THESIS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Secure and Privacy-Preserving Cloud-Assisted Computing

Georgia Tsaloli

Department of Computer Science & Engineering
Chalmers University of Technology
Gothenburg, Sweden, 2022

Secure and Privacy-Preserving Cloud-Assisted Computing

Georgia Tsaloli

ISBN 978-91-7905-664-3
Doktorsavhandlingar vid Chalmers tekniska högskola
Ny serie nr 5130
ISSN 0346-718X

Technical report no 215D
Department of Computer Science & Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 (0)31-772 1000

Author email: gtsaloli@gmail.com, tsaloli@chalmers.se

Gothenburg, Sweden, 2022

Secure and Privacy-Preserving Cloud-Assisted Computing
Georgia Tsaloli
Department of Computer Science & Engineering
Chalmers University of Technology

Abstract

Smart devices such as smartphones, wearables, and smart appliances collect significant amounts of data and transmit them over the network forming the Internet of Things (IoT). Many applications in our daily lives (*e.g.*, health, smart grid, traffic monitoring) involve IoT devices that often have low computational capabilities. Subsequently, powerful cloud servers are employed to process the data collected from these devices. Nevertheless, security and privacy concerns arise in cloud-assisted computing settings.

Collected data can be sensitive, and it is essential to protect their confidentiality. Additionally, outsourcing computations to untrusted cloud servers creates the need to ensure that servers perform the computations as requested and that any misbehavior can be detected, safeguarding security. Cryptographic primitives and protocols are the foundation to design secure and privacy-preserving solutions that address these challenges. This thesis focuses on providing privacy and security guarantees when outsourcing heavy computations on sensitive data to untrusted cloud servers. More concretely, this work:

(a) provides solutions for outsourcing the secure computation of the sum and the product functions in the multi-server, multi-client setting, protecting the sensitive data of the data owners, even against potentially untrusted cloud servers;

(b) provides integrity guarantees for the proposed protocols, by enabling anyone to verify the correctness of the computed function values. More precisely, the employed servers or the clients (depending on the proposed solution) provide specific values which are the proofs that the computed results are correct;

(c) designs decentralized settings, where multiple cloud servers are employed to perform the requested computations as opposed to relying on a single server that might fail or lose connection;

(d) suggests ways to protect individual privacy and provide integrity. More precisely, we propose a verifiable differentially private solution that provides verifiability and avoids any leakage of information regardless of the participation of some individual's sensitive data in the computation or not.

Keywords: secret sharing, cloud computing, differential privacy, privacy-preservation, verifiability, secure aggregation, privacy, decentralization

List of publications

This thesis is based on the following publications:

Paper A [TLM18] "Verifiable Homomorphic Secret Sharing"
Georgia Tsaloli, Bei Liang, Aikaterini Mitrokotsa
12-*th International Conference on Provable Security (ProvSec) 2018.*

Paper B [TM19] "Differential Privacy meets Verifiable Computation: Achieving Strong Privacy and Integrity Guarantees"
Georgia Tsaloli, Aikaterini Mitrokotsa
16-*th International Conference on Security and Cryptography (SECRYPT) 2019.*

Paper C [TBM20] "Practical and Provably Secure Distributed Aggregation: Verifiable Additive Homomorphic Secret Sharing"
Georgia Tsaloli , Gustavo Banegas, Aikaterini Mitrokotsa
Cryptography Open Access Journal 2020.
The above is an extended and elaborated version of the work that previously appeared in:
[TM20] "Sum It Up: Verifiable Additive Homomorphic Secret Sharing"
Georgia Tsaloli , Aikaterini Mitrokotsa
22-*th International Conference on Information Security and Cryptology (ICISC), 2019.*

Paper D [TLB+21] "DEVA: Decentralized, Verifiable Secure Aggregation for Privacy-Preserving Learning"
Georgia Tsaloli, Bei Liang, Carlo Brunetta, Gustavo Banegas, Aikaterini Mitrokotsa
24-*th Information Security Conference (ISC) 2021.*

Paper E [BTL+21] "Non-Interactive, Secure Verifiable Aggregation for Decentralized, Privacy-Preserving Learning"
Carlo Brunetta, **Georgia Tsaloli**, Bei Liang, Gustavo Banegas, Aikaterini Mitrokotsa
26-*th Australasian Conference on Information Security and Privacy (ACISP) 2021.*

Other publications

The following publications were published during my PhD studies. However, they are not appended to this thesis.

[LKT+20] "AnonFACES: Anonymizing Faces Adjusted to Constraints on Efficacy and Security"
Minh-Ha Le, Md Sakib Nizam Khan, **Georgia Tsaloli**, Niklas Carlsson, and Sonja Buchegger
19-*th ACM Workshop on Privacy in the Electronic Society (WPES), 2020*

[TLMD22] "Verifiable, Secure and Energy-Efficient Private Data Aggregation in Wireless Sensor Networks"
Georgia Tsaloli, Alejandro Lancho, Aikaterini Mitrokotsa and Giuseppe Durisi
27-th ACM Symposium on Access Control Models and Technologies (SACMAT), 2022

This thesis is dedicated to

my parents, Chrysoula and Manolis,

for their endless support.

" Óso ipárhei ouranós na petás. "
- My mom.

Acknowledgments

This part of the thesis is for all of you that made this work possible, and my day-to-day life better.

Firstly, I want to thank my supervisor Katerina for all the guidance and support throughout my Ph.D. journey. Five years (almost) is a lot, and I believe we have both managed to work well together despite the challenges. Thank you for your ideas, your flexibility towards all different cases, and your encouragement along the way. It has been an enjoyable experience with a lot of knowledge.

I also wish to thank my co-supervisor, Dave, and my Ph.D. examiner, Andrei. Thank you both for making my Ph.D. studies smooth and always having clear and concrete expectations. I also got help while looking for internship opportunities from Andrei; thank you for making this easier for me.

Thank you to Prof. Claudio Orlandi for accepting to be the faculty opponent during my defense, as well as the grading committee members Prof. Rei Safavi-Naini, Prof. Charalampos Papamanthou, Prof. Pascal Lafourcade, and Prof. Marija Furdek Prekratic for reviewing my work and participating in my defense.

Next, comes my Ph.D. buddy, Carlo. I give this title to you since you deserve it in many ways. Thank you for successfully handling all the craziness I carry as a person, for surviving business trips eating fancy food (despite you would very much have liked to cook it yourself instead), and especially for always showing me other perspectives throughout my Ph.D. studies. I hope we meet soon; I need to check how well your Greek has evolved :)

A special thank you goes to Bei. Having a hard-working person to brainstorm about research was a very good and motivating start for my studies. Despite needing some "time off" in our collaboration, I am very happy we did work together again and grateful for having your help. I am looking forward to visiting you in Beijing, to get the real Asian experience.

I feel lucky having met so nice colleagues and friends in Chalmers who made work a fun place for me. To the “old batch" Babis, Ivan, Aras, Valentin, Oliver, Fazeleh, Hannah, Romaric, Vagelis, Petros and Stavros, with whom we spent several Fridays after work until most of you “retired" :) To Nadja and Christos, thank you for being amazing office mates. Christos joined me also in the new office, making the final “post"-pandemic time enjoyable. Thank you to Dimitris who is a true neighbor, a skiing and a sailing partner, *i.e.,* a companion for all seasons. Thank you to Uddipana for spending time together in Switzerland, getting to know you better; it has been great. Thank you to Iulia with whom I shared a lot of fun times and discussions

during our WASP trips, back then when we were still young :) It is now time to thank Thomas, who is my coffee advisor. I started my Ph.D. drinking orange juice, tea, or hot chocolate, and I am now finishing having a proper espresso machine at home and drinking several cups of coffee per day; that is your fault :) Thank you for being a great complainer, for motivating me to go running (I will not), and, except for these, being an awesome person. “I am proud of you, great job". Thank you to past and present colleagues Bastian, Kalle, Gustavo, Amir, Beshr, Nasser, Francisco, Aljoscha, Kim, Huaifeng, Masoud, and Joris for creating a pleasant atmosphere at work and/or getting some Fika breaks together. Thank you also to Jenit, Nan, Sayantan, Johannes, Adrien, and Eriane whom I met in person when in St. Gallen; thank you for the nice time during my short visit. Thank you to Tomas, Magnus, Marina, Phillipas, Vincenzo, Ahmed, Elad, Olaf, Pedro, Miquel, and Yiannis S. for making Chalmers a nice workplace, and thank you to my co-authors Melek, Niklas, Sonja, Minh-Ha, Sakib, Hiraku, Alejandro, and Giuseppe for the interesting collaborations. Special thanks to Tomas who is the best manager you would wish to have. Last but not least, thank you to Eva, Marianne, Rebecca, Monica, Michael, Lasse, Rolf, and Clara. Thanks to you, everything works perfectly, you are all great.

Gladly, there has been a life outside work, and I was lucky to find amazing friends, with whom I have shared incredible times. Thank you to Vasiliki, Eva, Penelope, Maria, Yiannis, Dimitris T., Dimitris P., Iosif, Manos, and, of course, the little Christina, for all the philosophical discussions, trips, adventures, and beautiful moments we have spent together. Thank you to Santiago and Eirini, who are the masters of the “last-minute trips", and with whom I have shared crazy moments, endless discussions, and experiences; the best is to come. Thank you to Suvi for motivating my ski adventures, my walks in the nature, and many more. I am grateful to have all of you in my life, and you are the reason I can still survive the weather of this country. Thank you!

I want to also thank the newcomers, Fotini and Fanis, for spending time together and all the dance crew who try to teach me Swedish, in all ways :)

For my family, Chrisoula, Manolis, and Konstantinos: Το ξέρω ότι μάλλον θα θέλατε να είμαι κάπου εκεί, δίπλα, παραδίπλα σε κάποια πόλη μερικά χιλιόμετρα πιο κοντά, αλλά νιώθω την αγάπη σας και από εδώ και πάντα. Σας ευχαριστώ για όλα, και εύχομαι να είστε περήφανοι και υγιείς για να χαρείτε την ησυχία σας :)

Finally, I want to thank my partner in crime, my love, Hamdy. Thank you for making me a better human, and for being always here, there, everywhere for me against all odds. You are one-of-a-kind, you sparkle, thank you ♡.

Georgia Tsaloli
Göteborg, June 2022

Contents

Thesis Overview

Introduction

Our societies communicate, share experiences and feelings, and exchange information every day. Smart electronic devices are used immensely in our digital lives and are utilized to control, schedule, or automate functions and generally enhance users' way of life. These devices have a network connection and can aggregate users' data. Cellphones, smartwatches, sensors, and smart TVs hold user information, store and process data that are transmitted to other smart devices or the "cloud". In this paradigm, there exist several security and privacy challenges, such as how to handle these data in a privacy-preserving way, and also whether to trust the "cloud" to treat data as expected. Data are uploaded to the "cloud" because these smart devices with increased connectivity, namely IoT devices, often have a very low computational power. Therefore, they require more powerful entities such as cloud servers to handle and process their data.

For example, many applications such as smart metering, environmental monitoring, or computing statistics involve heavy computations that utilize data coming from IoT devices/users. Considering the smart metering instance, sensors installed in several households can collect the electricity consumption of a given area, upload the measured data to the "cloud" and request computations to be made. Cloud servers are expected to perform the analysis and provide back their computed results depending on what is requested.

However, handling data in the context of IoT-based applications involves several risks. Specifically, data can often contain sensitive information (*e.g.,* health information, biometric data, individual routines, etc.) that need to remain secret and, therefore, the processing of those data must satisfy some confidentiality requirements. In the smart metering example, electricity sensors' measurements can reveal when people are home, which devices they are using, and when. Similarly, in other applications data leakages can reveal the identity of a person or enable third parties to impersonate someone.

Moreover, IoT devices, due to their low computational capabilities, assign heavy computations that they cannot perform on the device itself to external well-equipped cloud servers. Thus, another important concern is that outsourcing computations to the "cloud", *i.e.,* powerful computers hosted over a network (Internet), can be problematic. These cloud servers are unknown and possibly untrusted, which leads to security risks such as whether the cloud servers should see the data in the clear and also, whether the computation that they are expected to perform is properly

executed. In other words, untrusted servers might reveal the private data of individuals or intervene and make modifications to the computed results according to their preferences.

Furthermore, since often the aggregation in IoT-based applications involves data coming from a population of clients, it is realistic to consider multiple clients involved in the cloud computing setting. In the literature, there is a lot of work on the problem of outsourcing computations. There exist scenarios considering a single client outsourcing the computation to a single server [BFR13a, CKV10, FGP14, GGP10, PMT17, PRV12, TC14]. Then, some works address the setting with multiple clients [FMNP16, GKL+15], whereas considering multiple servers for a single client has also been considered [ACG+14, WYG+17]. However, besides servers being potentially untrusted and behaving maliciously, assigning the computations to a single server can cause single points of failure. Recently, we experienced that even huge companies like Facebook and its subsidiaries can, despite rarely, suddenly go offline for hours [Dou21, Ric21, Ale21]. Therefore, employing multiple servers is realistic, and often a necessity; in this context, the setting with multiple clients and multiple servers has received limited attention.

Hence, in this work, the following entities are considered: *(i)* Clients: multiple clients which own sensitive data and do not interact with each other. They want to outsource the computation of a function over their joint inputs to the "cloud"; *(ii)* Cloud: External cloud servers with high computational power that perform calculations on the aggregated data to compute the requested functions (*e.g.,* statistical results); *(iii)* Receiver(s): The results of the computations are provided to one or more receivers which may be the clients or other entities (*e.g.,* an electricity company, a doctor, or a researcher). The considered setting is depicted in Figure I.1.

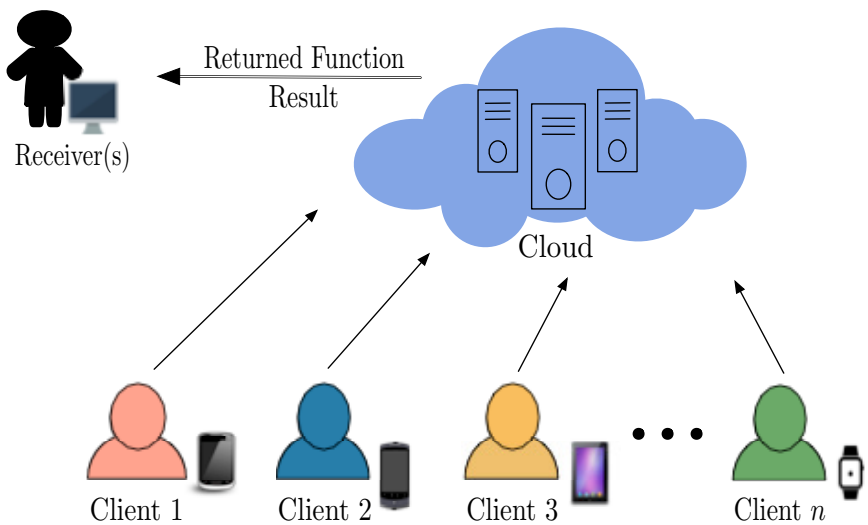

Figure I.1: Cloud servers collect data from multiple clients to calculate the requested function and return the results to the receiver(s).

The synopsis of the challenges that we encounter in the cloud computing paradigm, is as follows: *(i)* Security: cloud servers might be malicious and wish to modify the calculations and return wrong results. Furthermore, a single server might be under attack or out of service, causing single points of failure; *(ii)* Privacy: clients hold sensitive data that need to be protected, and moreover, even the actual computed result often needs to remain private; *(iii)* Limited resources: the aggregation is performed over large amounts of data that cannot be stored or processed on the resource-constraint devices and are then outsourced to cloud servers. Since servers might show malicious behavior, we need to provide guarantees about the correctness of the results, while not requiring more resources than the computation itself.

Thesis Scope The goals of this thesis are four-fold:

1. To design solutions for *outsourcing computations* to the cloud which safeguard that *sensitive data*, often coming from resource-constrained devices, *remain secret*. In other words, to enable the processing of the users' data [1] without compromising their privacy.
2. To construct mechanisms that ensure the *integrity of the computed results*, *i.e.,* provide *security guarantees* that the computed results have not been modified and correspond to the correct expected outputs.
3. To create *decentralized*[2] *systems* for cloud computations, allowing secure aggregation of users' data, while not relying on a single server. The aim is to *assign the computation into multiple servers*, thus, being able to overcome either a single server's failure or a set of one or more servers being untrusted.
4. To avoid any *leakage of information* about individuals in case auxiliary information can be combined with sensitive data.

Outline. This thesis is organized into two parts. The first part is the thesis overview. It consists of a high-level introduction, a background chapter that contains definitions and descriptions of concepts related to this thesis, a chapter introducing the thesis's objectives, and a chapter with a summary of the thesis's results and contributions. The second part is a collection of five papers (Chapters A-E) on secure and private cloud-assisted computing. In Chapters A and C, we explore the multi-client, multi-server scenario and provide constructions for securely computing the sum and the product of multiple secret inputs in the presence of potentially malicious cloud servers; providing also integrity guarantees for the computed output. In Chapter B, we examine the combination of differential privacy and verifiable computation in the context of outsourcing computations on sensitive datasets. Finally, in Chapters D and E, we design secure privacy-preserving aggregation protocols with direct application in the machine learning setting.

[1] Throughout the thesis the words "user" and "client" are used interchangeably.
[2] Throughout the thesis the words "decentralized" and "distributed" are used interchangeably.

II

Background

In this section, we present some introductory material related to the content of this thesis. We get more familiar to concepts such as secret sharing and homomorphic secret sharing, verifiable computation, and differential privacy.

II.1 Secret Sharing Schemes

Consider a large company that wants to securely store the “master" password of their vault on the cloud. Firstly, cloud providers might be untrusted. A way to protect against potentially untrusted cloud servers could be to encrypt the password [CLW+16, WCH+15]; however, this works only partially. More precisely, keeping multiple copies (for improving reliability) or only a single one for maximizing confidentiality when using encryption, cannot deal with problems such as data loss in a single server or availability failure. Secret sharing allows data holders to keep their information private and store them securely.

Likewise, personal information such as sensor data, passwords, biometric data, or other sensitive data can be protected using secret sharing schemes [Bei11, BC94]. Shamir [Sha79] and Blakley [BLA79] designed the first secret sharing schemes, based on polynomial interpolation and hyperplanes intersection, respectively. Secret sharing is a cryptographic technique that enables data owners to securely distribute pieces of their personal information among a distributed group or network, safeguarding their private data. Consider a data owner that holds a secret input x and m parties that comprise a network. Using a secret sharing scheme, a dealer (data owner or client) splits the secret x into m fragments, *i.e.,* m values $x_1, \ldots, x_m$. More precisely, the goal is to satisfy the following properties [BR07, CK93]:

1. any set of more than t fragments are enough to reconstruct the secret input x;
2. it is impossible to know the secret x, if knowing any t or fewer fragments $x_i, i \in [m]$.

Such a scheme is called **(t, m) threshold secret sharing scheme** and is illustrated in Figure II.1.

Homomorphic Secret Sharing. Homomorphic secret sharing (HSS) is the secret sharing analogue of homomorphic encryption [BCG+17,BGI+18]. Homomorphic secret sharing refers to one or multiple secret inputs that are split and distributed

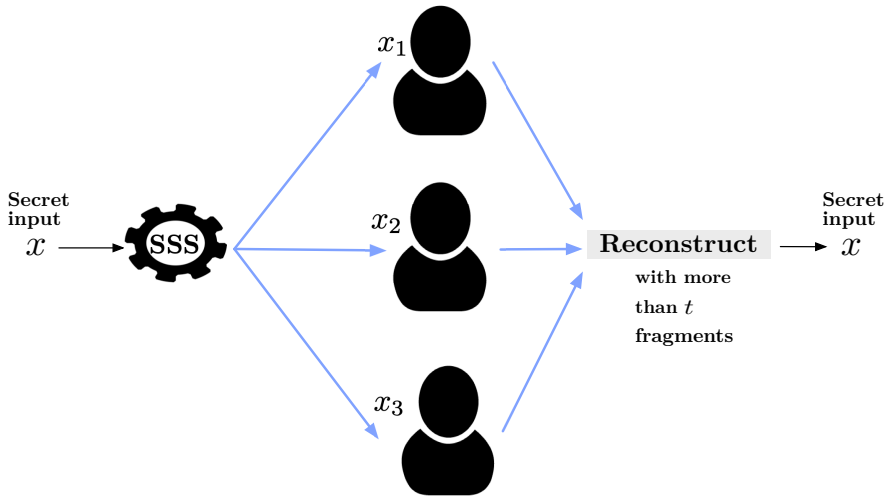

Figure II.1: A data owner breaks her secret x into pieces, namely x_1, x_2, x_3, using a threshold secret sharing scheme (SSS). If more than t pieces are known, the secret is reconstructed. Here $t = 2$.

into multiple servers. Each server performs local evaluations/computations on the received shares of the data (*i.e.,* pieces of the data), and generates an output share (*i.e.,* output value). Applying a decoding algorithm on the output shares provided by the servers, an HSS scheme results in an evaluation of a selected function f on the secret inputs. More concretely, given shares (*i.e.,* pieces) of an input x that are created with a (threshold) secret sharing algorithm (SSS), *i.e.,* Share, an evaluation algorithm Eval, and a decoder algorithm Dec, an HSS scheme is a 3-tuple of the probabilistic polynomial-time (PPT) algorithms (Share, Eval, Dec) which are defined as follows [BGI+18]:

1. Share$(1^\lambda; i; x)$: given a security parameter 1^λ, the index i of the data owner, and the corresponding secret input x, this algorithm outputs m shares of the secret input, namely $x_1, \ldots, x_m$.
2. Eval$(f; j, \{x_j^i\}_{i\in[n]})$: given a function f, the index j of the server, and a list of shares x_j^i of the secret input x^i, this algorithm outputs a value y_j.
3. Dec$(y_1, \ldots, y_m)$: given the outputs of the evaluation algorithm, the decoder algorithm outputs $f(x)$.

The algorithms (Share, Eval, Dec) should satisfy the following correctness and security requirements:

- **Correctness**: For any n secret inputs $x^1, \ldots, x^n$, the homomorphic secret

sharing (HSS) scheme should satisfy the following correctness requirement:

$$\Pr\left[\begin{array}{l}\forall i \in [n]\ \ (x_1^i, \ldots, x_m^i) \leftarrow \mathsf{Share}(1^\lambda; i; x^i), \\ \forall j \in [m]\ \ y_j \leftarrow \mathsf{Eval}(f; j, \{x_j^i\}_{i\in[n]}) : \\ \mathsf{Dec}(y_1, \ldots, y_m) = f(x^1, \ldots, x^n)\end{array}\right] = 1.$$

- **Security**: Let $T \subset [m]$ be the set of the corrupted servers with $|T| < m$. Consider the following semantic security challenge experiment:

  1. The adversary $\mathcal{A}$ gives $(i, x, x') \leftarrow \mathcal{A}(1^\lambda)$ to the challenger where $i \in [n]$, $x \neq x'$ and $|x| = |x'|$.
  2. The challenger picks a bit $b \in \{0, 1\}$ uniformly at random and computes $(\hat{x}_1, \ldots, \hat{x}_m) \leftarrow \mathsf{Share}(1^\lambda, i, \hat{x})$ where $\hat{x} = \begin{cases} x, \text{if } b = 0 \\ x', \text{otherwise} \end{cases}$.
  3. The adversary outputs a guess $b' \leftarrow \mathcal{A}((\hat{x}_j)_{j|s_j\in T})$, given the shares from the corrupted servers T.

  Let $\mathrm{Adv}(1^\lambda, \mathcal{A}, T) := \Pr[b = b'] - 1/2$ be the advantage of $\mathcal{A}$ in guessing b in the above experiment, where the probability is taken over the randomness of the challenger and of $\mathcal{A}$. The scheme (Share, Eval, Dec) is t-secure if for all $T \subset \{s_1, \ldots, s_m\}$ with $|T| \leq t$, and all PPT adversaries $\mathcal{A}$, it holds that $\mathrm{Adv}(1^\lambda, \mathcal{A}, T) \leq \varepsilon(\lambda)$ for some negligible $\varepsilon(\lambda)$.

Depending on the operation performed by the decoding algorithm, we can have an additive HSS scheme (*i.e.,* a summation of the values is performed), a multiplicative HSS [TLM18] (*i.e.,* a product of the values is computed) or other types of HSS. Naturally, one may consider a multiple input variant of HSS where the evaluation algorithm maps the j-th shares of all the inputs to an output share y_j and the decoder algorithm outputs the f evaluated on the multiple secret inputs instead of a single input. An homomorphic secret sharing scheme (HSS) for two secret inputs x^1 and x^2 is depicted in Figure II.2.

II.2 Verifiable Computation

Cloud servers are employed to perform computations on shared data but, unfortunately, they can instead make modifications before providing their results, output incorrect values, and therefore, break the security of our systems. We need to ensure the integrity of the output coming from the cloud servers, protecting the security of the assigned computations.

Verifiable computation (VC) is a mechanism that can be utilized for this purpose. More precisely, verifiable computation is comprised of techniques that allow the aforementioned resource-constrained IoT devices to securely outsource heavy computations to the servers, by guaranteeing that the returned results are *correct*. The verification can be accomplished either privately or publicly [AWH+18]. In general,

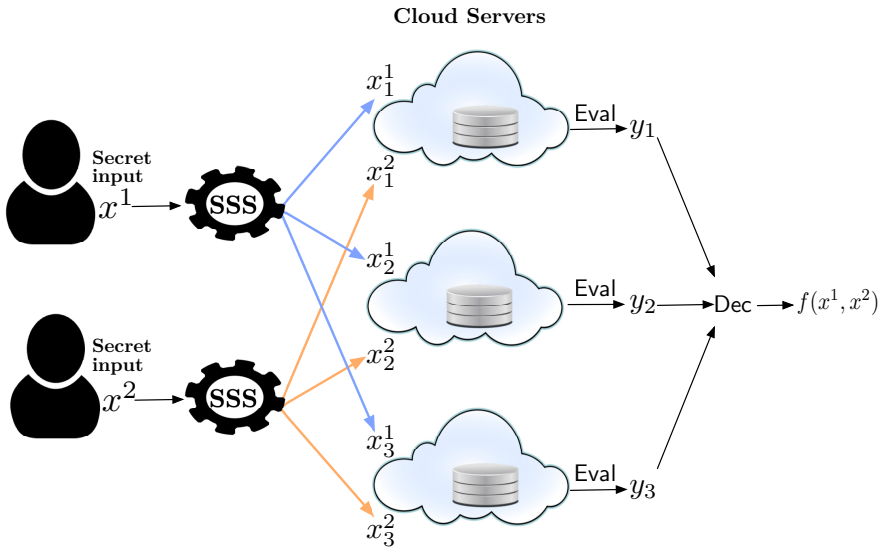

Figure II.2: Homomorphic Secret Sharing Scheme (HSS) for secret inputs x^1 and x^2.

verifiable computation is used to provide a way to verify the work of the untrusted "cloud".

Below, we describe the requirements that a *verifiable computation* mechanism must fulfill [YYV17, DD17]. The requirements are the following:

Requirements for Verifiable Computation (VC).

- **Verification Correctness**: This is the main requirement for VC. Verification correctness means that, given a verifiable computation scheme, it is impossible for an incorrect result to pass the verification check. In other words, an honest verifier will approve the result only if it is indeed *correct*.
- **Verification Privacy**: A verifiable computation solution must satisfy some privacy requirements. In detail, depending on the application scenario, verification confidentiality can be related either to the servers or the verifiers. Either the case, privacy is important for the *input*, *the computed result*, or even *both of them*.
- **High Verification Efficiency**: A verifiable computation scheme must require *less time than the actual computation*. Simply stated, a VC scheme must be practical to be useful.

Verifiable computation enhances the security of several schemes. For example, cloud storage service is a popular cloud service in which different security concerns may arise. Firstly, data confidentiality is safeguarded either with encryption or secret sharing. Luckily secret sharing can also protect against service availability failure (in case of a single server), data loss, or corruption. However, there exist more security

risks in this context. Precisely, there might be cheaters that wish to maliciously modify information or create collisions. VC schemes can provide solutions to this by detecting this behavior or even identifying who the cheaters are. Similarly, we can consider other applications such as electronic voting systems where the votes pass through the whole e-voting system and are expected to provide a correct election result without being jeopardized by political interests. An abstract illustration of a VC scheme for such a scenario is found in Figure II.3.

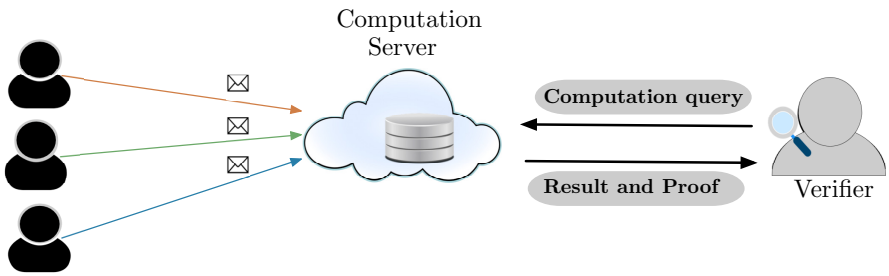

Figure II.3: Voters give their input/vote and everyone/verifier gets a proof that the announced voting result is correct.

II.3 Differential Privacy

More and more data from clients are outsourced for computations in order to acquire useful statistics such as which products people like, which devices they prefer to use and so on. However, this information must not reveal the sensitive data of the people that were part of the computation. In particular, in some cases, the resulted values of the computation itself can compromise the privacy of individuals.

For instance, Narayan *et al.* [NS08] showed that they could perform a linkage attack and reveal real identities of people by using a large, seemingly anonymized, dataset from Netflix and data from the Internet movie Database (IMDb). Furthermore, Sweeney [Lat15] identified the governor of Massachusetts by combining public health records with voter registration records. In such contexts, *differential privacy* is useful and counteracts these types of attacks.

Dwork *et al.* [DMNS06] presented the first *differential privacy* definition, a mathematical definition of privacy. Differential privacy makes it possible to generate useful results about a population of people without revealing sensitive information about a single entity. More precisely, the outcome of a computation remains the same regardless of whether an individual participates in the computation or not [DR14]. A formal definition for differential privacy is presented below:

Definition (ϵ-Differential Privacy [DMNS06]). A randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies ϵ-differential privacy if for any two adjacent inputs $d, d' \in \mathcal{D}$ and for any subset of outputs $S \subseteq \mathcal{R}$ it holds that:

$$Pr[\mathcal{M}(d) \in S] \leq e^{\epsilon} Pr[\mathcal{M}(d') \in S],$$

where ϵ denotes the privacy parameter which is defined in positive real numbers. A privacy promise can be considered strong when its ϵ is close to zero.

Consider a deterministic real-valued function $f : D \rightarrow \mathbb{R}$. A method commonly applied to approximate the function f with a differentially private mechanism is by applying additive noise incorporated in f's sensitivity S_f. Function f's sensitivity, *i.e.,* S_f, is defined as the maximum possible distance between the replies to queries (*i.e.,* $|f(d) - f(d')|$) addressed to any of the two neighboring databases (d and d'). Intuitively, larger sensitivity demands a stronger countermeasure. The Laplace noise mechanism is a differentially private mechanism often employed to achieve differential privacy, and is defined as follows: $\mathcal{M}(d) \triangleq f(d) + Laplace(\lambda)$.

Differential privacy becomes feasible with two ways: *(i)* applying noise to each of the dataset records or *(ii)* applying noise to the computed result to distort it. The main concern though, is the trade-off between data utility and individual privacy. We use an ϵ parameter when referring to differential privacy (*i.e.,* ϵ-differential privacy). The size of this value, which denotes the privacy loss, varies depending on the use cases and determines the relation between privacy-preservation and accuracy/utility.

Consider, for example, a query on how many people have a specific disease, namely disease X. Differential privacy ensures that having two neighboring datasets (*i.e.,* they only differ on a single entry) does not affect the query result too much to compromise privacy. This way, individuals are protected from any "harm" that they could maybe face due to their data being included in a specific dataset. See Figure II.4 for an illustration of this paradigm.

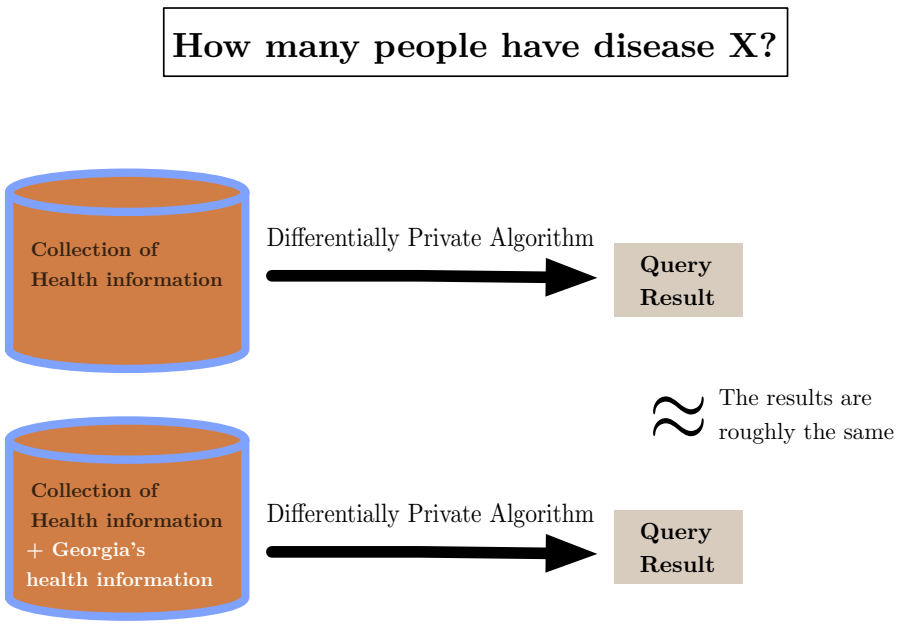

Figure II.4: Georgia's health information is protected thanks to the randomized mechanisms of differential privacy.

One way to verify that a response to a given query is indeed an ϵ-DP computation, is using the VFuzz language. VFuzz or Verifiable Fuzz is a modified version of the Fuzz language. It can statistically certify queries as differential private, without looking at the data, and, moreover, it can be translated safely to circuits, without causing any data leakage (in contrast to Fuzz) [NFPH15].

III

Thesis Objectives

This thesis explores the problem of secure and private cloud-assisted computing. In particular, this thesis investigates how to design secure systems that outsource computations to the cloud, preserving the privacy of the data owners and providing integrity guarantees for their outcomes. In detail, this thesis focuses on the following research questions:

RQ1. **Data privacy and decentralization.** How can we outsource joint computations on sensitive data coming from multiple clients to multiple untrusted servers without revealing confidential data, while also avoiding single points of failure?

RQ2. **Public verifiability.** How can we securely provide public verifiability (*i.e.,* confirmation of the integrity of the computed results), while keeping sensitive data private?

RQ3. **Individual privacy.** How can we design a protocol that does not leak any information about the participation of an individual in a specific dataset?

RQ4. **Data aggregation with user dropouts.** How can we design secure and privacy-preserving protocols for sensitive data aggregation that can handle users' dropouts and do not require any communication among users?

RQ1. Collecting data from resource-constrained devices creates the need for outsourcing heavy computations to the powerful cloud, and also for protecting the privacy of the used data. In this scenario, assigning the computations to a single cloud server can be risky, *e.g.,* there might be that the server fails (*i.e.,* single point of failure) leading to data loss or the server might try to compromise the privacy of the data and infer sensitive information about the clients (devices). Therefore, it is important to meet these challenges and provide solutions addressing them.

RQ2. Employing multiple servers can overcome single points of failure; however, servers can also try to adjust the computed results according to their interests. This means that, when outsourcing computations, we cannot rely solely on the given results. It is essential to make sure that the given outcome is useful and can be utilized as expected. Huge companies and organizations might earn or lose money from statistics and similar computations; thus, they need to be sure that they indeed receive what they requested. Addressing *RQ2*, aims to provide integrity guarantees

to anyone that wishes to get the computed results. In other words, anyone can confirm that the computed values received by the servers correspond to the correct result of their assigned computation.

RQ3. Although privacy-preserving mechanisms have been applied to protect private data, research has shown that combining auxiliary information together with a given dataset can reveal someone's identity or sensitive information about them [Sta10, BJ12]. Thus, it is vital to enable individuals to keep their private information secret while participating in some computations. We pose this research question to explore how to achieve this.

RQ4. In machine learning applications, sums of local updated parameters from users' devices are computed to train learning models. Private data are kept in mobile devices which can drop at any time. In this context, the problem of secure data aggregation while guaranteeing privacy-preservation of the users' inputs is interesting. This is because it allows, for example, different organizations (*e.g.,* hospitals) to collaboratively train such learning models, tolerating users' dropouts and preserving the privacy of their data.

We address these research questions in the remaining chapters of this thesis. The mapping of the research questions to the respective chapters is shown in Table III.1.

Table III.1: Research questions addressed in each chapter.

	Chapter A	Chapter B	Chapter C	Chapter D	Chapter E
RQ1	●	○	●	●	●
RQ2	●	●	●	●	●
RQ3	○	●	○	○	○
RQ4	○	○	○	●	○

IV

Thesis Contributions

This thesis tackles the research questions raised in Section III for providing privacy and security in cloud-assisted computing. Below, we describe how each chapter contributes to one or more of the listed research questions.

Chapter A: Verifiable Homomorphic Secret Sharing [TLM18]

In this paper, we consider multiple clients and multiple servers involved in the problem of outsourcing computations (*i.e.,* sum or product). Primarily, we introduce a general definition for what we call *verifiable homomoprhic secret sharing* (VHSS). A VHSS scheme enables n clients, which do not interact with each other, to assign their joint computations to m servers. Additionally, a VHSS scheme achieves verifiability, *i.e.,* confirmation/proof that the final computed value is correct.

Next, we propose a detailed solution on computing the sum of n clients' secret inputs. A set of m servers receive shares of each of the secret inputs, denoted by x_i, and perform local computations (without communicating). The combination of their computed partial results provides the desired final value that corresponds to the sum of the n secret values. This construction contributes to the *RQ1* by protecting sensitive clients' data (*i.e.,* giving shares of them instead of the actual values) as well as avoiding single points of failure by employing multiple servers.

Furthermore, we propose an instantiation of the multiplicative VHSS scheme. More precisely, we address the problem of outsourcing the product of n secret inputs (belonging to n clients) to m untrusted servers such that: *(i)* the product is computed without the servers having access to the private data but rather to shares of them, and *(ii)* anyone is able to verify that the resulted outcome (product) is correct by getting a proof that the final output is correct (public verifiability). This contributes to both *RQ1* and *RQ2*.

Statement of Personal Contributions. I am the main author of the paper. I was responsible for defining the general definitions for homomorphic secret sharing (HSS) and VHSS, designing the additive HSS and the VHSS schemes. I formulated the correctness, security and verifiability requirements, the corresponding theorems and the proofs of the two proposed instantiations.

Appeared in: 12*-th International Conference on Provable Security (ProvSec), 2018.*

Chapter B: Differential Privacy meets Verifiable Computation: Achieving Strong Privacy and Integrity Guarantees [TM19]

Large companies are often interested in statistical results over a population of people. Unfortunately, the data necessary to perform these computations are often sensitive data. This implies that there is a need to provide guarantees to the companies (or service providers) that the computation results are correct but also guaranteeing that no leakage of information about individuals exists.

This work, tackles this problem by firstly providing a formal definition of *publicly verifiable differentially private computation* ($\mathsf{VDPC}_{\mathsf{Pub}}$). Next, we suggest a detailed protocol that consists of the following entities: *(i)* a curator which collects sensitive data (*e.g.,* health related information), *(ii)* an analyst which is expected to perform the computations, and finally, *(iii)* a reader, which submits a query and wishes to get the response together with a proof of correctness. Our proposed solution is partitioned into two stages. The first stage comprises of the steps where the reader and the curator agree on the noise that needs to be added to have a differentially private value. In the second stage, we present the employment of a publicly verifiable differentially private computation scheme in our system. Achieving public verifiability contributes to the *RQ2*, whereas proposing a protocol to compute a differentially private function value ensures that there is no leakage of information regarding the participation of any individual (contributing to *RQ3*).

Statement of Personal Contributions. I am the main author of the paper. I was responsible for providing the definition of *publicly verifiable differentially private computation* ($\mathsf{VDPC}_{\mathsf{Pub}}$). I designed and formalized the proposed public VDPC protocol.

Appeared in: 16-*th International Conference on Security and Cryptography (SECRYPT), 2019.*

Chapter C: Practical and Provably Secure Distributed Aggregation: Verifiable Additive Homomorphic Secret Sharing [TBM20]

This work addresses the problem of outsourcing the computation of the sum function $f = x_1 + \ldots + x_n$ to m servers. More precisely, considering n secret inputs the goal is to assign multiple servers for computing the sum of the inputs satisfying the following: *(i)* the secret inputs of the clients remain private, *(ii)* the servers have access only to shares of the inputs (such that they cannot reconstruct the inputs themselves), and *(iii)* the proposed scheme must provide the ability to anyone to verify the correctness of the sum. We refer to this problem as *verifiable additive homomorphic secret sharing* (VAHSS) and we propose three different solutions to it.

In detail, we combine an additive homomorphic secret sharing scheme with three different verification approaches, aiming to capture different application scenarios. Each method for achieving *verifiability* employs a different primitive, and involves the generation of some partial proofs which are values that are used for satisfying the verifiability property. These partial proofs are either computed by the servers or by the clients (depending on the proposed construction).

The first VAHSS construction is based on homomorphic collision-resistant hash functions. In this solution, clients are only responsible for distributing shares of their secret inputs to each of the servers. The servers take care of all the computations related to the requested sum value, and additionally, use hash functions to generate the proof of correctness. Next, we employ linear homomorphic signatures to achieve verifiability in our second construction. In this case, the clients distribute shares as previously, the servers similarly compute what is needed for getting the sum value; however, the values needed for the verification are generated by the clients. Finally, the third construction combines additive homomorphic secret sharing (for clients to generate the shares of their input) with a threshold RSA signature scheme. This solution requires a threshold t of servers to mutually construct the proof required for verification using the threshold signature scheme. We provide experimental evaluation for all proposed constructions.

This work contributes to *RQ1* by safeguarding *(i)* the privacy of the clients' secret data, and *(ii)* outsourcing the sum computation to multiple servers. Achieving public verifiability in all three proposed constructions contributes to *RQ2*.

Statement of Personal Contributions. I am the main author of the paper. I was responsible for designing the three VAHSS proposed constructions. I formalized and proved the security, verifiability and correctness requirements of all three proposed constructions. I helped in the experimental evaluation, performed the theoretical analysis and contributed in the protype analysis of all three constructions.

Appeared in: *Cryptography Open Access Journal, 2020.*

Chapter D: DEVA: Decentralized, Verifiable Secure Aggregation for Privacy-Preserving Learning [TLB+21]

This paper focuses on the problem of secure data aggregation in the context of machine learning applications. Data are located in mobile devices (users) with low computation capacity, and need to be outsourced to the servers to train the learning model. To do this, sums of local updated parameters are computed, while any individual user's update is not revealed in the clear.

In this work, we perform secure aggregation of the users' secret data (parameters) without requiring any communication among the users. We define and describe the DECENTA problem and propose the protocol DEVA which is our proposed DECENTA solution. More precisely, the DEVA protocol computes the sum of the users' input, while also providing verifiability of the computed result. In other words, the outcome of the protocol is not only the computed sum value, but also additional values that need to match with the final result to successfully pass the verification check and confirm that the sum is correctly computed. The sum value can be computed even if not all servers are present, *i.e.,* only a threshold amount of servers is required to compute the final value. Additionally, users can drop at any time during the protocol execution, and our solution can handle users' dropout successfully. Users never provide their data in the clear during the protocol, protecting the confidentiality of the latter. We evaluate the performance of our DEVA protocol and present our findings compared also with the prior work. We

have provided a solution for outsourcing the sum computation to multiple untrusted servers, protecting users' sensitive data, providing public verifiability, and finally handling users' dropout. Therefore, we contribute to *RQ1*, *RQ2*, and *RQ4*.

Statement of Personal Contributions. I am the main author of the paper. I was responsible for defining the general framework of a DECENTA problem. Bei and I together designed and formalized the proposed DEVA protocol. I was responsible for defining and proving the verifiability and correctness properties of DEVA. I got the experimental results and provided the corresponding implementation analysis as well as illustrated the evaluation findings for the different parameters considered.

Appeared in: 24*-th Information Security Conference (ISC), 2021.*

Chapter E: Non-Interactive, Secure Verifiable Aggregation for Decentralized, Privacy-Preserving Learning [BTL$^+$21]

In machine learning applications, *e.g.,* in federated learning, users (devices) collaboratively train their models under the arrangements of a central server. Next, that server updates accordingly the global training model given the parameters from several devices. However, as we have discussed in this thesis, this might cause single points of failure. Additionally, there might exist different organizations with similar objectives that want to train their models collaboratively.

This work proposes a non-interactive protocol, namely NIVA, which enables multiple servers to aggregate the secret inputs coming from several users and perform the required computation (sum) to update the learning model. The data of the users remain private, and the users only provide their inputs (not in the clear) and are not required to participate in any other stage. Servers can be untrusted and, therefore, are also requested to provide integrity guarantees, *i.e.,* NIVA allows anyone to verify the correctness of the computed result. This way any malicious server that would try to bias the model updates cannot succeed. This solution aims to provide a decentralized approach by employing multiple servers, protect sensitive data, and provide security guarantees against any malicious behaving server. We provide an evaluation of NIVA with respect to the current state of the art. This paper contributes to *RQ1* and *RQ2*.

Statement of Personal Contributions. This paper resulted from joint discussions and ideas among the authors, whereas Carlo came up with the idea of the NIVA protocol. This paper was initially united with the paper in Chapter D. I helped in constructing the verifiability proof of NIVA, while also helping with providing the state of the art for this work.

Appeared in: 26*-th Australasian Conference on Information Security and Privacy (ACISP), 2021.*

V

Conclusion

This thesis consists of five chapters which address the problem of security and privacy in cloud-assisted settings in different ways. The summary of the findings in each chapter are depicted in Figure V.1.

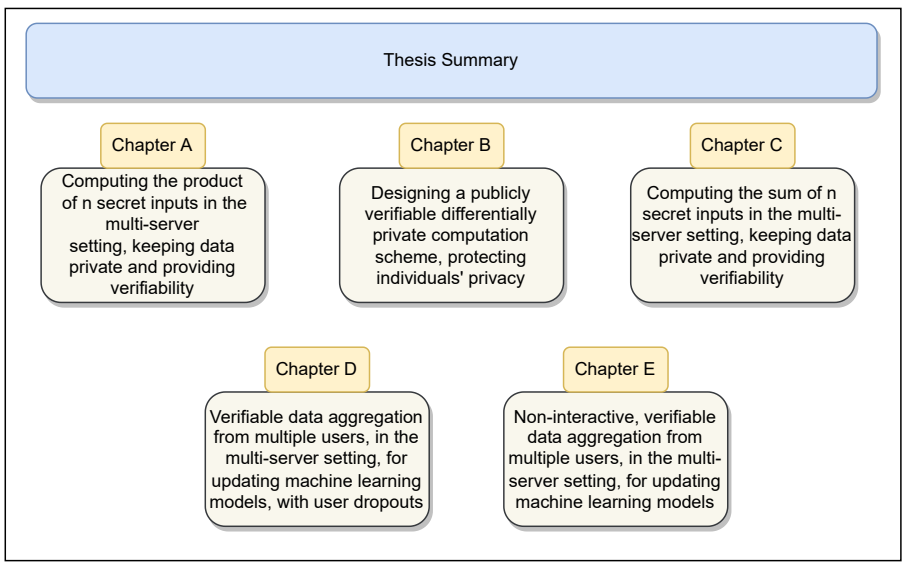

Figure V.1: Summary of results in each thesis chapter.

Throughput this work, several challenges were encountered, including *(i)* potentially malicious servers that want to modify the computed function results according to their wishes, *(ii)* the possibility that a single server might fail or lose connection, *(iii)* the need to outsource the assigned computations due to the limited resources of the devices that collect the sensitive data, while also keeping data private, *(iv)* and, finally, the necessity to provide integrity guarantees for the calculated values to counteract the existence of malicious servers in our settings.

To extend and improve our findings, one may consider the following future directions:

- Chapter A and C propose solutions for computing the product and the sum, respectively, of n secret inputs when considering multiple clients. A future direction for both chapters can be considered to address a scenario where one or more clients are untrusted, and therefore, design a solution that handles all involved parties being potentially malicious.
- Chapter B provides a publicly verifiable differentially private computation scheme. In this work, one may implement such a scheme to explore how different parameters provide different levels of privacy and utility trade-offs.
- Chapter D and E focus on secure data aggregation in a decentralized setting, providing verifiability against potentially malicious servers. An improvement of these works would be to extend the solutions to work more efficiently when considering vectors as secret inputs.