Privacy-Preserving Certification of Sustainability Metrics

Cetin Sahin, Brandon Kuczenski, Omer Egecioglu, Amr El Abbadi
University of California, Santa Barbara
{cetin, omer, amr}@cs.ucsb.edu, bkuczenski@bren.ucsb.edu

ABSTRACT

Companies are often motivated to evaluate their environmental sustainability, and to make public pronouncements about their performance with respect to quantitative sustainability metrics. Public trust in these declarations is enhanced if the claims are certified by a recognized authority. Because accurate evaluations of environmental impacts require detailed information about industrial processes throughout a supply chain, protecting the privacy of input data in sustainability assessment is of paramount importance. We introduce a new paradigm, called *privacy-preserving certification*, that enables the computation of sustainability indicators in a privacy-preserving manner, allowing firms to be classified based on their individual performance without revealing sensitive information to the certifier, other parties, or the public. In this work, we describe different variants of the certification problem, highlight the necessary security requirements, and propose a provably-secure novel framework that performs the certification operations under the management of an authorized, yet untrusted, party without compromising confidential information.

ACM Reference format:
Cetin Sahin, Brandon Kuczenski, Omer Egecioglu, Amr El Abbadi University of California, Santa Barbara {cetin, omer, amr}@cs.ucsb.edu, bkuczenski@bren.ucsb.edu . 2018. Privacy-Preserving Certification of Sustainability Metrics. In *Proceedings of Eighth ACM Conference on Data and Application Security and Privacy, Tempe, AZ, USA, March 19–21, 2018 (CODASPY '18),* 11 pages.
https://doi.org/10.1145/3176258.3176308

1 INTRODUCTION

Organizations are often motivated to make public disclosures about their environmental performance. These motivations may be inspired by regulatory requirements, marketing initiatives, or as part of a broader project of corporate sustainability. The landscape of environmental and sustainability claims is largely standardized, as exemplified by the ISO 14000 series of standards. Often environmental disclosures take the form of certifications, which establish that some agency has reviewed the claim and confirmed its validity. A prominent example is the ISO 14001 certification, which simply establishes that a firm has an established policy to review and correct its environmental performance. To make a quantitative evaluation about the ecological sustainability of a product or service, approaches that consider the full life cycle of the product are often used [38]. This form of analysis, known as life cycle assessment (LCA), is codified in the ISO 14044 standard [23].

Sustainability certification has been shown to lead to potentially significant operational improvements in environmental performance [35]. Firms with more significant environmental impacts are more likely to have high-quality environmental management systems [16]. Life cycle approaches can improve the quality of environmental disclosures [24] and also provide a framework for firms to take broader responsibility for the impacts of the products they make or sell [22].

The ISO 14020 series of standards governs environmental product declarations (EPDs), which include public assertions about the sustainability of products, based on ISO 14044-style life cycle evaluation [15, 32]. EPDs can include both externally certified claims and self-reported results. Certified results can include both "pass-fail" binary assertions about a product or process with regard to a set of criteria, known as "eco-labels," as well as detailed quantitative results [17].

The data sets that provide input to these computations express essential information about the operation of a process or production step [10]. A typical data point could be the quantity of electricity required to output a reference unit of some product. These data are often regarded as confidential and are typically concealed through aggregation with other data sets [41, 44]. Engagement with stakeholders and supply chain partners [36] is often required for effective consideration of life cycle environmental sustainability, which accentuates confidentiality concerns and may limit the scope of information included in the assessment [24].

Despite the importance of data privacy, the LCA community lacks a formal framework for managing private data, and very limited number of techniques exist for computing sustainability metrics that preserve the privacy of input data. In [29], Kerschbaum et al. introduce a framework for sustainability benchmarking with the help of an untrusted third-party, however, the proposed solution has an assumption that the participants do not collude with the third-party or each other which not might be realistic in the LCA community. This can result in significant risk to the privacy of individual data since small organizations might collude with each other to gain private information against big competitors or vice versa. We seek to apply recent developments in security and privacy to the problem of certification of environmental claims even in the presence of colluding parties. Specifically, we aim to confront the following challenges: 1) mutually competitive firms want to gain private knowledge about their environmental performance by comparing their environmental impact against a statistical metric, which is a function of the competitors' performance, such as an average or maximum; 2) an association of firms wants to enable its

CODASPY '18, March 19–21, 2018, Tempe, AZ, USA

ACM ISBN 978-1-4503-5632-9/18/03…$15.00
https://doi.org/10.1145/3176258.3176308

members to make public, validated claims about their individual environmental performance in comparison to a cohort or to the full group, based on private data.

The first of these can be achieved using existing secure multiparty computation (SMC) protocols (see Section 2). However, to the best of our knowledge, SMC has never been applied to the case of sustainability assessment in a completely secure manner. The second use case is novel and has the distinct requirements that parties be provided with certificates validating qualitative assertions about their inputs without the inputs being known, unlike most SMC solutions, these parties *not* communicate directly with one another, instead by interacting through a certifying authority.

In this paper, we formally define the *privacy preserving LCA certification* paradigm along with its goals, security and computation requirements. A certification is a quantitative evaluation of the result of such a computation, or an evaluation of a given contribution with respect to the result. Unlike in the SMC context where the individual parties involved need to know some if not all of the other parties involved, in the LCA context, communication with other parties might not be possible or is even desirable. Hence, we propose a novel privacy-preserving certification framework that enables an authorized party, referred to as *certifier*, to certify participants based on industrially well agreed on set of criteria or a common function without compromising any sensitive/confidential information to any other parties even in the presence of colluding parties. Although the certifier is authorized in the LCA context, it is not assumed to be trusted, which explicitly requires hiding inputs from the certifier as well. Moreover, the certifier might collude with some of the parties. Unlike previous proposals like [29], our approach is secure even if parties collude with the certifier. Our framework **does not** require parties to communicate with each other and aims to minimize the rounds of communication between the parties and the certifier. We now highlight some of the distinctive features of the LCA problem domain and our contributions.

Certification with no trusted entities

Even though the computation is performed by a certifying authority, it cannot be assumed to act as a *trusted*, unbiased authority, since the parties may not want to reveal their individual inputs to any other entity, including the certifying authority. In general, the certifying authority might need to perform complex computations and comparisons. It might be possible to perform such computations with an untrusted authority using advanced cryptographic tools like fully homomorphic encryption [18], but such techniques are known to be quite inefficient [40]. An established, computationally efficient approach for performing the complex computations required for certification is to use secure co-processors [3]. A secure co-processor is a tamper-proof hardware, which provides a non-transparent and isolated computation environment. It creates a trusted computing environment in hostile environments and prevents any unauthorized access. Because of these advantages, secure co-processors have been adapted in different contexts such as encrypted database querying [6, 7] and secure multiparty computations [25]. However, such hardware is limited in terms of computational resources and their straightforward deployment does not solve all the problems. The design of a secure and efficient framework is still a challenge.

Certification Operations

The certifier will perform secure *mean* and *quantile* computations (will be discussed later in detail) to make public or private announcements about parties. These are quantitive computations that allow the certifier to benchmark the performance of parties. To perform such computations, the certifier needs to perform secure comparison which requires a set of private cryptographic and secure operations. Performing these computations without compromising security and privacy constraints is a challenge in the LCA context.

Veracity of LCA data

When multiple parties want to perform a joint computation, the accuracy and usefulness of the computation rely on the correctness of the inputs. Verifying the correctness of the inputs is an important challenge in many contexts. The standard approach in the LCA context is the assumption of the correctness of the provided inputs, since the correctness of the inputs are verified via an audit after the computation [8, 43]. Therefore, the verification of inputs and the audition of data are beyond the scope of this paper. The main motivation is to perform computations securely.

We propose efficient algorithms to perform certification operations for the certification problems-mean, quantile- using the proposed framework. We show that the proposed algorithms are correct and secure with the assumption of honest-but-curious parties. Furthermore, we discuss the efficiency of our algorithms both empirically and analytically.

2 RELATED WORK

Secure multiparty protocols (SMC) are known for computing functions jointly over a set of inputs without revealing any information about the inputs. In brief, a set of n parties with private inputs $x_1, x_2, \ldots, x_n$ wish to compute a function $f(x_1, x_2, \ldots, x_n)$ jointly without revealing any x_i to any other party. After an execution of this function, the parties learn the correct output but nothing else, even if some parties try to obtain more information by colluding. There are two-party computation protocols that execute generic functions [34, 46], but these constructions rely on heavy cryptographic computations and may not be practical [12]. Privacy-preserving statistics using SMC have been well-studied under the scope of privacy-preserving data mining[11, 19, 26, 27, 33]. For example, Rmind [11] is a tool that computes well-known statistics privately such as average, mean, median, while [19] proposes a secure dot product computation using SMC.

Although SMC has a wide spectrum of applications, most applications require interactive communication among the parties. Certification on the other hand focuses on a performance evaluation using some statistical analysis. Our protocols differ from existing SMC approaches in that they do not require communication and data exchange among the parties, and instead require the involvement of an authorized (but untrusted) party in the computations to regulate certification policies.

Involvement of an authorized party requires the establishment of trust between the participants and the authority. Establishing trust with an untrusted party is not a new problem in the literature and several works in different contexts [4–7, 39] rely on trusted hardware based solutions, e.g. Trusted Platform Modules (TPMs) [2]

or secure co-processors [1, 3], to establish a trusted computing environment, which are shown to be quite efficient for specific applications [4, 6].

Unlike fully homomorphic encryption, which is computationally quite expensive, partial homomorphc encryption has been shown to be relatively efficient. Examples of partial homomorphic encryption are the additive homomorphic Paillier [37] and Quadratic Residues [20] public key cryptosystems and these will be explained in detail later in Section 4.2. The central component of our protocols is private comparison, which has been well studied previously [9, 13, 14, 28, 30, 42, 46]. Each technique is suitable to different settings. For example, while [42] performs comparison on encrypted data, [14] compares unencrypted values privately. It is important to note that providing a new private comparison technique is not in the scope of this paper, it is just one of the main building tools to develop our protocols for the certification problem. We adapted our private comparison protocol from Veugen's protocol [42] as discussed in Section 4.3. Several recent works [7, 12] also adopt Veugen's protocol to solve different problems. Bost et al. [12] construct machine learning classification protocols over encrypted data. On the other hand, Baldimtsi et al. [7] propose a framework, which also benefits from secure co-processors, that builds on top of searchable encryption techniques to return ranked results to queries. Our work follows in this tradition, and applies it to an important new domain, namely environmental certification.

To the best of our knowledge, the closest work to ours is [29]. In this work, Kerschbaum et al. propose a private benchmarking platform for environmental sustainability with the help of an untrusted third party. Although the overall setting seems similar to our setting, there are fundamental differences in the two approaches regarding the security of the systems. The assumption in [29] is that the parties do not collude with each other and the untrusted party. However, this is not a realistic assumption given the current competition in the market. The parties might collude with each other or with the untrusted party to gain private knowledge against the competitors. The proposed key management scheme in [29] either allows parties to share the same private key or distribute the private key among k parties which will later require at least t of them to be present to decrypt the output. In the case of key sharing, any party colluding with the untrusted party can reveal the private inputs of the other parties. Similarly, in the presence of t colluding parties, it is possible to infer the private inputs of others if the key distribution approach is applied. Our approach is secure against colluding parties. Additionally, the certification process heavily relies on private comparison of inputs. The proposed comparison protocol in [29] relies on [30] which ensures a weaker notion of security due to the usage of multiplicative hiding. Our protocols rely on semantically and cryptographically secure comparison protocols in the certification process.

3 PROBLEM DESCRIPTION

3.1 Privacy-Preserving Aggregation in LCA

Life Cycle Assessment (LCA) is critical for quantitative evaluations of the ecological sustainability of a product or service. The computation of results in LCA can be described as a series of matrix operations in which possible results are activity or output levels of industrial unit processes, quantities of emissions into the environment resulting from those processes, or measurements of environmental impact scores [21]. The calculation of any one of these values can be described as the inner product of a vector of input data with a weighting vector of environmental characteristics [31]. We formulate the private LCA aggregation problem as an inner product of two vectors

$$s = \mathbf{w} \cdot \mathbf{x} \tag{1}$$

where s is an LCA metric, each element x_i of the input vector $\mathbf{x}$ is one party's private contribution, and the weighting vector $\mathbf{w}$ is determined separately and may be either public or private. In this paper, for simplicity, $\mathbf{w}$ will be taken to be $\mathbf{1}$, so that s is the sum of the parties' inputs.

Consider an international trade group in steel manufacturing that wants to issue a report that documents the industry's environmental performance, such as the World Steel Organization's LCA study [45]. In the World Steel Organization, the certifier is managed by a committee, with representatives of the different manufacturers. All the manufacturers want to have a certifier, but since it has reps for different manufacturers, any given manufacturer cannot trust the certifier with its info. Conventionally, a report can only be prepared if the member firms share their confidential information with the trade group, allowing it to perform the aggregation and report the results. If instead the report were determined using privacy-preserving aggregation, the inputs would remain private, and firms could use the results privately for benchmarking their own performance, or publish the results, individually or together. However, the veracity of the results would be difficult to establish to the public.

We define a new problem, called *private certification*, in which an authorized party, referred to as *certifier*, can certify the participants' inputs based on a set of criteria or a common function without compromising any sensitive or confidential information. The output of this private computation may be announced by the certifier publicly or held private; however, the certifier should not learn any sensitive information during its execution. The certifier would need to be "trusted" by the public to compute and report results accurately, but may not be trusted by the parties with respect to the private data. In the private certification framework, unlike in traditional SMC, parties *are not required to communicate with each other*, but only with the certifier. The parties is not realistic nor desirable in the certification model, since the parties might not know each other, and may not want to communicate with each other.

We introduce two new privacy preserving certification problems, namely *mean* and *quantile* based, which allow firms to make public or private announcements about their inputs to a secure aggregation. Here we describe the constraints and requirements of the two certification methods. The correctness of the certification relies on the correctness of the inputs. As we mentioned earlier, the parties are honest-but-curious, i.e. they are honest about executing the protocol correctly, but curious to learn other inputs. Hence, we can assume that the provided inputs are correct, which is a standard assumption in the LCA context, since the correctness of inputs are verified via an audit after the computation (e.g. [8, 43]). Please note that in describing the functionality, we use inputs in the clear and

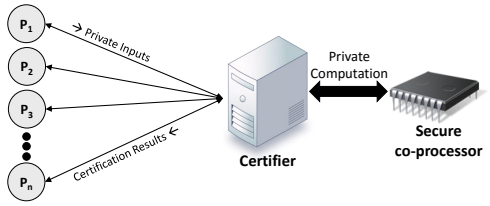


Figure 1: Overview of Framework Model

ignore cryptographic details. Later in Section 5, we will explain how to perform these certifications securely.

3.2 Mean Based Certification

In *mean based certification*, the certifier uses private aggregation to compute the average of a set of private inputs. Afterwards, the certifier compares each private input x_i with the average and performs the necessary certification operation, i.e. if a party generates less than the average, it can seek being labeled as more "eco-friendly" than its peers; otherwise it can forgo such labeling.

In mean based certification, the certifier computes the average of n inputs $x_1, x_2, x_3, ..., x_n$, and then certifies the parties either as *below* or *above* by comparing the individual values with the computed average value.

3.3 k-Quantile Based Certification

Grouping items into distinct groups based on predefined criteria is a well studied concept in statistics and can be utilized in different contexts. In the context of environmental impact assessment, this grouping technique provides performance information about a specific firm among the set of manufacturers. Being in the top quantile may be regarded as a prestigious certification that manufacturers can use to advertise their products with a greater confidence. By the nature of quantile based computation, the order information among the groups is revealed but it is hard to conclude which party is better inside the same group if the complete ranking information is hidden. It also allows parties increased flexibility to publish top performers' results while keeping others private.

In $k - quantile$ based certification, the certifier partitions the parties into k groups after ranking them based on the provided inputs. A party with the minimum input will be in the first group while a party with the maximum input will be in the k^{th} group.

4 SYSTEM MODEL AND BUILDING BLOCKS

We now describe the system model and basic building blocks used in the paper.

4.1 System Model

The proposed framework contains three main entities: *parties*, a *certifier*, and a *computation helper* as illustrated in Figure 1.

Parties. Parties are end-users which are the main data (input) providers to the system. In reality, parties are the competitors in manufacturing the same product or providing the same service. To demonstrate the superiority of their product or service, they would like to be certified by an authorized party. Parties are not aware of the other participant parties and do not communicate directly with each other.

Certifier. In this context, the authorized party is called the *certifier*. It is the main computation unit of the framework and it communicates with all registered parties during the computation. Each party has to register through the certifier to be able to join the certification process. The certifier is trusted in performing operations but at the same time it might be curious to learn some information about the parties' data. Therefore, the framework aims to preserve the confidentiality of inputs throughout the computation against the certifier and all other external adversaries. To achieve this goal, the computation is split between two non-colluding computation units: *the certifier itself and a computation helper.*

Computation Helper. The framework needs an additional computation unit other than the certifier to satisfy privacy constraints. It is called *computation helper.*

The computation helper aids the certifier compute the certification function. The helper and the certifier must not collude, otherwise, they can reveal the secret data. The helper can be a server from a different service provider or a secure, tamper-proof hardware that can be deployed on the certifier site. As depicted in Figure 1, the framework deploys a specialized secure co-processor like IBM 4764 PCI-X Cryptographic co-processor [3]. These processors have relatively low resources in terms of memory and computation power, and are invoked to compute relatively small computations. Secure co-processors provide a non-transparent and isolated computation environment which fits directly into our model. We assume that the supplier of the co-processor is different than the certifier and their marketing interests do not intersect. Several privacy preserving solutions using a secure co-processor have already been proposed in different contexts such as encrypted database querying[6, 7] and secure multiparty computations [25]. Our framework requires only one round of communication between the parties and the certifier. Once a party submits a private input to the certifier, all the remaining communication happens between the certifier and the secure co-processor (the computation helper). The availability of fast network communication between the certifier and the secure co-processor is another advantage of our design. When the secure co-processor is deployed at the certifier's site, it is realistic to assume negligible network latency, since communication usually happens in the order of 1 millisecond.

4.2 Cryptosystems

The certifier needs two additively homomorphic cryptosystems: Paillier [37] and Quadratic Residues(QR) [20]. The cryptosystem is called *partially homomorphic* if it supports either addition (additive homomorphic) or multiplication (multiplicative homomorphic). Both Paillier and QR are additively homomorphic which means given two encrypted ciphertexts, $Enc(m_1)$ and $Enc(m_2)$, the application of the additive homomorphic operation will result in the decryption of $Enc(m_1 + m_2)$.

The Paillier cryptosystem is based on the Decisional Composite Residuosity assumption [37]. We use $[\![m]\!]$ to denote the encryption of message m with the Paillier cryptosystem using a public-secret key pair $K_P = (PK_P, SK_P)$. The plaintext space of Paillier is $\mathbb{Z}_N$ where N is the public modulus of Paillier and its homomorphic property is $[\![m_1]\!].[\![m_2]\!] = [\![m_1 + m_2]\!]$. In addition, the Paillier cryptosystem

also supports multiplying ciphertext with a constant, which is actually the homomorphic summation of input with itself by n times. On the other hand, the plaintext space of Quadratic Residues (QR) is bits and $[m]$ denotes the encrypted bit m under QR. The key pair of QR is denoted by $K_{QR} = (PK_{QR}, SK_{QR})$. The homomorphic property of QR is $[m_1].[m_2] = [m_1 \oplus m_2]$.

Basically, Paillier implements the following three functions:

- $K_P(PK_P, SK_P) \leftarrow KEYGEN_{PL}(\lambda)$ generates a key pair. Note that λ is a security parameter.
- $[\![m]\!] \leftarrow$ encPL(m, PK_P) encrypts plaintext m using public key PK_P and outputs encrypted ciphertext $[\![m]\!]$.
- $m \leftarrow$ decPL$([\![m]\!], SK_P)$ decrypts given ciphertext $[\![m]\!]$ using secret key SK_P and outputs m in the clear.

Similarly, QR implements the following functions:

- $K_{QR}(PK_{QR}, SK_{QR}) \leftarrow KEYGEN_{QR}(\lambda)$ generates a key pair.
- $[m] \leftarrow$ encQR(m, PK_{QR}) encrypts clear bit m using public key PK_{QR} and outputs encrypted ciphertext $[m]$.
- $m \leftarrow$ decQR$([m], SK_{QR})$ decrypts given ciphertext $[m]$ using secret key SK_{QR} and outputs m in the clear.

For the simplicity, we will omit including keys and security parameters in the function parameters in the rest of the paper. The reason for using two homomorphic cryptosystems is efficiency but only one cryptosystem can be used as long as it is homomorphic and semantically secure.

4.3 Comparison of Encrypted Data

A primitive module used by many of the problems addressed in this paper is "comparison". Take mean certification as an example. The certifier can compute the average using the homomorphic encryption scheme. However, the next step is challenging: the certifier has to compare secret values against the average without learning any information about neither the average nor the secret values. There is no efficient and secure way for a certifier to perform the comparison herself. Therefore, we need a *collaboration of two parties* such that both will not know the values, but *together* they will be able to do the comparison. The proposed framework fits this requirement and the certifier is able to perform the comparison protocol with the help of a computation helper.

The certifier has two encrypted numbers $[\![a]\!] \leftarrow$encPL(a) and $[\![b]\!] \leftarrow$encPL(b) of ℓ bits and the computation helper has private keys SK_P and SK_{QR}. Both $[\![a]\!]$ and $[\![b]\!]$ are sent by parties. The goal of the comparison protocol is to decide whether $a \leq b$ without revealing the actual values of a and b to neither the certifier or the computation helper. Our comparison protocol is adapted from Veugen's [42] protocol. The main idea is to compute $2^\ell + b - a$ and check the most significant bit $(\ell + 1)$. If the most significant bit equals 1, then $a \leq b$, otherwise $a > b$. As a result of the protocol, the certifier gets the result of the comparison encrypted and the computation helper never learns the actual results of the inputs. Veugen's protocol has also been adapted and slightly modified by two recent works [7, 12].

To perform certification either with public or private outputs in our certification framework, we introduce two private comparison protocols, namely PRIVATECOMPARE and ENCRYPTEDPCOMPARE. They both takes encrypted inputs but PRIVATECOMPARE announces the output of the comparison publicly, while ENCRYPTEDPCOMPARE keeps the result of the comparison secret. Both protocols require joint computations between the two parties, and both of them are secure under the honest-but-curious security model.

Protocol 1 Two party private comparison with Public Output

Input A: $[\![a]\!]$, $[\![b]\!]$, PK_P, PK_{QR}, and SK_{QR}
Input B: SK_P
Output: bit t where $t = a \leq b$

1: **procedure** PrivateCompare($[\![a]\!]$, $[\![b]\!]$)
2: A: $[\![x]\!] \leftarrow [\![b]\!].[\![2^\ell]\!].[\![a]\!]^{-1}$ *mod* N ▷ $x \leftarrow b + 2^\ell - a$
3: A chooses a random number $r \leftarrow \{0,1\}^{\ell+\sigma}$
4: A: $[\![z]\!] \leftarrow [\![x]\!].[\![r]\!]$ *mod* N
5: A sends $[\![z]\!]$ to B
6: B: $z \leftarrow$ decPL$([\![z]\!])$
7: A: $c \leftarrow r$ *mod* 2^ℓ
8: B: $d \leftarrow z$ *mod* 2^ℓ
9: A and B privately compute the encrypted bit $[t']$ such that $t' = (d < c)$
10: A: $[r_{\ell+1}] \leftarrow$ encQR$(r_{\ell+1})$ and sends $[r_{\ell+1}]$ to B
11: B: $[z_{\ell+1}] \leftarrow$ encQR$(z_{\ell+1})$
12: B: $[t] \leftarrow [z_{\ell+1}].[r_{\ell+1}].[t']$ ▷ $t \leftarrow z_{\ell+1} \oplus r_{\ell+1} \oplus t'$
13: B sends $[t]$ to A
14: A: $t \leftarrow$ decryptQR(t)
15: **return** t

PRIVATECOMPARE compares two encrypted inputs and announces the result of the comparison publicly. The details of the protocol is summarized in Protocol 1. It is a joint computation of two parties, the certifier and the computation helper. The certifier has two encrypted numbers $[\![a]\!]$ and $[\![b]\!]$ and owns public keys PK_P, PK_{QR} and secret key SK_{QR}. On the other hand, the computation helper owns the secret key for Paillier, SK_P. The certifier initially computes $[\![x]\!] \leftarrow [\![b]\!].[\![2]\!]^\ell.[\![a]\!]^{-1}$ *mod* N and then hides it with a randomly chosen number, r. r should contain σ more bits than x. Next, the certifier sends $[\![z]\!]$ to the computation helper. Note that unless x was hidden by r, the computation helper could easily learn the comparison result. After receiving $[\![z]\!]$, the computation helper decrypts it and computes $d \leftarrow z$ *mod* 2^ℓ. In the meantime, the certifier computes $c \leftarrow r$ *mod* 2^ℓ. Then, the certifier and the computation helper cooperate to compare c and d $(t' \equiv d < c)$ using a private input comparison protocol. Although Veugen also proposes a private integer comparison protocol in [42], Bost et al. [12] suggest using the DGK protocol[14] for better practicality. This private integer comparison procedure is a sub-procedure in the protocol and either of the proposed protocols can be used in this protocol. After the execution of the private input comparison, the computation helper receives the encrypted bit $[t']$ as a result. Later, the certifier encrypts and sends the $(\ell+1)^{th}$ bit of r, $[r_{\ell+1}]$ to the computation helper. Finally, the computation helper computes the most significant bit of z by computing $[t] \leftarrow [z_{\ell+1}].[r_{\ell+1}].[t']$ and sends $[t]$ to the certifier. By using private key SK_{QR}, the certifier decrypts $[t]$ and announces t publicly.

Unlike PRIVATECOMPARE, ENCRYPTEDPCOMPARE aims to return both the comparison result and its negation privately. ENCRYPTEDPCOMPARE is summarized in Protocol 2. As in PRIVATECOMPARE, ENCRYPTEDPCOMPARE also requires the cooperation

Protocol 2 Two party private comparison with Private Output

Input A: $[\![a]\!]$, $[\![b]\!]$, PK_P, and PK_{QR}
Input B: SK_P and SK_{QR}
Output A: Encrypted Integer $[\![t]\!]$ where $(t = 1) \equiv a \leq b$
9: **procedure** EncryptedPCompare($[\![a]\!]$, $[\![b]\!]$)
Run the steps 2-9 of Protocol 1
10: A: $[r_{\ell+1}] \leftarrow$ encQR($r_{\ell+1}$)
11: B: $[z_{\ell+1}] \leftarrow$ encQR($z_{\ell+1}$) and sends $[z_{\ell+1}]$ to A
12: A: $[t] \leftarrow [z_{\ell+1}].[r_{\ell+1}].[t']$ ▷ $t \leftarrow z_{\ell+1} \oplus r_{\ell+1} \oplus t'$
Run re-encryption procedure
13: $[\![t]\!]$, $[\![\overline{t}]\!] \leftarrow$ ReEncForPL($[t]$) from Protocol 3

of both the certification and the computation helper. Although the protocols appear quite similar, they feature crucial differences in terms of the initial setup and the computation. The certifier owns two encrypted numbers-$[\![a]\!]$, $[\![b]\!]$- and public keys for both Paillier and QR cryptosystem, PK_P and PK_{QR}. On the other hand, the computation helper owns private keys for both Paillier and QR, SK_P and SK_{QR}. Until the private integer comparison, both the certifier and the computation helper follow the same procedures as they execute in Protocol 1 (line 2 to 9). Once line 9 is executed, i.e. the certifier and the computation helper have privately computed the encrypted bit $[t']$ such that $(t' = 1) \equiv (d < c)$, the certifier receives the result of the comparison encrypted $[t']$, and computes $[r_{\ell+1}]$. In the meantime, the computation helper encrypts $[z_{\ell+1}]$ and sends it to the certifier. Finally, the certifier computes $[t] \leftarrow [z_{\ell+1}].[r_{\ell+1}].[t']$ and has the result encrypted. The result is encrypted with the QR cryptosystem. Thus, the certifier and the computation helper jointly run the re-encryption protocol that returns both the resulting bit and its negate to the certifier encrypted under Paillier, i.e. $(t = 1 \equiv a \leq b) \Longleftrightarrow [\![t]\!]=[\![1]\!]$ and $[\![\overline{t}]\!]=[\![0]\!]$.

4.4 Re-encryption From QR to Paillier

PRIVATECOMPARE generates the result of the comparison encrypted under the QR cryptosystem (line 12 of Protocol 1). The plaintext space of QR is a bit, i.e. the result is either the encryption of 0 or 1. Although it is enough for learning the result of the comparison, to rank the inputs privately, our quantile based certification protocol needs to keep counters for comparison results without actually knowing the result. Therefore, we need to re-encrypt the resulting comparison bit to a corresponding integer value which is encrypted with Paillier. Re-encryption from the QR scheme to Paillier is performed such that the value of an encrypted bit is not revealed to any of the parties. Our implementation is adapted from [7] and slightly modified to meet the additional requirements. As presented in Protocol 3, to re-encrypt encrypted bit $[m]$, the certifier selects a random secret bit r, and then computes $[s_r]$=$[m].[0]$ and $[s_{1-r}]$=$[m].[1]$. The certifier sends $[s_r]$ and $[s_{1-r}]$ to the computation helper, thus, independently of the value of m, the computation helper receives the encryption of 0 and 1 every time. Then, the computation helper decrypts s_r and s_{1-r} under Paillier encryption, and sends $[\![s_r]\!]$ and $[\![s_{1-r}]\!]$ back together with their negates $[\![\overline{s_r}]\!]$, $[\![\overline{s_{1-r}}]\!]$ to the certifier in the same order as it received them. Since the certifier knows r, it uses $[\![s_r]\!]$ and $[\![\overline{s_r}]\!]$. $[\![\overline{s_{1-r}}]\!]$ and $[\![\overline{s_{1-r}}]\!]$ are disregarded.

Note that our comparison and re-encryption protocols are correct and secure. Due to space constraints, we omit further details, but

Protocol 3 Re-encrypt from QR to Paillier

Input A: $[m]$, PK_P, and PK_{QR}
Input B: SK_P and SK_{QR}
Output: $[\![m]\!]$ where $[\![m]\!] = [\![1]\!]$ if $m \equiv 1$. Else, $[\![m]\!] = [\![0]\!]$.
1: **procedure** ReEncForPL($[m]$)
2: A chooses a random bit $r \leftarrow \{0, 1\}$
3: A: $[s_r] \leftarrow [m].[0]$ ▷ $s_r \leftarrow m \oplus 0$
4: A: $[s_{1-r}] \leftarrow [m].[1]$ ▷ $s_{1-r} \leftarrow m \oplus 1$
5: A sends $[s_0]$ and $[s_1]$ to B
6: B: $s_0 \leftarrow$ decrQR($[s_0]$)
7: B: $[\![s_0]\!] \leftarrow$ encPL(s_0), $[\![\overline{s_0}]\!] \leftarrow$ encPL($s_0 \oplus 1$)
8: B: $s_1 \leftarrow$ decQR($[s_1]$)
9: B: $[\![s_1]\!] \leftarrow$ encPL(s_1), $[\![\overline{s_1}]\!] \leftarrow$ encPL($s_1 \oplus 1$)
10: B sends $[\![s_0]\!]$, $[\![s_1]\!]$, and their negates to A in the same order as received, i.e ($[\![s_0]\!]$, $[\![s_1]\!]$, $[\![\overline{s_0}]\!]$, $[\![\overline{s_1}]\!]$)
11: A: $[\![m]\!] \leftarrow [\![s_r]\!]$ and $[\![\overline{m}]\!] \leftarrow [\![\overline{s_r}]\!]$

the intuitions for the correctness and the security can be found in [7, 42].

5 CERTIFICATION PROTOCOLS

This section outlines how to deploy and perform the certification operations described in Section 3 in a privacy-preserving manner on top of the proposed framework model. Basically, n parties want to be certified through a certifier. To satisfy security guarantees, the computation helper, an on-site secure co-processor, helps the certifier execute protocols securely. Note that each certification problem has its own computation and security requirements, and these are highlighted explicitly. For simplicity, we assume all n parties join the computation.

5.1 Private Mean Based Certification

To perform *mean based certification*, the certifier needs to overcome two main challenges: (1) computing the average of n encrypted ciphertexts, (2) comparing each private input with the computed average privately.

Initialization. The secure co-processor executes the $K_P \leftarrow KEYGEN_P$ function to generate a key pair for the Paillier cryptosystem and shares public key PK_P with the certifier. Then, the certifier executes the key generation algorithm for QR, $K_{QR} \leftarrow KEYGEN_{QR}$. After key generation, the certifier sends PK_P to all parties and sends PK_{QR} to the secure co-processor.

Security Requirements. The individual inputs x_i will be kept confidential throughout the certification. In addition to this, the average value of the provided inputs must also be hidden from both the certifier and the secure co-processor. The final result of the computation will be made public. The system should also be secure against the existence of colluding parties.

Protocol. The parties encrypt their inputs with the Paillier cryptosystem using public key PK_P, $[\![x_i]\!] \leftarrow$encPL(x_i), and send the encrypted ciptertexts to the certifier. After receiving n inputs $[\![x_1]\!]$, $[\![x_2]\!]$, ..., $[\![x_n]\!]$, the certifier executes the MEAN-CERTIFY algorithm which is presented in Protocol 4. The protocol starts by computing the summation of the private inputs. Using the homomorphic property of Paillier, the certifier computes $[\![s]\!] \leftarrow [\![s]\!].X[i]$ *mod* N. This operation yields $s \leftarrow s + x_i$ and after this is executed on all

Protocol 4 Mean Certification

Input Party$_i$: x_i and PK_P
Input Certifier: SK_{QR}, and PK_P
Input Secure Coprocessor: SK_P and PK_{QR}
Output: c_i (certification result for each party)

1: **procedure** SendToCertifier(x_i)
2: $[\![x_i]\!] \leftarrow$ encPL(x_i)
3: Send $[\![x_i]\!]$ to the certifier
4: Each party executes SendToCertifier(x_i)

After receiving all inputs, $X[1..n] = \{[\![x_1]\!], [\![x_2]\!],...,[\![x_n]\!]\}$, the certifier executes the following procedure.

5: **procedure** Mean-Certify($X[1..n]$)
Compute the sum of inputs
6: $[\![s]\!] \leftarrow X[1]$
7: **for** $i \leftarrow 2$ to n **do**
8: $[\![s]\!] \leftarrow [\![s]\!].X[i] \ mod \ N$ ▷ $s \leftarrow s + x_i$
Note that $s = \sum_{i=1}^{n} x_i$
9: **for** $i \leftarrow 1$ to n **do**
10: $[\![\tilde{x_i}]\!] \leftarrow X[i]^n \ mod \ N$ ▷ $\tilde{x_i} \leftarrow n \times x_i$
11: $t_i \leftarrow$ PrivateCompare($[\![s]\!]$, $[\![\tilde{x_i}]\!]$)
12: **if** $t_i == 1$ **then**
13: $c_i \leftarrow$ Above
14: **else**
15: $c_i \leftarrow$ Below
16: Sends c_i to P_i

inputs, the resulting computation will be the summation of all inputs encrypted with Paillier, $[\![s]\!]$. The Paillier cryptosystem does not support a division operation. Rather than computing the average, i.e. dividing the summation by n, the certifier normalizes the inputs by multiplying them by the number of participants, i.e. $[\![\tilde{x_i}]\!] \leftarrow [\![x_i]\!]^n \ mod \ N$. Recall that our main goal is to compare x_i with the average, ie, $x_i \leq sum/n$. The basic idea for this comparison is $x_i \leq \frac{sum}{n} \equiv x_i * n \leq sum$ where $\frac{sum}{n}$ is the average. Recall that the Paillier cryptosystem supports multiplying ciphertext with a constant. After normalizing the input, the certifier and the secure co-processor jointly execute the PrivateCompare function, introduced in Section 4.3, to compare $[\![s]\!]$ with the normalized input $[\![\tilde{x_i}]\!]$. The result of this comparison is known by the certifier in the clear and the certification is completed by labeling the party with input x_i as *above* or *below*.

Correctness. The certifier computes the summation of n private inputs using the additive homomorphic operation of Paillier which executes the summation operation over ciphertexts. Because of the homomorphic property of Paillier, lines 3 through 5 of Protocol 4 compute the encrypted summation of n inputs. Each encrypted input $[\![x_i]\!]$ is normalized by taking the power of n under modular arithmetic, which is equivalent to $[\![\tilde{x_i}]\!] \leftarrow [\![x_i \times n]\!]$ due to the Paillier properties. The comparison of each private input with the average of n private input is equal to the comparison of normalized input with the summation of n inputs, i.e. $\frac{sum}{n} \leq x_i \equiv sum \leq x_i \times n$ where $sum \leftarrow \sum_{i=1}^{n} x_i$. After executing the private comparison, a party is certified as *above* if $sum \leq x_i * n$. Otherwise, the label is *below*.

Intuition of Security Proof. The certifier receives the inputs encrypted with Paillier from the parties. Since, it does not own the secret key SK_P, it cannot decrypt and learn the actual inputs. The homomorphic addition is semantically secure due to the Paillier cryptosystem, and the certifier computes the summation encrypted under Paillier. The comparison protocol is already proved secure [42] and does not reveal any information. Recall that the only restriction on collusion is between the certifier and the helper. Hence, we need to prove that collusion between the certifier and any number of parties will not reveal any private parties. Assume $n-1$ parties collude with the certifier except party P_1. In such a case, the certifier has $x_2, x_3, ..., x_n$ in the clear and x_1 encrypted, i.e. $[\![x_1]\!]$. Throughout the computation, the certifier computes *sum* encrypted which is denoted as $[\![s]\!]$ in Protocol 4. The certifier can compute $C_{sum} = \sum_{i=2}^{n} x_i$ in the clear. If *sum* was in clear, knowing C_{sum} would help computing $s - C_{sum} \equiv x_1$. However, since s is encrypted, the subtraction results in $[\![s]\!].[\![C_{sum}]\!]^{-1} \equiv [\![s - C_{sum}]\!] \equiv [\![x_1]\!]$. As it can be easily inferred from the result, the colluding parties do not provide any useful information to the certifier to reveal x_1. Hence, our private mean based certification protocol is secure even under the existence of colluding parties, since neither the certifier nor the secure co-processor learn any intermediary results throughout the computation.

5.2 Private k-Quantile Certification

To split parties into distinct groups privately, the quantile based certification is performed. Although the quantile computation is directly related to the ranking of a set of inputs, the certifier computes the k-quantiles privately without learning the ordering of the private inputs.

Initialization. The secure co-processor generates key pairs, K_P and K_{QR}, for both Paillier and QR. It owns *both* private keys SK_P and SK_{QR}, and sends the public keys, PK_P and PK_{QR}, to the certifier. Then, the certifier shares the public key for Paillier, PK_P, with the parties. We assume that the certifier knows the parameter k.

Security Requirements. Throughout the certification process, the individual inputs should be kept secret as in prior certifications. By the nature of quantile computations, the order information among different groups, i.e. the order of the parties in different groups, will be revealed. However, the ordering information of parties inside the same quantile group should not be revealed. We assume that the parties do not collude in this certification method[1].

Protocol. Parties encrypt their inputs with Paillier, $[\![x_i]\!]$, and send them to the certifier. After receiving n inputs, the certifier executes the QUANTILE-CERTIFY algorithm which is presented in Protocol 5. At a high-level, the protocol privately compares each possible pair securely. The aim is to construct a private comparison matrix, where the pairwise comparison is hidden from the certifier using Paillier. Later, by computing the sum of each column in the comparison matrix, the certifier figures out the rank of the corresponding parties, which will be used later to split parties into k groups.

The public pairwise comparisons of all pairs reveal the order of the inputs, which obviously violates the security constraint. Therefore, the protocol initially performs private pairwise comparison

[1]This is a natural problem of quantile based private grouping. If the parties from neighbor groups collude with each other, due to the ordering, it might be possible to reveal the input of non-colluding party in one of these groups.

Protocol 5 k-Quantile Certification

Input Party$_i$: x_i and PK_P
Input Certifier: k, PK_P, and PK_{QR}
Input Secure Coprocessor: SK_{QR}, SK_P and PK_{QR}
Output: c_i (certification result for each party)

1: Each party executes SendToCertifier(x_i) from Protocol 4
After receiving all inputs, $X[1..n] = \{[\![x_1]\!],...,[\![x_n]\!]\}$, the certifier executes the following.
2: **procedure** Quantile-Certify($X[1..n]$, k)
Private pairwise comparisons of inputs
3: $C[1..n][1..n] \leftarrow$empty
4: **for** $i \leftarrow 1$ to n **do**
5: **for** $j \leftarrow i+1$ to n **do**
6: $[t_{ij}] \leftarrow$ EncryptedPCompare($X[i]$, $X[j]$) ▷ $t_{ij} \leftarrow x_i \leq x_j$
7: $[\![t_{ij}]\!]$, $[\![\overline{t_{ij}}]\!] \leftarrow$ ReEncForPL($[t_{ij}]$)
8: $C[i][j] \leftarrow [\![t_{ij}]\!]$ and $C[j][i] \leftarrow [\![\overline{t_{ij}}]\!]$
9: $c[1..n] \leftarrow$ empty
10: **for** $j \leftarrow 1$ to n **do**
11: $[\![sum]\!] \leftarrow [\![0]\!]$
12: **for** $i \leftarrow 1$ to n **do**
13: **if** $i \neq j$ **then**
14: $[\![sum]\!] \leftarrow [\![sum]\!].C[i][j]$ *mod* N
15: $c[j] \leftarrow [\![sum]\!]$
16: Choose a random permutation π over $\{1, .., n\}$
17: **for** $i \leftarrow 1$ to n **do**
18: $c_\pi[i] \leftarrow c[\pi(i)]$
19: $R[1..n] \leftarrow$ComputeBin($c_\pi[]$, n, k)
20: **for** $i \leftarrow 1$ to n **do**
21: $j \leftarrow \pi^{-1}(i)$
22: $c_j \leftarrow R[i]$
23: Send c_j to party P_j
24: **procedure** ComputeBin($V[1..n]$, n, k)
25: $R[1..n] \leftarrow$ empty
26: **for** $i \leftarrow 1$ to n **do**
27: $v \leftarrow$ decPL($V[i]$)
28: $R[i] \leftarrow \lceil \frac{n-v}{n/k} \rceil$
29: **return** R

for all pairs using EncryptedPCompare introduced earlier in Section 4.3 which returns the resulting bit $[t_{ij}]$ of comparison $x_i \leq x_j$ encrypted. Using the re-encryption function, the resulting bit is transformed to a Paillier scheme. Additionally, the negation of the result is also provided to the certifier. This will allow the certifier to construct a private comparison matrix C_{ij} where $i, j \in \{1, .., n\}$ as shown is Figure 2. Briefly, if $x_i \leq x_j$, then the comparison will return $[\![t_{ij}]\!]$= 1 and $[\![\overline{t_{ij}}]\!]$= 0. These results are stored in the indexes of C_{ij} and C_{ji}. Consider an example in Figure 2 where $x_1 = 3$ and $x_3 = 5$. The comparison of x_1 and x_3 is $x_1 \leq x_3 \equiv 3 \leq 5 \equiv 1$. Hence, $C_{13} = 1$ and $C_{31} = 0$. After comparing all pairs, the certifier computes the columnwise summation of all entries in the comparison matrix using homomorphic addition. The columnwise summation will give the number of ones, i.e. the input is greater than or equal to how many other inputs. Therefore, the summed values in the resulting vector show the ranking among n parties, i.e. if the entry

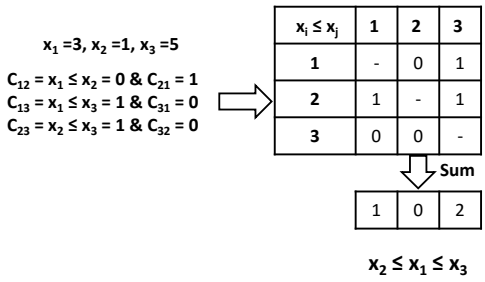


Figure 2: Private Comparison for Ordering

in index i of the resulting vector is 0, that means the input x_i is the minimum input. If it is $n-1$, that means x_i is greater than all other inputs and it is the maximum. Consider the example in Figure 2. After the columnwise summation, the resulting vector is $< 1, 0, 2 >$, which means x_i is greater than one input, x_2 is not greater than or equal to any of the other inputs, and x_3 is greater than equal to two parties. Hence, the resulting vector shows the ranking of the corresponding inputs. Recall the certifier does not own SK_P; thus, it cannot learn the ordering information. To prevent the secure co-processor from learning the order of the values in the resulting vector $c[1..n]$, the certifier applies a random permutation π. The i^{th} element of c is stored at index $\pi(i)$, $c_\pi[i] \leftarrow c[\pi(i)]$. Then, the permuted result vector is sent to the secure co-processor. The secure co-processor decrypts the entries in the permuted resulting vector, and computes the group of the inputs. After computing groups for all inputs, the secure co-processor returns the group vector, R, to the certifier. The certifier can compute $j \leftarrow \pi^{-1}(i)$ which represents the j^{th} index in the unpermuted order. After unpermuting the orders, the certifier returns the corresponding results to the parties, $c_j \leftarrow R[j]$, where c_j is the quantile rank of the j^{th} party.

Correctness. Th certifier first compares all pairs and constructs a comparison matrix such that $\forall i, j x_i \leq x_j \Leftrightarrow C_{ij} \leftarrow 1$ and $C_{ji} \leftarrow 0$ where $i \neq j$. The comparison can be one of the followings: (1) $x_i < x_j$, (2) $x_i = x_j$, and (3) $x_i > x_j$. For cases 1 and 3, the numbers are distinct, and the output of comparisons are $C_{ij} \leftarrow 1$ and $C_{ij} \leftarrow 0$, respectively. In case 2, the numbers are equal and it returns $C_{ij} \leftarrow 1$. In this case, $C_{ji} \leftarrow 0$. This means x_i is not greater than x_j. Although $x_i = x_j$, the comparison selects x_j greater and ranks it higher. The columnwise summation of the comparison matrix will form a resulting vector which shows the ranking of the inputs among all n parties. The smallest input will have an entry of 0 and the maximum input will have an entry of $n-1$ which says this input is greater than or equal to $n-1$ other entries. Thus, the resulting vector will have entries from 0 to $n-1$ which are the ranks of the inputs. The correctness of the rest of the protocol is straightforward. The resulting vector has entries $0, 1, ..., n-1$ in some order. The secure co-processor decrypts the entries and split inputs into k groups (quantiles) based on their order among the n parties. For example, the inputs with entries $0, 1, ..., k-1$ will be in the first group.

Intuition of Security Proof. The certifier receives the inputs encrypted with Paillier. The certifier initially compares all pairs using the function ENCRYPTEDPCOMPARE which is followed by the execution of the re-encryption function. The private comparison and re-encryption functions are already proved secure in [7] and

they do not reveal any information. The results of the pairwise comparisons are encrypted with Paillier and the certifier cannot decrypt the results due to its lack of knowledge of the private key, SK_P. To rank the inputs, the certifier computes the columnwise summation of the comparison matrix using the additive homomorphic properties of Paillier. Therefore, it does not learn any information about the inputs and the pairwise comparisons. On the other hand, the secure co-processor receives the resulting vector permuted. Although it decrypts entries in the permuted vector, it cannot infer any information about the relationship between the results and the parties, since it does not know the permutation. At the end of the certification, groups(quantiles) of parties are public, but neither the certifier nor the secure co-processor learn any information about the ordering of parties inside the same group. Thus, the quantile based certification is secure.

5.3 Private Certification with Private Outputs

So far, the certification results are made public. As was discussed earlier, mutually competitive firms might want to gain private knowledge about their performances without revealing the result of the certification to the certifier, the computation helper and other parties. We now describe necessary modifications to perform such certifications with private outputs.

5.3.1 Mean Based Certifications with Private Outputs. The framework initializes the same setup as in the corresponding certification with public outputs except that a key pair K_{QR} is generated by the secure co-processor. SK_{QR} is only owned by the secure co-processor and PK_{QR} is shared with the certifier. To compare the private input with the encrypted threshold value, the certifier invokes the EncryptedPCompare function from Protocol 2 until line 12 instead of the PRIVATECOMPARE function inside the MEAN-Certify function. Line 12 from the EncryptedPCompare function returns the result of the comparison encrypted with QR, $[t]$, to the certifier. Since, the certifier does not own SK_{QR}, it cannot decrypt and learn the result of the comparison. The certifier sends the resulting bits to the parties encrypted. The parties do not own the secret key SK_{QR}, thus, they need help from the secure co-processor to learn the actual results. To prevent the certifier and the secure co-processor from learning the actual results, the parties randomize their inputs by applying the same logic as in Protocol 3. In brief, each party chooses a random bit, r, and then computes $s_r \leftarrow [t].[0]$ and $s_{1-r} \leftarrow [t].[1]$. Both s_r and s_{1-r} are independent from the value of the resulting bit t. Each party sends their s_r and s_{1-r} to the certifier and the certifier sends them to the secure co-processor. The secure co-processor decrypts both of them and returns the unencrypted results to the certifier in the order received. The certifier also does the same and sends the unencrypted s_r and s_{1-r} to the corresponding party. Since the party knows r, it selects the correct result. If the result is 1, the party knows the label is *above*; otherwise, it is *below*.

5.3.2 Quantile based Certification with Private Outputs. The framework uses the same setup introduced in Section 5.2. The certifier executes the QUANTILE-CERTIFY functions as it is until line 21 in Protocol 5, where the secure co-processor computes the groups (quantiles) of inputs based on their order. After the secure co-processor computes the groups, it encrypts the entries of R using the COMPUTEBIN function, which are the group numbers (quantiles) of the inputs, with Paillier. Then, the secure co-processor returns R to the certifier. Since the certifier does not own SK_P, it cannot decrypt and learn which party is placed in which group. The certifier executes the rest of the protocol as is and sends the results to the parties encrypted. The parties do not have the secret key SK_P. Therefore, they need help from the secure co-processor. To hide the real results (c in this case), each party selects a large enough random number r, and executes $[\![s]\!] \leftarrow [\![c]\!].[\![r]\!]$ which is equivalent to $[\![s]\!] \leftarrow [\![c + r]\!]$. Then, each party sends their inputs to the certifier and the certifier also sends these inputs to the secure co-processor. After decrypting $[\![s]\!]$, the secure co-processor sends s to the certifier in the clear. Note that since the random number r is hidden from both the certifier and the secure co-processor, they cannot learn the actual group number of the party. The certifier sends s back to the corresponding party. Upon receiving s, a party executes $c \leftarrow s - r$ and learns the group of the party.

6 PERFORMANCE

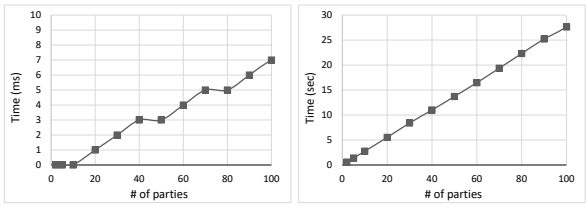
To show the performance analysis of our framework and algorithms, in this section, we present both empirical and complexity analysis for both the mean and k-quantile certifications.

6.1 Complexity Analysis

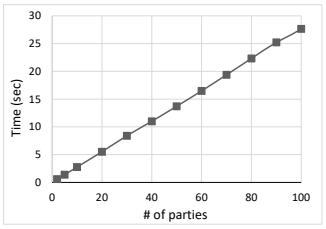
Mean and quantile based certifications rely on comparing encrypted data. This paper proposes two comparisons protocols, PRIVATECOMPARE and ENCRYPTEDPCOMPARE, which are adapted from Veugen's [42] protocol. Veugen discusses the complexity analysis of the encrypted comparison protocol and shows that encrypted comparison has a very low computation complexity. The main computation complexity occurs while two private integers are being compared. In the same paper, Veugen proposes a Lightweight Secure Integer Comparison (LSIC) which requires l rounds of communications plus half a round at the beginning. Our prototype also implements the LSIC algorithm to compare two integers privately. Both PRIVATECOMPARE and ENCRYPTEDPCOMPARE have one more round for transferring z and $[t]$. Therefore, our comparison protocols require $l + 1.5$ rounds of communications between the certifier and the computation helper (e.g. assuming a 32-bit integer domain: 33.5). In addition, the re-encryption procedure requires one round of communications.

6.2 Empirical Analysis

We implemented a prototype of the proposed framework in Java. The certifier is run on a Windows machine with i5-2320 3 GHZ CPU and 8 GB memory. On the other hand, the computation helper is run on a machine running Linux with Intel Xeon(R) E31235 3.20 GHZ CPU and 32 GB memory. Both machines are on the same network and the average latency between them is 0.1 ms. The parties are run on the same machine with the certifier. The data domain is 32-bit integers. The conducted experiments measure the execution time to evaluate system performance by varying the number of participating parties. The size of the keys for both the Paillier and the QR cryptosystems are set to 2048 bits.

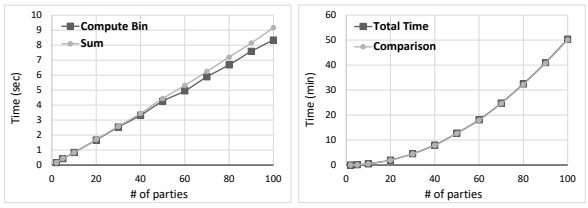


(a) Homomorphic sum **(b) Time spent in comparisons**



(c) Total execution time

Figure 3: Results of Private Mean Certification



(a) Homomorphic Sum and Bin Computation **(b) Time spent in comparisons**

Figure 4: Results of Private Quantile Certification

6.2.1 Mean Certification. The mean certification initially computes the average of inputs, and then compares each input with the average. Figure 3(a) and 3(c) present the execution times for homomorphic summation and total certification times, respectively.

Homomorphic summation is performed with modular multiplication. It is cheaper compared to encryption and decryption and this is also validated in our experiments. For very small number of parties, the average computation is performed in 0 or 1 ms. In the worst case, the homomorphic summation takes 7 ms (number of parties = 100). These results are very promising for other privacy-preserving database applications which need to perform aggregate operations as part of query executions.

The execution time of the mean based certification is dominated by the comparisons with the average (Figure 3(b)). The mean certification requires n comparisons against the computed average. In our implementation, the comparisons are sequential, therefore, both total execution time and time spent in comparisons have linear behavior. As the number of participating parties increases, the total execution time also increases. It is possible to perform comparisons in parallel which will decrease the total execution time, though this paper does not discuss and implement parallelism. Even without such an optimization, the total certification times take seconds, with a maximum of 27.6 seconds when 100 parties participate. This is still well below a minute, and hence for many applications, especially environmental certification, is very reasonable.

6.2.2 4-Quantile Certification. The quantile certification requires pair-wise comparison of each input data, which requires $(n^2 - n)/2$ comparisons. This quadratic behavior causes longer certification times as the number of participants increases as depicted in Figure 4(b). The other important sub-procedures inside the quantile certification protocol are the homomorphic summation of comparison values and the grouping computations. We set k to 4, that means the parties are split into 4 groups. The computation helper maps parties into groups in linear time. On the other hand, to get the final scores encrypted, the certifier performs $n^2 - n$ homomorphic summations before computing the bins. The certification of 20 participants is performed within 2 minutes though it takes slightly more than 50 minutes when there are 100 participants. Since this is an off-line operation, such execution times are reasonable. However, it is expected to have 20-30 participants most of the time and the quantile based certification can be done within a few minutes in such settings, which is pretty efficient.

Discussion. Our algorithms and framework enable achieving significant functionality with reasonable computation performance without sacrificing any performance. Our evaluations show the advantage of the usage of secure co-processors as a computation helper on site. Recall that the average network latency between the certifier and the computation helper is 0.1 ms in our experiments, which makes the cost of rounds of interactions among two parties negligible compared to the computation cost. An on site secure co-processor also makes the network transmission time negligible. Recall that the encrypted comparison operations require $l + 1.5$ rounds of communication and the mean certification requires n comparisons while the k-quantile comparison requires $(n^2 - n)/2$ comparisons, which makes $n(l + 1.5)$ and $(n^2 - n)(l + 1.5)/2$ rounds of communications, respectively. A setting where there is a non-negligible latency between the certifier and the computation helper will result in drastic performance degradation. Therefore, a secure co-processor perfectly fits the proposed model.

7 CONCLUSION

In this paper, we formally define the *privacy preserving certification* paradigm to evaluate the environmental impacts of industrial processes privately and propose solutions for two certification problems-mean, quantile. To perform privacy preserving certifications without compromising any sensitive information, we propose a framework, which enables a certifier to certify parties based on a well agreed upon set of criteria under realistic network setting. The paper also presents efficient and provably secure algorithms for the certification. Our prototype demonstrates that the proposed approach is not only secure but also efficient and practical.

Although the certification process is typically performed off-line, and hence might not require strict time constraints to complete the certification process, other applications might require instant feedback or certification based on the input, e.g., privacy preserving online auction system. Our framework is shown to be efficient for such application scenarios as well.

ACKNOWLEDGMENT

This work is partly funded by NSF grants CNS-1528178 and CCF-1442966.

REFERENCES

[1] Sean W. Smith and Steve Weingart (Eds.). 1999. Building a High-performance, Programmable Secure Coprocessor. *Comput. Netw.* 31, 9 (April 1999), 831–860. http://dl.acm.org/citation.cfm?id=324119.324128

[2] 2011. TPM Main Specification. (March 2011). http://www.trustedcomputinggroup.org/tpm-main-specification/.

[3] 2012. IBM 4764 product and PCIXCC feature overview. (March 2012). https://www-03.ibm.com/security/cryptocards/pcixcc/overview.shtml.

[4] Arvind Arasu, Spyros Blanas, Ken Eguro, Raghav Kaushik, Donald Kossmann, Ravi Ramamurthy, and Ramaratnam Venkatesan. 2013. Orthogonal security with cipherbase. In *Proc. of the 6th CIDR, Asilomar, CA*.

[5] Michael Backes, Aniket Kate, Matteo Maffei, and Kim Pecina. 2012. ObliviAd: Provably Secure and Practical Online Behavioral Advertising. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy (SP '12)*. IEEE Computer Society, Washington, DC, USA, 257–271. DOI: https://doi.org/10.1109/SP.2012.25

[6] Sumeet Bajaj and Radu Sion. 2011. TrustedDB: a trusted hardware based database with privacy and data confidentiality. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2011, Athens, Greece, June 12-16, 2011*. 205–216. DOI: https://doi.org/10.1145/1989323.1989346

[7] Foteini Baldimtsi and Olga Ohrimenko. 2015. Sorting and Searching Behind the Curtain. In *Financial Cryptography and Data Security - 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers*. 127–146. DOI: https://doi.org/10.1007/978-3-662-47854-7_8

[8] Carsten Baum, Ivan DamgÃërd, and Claudio Orlandi. 2014. Publicly Auditable Secure Multi-Party Computation. *Security and Cryptography for Networks* (2014), 175–196. DOI: https://doi.org/10.1007/978-3-319-10879-7_11

[9] Mihir Bellare, Viet Tung Hoang, Sriram Keelveedhi, and Phillip Rogaway. 2013. Efficient garbling from a fixed-key blockcipher. In *Security and Privacy (SP), 2013 IEEE Symposium on*. IEEE, 478–492.

[10] Beth R. Beloff, Jeanette M. Schwarz, and Earl Beaver. 2002. Use Sustainability Metrics to Guide Decision-Making. *Chemical Engineering Progress* 98, 7 (July 2002), 58–63.

[11] Dan Bogdanov, Liina Kamm, Sven Laur, and Ville Sokk. 2014. *Rmind: a tool for cryptographically secure statistical analysis*. Technical Report. Cryptology ePrint Archive, Report 2014/512.

[12] Raphael Bost, Raluca Ada Popa, Stephen Tu, and Shafi Goldwasser. 2015. Machine Learning Classification over Encrypted Data. In *22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2014*. http://www.internetsociety.org/doc/machine-learning-classification-over-encrypted-data

[13] Ivan Damgård, Martin Geisler, and Mikkel Krøigaard. 2007. Efficient and secure comparison for on-line auctions. In *Information security and privacy*. Springer, 416–430.

[14] Ivan Damgård, Martin Geisler, and Mikkel Krøigaard. 2009. A correction to 'efficient and secure comparison for on-line auctions'. *IJACT* 1, 4 (2009), 323–324. DOI: https://doi.org/10.1504/IJACT.2009.028031

[15] Adriana Del Borghi. 2012. LCA and communication: Environmental Product Declaration. *The International Journal of Life Cycle Assessment* 18, 2 (Oct 2012), 293–295. DOI: https://doi.org/10.1007/s11367-012-0513-9

[16] Magali Delmas and Vered Doctori Blass. 2010. Measuring corporate environmental performance: the trade-offs of sustainability ratings. *Bus. Strat. Env.* 19, 4 (Apr 2010), 245–260. DOI: https://doi.org/10.1002/bse.676

[17] Annik Magerholm Fet and Christofer Skaar. 2006. Eco-labeling, Product Category Rules and Certification Procedures Based on ISO 14025 Requirements (6 pp). *The International Journal of Life Cycle Assessment* 11, 1 (Jan 2006), 49–54. DOI: https://doi.org/10.1065/lca2006.01.237

[18] Craig Gentry. 2009. Fully homomorphic encryption using ideal lattices. In *STOC '09: Proceedings of the 41st annual ACM symposium on Theory of computing*. 169–178.

[19] Bart Goethals, Sven Laur, Helger Lipmaa, and Taneli Mielikäinen. 2005. On private scalar product computation for privacy-preserving data mining. In *Information Security and Cryptology–ICISC 2004*. Springer, 104–120.

[20] Shafi Goldwasser and Silvio Micali. 1984. Probabilistic encryption. *Journal of computer and system sciences* 28, 2 (1984), 270–299.

[21] Reinout Heijungs and Sangwon Suh. 2002. *The computational structure of life cycle assessment*. Vol. 11. Springer Science & Business Media.

[22] Eva Heiskanen. 2002. The institutional logic of life cycle thinking. *Journal of Cleaner Production* 10, 5 (Oct 2002), 427–437. DOI: https://doi.org/10.1016/s0959-6526(02)00014-8

[23] ISO 14044. 2006. *Environmental management — Life cycle assessment — Requirements and guidelines*. ISO, Geneva, Switzerland.

[24] Josef Kaenzig, Damien Friot, Myriam SaadÃĪ, Manuele Margni, and Olivier Jolliet. 2010. Using life cycle approaches to enhance the value of corporate environmental disclosures. *Bus. Strat. Env.* 20, 1 (Dec 2010), 38–54. DOI: https://doi.org/10.1002/bse.667

[25] Jonathan Katz. 2007. Universally composable multi-party computation using tamper-proof hardware. In *Advances in Cryptology-EUROCRYPT 2007*. Springer, 115–128.

[26] Florian Kerschbaum. 2008. Practical Privacy-Preserving Benchmarking. In *Proceedings of The IFIP TC-11 23rd International Information Security Conference, IFIP 20th World Computer Congress, IFIP SEC 2008, September 7-10, 2008, Milano, Italy*. 17–31. DOI: https://doi.org/10.1007/978-0-387-09699-5_2

[27] Florian Kerschbaum. 2011. Secure and Sustainable Benchmarking in Clouds - A Multi-Party Cloud Application with an Untrusted Service Provider. *Business & Information Systems Engineering* 3, 3 (2011), 135–143. DOI: https://doi.org/10.1007/s12599-011-0153-9

[28] Florian Kerschbaum, Debmalya Biswas, and Sebastiaan de Hoogh. 2009. Performance Comparison of Secure Comparison Protocols. In *Database and Expert Systems Applications, DEXA, International Workshops, Linz, Austria, August 31-September 4, 2009, Proceedings*. 133–136. DOI: https://doi.org/10.1109/DEXA.2009.37

[29] Florian Kerschbaum, Jens Strüker, and Thomas G. Koslowski. 2011. Confidential Information-Sharing for Automated Sustainability Benchmarks. In *Proceedings of the International Conference on Information Systems, ICIS 2011, Shanghai, China, December 4-7, 2011*. http://aisel.aisnet.org/icis2011/proceedings/breakthroughideas/4

[30] Florian Kerschbaum and Orestis Terzidis. 2006. Filtering for Private Collaborative Benchmarking. In *Emerging Trends in Information and Communication Security, International Conference, ETRICS 2006, Freiburg, Germany, June 6-9, 2006, Proceedings*. 409–422. DOI: https://doi.org/10.1007/11766155_29

[31] Brandon Kuczenski. 2015. Partial ordering of life cycle inventory databases. *The International Journal of Life Cycle Assessment* 20, 12 (Oct 2015), 1673–1683. DOI: https://doi.org/10.1007/s11367-015-0972-x

[32] K.M. Lee and H.D. Stensel. 1999. ISO standards on environmental labels and declarations and its implications on the market. *Proceedings First International Symposium on Environmentally Conscious Design and Inverse Manufacturing* (1999). DOI: https://doi.org/10.1109/ecodim.1999.747664

[33] Yehuda Lindell and Benny Pinkas. 2008. Secure Multiparty Computation for Privacy-Preserving Data Mining. *IACR Cryptology ePrint Archive* 2008 (2008), 197. http://eprint.iacr.org/2008/197

[34] Dahlia Malkhi, Noam Nisan, Benny Pinkas, Yaron Sella, and others. 2004. Fairplay-Secure Two-Party Computation System.. In *USENIX Security Symposium*, Vol. 4. San Diego, CA, USA.

[35] David Morrow and Dennis Rondinelli. 2002. Adopting Corporate Environmental Management Systems:. *European Management Journal* 20, 2 (Apr 2002), 159–171. DOI: https://doi.org/10.1016/s0263-2373(02)00026-9

[36] Katsuyuki Nakano and Masahiko Hirao. 2011. Collaborative activity with business partners for improvement of product environmental performance using LCA. *Journal of Cleaner Production* 19, 11 (Jul 2011), 1189–1197. DOI: https://doi.org/10.1016/j.jclepro.2011.03.007

[37] Pascal Paillier. 1999. Public-key Cryptosystems Based on Composite Degree Residuosity Classes. In *Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques (EUROCRYPT'99)*. Springer-Verlag, Berlin, Heidelberg, 223–238. http://dl.acm.org/citation.cfm?id=1756123.1756146

[38] G. Rebitzer, T. Ekvall, R. Frischknecht, D. Hunkeler, G. Norris, T. Rydberg, W. -P. Schmidt, S. Suh, B. P. Weidema, and D. W. Pennington. 2004. Life cycle assessment: Part 1: Framework, goal and scope definition, inventory analysis, and applications. *Environ. Int.* 30, 5 (July 2004), 701–720. DOI: https://doi.org/10.1016/j.envint.2003.11.005

[39] Nuno Santos, Rodrigo Rodrigues, Krishna P. Gummadi, and Stefan Saroiu. 2012. Policy-Sealed Data: A New Abstraction for Building Trusted Cloud Services. In *Presented as part of the 21st USENIX Security Symposium (USENIX Security 12)*. USENIX, Bellevue, WA, 175–188. https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/santos

[40] Bruce Schneier. 2009. Homomorphic Encryption Breakthrough. (2009). http://www.schneier.com/blog/archives/2009/07/homomorphic_enc.html, 2009.

[41] UNEP/SETAC. 2011. *Global Guidance Principles for Life Cycle Assessment Databases*. Technical Report. United Nations Environment Programme.

[42] Thijs Veugen. 2011. Comparing encrypted data. *Multimedia Signal Processing Group, Delft University of Technology, The Netherlands and TNO Information and Communication Technology, Delft, Tech. Rep* (2011).

[43] Cong Wang, Sherman S.M. Chow, Qian Wang, Kui Ren, and Wenjing Lou. 2013. Privacy-Preserving Public Auditing for Secure Cloud Storage. *IEEE Trans. Comput.* 62, 2 (Feb 2013), 362–375. DOI: https://doi.org/10.1109/tc.2011.245

[44] Gregor Wernet, Stavros Papadokonstantakis, Stefanie Hellweg, and Konrad HungerbÃijhler. 2009. Bridging data gaps in environmental assessments: Modeling impacts of fine and basic chemical production. *Green Chem.* 11, 11 (2009), 18–26. DOI: https://doi.org/10.1039/b905558d

[45] World Steel Association. 2011. *Life cycle inventory study for steel products*. Technical Report. World Steel Association.

[46] Andrew Chi-Chih Yao. 1982. Protocols for secure computations. In *FOCS*, Vol. 82. 160–164.