*, 15(4):27–75, 1993. URL: https://doi.org/10.1109/85.238389, doi:10.1109/85.238389.

[Wan15] Qinglong Wang. Efficient k-out-of-n oblivious transfer protocol. *IACR Cryptology ePrint Archive*, 2015:218, 2015. URL: http://eprint.iacr.org/2015/218.

[WGA06] Dirk Westhoff, Joao Girão, and Mithun Acharya. Concealed data aggregation for reverse multicast traffic in sensor networks: Encryption, key distribution, and routing adaptation. *IEEE Trans. Mob. Comput.*, 5(10):1417–1431, 2006. URL: https://doi.org/10.1109/TMC.2006.144.

[WMLD04] Yongdong Wu, Di Ma, Tieyan Li, and Robert H Deng. Classify encrypted data in wireless sensor networks. In *IEEE 60th Vehicular Technology Conference, 2004. VTC2004-Fall. 2004*, volume 5, pages 3236–3239. IEEE, 2004.

[Wol09] Stephen D. Wolthusen. Overcast: Forensic discovery in cloud environments. In Oliver Goebel, Ralf Ehlert, Sandra Frings, Detlef Günther, Holger Morgenstern, and Dirk Schadt, editors, *IMF 2009, Fifth International Conference on IT Security Incident Management and IT Forensics, Stuttgart, Germany, 15-17 September 2009*, pages 3–9. IEEE Computer Society, 2009. URL: https://doi.org/10.1109/IMF.2009.21.

[YWPZ08] Qingsong Ye, Huaxiong Wang, Josef Pieprzyk, and Xian-Mo Zhang. Efficient disjointness tests for private datasets. In Yi Mu, Willy Susilo, and Jennifer Seberry, editors, *Information Security and Privacy, 13th Australasian Conference, ACISP 2008, Wollongong, Australia, July 7-9, 2008, Proceedings*, volume 5107 of *Lecture Notes in Computer Science*, pages 155–169. Springer, 2008. URL: https://doi.org/10.1007/978-3-540-70500-0_12.

[YZFC15] Qiuwei Yang, Xiaogang Zhu, Hongjuan Fu, and Xiqiang Che. Survey of security technologies on wireless sensor networks. *J. Sensors*, 2015:842392:1–842392:9, 2015. URL: https://doi.org/10.1155/2015/842392.

[ZPH+17] Jan Henrik Ziegeldorf, Jan Pennekamp, David Hellmanns, Felix Schwinger, Ike Kunze, Martin Henze, Jens Hiller, Roman Matzutt, and Klaus Wehrle. Bloom: Bloom filter based oblivious outsourced matchings. *BMC Medical Genomics*, 10(2):44, Jul 2017. URL: https://doi.org/10.1186/s12920-017-0277-y.

[ZYH14] Qiang Zhou, Geng Yang, and Liwen He. A secure-enhanced data aggregation based on ECC in wireless sensor networks. *Sensors*, 14(4):6701–6721, 2014. URL: https://doi.org/10.3390/s140406701, doi:10.3390/s140406701.

[ZYS+14] Hongli Zhang, Lin Ye, Jiantao Shi, Xiaojiang Du, and Mohsen Guizani. Verifying cloud service-level agreement by a third-party auditor. *Security and Communication Networks*, 7(3):492–502, 2014. URL: https://doi.org/10.1002/sec.740.